

# **EMC<sup>®</sup> Documentum<sup>®</sup> Platform REST Services**

**Version 7.3**

## **Reference Guide**

EMC Corporation  
*Corporate Headquarters*  
Hopkinton, MA 01748-9103  
1-508-435-1000  
[www.EMC.com](http://www.EMC.com)

**Legal Notice**

Copyright © 2013 – 2016 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com. Adobe and Adobe PDF Library are trademarks or registered trademarks of Adobe Systems Inc. in the U.S. and other countries. All other trademarks used herein are the property of their respective owners.

**Documentation Feedback**

Your opinion matters. We want to hear from you regarding our product documentation. If you have feedback about how we can make our documentation better or easier to use, please send us your feedback directly at [ECD.Documentation.Feedback@emc.com](mailto:ECD.Documentation.Feedback@emc.com)

# Table of Contents

---

<b>Preface</b>	35
<b>Chapter 1 Overview</b>	37
Hypermedia-driven API	37
<b>Chapter 2 Resources</b>	41
Organization of Resource Reference Documentation	41
Additional Services Installation	43
Collaboration Service	43
Search Service	43
All Versions	44
Resource Relationships	44
Feed	44
Link Relations	44
Operations	44
Get Versions	45
Supported HTTP Method	45
Request Media Types	45
Request Query Parameters	45
Request Headers	46
Request Body	46
Response Headers	47
Response Media Types	47
Response Status	47
Response Body	47
Check in an object	47
Supported HTTP Method	47
Request Media Types	48
Request Query Parameters	48
Request Headers	57
Request Body	57
Request Payload Samples	57
Request Payload Samples	58
Response Headers	59
Response Media Types	59
Response Status	59
Response Body	59
ACL(s)	60
ACL	60
Resource Relationships	60
Feed	60
Link Relations	60
Operations	60
Get an ACL Object Instance	61
HTTP Method	61
Request Media Types	61
Request Query Parameters	61

Request Headers .....	61
Request Body .....	61
Response Headers .....	62
Response Media Types .....	62
Response Status .....	62
Response Body .....	62
Update the ACL Object Instance .....	65
HTTP Method .....	65
Request Media Types .....	65
Request Query Parameters .....	65
Request Headers .....	66
Request Body .....	66
Response Headers .....	67
Response Media Types .....	67
Response Status .....	67
Response Body .....	67
Delete the ACL Object Instance .....	69
HTTP Method .....	69
Request Media Types .....	69
Request Query Parameters .....	69
Request Headers .....	70
Request Body .....	70
Response Headers .....	70
Response Media Types .....	70
Response Status .....	71
Response Body .....	71
ACLs Collection .....	71
Resource Relationships .....	71
Feed .....	71
Link Relations .....	71
Operations .....	72
Get the ACLs Collection .....	72
HTTP Method .....	72
Request Media Types .....	72
Request Query Parameters .....	72
Request Headers .....	73
Request Body .....	73
Response Headers .....	73
Response Media Types .....	73
Response Status .....	73
Response Body .....	73
Create an ACL .....	75
Supported HTTP Method .....	75
Request Media Types .....	76
Request Query Parameters .....	76
Request Headers .....	76
Request Body .....	76
Response Headers .....	77
Response Media Types .....	77
Response Status .....	78
Response Body .....	78
ACL Associations .....	80
Resource Relationships .....	80
Feed .....	80
Link Relation .....	80
Operations .....	80
List All Available Associated SysObjects .....	81
HTTP Method .....	81

Request Media Types .....	81
Request Query Parameters .....	81
Request Headers .....	81
Request Body .....	81
Response Headers .....	82
Response Media Types .....	82
Response Status .....	82
Response Body .....	82
Aspect Type(s) .....	84
Aspect Type .....	84
Resource Relationships .....	84
Link Relation .....	84
Operations .....	84
Get an Aspect Type .....	84
HTTP Method .....	85
Request Media Types .....	85
Request Query Parameters .....	85
Request Headers .....	85
Request Body .....	85
Response Headers .....	85
Response Media Types .....	85
Response Status .....	86
Response Body .....	86
Aspect Types .....	89
Resource Relationships .....	89
Feed .....	89
Link Relation .....	89
Operations .....	90
Get Aspect Types .....	90
HTTP Method .....	90
Request Media Types .....	90
Request Query Parameters .....	90
Request Headers .....	90
Request Body .....	91
Response Headers .....	91
Response Media Types .....	91
Response Status .....	91
Response Body .....	91
Object Aspect(s) .....	92
Object Aspects .....	92
Resource Relationships .....	92
Link Relation .....	92
Operations .....	92
Get Attached Aspects .....	93
HTTP Method .....	93
Server Accepted Request Media Types .....	93
Query Parameters .....	93
Request Headers .....	93
Request Body .....	93
Response Headers .....	93
Supported Response Media Types .....	94
Response Status .....	94
Response Body .....	94
Attach Aspects to a SysObject .....	96
HTTP Method .....	96
Server Accepted Request Media Types .....	96
Query Parameters .....	96
Request Headers .....	96

Request Body .....	96
Response Headers .....	97
Supported Response Media Types .....	97
Response Status .....	97
Response Body .....	97
Object Properties Attached via Aspects .....	97
Object Aspect .....	98
Resource Relationships .....	98
Link Relation .....	98
Operations .....	98
Detach an Aspect .....	98
HTTP Method .....	99
Server Accepted Request Media Types .....	99
Query Parameters .....	99
Request Headers .....	99
Request Body .....	99
Response Headers .....	99
Supported Response Media Types .....	99
Response Status .....	99
Response Body .....	100
Batches .....	100
Resource Relationships .....	100
Link Relations .....	101
Operations .....	101
Create and Execute a Batch Request .....	101
Supported HTTP Method .....	101
Request Media Types .....	101
Request Query Parameters .....	101
Request Headers .....	101
Request Body .....	102
Request Samples .....	104
Multipart Request .....	105
Multipart Request Samples .....	108
Response Headers .....	111
Response Media Types .....	111
Response Status .....	111
Exceptions .....	111
Response Body .....	112
Batch Capabilities .....	116
Resource Relationships .....	116
Link Relations .....	116
Operations .....	116
Get Batch Capabilities .....	116
Supported HTTP Method .....	117
Request Media Types .....	117
Request Query Parameters .....	117
Request Headers .....	117
Request Body .....	117
Response Headers .....	117
Response Media Types .....	117
Response Status .....	117
Response Body .....	118
Cabinet(s) .....	120
Cabinet .....	120
Resource Relationships .....	120
Link Relations .....	120
Operations .....	121
Get a Cabinet .....	121

Supported HTTP Method.....	121
Request Media Types .....	121
Request Query Parameters .....	121
Request Headers .....	121
Request Body .....	122
Response Headers .....	122
Response Media Types.....	122
Response Status.....	122
Response Body .....	122
Update a Cabinet.....	122
Supported HTTP Method.....	122
Request Media Types .....	122
Request Query Parameters .....	123
Request Headers .....	123
Request Body .....	123
Response Headers .....	123
Response Media Types.....	123
Response Status.....	123
Response Body .....	124
Delete a Cabinet .....	124
Supported HTTP Method.....	124
Request Media Types .....	124
Request Query Parameters .....	124
Request Headers .....	125
Request Body .....	125
Response Headers .....	125
Response Media Types.....	125
Response Status.....	125
Response Body .....	126
Cabinets.....	127
Resource Relationships .....	127
Feed .....	127
Link Relations .....	127
Operations .....	127
Get Cabinets.....	128
Supported HTTP Method.....	128
Request Media Types .....	128
Request Query Parameters .....	128
Request Headers .....	129
Request Body .....	129
Response Headers .....	129
Response Media Types.....	129
Response Status.....	129
Response Body .....	130
Create a Cabinet .....	130
Supported HTTP Method.....	130
Request Media Types .....	130
Request Query Parameters .....	130
Request Headers .....	131
Request Body .....	131
Response Headers .....	131
Response Media Types.....	131
Response Status.....	131
Response Body .....	132
Checked Out Objects .....	133
Resource Relationships .....	133
Feed .....	133
Link Relations .....	133

Operations .....	133
Get Checked Out Objects .....	134
Supported HTTP Method.....	134
Request Media Types .....	134
Request Query Parameters .....	134
Request Headers .....	136
Request Body .....	136
Response Headers .....	136
Response Media Types.....	136
Response Status.....	136
Response Body .....	137
Comment(s) .....	138
Comment.....	138
Link Relations .....	138
Operations .....	138
Get a Comment .....	138
Supported HTTP Method.....	138
Request Media Types .....	139
Request Query Parameters .....	139
Request Headers .....	139
Request Body .....	139
Response Headers .....	139
Response Media Types.....	139
Response Status.....	139
Response Body .....	140
Delete a Comment .....	141
Supported HTTP Method.....	141
Request Media Types .....	141
Request Query Parameters .....	141
Request Headers .....	141
Request Body .....	141
Response Headers .....	141
Response Media Types.....	142
Response Status.....	142
Response Body .....	142
Comments .....	142
Feed .....	142
Link Relations .....	142
Operations .....	143
Get Root-level Comments .....	143
Supported HTTP Method.....	143
Request Media Types .....	143
Request Query Parameters .....	143
Request Headers .....	143
Request Body .....	144
Response Headers .....	144
Response Media Types.....	144
Response Status.....	144
Response Body .....	144
Create a Root-Level Comment.....	147
Supported HTTP Method.....	147
Request Media Types .....	147
Request Query Parameters .....	147
Request Headers .....	147
Request Body .....	147
Response Headers .....	147
Response Media Types.....	148
Response Status.....	148



Response Body .....	148
Comment Replies .....	149
Feed .....	149
Link Relations .....	149
Operations .....	149
Get Replies.....	150
Supported HTTP Method.....	150
Request Media Types .....	150
Request Query Parameters .....	150
Request Headers .....	150
Request Body .....	150
Response Headers .....	150
Response Media Types.....	151
Response Status.....	151
Response Body .....	151
Create a Reply to a Comment .....	153
Supported HTTP Method.....	153
Request Media Types .....	153
Request Query Parameters .....	153
Request Headers .....	154
Request Body .....	154
Response Headers .....	154
Response Media Types.....	154
Response Status.....	155
Response Body .....	155
Content(s) .....	157
Content.....	157
Resource Relationships .....	157
Link Relations .....	157
Operations .....	158
Get Content.....	158
Content URL return policy .....	158
Supported HTTP Method.....	159
Request Media Types .....	159
Request Query Parameters .....	159
Request Headers .....	160
Request Body .....	160
Response Headers .....	161
Response Media Types.....	161
Response Status.....	161
Response Body .....	161
Delete Content .....	161
Supported HTTP Method.....	161
Request Media Types .....	161
Request Query Parameters .....	162
Request Headers .....	162
Request Body .....	162
Response Headers .....	162
Response Media Types.....	162
Response Status.....	163
Response Body .....	163
Contents .....	164
Resource Relationships .....	164
Feed .....	164
Link Relations .....	164
Operations .....	165
Get Content.....	165
Supported HTTP Method.....	165

Request Media Types .....	165
Request Query Parameters .....	165
Request Headers .....	166
Request Body .....	166
Response Headers .....	166
Response Media Types .....	166
Response Status .....	166
Response Body .....	167
Create Content .....	167
Supported HTTP Method .....	167
Request Media Types .....	167
Request Query Parameters .....	167
Request Headers .....	169
Request Body .....	169
Response Headers .....	169
Response Media Types .....	169
Response Status .....	170
Response Body .....	170
Content Media .....	171
Resource Relationships .....	171
Link Relations .....	171
Operations .....	171
Get a Content Media from the REST Server .....	171
Supported HTTP Method .....	171
Request Media Types .....	171
Request Query Parameters .....	172
Request Headers .....	172
Request Body .....	172
Response Headers .....	172
Response Media Types .....	173
Response Status .....	173
Response Body .....	173
Current User .....	174
Resource Relationships .....	174
Link Relations .....	174
Operations .....	175
Get a Current User .....	175
Supported HTTP Method .....	175
Request Media Types .....	175
Request Query Parameters .....	175
Request Headers .....	175
Request Body .....	175
Response Headers .....	176
Response Media Types .....	176
Response Status .....	176
Response Body .....	176
Current User Preference(s) .....	177
Current User Preference .....	177
Resource Relationships .....	177
Link Relation .....	177
Operations .....	177
Get Current User Preference Settings .....	178
Supported HTTP Method .....	178
Request Media Types .....	178
Request Query Parameters .....	178
Request Headers .....	178
Request Body .....	178
Response Headers .....	178

Response Media Types.....	179
Response Status.....	179
Response Body.....	179
Update User Preference Settings .....	180
Supported HTTP Method.....	180
Request Media Types .....	180
Request Query Parameters .....	180
Request Headers .....	180
Request Body .....	181
Response Headers .....	181
Response Media Types.....	182
Response Status.....	182
Response Body.....	182
Delete User Preference Settings .....	183
Supported HTTP Method.....	183
Request Media Types .....	183
Request Query Parameters .....	183
Request Headers .....	184
Request Body .....	184
Response Headers .....	184
Response Media Types.....	184
Response Status.....	184
Response Body.....	184
Current User Preferences .....	185
Resource Relationships .....	185
Feed .....	185
Link Relation.....	185
Operations .....	185
Get Current User Preferences Collection .....	186
Supported HTTP Method.....	186
Request Media Types .....	186
Request Query Parameters .....	186
Request Headers .....	186
Request Body .....	186
Response Headers .....	186
Response Media Types.....	187
Response Status.....	187
Response Body.....	187
Create a User Preference .....	189
Supported HTTP Method.....	189
Request Media Types .....	189
Request Query Parameters .....	189
Request Headers .....	189
Request Body .....	189
Response Headers .....	190
Response Media Types.....	190
Response Status.....	190
Response Body.....	190
Current Version.....	192
Resource Relationships .....	192
Link Relations .....	192
Operations .....	192
Get the Current Version .....	192
Supported HTTP Method.....	193
Request Media Types.....	193
Request Query Parameters .....	193
Request Headers .....	193
Request Body .....	193

Response Headers .....	193
Response Media Types.....	193
Response Status.....	194
Response Body .....	194
Document .....	195
Resource Relationships .....	195
Link Relations .....	195
Operations .....	195
Get a Document .....	196
Supported HTTP Method.....	196
Request Media Types.....	196
Request Query Parameters .....	196
Request Headers .....	196
Request Body .....	196
Response Headers .....	197
Response Media Types.....	197
Response Status.....	197
Response Body .....	197
Update a Document.....	197
Supported HTTP Method.....	197
Request Media Types .....	197
Request Query Parameters .....	197
Request Headers .....	198
Request Body .....	198
Response Headers .....	198
Response Media Types.....	198
Response Status.....	198
Response Body .....	198
Delete a Document .....	199
Supported HTTP Method.....	199
Request Media Types .....	199
Request Query Parameters .....	199
Request Headers .....	199
Request Body .....	200
Response Headers .....	200
Response Media Types.....	200
Response Status.....	200
Response Body .....	200
DQL .....	201
Resource Relationships .....	201
Feed .....	201
Link Relations .....	201
Operations .....	202
Perform a read-only DQL query .....	202
Supported HTTP Method.....	202
Request Media Types.....	202
Request Query Parameters .....	202
Request Headers .....	203
Request Body .....	203
Response Headers .....	204
Response Media Types.....	204
Response Status.....	204
Response Body .....	204
Perform a long read-only DQL query .....	205
Supported HTTP Method.....	205
Request Media Types.....	205
Request Query Parameters .....	205
Request Headers .....	206

Request Body .....	206
Response Headers .....	207
Response Media Types.....	207
Response Status.....	207
Response Body .....	207
Folder .....	209
Resource Relationships .....	209
Link Relations .....	209
Operations .....	210
Get a Folder .....	210
Supported HTTP Method.....	210
Request Media Types .....	210
Request Query Parameters .....	210
Request Headers .....	211
Request Body .....	211
Response Headers .....	211
Response Media Types.....	211
Response Status.....	211
Response Body .....	211
Update a Folder.....	211
Supported HTTP Method.....	212
Request Media Types .....	212
Request Query Parameters .....	212
Request Headers .....	212
Request Body .....	212
Response Headers .....	212
Response Media Types.....	212
Response Status.....	213
Response Body .....	213
Delete a Folder .....	213
Supported HTTP Method.....	213
Request Media Types .....	213
Request Query Parameters .....	213
Request Headers .....	214
Request Body .....	214
Response Headers .....	214
Response Media Types.....	215
Response Status.....	215
Response Body .....	215
Folder Child Documents .....	216
Resource Relationships .....	216
Feed .....	216
Link Relations .....	216
Operations .....	216
Get Child Documents under a Folder.....	217
Supported HTTP Method.....	217
Request Media Types .....	217
Request Query Parameters .....	217
Request Headers .....	219
Request Body .....	219
Response Headers .....	219
Response Media Types.....	219
Response Status.....	219
Response Body .....	220
Create a Document Under a Folder .....	220
Supported HTTP Method.....	220
Request Media Types .....	220
Request Query Parameters .....	220

Request Headers .....	222
Request Body .....	222
Response Headers .....	223
Response Media Types .....	224
Response Status .....	224
Response Body .....	224
Import Document Metadata and Contents .....	224
Supported HTTP Method .....	224
Request Media Types .....	224
Request Query Parameters .....	225
Request Headers .....	228
Request Body .....	229
Response Headers .....	231
Response Media Types .....	231
Response Status .....	231
Response Body .....	231
Folder Child Folders .....	233
Resource Relationships .....	233
Feed .....	233
Link Relations .....	233
Operations .....	233
Get Child Folders under a Folder .....	234
Supported HTTP Method .....	234
Request Media Types .....	234
Request Query Parameters .....	234
Request Headers .....	235
Request Body .....	235
Response Headers .....	236
Response Media Types .....	236
Response Status .....	236
Response Body .....	236
Create a Child Folder under a Folder .....	236
Supported HTTP Method .....	237
Request Media Types .....	237
Request Query Parameters .....	237
Request Headers .....	237
Request Body .....	237
Response Headers .....	237
Response Media Types .....	238
Response Status .....	238
Response Body .....	238
Folder Child Link(s) .....	239
Folder Child Link .....	239
Resource Relationships .....	239
Link Relations .....	239
Operations .....	239
Retrieve a folder link .....	239
Supported HTTP Method .....	240
Request Media Types .....	240
Request Query Parameters .....	240
Request Headers .....	240
Request Body .....	240
Response Headers .....	240
Response Media Types .....	240
Response Status .....	240
Response Body .....	241
Unlink an Object from a Parent Folder .....	241
Supported HTTP Method .....	241

Request Media Types .....	241
Request Query Parameters .....	241
Request Headers .....	242
Request Body .....	242
Response Headers .....	242
Response Media Types .....	242
Response Status .....	242
Response Body .....	242
Folder Child Links .....	242
Resource Relationships .....	242
Feed .....	243
Link Relations .....	243
Operations .....	243
Retrieve child links .....	243
Supported HTTP Method .....	244
Request Media Types .....	244
Request Query Parameters .....	244
Request Headers .....	244
Request Body .....	245
Response Headers .....	245
Response Media Types .....	245
Response Status .....	245
Response Body .....	245
Link an Object to a Folder .....	245
Supported HTTP Method .....	246
Request Media Types .....	246
Request Query Parameters .....	246
Request Headers .....	246
Request Body .....	246
Response Headers .....	246
Response Media Types .....	247
Response Status .....	247
Response Body .....	247
Folder Child Objects .....	248
Resource Relationships .....	248
Feed .....	248
Link Relations .....	248
Operations .....	248
Get Folder Child Objects .....	249
Supported HTTP Method .....	249
Request Media Types .....	249
Request Query Parameters .....	249
Request Headers .....	251
Request Body .....	251
Response Headers .....	251
Response Media Types .....	251
Response Status .....	251
Response Body .....	252
Create an Object under a Folder .....	252
Supported HTTP Method .....	252
Request Media Types .....	252
Request Query Parameters .....	252
Request Headers .....	253
Request Body .....	253
Response Headers .....	253
Response Media Types .....	253
Response Status .....	253
Response Body .....	254

Import SysObject Metadata and Contents.....	254
Supported HTTP Method.....	254
Request Media Types.....	254
Request Query Parameters.....	254
Request Headers.....	258
Request Body.....	259
Response Headers.....	260
Response Media Types.....	260
Response Status.....	260
Response Body.....	261
Copy an Object to a Folder.....	261
Supported HTTP Method.....	262
Request Media Types.....	262
Request Query Parameters.....	262
Request Headers.....	262
Request Body.....	262
Response Headers.....	264
Response Media Types.....	264
Response Status.....	264
Response Body.....	265
Format(s).....	266
Format.....	266
Resource Relationships.....	266
Link Relations.....	266
Operations.....	266
Get a Format.....	266
Supported HTTP Method.....	266
Request Media Types.....	267
Request Query Parameters.....	267
Request Headers.....	267
Request Body.....	267
Response Headers.....	267
Response Media Types.....	267
Response Status.....	268
Response Body.....	268
Formats.....	269
Resource Relationships.....	269
Feed.....	269
Link Relations.....	269
Operations.....	269
Get Formats.....	270
Supported HTTP Method.....	270
Request Media Types.....	270
Request Query Parameters.....	270
Request Headers.....	270
Request Body.....	270
Response Headers.....	271
Response Media Types.....	271
Response Status.....	271
Response Body.....	271
Group(s).....	272
Group.....	272
Resource Relationships.....	272
Link Relations.....	272
Operations.....	273
Get a Group.....	273
Supported HTTP Method.....	273
Request Media Types.....	273



Request Query Parameters .....	273
Request Headers .....	273
Request Body .....	274
Response Headers .....	274
Response Media Types.....	274
Response Status.....	274
Response Body .....	274
Update a Group .....	276
Supported HTTP Method.....	276
Request Media Types .....	276
Request Query Parameters .....	276
Request Headers .....	277
Request Body .....	277
Response Headers .....	277
Response Media Types.....	277
Response Status.....	277
Response Body .....	278
Delete a Group .....	279
Supported HTTP Method.....	279
Request Media Types .....	279
Request Query Parameters .....	280
Request Headers .....	280
Request Body .....	280
Response Headers .....	280
Response Media Types.....	280
Response Status.....	280
Response Body .....	280
Groups .....	280
Resource Relationships .....	280
Feed .....	281
Link Relations .....	281
Operations .....	281
Get Groups .....	281
Supported HTTP Method.....	281
Request Media Types .....	282
Request Query Parameters .....	282
Request Headers .....	283
Request Body .....	283
Response Headers .....	283
Response Media Types.....	283
Response Status.....	283
Response Body .....	283
Create a Group.....	284
Supported HTTP Method.....	284
Request Media Types .....	284
Request Query Parameters .....	284
Request Headers .....	284
Request Body .....	284
Response Headers .....	285
Response Media Types.....	285
Response Status.....	285
Response Body .....	285
Group User(s) .....	286
Group User .....	286
Resource Relationships .....	286
Link Relation.....	286
Operations .....	286
Removing a User .....	286

Supported HTTP Method .....	286
Request Media Types .....	287
Request Query Parameters .....	287
Request Headers .....	287
Request Body .....	287
Response Headers .....	287
Response Media Types .....	287
Response Status .....	287
Response Body .....	287
Group Users .....	288
Resource Relationships .....	288
Feed .....	288
Link Relations .....	288
Operations .....	288
Get Group Users .....	289
Supported HTTP Method .....	289
Request Media Types .....	289
Request Query Parameters .....	289
Request Headers .....	289
Request Body .....	289
Response Headers .....	290
Response Media Types .....	290
Response Status .....	290
Response Body .....	290
Add a User Member .....	290
Supported HTTP Method .....	290
Request Media Types .....	291
Request Query Parameters .....	291
Request Headers .....	291
Request Body .....	291
Response Headers .....	291
Response Media Types .....	291
Response Status .....	292
Response Body .....	292
Home Document .....	293
Resource Relationships .....	293
Link Relations .....	293
Operations .....	293
Get Home Document .....	294
Supported HTTP Method .....	294
Request Media Types .....	294
Request Query Parameters .....	294
Request Headers .....	294
Request Body .....	294
Response Headers .....	294
Response Media Types .....	295
Response Status .....	295
Response Body .....	295
Lightweight Objects .....	297
Feed .....	297
Resource Relationships .....	297
Link Relations .....	297
Operations .....	297
Create a Lightweight Object .....	298
Supported HTTP Method .....	298
Request Media Types .....	298
Request Query Parameters .....	298
Request Headers .....	298

Request Body .....	298
Response Headers .....	299
Response Media Types.....	299
Response Status.....	299
Response Body .....	299
Get the Lightweight Objects of a Shared Object.....	300
Supported HTTP Method.....	300
Request Media Types .....	300
Request Query Parameters .....	300
Request Headers .....	301
Request Body .....	301
Response Headers .....	301
Response Media Types.....	301
Response Status.....	301
Response Body .....	301
Lightweight Object Parent.....	304
Feed .....	304
Resource Relationships .....	304
Link Relations .....	304
Operations .....	304
Get the Parent Shareable Object .....	304
Supported HTTP Method.....	305
Request Media Types .....	305
Request Query Parameters .....	305
Request Headers .....	305
Request Body .....	305
Response Headers .....	305
Response Media Types.....	305
Response Status.....	305
Response Body .....	306
Repair a Lightweight Object .....	306
Supported HTTP Method.....	306
Request Media Types .....	306
Request Query Parameters .....	307
Request Headers .....	307
Request Body .....	307
Response Headers .....	307
Response Media Types.....	307
Response Status.....	307
Response Body .....	308
Lightweight Object Materialization.....	309
Feed .....	309
Resource Relationships .....	309
Link Relations .....	309
Operations .....	309
Materialize the Lightweight Object .....	309
Supported HTTP Method.....	310
Request Media Types .....	310
Request Query Parameters .....	310
Request Headers .....	310
Request Body .....	310
Response Headers .....	310
Response Media Types.....	310
Response Status.....	310
Response Body .....	311
Dematerialize a Lightweight Object .....	311
Supported HTTP Method.....	311
Request Media Types .....	311

Request Query Parameters .....	312
Request Headers .....	312
Request Body .....	312
Response Headers .....	312
Response Media Types.....	312
Response Status.....	312
Response Body .....	312
Lock .....	313
Resource Relationships .....	313
Link Relations .....	313
Operations .....	313
Check out (lock) an object .....	313
Supported HTTP Method.....	313
Request Media Types .....	313
Request Query Parameters .....	314
Request Headers .....	314
Request Body .....	314
Response Headers .....	314
Response Media Types.....	314
Response Status.....	314
Response Body .....	315
Cancel the check-out operation (unlock) on an object. ....	315
Supported HTTP Method.....	315
Request Media Types .....	315
Request Query Parameters .....	315
Request Headers .....	315
Request Body .....	315
Response Headers .....	315
Response Media Types.....	316
Response Status.....	316
Response Body .....	316
Network Location(s) .....	317
Network Location.....	317
Resource Relationships .....	317
Link Relations .....	317
Operations .....	317
Get a Network Location .....	317
Supported HTTP Method.....	318
Request Media Types .....	318
Request Query Parameters .....	318
Request Headers .....	318
Request Body .....	318
Response Headers .....	318
Response Media Types.....	318
Response Status.....	319
Response Body .....	319
Network Locations .....	320
Resource Relationships .....	320
Feed .....	320
Link Relations .....	320
Operations .....	320
Get Network Locations .....	321
Supported HTTP Method.....	321
Request Media Types .....	321
Request Query Parameters .....	321
Request Headers .....	321
Request Body .....	322
Response Headers .....	322

Response Media Types.....	322
Response Status.....	322
Response Body.....	322
Parent Link(s).....	323
Parent Link .....	323
Resource Relationships .....	323
Link Relations .....	323
Operations .....	323
Retrieve a folder link.....	323
Supported HTTP Method.....	324
Request Media Types .....	324
Request Query Parameters .....	324
Request Headers .....	324
Request Body .....	324
Response Headers .....	324
Response Media Types.....	324
Response Status.....	324
Response Body.....	325
Move an Object from one Folder to another .....	325
Supported HTTP Method.....	325
Request Media Types .....	325
Request Query Parameters .....	325
Request Headers .....	326
Request Body .....	326
Response Headers .....	326
Response Media Types.....	326
Response Status.....	326
Response Body.....	327
Unlink an Object from a Parent Folder .....	327
Supported HTTP Method.....	327
Request Media Types .....	327
Request Query Parameters .....	327
Request Headers .....	327
Request Body .....	327
Response Headers .....	328
Response Media Types.....	328
Response Status.....	328
Response Body.....	328
Parent Links .....	328
Resource Relationships .....	328
Feed .....	328
Link Relations .....	329
Operations .....	329
Get Parent Links.....	329
Supported HTTP Method.....	329
Request Media Types .....	329
Request Query Parameters .....	330
Request Headers .....	330
Request Body .....	330
Response Headers .....	330
Response Media Types.....	330
Response Status.....	330
Response Body.....	331
Link an Object to a New Folder .....	331
Supported HTTP Method.....	331
Request Media Types .....	331
Request Query Parameters .....	331
Request Headers .....	331

Request Body .....	332
Response Headers .....	332
Response Media Types.....	332
Response Status.....	332
Response Body .....	332
Permissions .....	333
Feed .....	333
Resource Relationships .....	333
Link Relations .....	333
Operations .....	333
Get the Permission View Instance .....	333
Supported HTTP Method.....	334
Request Media Types.....	334
Request Query Parameters .....	334
Request Headers .....	334
Request Body .....	335
Response Headers .....	335
Response Media Types.....	335
Response Status.....	335
Response Body .....	335
Permission Set.....	337
Resource Relationships .....	337
Feed .....	337
Link Relation.....	337
Operations .....	338
Get the Permission Set Instance .....	338
HTTP Method .....	338
Request Media Types.....	338
Request Query Parameters .....	338
Request Headers .....	338
Request Body .....	338
Response Headers .....	339
Supported Response Media Types .....	339
Response Status.....	339
Response Body .....	339
Update the Permission Set Instance.....	341
HTTP Method .....	343
Request Media Types.....	343
Request Query Parameters .....	343
Request Headers .....	344
Request Body .....	344
Response Headers .....	345
Response Media Types.....	345
Response Status.....	345
Response Body .....	345
User Permission Set .....	347
Feed .....	347
Resource Relationships .....	347
Link Relation.....	347
Operations .....	348
Get the User Permission Set.....	348
HTTP Method .....	348
Request Media Types.....	348
Request Query Parameters .....	348
Request Headers .....	349
Request Body .....	349
Response Headers .....	349
Response Media Types.....	349

Response Status.....	349
Response Body .....	349
Product Information .....	351
Resource Relationships .....	351
Link Relations .....	351
Operations .....	351
Get Product Information .....	351
Supported HTTP Method .....	351
Request Media Types .....	352
Request Query Parameters .....	352
Request Headers .....	352
Request Body .....	352
Response Headers .....	352
Response Media Types.....	352
Response Status.....	352
Response Body .....	352
Relation(s).....	354
Relation .....	354
Resource Relationships .....	354
Link Relations .....	354
Operations .....	355
Get a Relation.....	355
Supported HTTP Methods .....	355
Request Media Types .....	355
Request Query Parameters .....	355
Request Headers .....	355
Request Body .....	355
Response Headers .....	356
Response Media Types.....	356
Response Status.....	356
Response Body .....	356
Delete a Relation .....	356
Supported HTTP Method.....	356
Request Media Types .....	356
Request Query Parameters .....	356
Request Headers .....	357
Request Body .....	357
Response Headers .....	357
Response Media Types.....	357
Response Status.....	357
Response Body .....	357
Relations .....	357
Resource Relationships .....	357
Feed .....	358
Link Relations .....	358
Operations .....	358
Get Relations.....	358
Supported HTTP Method.....	358
Request Media Types .....	358
Request Query Parameters .....	359
Request Headers .....	360
Request Body .....	360
Response Headers .....	360
Response Media Types.....	360
Response Status.....	360
Response Body .....	360
Create a Relation .....	361
Supported HTTP Method.....	361

Request Media Types .....	361
Request Query Parameters .....	361
Request Headers .....	361
Request Body .....	361
Response Headers .....	362
Response Media Types .....	362
Response Status .....	362
Response Body .....	362
Relation Type(s) .....	363
Relation Type .....	363
Resource Relationships .....	363
Link Relations .....	363
Operations .....	363
Get a Relation Type .....	364
Supported HTTP Method .....	364
Request Media Types .....	364
Request Query Parameters .....	364
Request Headers .....	364
Request Body .....	364
Response Headers .....	364
Response Media Types .....	364
Response Status .....	365
Response Body .....	365
Relation Types .....	366
Resource Relationships .....	366
Feed .....	366
Link Relations .....	366
Operations .....	366
Get Relation Types .....	367
Supported HTTP Method .....	367
Request Media Types .....	367
Request Query Parameters .....	367
Request Headers .....	367
Request Body .....	367
Response Headers .....	368
Response Media Types .....	368
Response Status .....	368
Response Body .....	368
Repository .....	369
Resource Relationships .....	369
Link Relations .....	369
Operations .....	370
Get a Repository .....	370
Supported HTTP Method .....	370
Request Media Types .....	370
Request Query Parameters .....	371
Request Headers .....	371
Request Body .....	371
Response Headers .....	371
Response Media Types .....	371
Response Status .....	371
Response Body .....	371
Repositories .....	374
Resource Relationships .....	374
Feed .....	374
Link Relations .....	374
Operations .....	374
Get Repositories .....	375



Supported HTTP Method .....	375
Request Media Types .....	375
Request Query Parameters .....	375
Request Headers .....	375
Request Body .....	375
Response Headers .....	375
Response Media Types .....	376
Response Status .....	376
Response Body .....	376
Search .....	377
Resource Relationships .....	377
Feed .....	377
Link Relations .....	377
Operations .....	378
Full-text Search Using URI Parameters .....	378
Supported HTTP Methods .....	378
Request Media Types .....	378
Request Query Parameters .....	378
Request Headers .....	381
Request Body .....	382
Response Headers .....	382
Response Media Types .....	382
Response Status .....	382
Response Body .....	382
Full-text Search Using Query Document and URI Parameters .....	386
Supported HTTP Methods .....	386
Request Media Types .....	386
Request Query Parameters .....	386
Request Headers .....	386
Request Body .....	387
Response Media Types .....	387
Response Headers .....	387
Response Status .....	387
Response Body .....	388
Search with Facets .....	392
Facets Using Query Parameters .....	392
Facet Using Query Document .....	394
Saved Search(es) .....	402
Saved Search .....	402
Feed .....	402
Resource Relationships .....	402
Link Relations .....	402
Supported HTTP Methods .....	403
Operations .....	403
Get a Saved Search .....	403
Supported HTTP Method .....	403
Request Media Types .....	403
Request Query Parameters .....	403
Request Headers .....	403
Request Body .....	403
Response Headers .....	403
Response Media Types .....	404
Response Status .....	404
Response Body .....	404
Update a Saved Search .....	406
Supported HTTP Method .....	406
Request Media Types .....	406
Request Query Parameters .....	406

Request Headers .....	406
Request Body .....	406
Response Status.....	407
Response Headers .....	408
Response Body .....	408
Delete a Saved Search .....	410
Supported HTTP Methods .....	410
Request Headers .....	411
Request Body .....	411
Response Status.....	411
Saved Searches .....	411
Feed .....	411
Resource Relationships .....	411
Link Relations .....	412
Supported HTTP Methods .....	412
Operations .....	412
Get a Saved Searches Collection.....	412
Request Method .....	412
Request Query Parameters .....	412
Request Headers .....	412
Request Body .....	413
Response Status.....	413
Response Headers .....	413
Response Media Type .....	413
Example.....	413
Create a Saved Search .....	415
Supported HTTP Method.....	415
Request Media Type .....	415
Request Query Parameters .....	415
Request Headers .....	415
Request Body .....	416
Response Headers .....	417
Response Media Type .....	417
Response Status.....	417
Response Body .....	417
Saved Search Execution.....	420
Resource Relationships .....	420
Feed .....	420
Link Relations .....	420
Operations .....	420
Execute a Saved Search .....	421
Supported HTTP Method.....	421
Request Media Types .....	421
Request Query Parameters .....	421
Request Headers .....	422
Request Body .....	422
Response Headers .....	423
Response Media Types.....	423
Response Status.....	423
Response Body .....	423
Saved Search Results .....	426
Resource Relationships .....	426
Feed .....	426
Link Relations .....	426
Operations .....	427
Enable and Refresh Saved Results.....	427
Supported HTTP Method.....	427
Request Media Types .....	427
Request Query Parameters .....	427

Request Headers .....	428
Request Body .....	428
Response Headers .....	430
Response Media Types .....	430
Response Status .....	430
Response Body .....	430
Get Saved Results .....	431
Supported HTTP Method .....	431
Request Media Types .....	431
Request Query Parameters .....	431
Request Headers .....	431
Request Body .....	431
Response Headers .....	432
Response Media Types .....	432
Response Status .....	432
Response Body .....	432
Disable Saved Results .....	432
Supported HTTP Method .....	432
Request Media Types .....	433
Request Query Parameters .....	433
Request Headers .....	433
Request Body .....	433
Response Headers .....	433
Response Media Types .....	433
Response Status .....	433
Response Body .....	433
Saved Search Template(s) .....	434
Saved Search Template .....	434
Resource Relationships .....	435
Feed .....	435
Link Relations .....	435
Supported HTTP Methods .....	436
Operations .....	436
Get a Search Template .....	436
Supported HTTP Method .....	436
Request Media Types .....	436
Request Query Parameters .....	436
Request Headers .....	436
Request Body .....	436
Response Headers .....	437
Response Media Types .....	437
Response Status .....	437
Response Body .....	437
Delete a Search Template .....	441
HTTP Method .....	441
Request Media Types .....	441
Request Query Parameters .....	441
Request Headers .....	441
Request Body .....	441
Response Headers .....	441
Response Media Types .....	442
Response Status .....	442
Response Body .....	442
Saved Search Templates .....	442
Resource Relationships .....	442
Feed .....	442
Link Relations .....	443
Supported HTTP Methods .....	443

Operations .....	443
Create a Search Template .....	443
Request Method .....	443
Request Query Parameters .....	443
Request Headers .....	443
Request Media Types .....	444
Request Body .....	444
Response Status .....	446
Response Headers .....	446
Response Media Types .....	446
Saved Search as Search Template .....	449
HTTP Method .....	449
Request Media Types .....	449
Request Query Parameters .....	449
Request Headers .....	449
Request Body .....	450
Response Headers .....	450
Response Media Types .....	450
Response Status .....	451
Response Body .....	451
Get Search Templates .....	454
HTTP Method .....	454
Request Media Types .....	454
Request Query Parameters .....	454
Request Headers .....	454
Request Body .....	454
Response Headers .....	455
Response Media Types .....	455
Response Status .....	455
Response Body .....	455
Search Template Execution .....	458
Resource Relationships .....	458
Feed .....	458
Link Relations .....	458
Supported HTTP Methods .....	458
Operations .....	459
Get Search Results of an Instantiated Search Template .....	459
HTTP Method .....	459
Request Media Types .....	459
Request Query Parameters .....	459
Request Headers .....	459
Request Body .....	460
Response Headers .....	462
Response Media Types .....	462
Response Status .....	462
Response Body .....	462
Sub-Group(s) .....	465
Sub Group .....	465
Resource Relationships .....	465
Link Relation .....	465
Operations .....	465
Remove a Sub Group .....	465
Supported HTTP Method .....	465
Request Media Types .....	466
Request Query Parameters .....	466
Request Headers .....	466
Request Body .....	466
Response Headers .....	466

Response Media Types.....	466
Response Status.....	466
Response Body.....	466
Sub Groups.....	467
Resource Relationships.....	467
Feed.....	467
Link Relations.....	467
Operations.....	467
Add a Sub Group.....	468
Supported HTTP Method.....	468
Request Media Types.....	468
Request Query Parameters.....	468
Request Headers.....	468
Request Body.....	468
Response Headers.....	469
Response Media Types.....	469
Response Status.....	469
Response Body.....	469
Get Sub Groups.....	469
Supported HTTP Method.....	469
Request Media Types.....	469
Request Query Parameters.....	470
Request Headers.....	470
Request Body.....	470
Response Headers.....	470
Response Media Types.....	470
Response Status.....	471
Response Body.....	471
SysObject.....	472
Resource Relationships.....	472
Link Relations.....	472
Operations.....	476
Get an Object.....	476
Supported HTTP Method.....	476
Request Media Types.....	476
Request Query Parameters.....	476
Request Headers.....	476
Request Body.....	477
Response Headers.....	477
Response Media Types.....	477
Response Status.....	477
Response Body.....	477
Delete an Object.....	477
Supported HTTP Method.....	477
Request Media Types.....	478
Request Query Parameters.....	478
Request Headers.....	479
Request Body.....	479
Response Headers.....	479
Response Media Types.....	479
Response Status.....	480
Response Body.....	480
Update an Object.....	480
Supported HTTP Method.....	480
Request Media Types.....	480
Request Query Parameters.....	480
Request Headers.....	480
Request Body.....	481

Response Headers .....	481
Response Media Types.....	481
Response Status.....	481
Response Body .....	481
Type(s).....	482
Type .....	482
Resource Relationships .....	482
Link Relations .....	482
Operations .....	482
Get a Type.....	483
HTTP Method .....	483
Request Media Types .....	483
Request Query Parameters .....	483
Request Headers .....	484
Request Body .....	484
Response Headers .....	484
Response Media Types.....	484
Response Status.....	484
Response Body .....	485
Types.....	488
Resource Relationships .....	488
Feed .....	488
Link Relations .....	488
Operations .....	488
Get Types.....	489
Supported HTTP Method.....	489
Request Media Types .....	489
Request Query Parameters .....	489
Request Headers .....	491
Request Body .....	491
Response Headers .....	491
Response Media Types.....	492
Response Status.....	492
Response Body .....	492
User(s) .....	495
User.....	495
Resource Relationships .....	495
Link Relations .....	495
Operations .....	496
Get a User .....	496
Supported HTTP Method.....	496
Request Media Types .....	496
Request Query Parameters .....	496
Request Headers .....	496
Request Body .....	496
Response Headers .....	496
Response Media Types.....	497
Response Status.....	497
Response Body .....	497
Update a User .....	499
Supported HTTP Method.....	499
Request Media Types .....	499
Request Query Parameters .....	499
Request Headers .....	499
Request Body .....	500
Response Headers .....	500
Response Media Types.....	500
Response Status.....	500

Response Body .....	501
Delete a User .....	502
Supported HTTP Method .....	502
Request Media Types .....	503
Request Query Parameters .....	503
Request Headers .....	503
Request Body .....	503
Response Headers .....	503
Response Media Types .....	503
Response Status .....	503
Response Body .....	503
Users .....	503
Resource Relationships .....	504
Feed .....	504
Link Relations .....	504
Operations .....	504
Get Users .....	504
Supported HTTP Method .....	505
Request Media Types .....	505
Request Query Parameters .....	505
Request Headers .....	505
Request Body .....	505
Response Headers .....	505
Response Media Types .....	506
Response Status .....	506
Response Body .....	506
Create a User .....	506
Supported HTTP Method .....	506
Request Media Types .....	506
Request Query Parameters .....	506
Request Headers .....	507
Request Body .....	507
Response Headers .....	507
Response Media Types .....	507
Response Status .....	508
Response Body .....	508
User Default Folder .....	509
Resource Relationships .....	509
Link Relations .....	509
Operations .....	509
Get a User Default Folder .....	509
Supported HTTP Method .....	509
Request Media Types .....	510
Request Query Parameters .....	510
Request Headers .....	510
Request Body .....	510
Response Headers .....	510
Response Media Types .....	510
Response Status .....	510
Response Body .....	511
Value Assistance .....	512
Resource Relationships .....	512
Feed .....	512
Link Relations .....	512
Supported HTTP Methods .....	512
Operations .....	512
Get the Value Assistance of the Type .....	512
Request URI Query .....	513

	Request Headers .....	513
	Request Media Types .....	513
	Request Body .....	513
	Response Status .....	514
	Response Headers .....	514
	Response Media Types .....	514
	Response Body .....	514
	Virtual Document Nodes .....	516
	Resource Relationships .....	516
	Feed .....	516
	Link Relations .....	516
	Supported HTTP Methods .....	516
	Operations .....	517
	Get a Virtual Document Hierarchy.....	517
	Request URI Query.....	517
	Request Headers .....	518
	Request Media Types .....	518
	Request Body .....	518
	Response Status.....	518
	Response Headers .....	518
	Response Media Types.....	518
	Response Body .....	519
<b>Appendix A</b>	<b>Link Relations .....</b>	<b>527</b>
<b>Appendix B</b>	<b>REST Common Definition - URI Request Query Parameters .....</b>	<b>533</b>
<b>Appendix C</b>	<b>REST Common Definition - HTTP Status Codes .....</b>	<b>539</b>
<b>Appendix D</b>	<b>REST Common Definition - HTTP Headers .....</b>	<b>541</b>
<b>Appendix E</b>	<b>AQL .....</b>	<b>545</b>
	Expressions in AQL .....	550
	Fulltext Expression .....	552
	Property Expression .....	552
	Property List Expression .....	553
	Property Range Expression .....	554
	Relative Date Expression.....	555
	Expressions as Templates .....	555
	Facet in AQL .....	556
	Grouping Strategy .....	556
	Facet Ordering .....	559



## List of Figures

Figure 1.	Core Resources in Documentum Platform REST Services 7.3 .....	39
Figure 2.	Resource Relationship Sample.....	42

## List of Tables

Table 1.	Public Link Relations .....	527
Table 2.	Link Relations Introduced in Documentum Platform REST Services .....	528
Table 3.	URI Request Query Parameters .....	533
Table 4.	.....	539

# Preface

---

This document is a reference to all resources shipped in EMC Documentum Platform REST Services (also known as core resources). It provides information on how to use each service operation.

For general information about Documentum Platform REST Services and developing your own REST resources, see the *Documentum Platform REST Services Development Guide*.

## Intended Readership

This document is intended for software developers and architects who are building REST web services to interact with the Documentum platform.

## Related Documentation

The following documentation provides additional information:

- *EMC Documentum Platform REST Services Development Guide*
- *EMC Documentum Platform REST Services Release Notes*

There is experimental API documentation, written in RADL, that is available to users in both the SDK and the REST war.

## Revision History

### Revision history

Revision Date	Description
November 2016	Initial publication.



## Overview

EMC Documentum Platform REST Services, which is also referred to as Documentum Platform REST Services, is a set of RESTful web service interfaces that interact with the Documentum platform.

Being developed in a purely RESTful style, Documentum Platform REST Services is hypertext-driven, server-side stateless, and content negotiable. This provides you with high efficiency and simplicity in programming, and it also makes all services easy to consume. These advantages make Documentum Platform REST Services the optimal choice for Web 2.0 applications and mobile applications to interact with Documentum repositories.

Documentum Platform REST Services models objects in Documentum repositories as resources. This guide is designed as a resource catalog that focuses on the following:

- How resources are related to one another. For more information, see [Hypermedia-driven API](#)
- General description of each resource
- Operations you can perform on each resource

## Hypermedia-driven API

Documentum Platform REST Services is a purely hypermedia-driven REST API. The physical Uniform Resource Identifiers (URI) used to access the service's resources are discoverable at runtime by using link relations instead of hard-coded values. Hypermedia is an important aspect of REST. It allows you to build services that separate the clients and servers, which allows them to evolve independently.

Before consuming Documentum Platform REST Services, a REST client requires the URI of the entry point for the service. The [Home document](#) is the entry point of Documentum Platform REST Services. The URI of the Home document is the only permanent URI path that the REST client must bookmark: `/services`.

Issuing an HTTP GET request for the entry point returns a Home Document that contains link relations for all of the top-level resources exposed by the Documentum REST Service. By default, the resources exposed include the [Repositories](#) collection and the [Product Information](#) resource.

Here is an example of a default Home Document:

```
{
  resources: {
    http://identifiers.emc.com/linkrel/repositories: {
      href: "http://localhost:8080/dctm-rest/repositories"
```

```
hints: {
  allow:
    "GET"
  representations:
    "application/xml"
    "application/json"
    "application/atom+xml"
    "application/vnd.emc.documentum+json"
}
}
about: {
  href: "http://localhost:8080/dctm-rest/product-info"
  hints: {
    allow:
      "GET"
    representations:
      "application/xml"
      "application/json"
      "application/vnd.emc.documentum+xml"
      "application/vnd.emc.documentum+json"
    }
  }
}
```

The key for each resource is an immutable link relation name. The link relation name defines the relation to the target resource. Each resource has an `href` attribute that contains the URI of the resource.

**Note:** An `href` does not have to be the same from one REST API version to another, which makes it more important to avoid hard-coding URIs in your client applications.

Each link relation name such as `self`, `edit`, `about`, and so on, provides the client with unchangeable identifiers that identify the resource that it is referencing. In the following example, the `delete` link relation indicates that the purpose of the `href` link is to delete the current resource.

```
"links": [
{
  "rel": " http://identifiers.emc.com/linkrel/delete",
  "href": "http://localhost:8080/dctm-rest/repositories/acme02"
}
]
```

The logical relations between Documentum Core resources are managed using either link relations or feed-to-entry containership. The following diagram lists Documentum Core resources from a logical relation point of view.

Figure 1. Core Resources in Documentum Platform REST Services 7.3



As you can see in the diagram above, the Home Document resource is the starting point for everything. For example, to obtain the representation of a Repository resource, you must issue an HTTP GET request for the Home Document in order to discover the Repositories collection. Then you can issue a GET request for the Repositories collection, which returns all of the links pointing to single Repository resources. You can search the links that are returned for one that points to the Repository resource you want.

Link relations in the Repository resource gives you information about what operations you can perform on that resource. This approach determines the way that a hypermedia RESTful web service interacts with clients.

**Note:** The diagram provides a visual overview of Documentum Core resources and their relationships. It is not intended to be used as a reference to all relationships among Core resources. For example, a Document resource contains a link relation to the **type** resource, but this relationship does not appear in the diagram. Furthermore, it does not mean that you must follow the traversal

path shown in the diagram to get to a resource. For example, you can also get to a **SysObject** resource from the DQL query resource.

The following chapters in the *Documentum Platform REST Services Development Guide* provide more detailed examples on how hypermedia links work in resource discovery and operation execution:

- Explore Documentum Platform REST Services
- Tutorial: Consume REST Services Programmatically



## Resources

### Organization of Resource Reference Documentation

These guidelines apply to each resource reference documentation entry:

- All method parameters are optional unless otherwise noted.
- The `DELETE` and `GET` methods do not have a Request body.

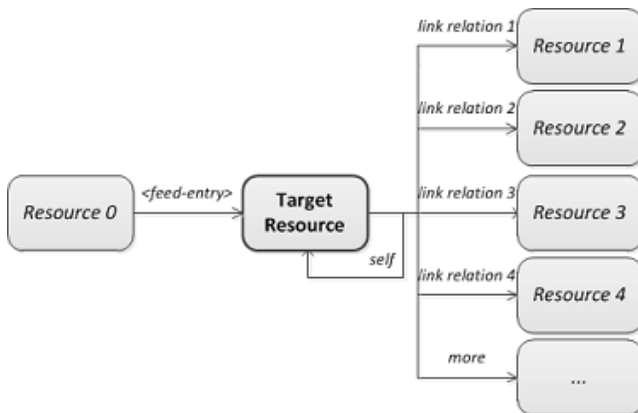
#### About the Resource URIs:

Resource URIs are used to identify the physical locations of server resources. According to REST services best practices, resource URIs should be discovered using hypermedia, such as link relations and the `Response Location` header. The only exception is the root entry URI of the Home Document.

You are encouraged to use hypermedia to discover your Documentum resources from the client side. Using hypermedia is far better than attempting to construct the URI in your client code. To help you with your REST client programming, there are a number of Documentum client samples available on [GitHub](#).

#### About the Diagrams in Resource Relationships:

- The target resource is presented in bold.
- Link relations are presented as strings on arrows.
- The resource to the left of the target resource (resource 0 in the sample) is the resource that you can use to navigate to the target resource using the link relation presented on the arrow. There can be more than one resource you can use to navigate to the target resource. However, only one is presented in the diagrams.
- Resources to the right of the target resource (resources 1 to 4 in the sample) are those that you can reach from the target resource. The diagrams displays at most four resources to the right of the target resource. Other resources, when they exist, are presented as suspension points such as (...).
- The string labelled as `more` on arrows is not a link relation. It is an indication that you can reach more than four resources from the target resource.
- The string `<feed-entry>` on arrows is not a link relation. It is an indication that two resources are linked via feed-to-entry containership.

**Figure 2. Resource Relationship Sample**

**Note:** The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119](#).

# Additional Services Installation

Most of Documentum Platform REST Services can be used out-of-the-box when the Content Server repositories and the REST server are installed. However, there are some REST services that require the installation of additional services on Content Server.

## Collaboration Service

The commenting function is dependent on the Collaboration Services module. Therefore, the Collaboration Services Archive (DAR) files must be installed in the repository prior to using any of the following resources:

- [Comment, page 138](#)
- [Comments, page 142](#)
- [Comment Replies, page 149](#)

## Search Service

The full-text search function is dependent on the deployment and configuration of full-text indexing on Documentum repositories. The following resources are only available when full-text indexing is enabled on the target repositories:

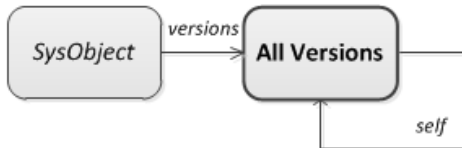
- [Search, page 377](#)
- [Saved Searches, page 411](#)
- [Saved Search, page 402](#)
- [Saved Search Execution, page 420](#)
- [Saved Search Templates, page 442](#)
- [Saved Search Template, page 434](#)
- [Search Template Execution, page 458](#)
- DQL with full-text query using the parameter *q* with [Get Versions, page 45](#), [Get Checked Out Objects, page 134](#), [Get Folder Child Objects, page 249](#), and [Get Child Documents under a Folder, page 217](#)

# All Versions

The All Versions resource represents a collection of all versions for a given non-folder SysObject.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [SysObject](#), page 472 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
URI of the versions	All versions	Server's current time	Yes

Entry	Entry ID	Entry Title	Entry Updated
Each entry represents a specific version of the given SysObject.	URI of the version	object_name for the specific version of the object	r_modify_date for the specific version of the object

## Link Relations

The following table lists link relations for the All Versions resource.

Link Relation	Description	Resource Reference
self	This All Versions resource	<a href="#">All Versions</a> , page 44
first, last, next, previous	Pagination links	<a href="#">All Versions</a> , page 44

## Operations

The All Versions resource supports the following HTTP methods.

Method	Description
GET	Retrieves all versions of a given SysObject.
POST	Checks in a given SysObject.

## Get Versions

Retrieve all versions of a given SysObject.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

The following common query parameters are supported:

- inline
- sort
- page
- items-per-page
- include-total
- view
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following version-related query parameters are also supported:

Variable	Description	Data type	Default value
version-label	<p>Only versions with the specified version label are returned in the result.</p> <p>A request can contain multiple version-label variables. Objects whose version-label matches any of those are returned.</p>	string	null
latest-versions-only	Specifies whether or not to show the latest versions only.	boolean	false
q	<p>Specifies the search criterion with a full-text expression in simple search language.</p> <p>Parameter q must be encoded because it may contain non-English locale characters.</p> <p><b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.</p>	String	No search criteria is used for the search.

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
404 - Object not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of all or a subset of versions of a given SysObject.

- Each object may contain all or a set of properties of the versions, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those items to which you have access.
- Pagination is supported.
- By default, the results are sorted by the `i_latest_flag` and `creation-date` properties in descending order.

## Check in an object

Check in the SysObject resource that has been checked out.

## Supported HTTP Method

POST

## Request Media Types

When the client tries to check in both the metadata and one or more pieces of content, the `Content-Type` must be one of the following:

- `multipart/form-data`
- `multipart/mixed`

In addition to the above, the `Content-Type` of the first part must either:

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

When the client tries to check in more than one piece of content without metadata, the `Content-Type` must be one of the following:

- `multipart/form-data`
- `multipart/mixed`

When the client tries to check in the metadata only, the `Content-Type` must be one of the following:

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

For content types other than any of the types mentioned above, the server treats the operation as a single *content-only* check-in.

## Request Query Parameters

This operation supports the following query parameters.

Variable	Description	Data type	Value range	Default value
<code>content-count</code>	<p>Specifies how many pieces of content are submitted. The metadata part is excluded from the count.</p> <p>When more than one piece of content exists, the total content count must be provided. Otherwise, only the first content type is used. For example, when you import an</p>	integer	$\geq 0$	0



Variable	Description	Data type	Value range	Default value
	object with one object metadata part and five binary content types, then the value of the <i>content-count</i> parameter is 5.			
<i>all-primary</i>	Specifies whether to import all contents using the same type as the primary content type for different pages (page 0, 1, 2...), or one primary content type, and others as renditions for the first page, which is page 0	boolean	true or false	true
<i>content-length</i>	<p>Specifies the byte count of the contents to be uploaded.</p> <p>This parameter is optional.</p> <p>A comma separated list can be used to specify multiple values for this parameter. For example:  <code>content-length=1024,2048,,4096</code>  . The blank value shown above is 0.</p> <p>When the <i>all-primary</i> parameter is false, the first value in the comma separated</p>	string	Comma separated string of integer number values	null

Variable	Description	Data type	Value range	Default value
	list is used for the <i>content-length</i> of the primary content, and all other values in the list are ignored.			
<i>version-policy</i>	<p>Specifies the version policy by using one of the following valid values:</p> <ul style="list-style-type: none"> <li>• <i>next-major</i> - the object receives the next major version number.</li> <li>• <i>next-minor</i> - the object receives the next minor version number.</li> <li>• <i>branch-version</i> - the object receives a branch version number.</li> <li>• <i>none</i> - no version policy is specified. Content Server makes the decision.</li> </ul>	string	A string containing any one of the valid values: <i>next-major</i> , <i>next-minor</i> , <i>branch-version</i> , or <i>none</i> .	null

Variable	Description	Data type	Value range	Default value
<i>retain-lock</i>	<p>Specifies whether or not to keep the object checked out and locked by the user after the new version is saved.</p> <ul style="list-style-type: none"> <li>• <code>true</code> - keep the lock after the new version is saved.</li> <li>• <code>false</code> - the object is not locked after the new version is saved.</li> </ul>	boolean	true or false	false
<i>version-label</i> *	<p>Specifies one or more version labels of the checked in object with the following constraints:</p> <ul style="list-style-type: none"> <li>• By default, numeric version labels, such as 1.1, 2.1.0.1, 5.0 are not allowed.</li> <li>• You cannot set this parameter to <code>CURRENT</code>. To make this check-in the current version, use the <code>make-current</code> operation.</li> </ul>	string	One or more version labels of the checked in object	null
<i>comment</i>	Specifies the comment for a check-in operation.	string	Any valid string	null

Variable	Description	Data type	Value range	Default value
<i>page</i>	Specifies the content page number with any non-negative number adhering to the constraints defined by Content Server.	integer	$\geq 0$	0
<i>format</i>	<p>Specifies the format when checking in more than one type of content. A comma separated string can be used to specify multiple values for the <code>format</code> parameter. For example: <code>format=crtext,html,pub_html</code>.</p> <p>The format list must match the content sequence. When you don't want to specify the format for a piece of content you can leave it blank in the comma separated list but you must still include the comma as a placeholder.</p> <p>For example:  <code>format=crtext,,html</code></p> <p>When the format for a piece of content is not defined, the server uses the content media</p>	string	Comma separated string	null

Variable	Description	Data type	Value range	Default value
	<p>type to select the format.</p> <p>When the <i>all-primary</i> parameter is true and <i>page</i>=0, all primary content types must have the same format. In this case, when you define more than one format, they must be the same</p> <p>For example: format =crtext, crtext, crtext .</p> <p>When the <i>all-primary</i> parameter is true and the <i>page</i> parameter is greater than 0, you can define one format for all of the primary content. For example: format =crtext. In this case, all format values must be the same as the format of the primary content of the first page, which is page 0.</p>			
<i>make-current</i>	Determines whether or not to make this check-in the current version.	boolean	true or false	true

Variable	Description	Data type	Value range	Default value
<i>content</i> <i>-charset</i>	<p>This parameter defines the parse behavior of the server when sanitizing content. A comma separated string can be used to specify multiple values for this parameter. For example:</p> <pre>content -charset=UTF -8,GBK,ISO -8859-1</pre> <p>When the <i>content</i> <i>-charset</i> parameter is not set and content is in HTML format, the server tries to get a value for the <i>content</i> <i>-charset</i> parameter from the metadata information of the content itself.</p> <p>When the metadata has a valid charset, then it is used to sanitize the content. Otherwise, the configuration default charset is used for the <i>content</i> <i>-charset</i> parameter when a value for this parameter and the metadata</p>	string	Comma separated string of integer number values	null

Variable	Description	Data type	Value range	Default value
	<p>information are not found.</p> <p><b>Note:</b> When the charset and content do not match, the uploaded content may have an incorrectly encoded value.</p>			
<i>modifier</i>	<p>Specifies the modifier of the contents. For the primary content, the modifier is ignored.</p> <p>For example, when you import with the <i>all-primary</i> parameter set to false, the value of the modifier can be: <code>modifier =, mod1, mod2, mod3</code>.</p> <p>That means the first (or primary) content type has no modifier, and the second rendition's modifier is <code>mod1</code>, the third is <code>mod2</code> and so on.</p> <p>Setting the <i>all-primary</i> parameter to true, or when <code>content-count ≤ 1</code> causes all modifiers to be ignored.</p>	string	Comma separated string	null
* Repeatable parameter				

**Note:** Here are some additional details about the variables listed above:

- *format*

The *format* list must match the content sequence. When you don't want to specify the format for a piece of content you can leave it blank in the comma separated list but you must still include the comma as a placeholder.

For example: `format=crtext,,html`

When the format for contents is not defined, the server uses the content media type to select the format.

When the *all-primary* parameter is `true`, all primary content types must have the same format. In this case, when you define more than one format, they must be the same

For example: `format=crtext,crtext,crtext`. You can also define one format for all of the primary content: `format=crtext`

- *content-charset*

When the metadata has a valid charset, then it is used to sanitize the content.

When the *content-charset* is not provided and the content is in HTML format, the server tries to get a value for the *content-charset* parameter from the metadata information of the content itself.

The configuration default charset is used when the *content-charset* parameter and the metadata information are not found

When the charset and the content do not match, the uploaded content may have an incorrectly encoded value.

- *content-length*

When the *all-primary* parameter is set to `false`, the first value of in the *content-length* parameter is used for the primary piece of content, the other values in the comma separated string are ignored.

When storing content to Centera storage, the *content-length* parameter is mandatory and must be provided.

For other storage mediums, such as harddisk, the *content-length* parameter is optional.

In all cases, including when the *content-length* parameter is not required, if a value for it is set, the value must be an accurate byte count of the contents to be uploaded.

- *content-count*

When you import more than one piece of contents, the REST server must know how many content pieces there are in total. The total number of content pieces cannot be obtained until the end of the multipart stream. Therefore, the client must specify this number in the value of the *content-count* parameter.

When the value set in the *content-count* parameter is greater than the actual number of content pieces, an exception is thrown because the system expects more content pieces.

When the value set in the *content-count* parameter is less than the actual number of content pieces, all content pieces that exist in addition to the number of pieces set in the *content-count* parameter are ignored.



The value provided by the *content-count* must be accurate.

## Request Headers

- Authorization
- Accept
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The Request body contains one of or both of the following:

- The object metadata in XML or JSON
- The pieces of binary content

## Request Payload Samples

### Example 2-1. Metadata and Binary Content in XML

```
Content-Type: multipart/form-data; boundary=314159265358979
--314159265358979
Content-Type: application/vnd.emc.documentum+xml
<document xsi:type="dm_document"
  definition="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
  acme01/types/dm_document">
  <properties xsi:type="dm_document-properties">
    <object_name>Blank PowerPoint Presentation</object_name>
    ...
  </properties>
</document>

--314159265358979
Content-Disposition: form-data; name=binary1
Content-Type: text/plain

This is content
--314159265358979--
```

### Example 2-2. Check in Metadata and Binary Content in XML

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+xml

<object>
  <properties>
    <object_name>xyz</object_name>
  </properties>
</object>

--314159265358979
Content-Disposition: form-data; name=binary
```

```
Content-Type: text/plain

This is the first content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the second content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the third content part

--314159265358979--
```

### Example 2-3. Check in Metadata and Binary Content in JSON

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+json
{
  "properties":{
    "object_name":"xyz"
  }
}

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the first content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the second content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the third content part

--314159265358979--
```

## Request Payload Samples

### Example 2-4. Checkin Binary Contents without Metadata

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the first content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

```
This is the second content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the third content part

--314159265358979--
```

## Response Headers

- Content-Type
- Location

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Checkin successful  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
403 - Permission denied; no permission to delete the object  
404 - Object not found  
406 - Not Acceptable  
409 - Conflict. For example, the object has not been checked out  
415 - Unsupported Media Type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the object instance. The successful response contains all properties of the checked in object. Permission set is not returned.

# ACL(s)

## ACL

The ACL resource represents the object instance for an Access Control List (ACL).

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [ACLs Collection, page 71](#) and [About the Diagram](#).

## Feed

Is feed? No.

## Link Relations

The following table lists link relations for the ACL resource.

Link Relation	Description	Resource Reference
self	This ACL resource	<a href="#">ACL, page 60</a>
edit	Edit this ACL resource	<a href="#">ACL, page 60</a>
<a href="http://identifiers.emc.com/linkrel/delete">http://identifiers.emc.com/linkrel/delete</a>	Delete this ACL resource	<a href="#">ACL, page 60</a>
<a href="http://identifiers.emc.com/linkrel/associations">http://identifiers.emc.com/linkrel/associations</a>	Retrieve the sysobjects associated with the ACL	<a href="#">ACL Associations, page 80</a>

## Operations

The ACL resource supports the following HTTP methods.

Method	Description
POST	Updates the attributes of the ACL object instance

Method	Description
GET	Retrieves the attributes and other information of the ACL object instance
DELETE	Deletes the ACL object instance from the repository

## Get an ACL Object Instance

Retrieves the attributes and other information of the ACL object instance. Attributes are returned as embedded elements in the Response message body. Other information is available within the link relations of the Response message body.

### HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

- view
- links
- For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

The properties block contains the following two sets of data:

- The metadata of this ACL
- ACL entries.

ACL entries are not presented one by one. Instead, an ACL resource displays the details of the entries in the following six elements:

- `r_accessor_name`
- `r_accessor_permit`
- `r_accessor_xpermit`
- `r_is_group`
- `r_permit_type`
- `r_application_permit`

These six elements contain the same number of items, which is also the number of entries in the ACL. The combination of items and elements positioning creates a key value pair that represents one ACL entry.

In the following code sample, the combination of all first items indicates that the accessor `dm_world`, which is an individual user where `r_is_group` is `false`, has Read basic permission (`r_accessor_permit` is 3), as well as EXECUTE\_PROCEDURE and CHANGE\_LOCATION extended permissions (`r_accessor_xpermit` is 0) on the objects to which this ACL is attached. It also indicates that the entry's permit type is AccessPermit (`r_permit_type` is 0).

**Example 2-5. XML Response**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+xml; charset=UTF-8

```

<?xml version='1.0' encoding='UTF-8'?>
<acl
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_acl"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl">
  <properties>
    <object_name>dm_4500000580000d07</object_name>
    <description>dm_4500000580000d07</description>
    <owner_name>dave</owner_name>
    <r_is_internal>true</r_is_internal>
    <r_accessor_name>
      <item>dm_world</item>
      <item>dm_owner</item>
      <item>dm_group</item>
    </r_accessor_name>
    <r_accessor_permit>
      <item>3</item>
      <item>7</item>
      <item>5</item>
    </r_accessor_permit>
    <r_accessor_xpermit>
      <item>0</item>
      <item>0</item>
      <item>3</item>
    </r_accessor_xpermit>
    <r_is_group>
      <item>>false</item>
      <item>>false</item>
      <item>>false</item>
    </r_is_group>
    <globally_managed>>false</globally_managed>
    <acl_class>0</acl_class>
    <r_has_events>>false</r_has_events>
    <r_permit_type>
      <item>0</item>
      <item>0</item>
      <item>0</item>
    </r_permit_type>
    <r_application_permit>
      <item></item>
      <item></item>
      <item></item>
    </r_application_permit>
    <i_has_required_groups>>false</i_has_required_groups>
    <i_has_required_group_set>>false</i_has_required_group_set>
    <i_has_access_restrictions>>false</i_has_access_restrictions>
    <r_template_id>0000000000000000</r_template_id>
    <r_alias_set_id>0000000000000000</r_alias_set_id>
    <i_partition>0</i_partition>
    <i_is_replica>>false</i_is_replica>
    <i_vstamp>0</i_vstamp>
    <r_object_id>4500000580000d07</r_object_id>
  </properties>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/
      REPO/acls/4500000580000d07"/>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/
      REPO/acls/4500000580000d07"/>
  </links>

```

```
<link rel="http://identifiers.emc.com/linkrel/associations"
      href="http://localhost:8080/dctm-rest/repositories/
        REPO/acls/4500000580000d07/associations"/>
<link rel="http://identifiers.emc.com/linkrel/delete"
      href="http://localhost:8080/dctm-rest/repositories/
        REPO/acls/4500000580000d07"/>
</links>
</acl>
```

**Example 2-6. JSON Response**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```
{
  "name": "acl",
  "type": "dm_acl",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl",
  "properties": {
    "object_name": "dm_4500000580000d07",
    "description": "dm_4500000580000d07",
    "owner_name": "dave",
    "r_is_internal": true,
    "r_accessor_name": [
      "dm_world",
      "dm_owner",
      "dm_group"
    ],
    "r_accessor_permit": [
      3,
      7,
      5
    ],
    "r_accessor_xpermit": [
      0,
      0,
      3
    ],
    "r_is_group": [
      false,
      false,
      false
    ],
    "globally_managed": false,
    "acl_class": 0,
    "r_has_events": false,
    "r_permit_type": [
      0,
      0,
      0
    ],
    "r_application_permit": [
      "",
      "",
      ""
    ],
    "i_has_required_groups": false,
    "i_has_required_group_set": false,
    "i_has_access_restrictions": false,
    "r_template_id": "0000000000000000",
    "r_alias_set_id": "0000000000000000",
    "i_partition": 0,
  }
}
```



```

    "i_is_replica": false,
    "i_vstamp": 0,
    "r_object_id": "4500000580000d07"
  },
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000d07"
    },
    {
      "rel": "edit",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000d07"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/delete",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000d07/"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/associations",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000d07/associations"
    }
  ]
}

```

## Update the ACL Object Instance

Updates the ACL with specified attributes. When a client provides a request body that contains an undefined or non-editable attributes, the server throws an exception. An ACL object can only be modified by its owner or the superuser.

The Server throws an exception when the update Request provides attributes whose values do not have matching positions in the repeating attributes, such as *r\_accessor\_name* and *r\_accessor\_permit*.

## HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

A single XML or JSON object instance. Only the following properties can be put into the Request message body:

- description
- acl\_class
- r\_accessor\_name
- r\_accessor\_permit
- r\_accessor\_xpermit
- r\_application\_permit
- r\_permit\_type

### Note:

- Except for description and acl\_class, all editable properties are prefixed with r\_, and must contain the same number of items. Otherwise, the REST server throws an exception.
- Every ACL has two built-in accessors: dm\_world and dm\_owner. The default permission value of these two accessors is defined by the Content Server configuration.
- To update an ACL template instance (acl\_class=2), you must change it to a regular (acl\_class=0) ACL, public (acl\_class=3) ACL, or template (acl\_class=1) ACL first. Additionally, you cannot change a regular, public, or template ACL into an ACL template instance.

### Example 2-7. XML Request

```
<?xml version='1.0' encoding='UTF-8'?>
<acl xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_acl">
  <properties>
    <r_accessor_name>
      <item>{accessor-name_1}</item>
      <item>{accessor-name_2}</item>
    </r_accessor_name>
    <r_accessor_permit>
      <item>{accessor-permit_1}</item>
      <item>{accessor-permit_1}</item>
    </r_accessor_permit>
    <r_permit_type>
      <item>{permit-type_1}</item>
      <item>{permit-type_1}</item>
    </r_permit_type>
    ...
  </properties>
</acl>
```

**Example 2-8. JSON Request**

```
{
  "name": "acl",
  "type": "dm_acl",
  "properties": {
    "r_accessor_name": [
      "{accessor-name_1}",
      "{accessor-name_1}"
    ],
    "r_accessor_permit": [
      {accessor-permit_1},
      {accessor-permit_2}
    ],
    "r_permit_type": [
      {permit-type_1},
      {permit-type_2}
    ]
    .....
  }
}
```

**Response Headers**

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

**Response Media Types**

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

**Response Status**

- 200 Updated successfully
- 400, 401, 403, 404, 406, 409, 415, 500
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

**Response Body**

When the Response is successful, it contains all the attributes of the updated object.

**Example 2-9. XML Response**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+xml; charset=UTF-8

```

<?xml version='1.0' encoding='UTF-8'?>
<acl xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="dm_acl" definition="{typeResourceUri}">
  <properties>
    <object_name>{object-name}</object_name>
    <description>{description of this object}</description>
    <owner_name>{object-owner}</owner_name>
    <r_accessor_name>
      <item>dm_world</item>
      <item>dm_owner</item>
      <item>{accessor-name_1}</item>
      <item>{accessor-name_2}</item>
    </r_accessor_name>
    <r_accessor_permit>
      <item>3</item>
      <item>7</item>
      <item>{accessor-permit_1}</item>
      <item>{accessor-permit_2}</item>
    </r_accessor_permit>
    <r_permit_type>
      <item>0</item>
      <item>0</item>
      <item>{permit-type_1}</item>
      <item>{permit-type_2}</item>
    </r_permit_type>
    ...
  </properties>
  <links>
    <link rel="self" href="{aclResourceUri}"/>
    <link rel="edit" href="{aclResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="{aclResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/associations"
      href="{aclAssociationsResourceUri}"/>
  </links>
</acl>

```

### Example 2-10. JSON Response

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```

{
  "name": "acl",
  "type": "dm_acl",
  "definition": "{typeResourceUri}",
  "properties": {
    "object_name": "{object-name}",
    "description": "{description of this object}",
    "owner_name": "{object-owner}",
    "r_accessor_name": [
      "dm_world",
      "dm_owner",
      "{accessor-name_1}",
      "{accessor-name_2}"
    ],
    "r_accessor_permit": [
      3,
      7,
      {accessor-permit_1},
      {accessor-permit_2}
    ],
    "r_permit_type": [

```

```

        0,
        0,
        {permit-type_1},
        {permit-type_2}
    ],
    ...
},
"links": [
    {
        "rel": "self",
        "href": "{aclResourceUri}"
    },
    {
        "rel": "edit",
        "href": "{aclResourceUri}"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/delete",
        "href": "{aclResourceUri}"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/associations",
        "href": "{aclAssociationsResourceUri}"
    }
]
}

```

## Delete the ACL Object Instance

Deletes the ACL object instance. When deletion fails, an exception is thrown.

## HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data Type	Value Range
force	Indicates whether or not to delete the ACL even when it is associated with other objects in the repository.  After the deletion, objects associated	boolean	true: Destroys the ACL object even if it is referenced by other objects in the repository. Only the super user has the necessary permissions to force delete an ACL object.

Variable	Description	Data Type	Value Range
	<p>with this ACL have no referenced ACL, which means that the following constraints apply to these objects:</p> <ul style="list-style-type: none"><li>• These objects become read-only</li><li>• These objects are accessible only to the object's owner and users with the <code>SuperUser</code> privilege.</li><li>• Only users with the <code>SuperUser</code> privilege can delete these objects.</li></ul>		<p><code>false</code>: Does not destroy the ACL object if it is referenced by other objects.</p> <p>Default value is <code>false</code>.</p>

### Request Headers

- Authorization
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

N/A

### Response Media Types

N/A

## Response Status

- 204 Deleted successfully
- 400, 401, 403, 500
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

N/A

## ACLs Collection

The ACLs Collection represents the collection of all Access Control Lists (ACLs) in the repository that are available to the user that is currently logged into the system.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Is feed? Yes.

Feed id	Feed Title	Feed Updated	Support POST
<Feed URI>	ACLs	The Server's current time	Yes

Entry	Entry id	Entry Title	Entry Updated
An ACL object	The object id of the ACL entry	The object name of the ACL entry	The Server's current time

## Link Relations

The following table lists link relations for the ACL resource.

Link Relation	Description	Resource Reference
self	This ACL resource	<a href="#">ACLs Collection, page 71</a>
first, last, next, previous	Pagination links	<a href="#">ACLs Collection, page 71</a>

## Operations

The ACL resource supports the following HTTP methods.

Method	Description
POST	Creates an ACL
GET	Retrieves ACLs that are available to the current user from a repository

### Get the ACLs Collection

Lists all the available ACLs in the repository for the current user.

#### HTTP Method

GET

#### Request Media Types

N/A

#### Request Query Parameters

- inline
- sort
- page
- items-per-page
- include-total
- view
- links
- filter
- For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).



## Request Headers

- Accept
- Authorization
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-11. XML Response

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+xml;charset=UTF-8

```
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/acls</id>
  <title>Acls</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-06-09T10:20:41.905+00:00</updated>
```

```

<dm:page
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
</dm:page>
<dm:items-per-page
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">2
</dm:items-per-page>
<link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
  /acsls?filter=owner_name=%27dave%27%20and%20r_is_internal=false
  &items-per-page=2"/>
<link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/
  acsls?filter=owner_name=%27dave%27%20and%20r_is_internal=false
  &items-per-page=2&page=2"/>
<link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/
  acsls?filter=owner_name=%27dave%27%20and%20r_is_internal=false
  &items-per-page=2&page=1"/>
<entry>
  <id>4500000580001d18</id>
  <title>test_acl_1377352452</title>
  <updated>2015-06-09T10:20:41.906+00:00</updated>
  <published>2015-06-09T10:20:41.906+00:00</published>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/acsls/
    4500000580001d18"/>
  <content type="application/vnd.emc.documentum+xml"
    src="http://localhost:8080/dctm-rest/repositories/REPO/acsls/4500000580001d18"/>
</entry>
<entry>
  <id>4500000580001d58</id>
  <title>3fb1de-8f77-4064-92a8-90d8950676</title>
  <updated>2015-06-09T10:20:41.906+00:00</updated>
  <published>2015-06-09T10:20:41.906+00:00</published>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/acsls/
    4500000580001d58"/>
  <content type="application/vnd.emc.documentum+xml"
    src="http://localhost:8080/dctm-rest/repositories/REPO/acsls/4500000580001d58"/>
</entry>
</feed>

```

**Example 2-12. JSON Response**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/acsls",
  "title": "ACLs",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-06-11T03:29:15.085+00:00",
  "page": 1,
  "items-per-page": 2,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acsls?
        filter=owner_name=%27dave%27%20and%20r_is_internal=false
        &items-per-page=2"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/acsls?

```

```

        filter=owner_name=%27dave%27%20and%20r_is_internal=false
        &items-per-page=2&page=2"
    },
    {
        "rel": "first",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls?
        filter=owner_name=%27dave%27%20and%20r_is_internal=false
        &items-per-page=2&page=1"
    }
],
"entries": [
    {
        "id": "4500000580001d37",
        "title": "test_acl_1913536617",
        "updated": "2015-06-11T03:29:15.127+00:00",
        "published": "2015-06-11T03:29:15.127+00:00",
        "links": [
            {
                "rel": "edit",
                "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/
                4500000580001d37"
            }
        ],
        "content": {
            "type": "application/vnd.emc.documentum+json",
            "src": "http://localhost:8080/dctm-rest/repositories/REPO/acls/
            4500000580001d37"
        }
    },
    {
        "id": "4500000580001d58",
        "title": "restAcl8f62ed43-f9",
        "updated": "2015-06-11T03:29:15.130+00:00",
        "published": "2015-06-11T03:29:15.130+00:00",
        "links": [
            {
                "rel": "edit",
                "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/
                4500000580001d58"
            }
        ],
        "content": {
            "type": "application/vnd.emc.documentum+json",
            "src": "http://localhost:8080/dctm-rest/repositories/REPO/acls/
            4500000580001d58"
        }
    }
]
}

```

## Create an ACL

Create a new ACL in the repository.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

- An XML or JSON message representing the ACL object that is created.
- The Server throws an exception when undefined or read-only attributes are provided by the client.
- An ACL instance cannot be created manually. In other words, you cannot create an ACL with the value of the `acl_class` attribute set to 2. An ACL instance is automatically created when an ACL template is applied to a sysobject by the owner of the repository.
- The `r_object_type` attribute is not set because its value is `dm_acl` by default.

### Example 2-13. XML Request

```
<?xml version='1.0' encoding='UTF-8'?>
<acl xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_acl"
    definition="{typeResourceUri}">
  <properties>
    <object_name>{object-name}</object_name>
    <description>{description of this object}</description>
    <owner_name>{object-owner}</owner_name>
    <r_accessor_name>
      <item>dm_world</item>
      <item>dm_owner</item>
      <item>{accessor-name_1}</item>
      <item>{accessor-name_2}</item>
    </r_accessor_name>
    <r_accessor_permit>
      <item>3</item>
      <item>7</item>
      <item>{accessor-permit_1}</item>
      <item>{accessor-permit_2}</item>
    </r_accessor_permit>
    <r_permit_type>
      <item>0</item>
      <item>0</item>
      <item>{permit-type_1}</item>
```

```

        <item>{permit-type_2}</item>
      </r_permit_type>
      ... ..
    </properties>
  </acl>

```

#### Example 2-14. JSON Request

```

{
  "name": "acl",
  "type": "dm_acl",
  "definition": "{typeResourceUri}",
  "properties": {
    "object_name": "{object-name}",
    "description": "{description of this object}",
    "owner_name": "{object-owner}",
    "r_accessor_name": [
      "dm_world",
      "dm_owner",
      "{accessor-name_1}",
      "{accessor-name_2}"
    ],
    "r_accessor_permit": [
      3,
      7,
      {accessor-permit_1},
      {accessor-permit_2}
    ],
    "r_permit_type": [
      0,
      0,
      {permit-type_1},
      {permit-type_2}
    ],
    ... ..
  }
}

```

#### Response Headers

- Content-Type
- Location
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

#### Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 201 Created successfully
- 400, 401, 403, 404, 406, 409, 415, 500
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

Every ACL has two built-in accessors:

- dm\_world
- dm\_owner

Even when these two accessors and their permissions are not provided by the client, the server grants them default permissions in the ACL object. The default permission value for each of these two accessors is defined by the configuration of the Content Server.

### Example 2-15. XML Response

Status Code: 201 Created

Location: <http://localhost/dctm-rest/repositories/REPO/acls/45000005800019c1>

Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```
<?xml version='1.0' encoding='UTF-8'?>
<acl xmlns="http://identifiers.emc.com/vocab/documentum"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_acl"
      definition="{typeResourceUri}">
  <properties>
    <object_name>{object-name}</object_name>
    <description>{description of this object}</description>
    <owner_name>{object-owner}</owner_name>
    <r_accessor_name>
      <item>dm_world</item>
      <item>dm_owner</item>
      <item>{accessor-name_1}</item>
      <item>{accessor-name_2}</item>
    </r_accessor_name>
    <r_accessor_permit>
      <item>3</item>
      <item>7</item>
      <item>{accessor-permit_1}</item>
      <item>{accessor-permit_2}</item>
    </r_accessor_permit>
    <r_permit_type>
      <item>0</item>
      <item>0</item>
      <item>{permit-type_1}</item>
      <item>{permit-type_2}</item>
    </r_permit_type>
    ... ..
  </properties>
  <links>
    <link rel="self" href="{aclResourceUri}"/>
    <link rel="edit" href="{aclResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
          href="{aclResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/associations"
```

```

        href="{aclAssociationsResourceUri}"/>
    </links>
</acl>

```

### Example 2-16. JSON Response

Status Code: 201 Created

Location: <http://localhost/dctm-rest/repositories/REPO/acls/45000005800019c1>

Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```

{
  "name": "acl",
  "type": "dm_acl",
  "definition": "{typeResourceUri}",
  "properties": {
    "object_name": "{object-name}",
    "description": "{description of this object}",
    "owner_name": "{object-owner}",
    "r_accessor_name": [
      "dm_world",
      "dm_owner",
      "{accessor-name_1}",
      "{accessor-name_2}"
    ],
    "r_accessor_permit": [
      3,
      7,
      {accessor-permit_1},
      {accessor-permit_2}
    ],
    "r_permit_type": [
      0,
      0,
      {permit-type_1},
      {permit-type_2}
    ],
    ... ..
  },
  "links": [
    {
      "rel": "self",
      "href": "{aclResourceUri}"
    },
    {
      "rel": "edit",
      "href": "{aclResourceUri}"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/delete",
      "href": "{aclResourceUri}"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/associations",
      "href": "{aclAssociationsResourceUri}"
    }
  ]
}

```

# ACL Associations

The ACL Associations resource represents a collection of all the sysobjects that are associated with a specified ACL.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [ACL, page 60](#) and [About the Diagram](#).

## Feed

Is feed? Yes.

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the ACL Associations resource	Objects	Server's current time	<a href="#">SysObject, page 472</a>	No

Entry	Entry ID	Entry Title	Entry Updated
<a href="#">SysObject, page 472</a>	The ObjectID of a SysObject	The object name of SysObject	The value of the r_modify_date

## Link Relation

The following table lists general link relations in the ACL Associations resource:

Link Relation	Description	Resource Reference
self	The ACL Associations collection	<a href="#">ACL Associations, page 80</a>
first, last, next, previous	Pagination links	<a href="#">ACL Associations, page 80</a>

## Operations

The ACL Association resource supports the following HTTP method:



Method	Description
GET	Get all SysObjects associated with a specified ACL.

## List All Available Associated SysObjects

List all available sysobjects that are associated with a specific ACL object.

### HTTP Method

GET

### Request Media Types

N/A.

### Request Query Parameters

This method supports the following common query parameters:

- inline
- sort
- page
- items-per-page
- include-total
- view
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

### Request Body

N/A.

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Invalid syntax or missing a required value  
404 - ACL not found  
406 - Not Acceptable  
415 - Unsupported media type  
500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-17. XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>{aclAssociationsResourceUri}</id>
  <title>Objects</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>{current_time}</updated>
  <dm:page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">{page}
  </dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">{items-per-page}
  </dm:items-per-page>
  <link rel="self" href="{aclAssociationsResourceUri}"/>
  .....pagination links.....
  <entry>
    <id>{objectResourceUri}</id>
    <title>{object_name}</title>
    <author>
      <name>{author_name}</name>
      <uri>{userResourceUri}</uri>
    </author>
```

```

        <summary>{r_object_type} {r_object_id}</summary>
        <updated>{r_modify_date} or {current_time}</updated>
        <published>{creation-date} or {current_time}</published>
        <link rel="edit" href="{objectResourceUri}"/>
        <content type="application/vnd.emc.documentum+xml"
            src="{objectResourceUri}"/>
    </entry>
    .....
</feed>

```

### Example 2-18. JSON Response

```

{
  "id": "{aclAssociationsResourceUri}",
  "title": "Objects",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "{current_time}",
  "page": {page},
  "items-per-page": {items-per-page},
  "links": [
    {
      "rel": "self",
      "href": "{aclAssociationsResourceUri}"
    },
    .....pagination links.....
  ],
  "entries": [
    {
      "id": "{objectResourceUri}",
      "title": "{object_name}",
      "author": [
        {
          "name": "{author_name}",
          "uri": "{userResourceUri}"
        }
      ],
      "summary": "{r_object_type} {r_object_id}",
      "updated": "{r_modify_date} or {current_time}",
      "published": "{creation-date} or {current_time}",
      "links": [
        {
          "rel": "edit",
          "href": "{objectResourceUri}"
        }
      ],
      "content": {
        "type": "application/vnd.emc.documentum+json",
        "src": "{objectResourceUri}"
      }
    },
    .....
  ]
}

```

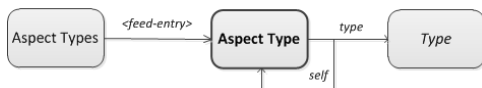
# Aspect Type(s)

## Aspect Type

The Aspect Type resource represents an aspect type in Content Server.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Aspect Types, page 89](#) and [About the Diagram](#).

## Link Relation

The following table lists general link relations in the Aspect Type resource.

Link Relation	Description	Resource Reference
self	This Aspect type.	<a href="#">Aspect Type, page 84</a>
type [1]	When an aspect type defines its own properties, this link relation points to those properties.	<a href="#">Type, page 482</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:  <code>http://identifiers.emc.com/linkrel/</code>		

## Operations

The Aspect Type resource supports the following HTTP method.

Method	Description
GET	Get an aspect type.

## Get an Aspect Type

Get the metadata of an Aspect Type resource.

## HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following request query parameters:

- view, inline
- locale

A string that specifies the value of a valid locale for the Request. For example, it can be used to get the label text of an aspect type object. The default value is `null`.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied  
404 - Aspect type not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the Aspect Type resource:

- When the aspect type defines additional properties, the `type` link relation points to the Type resource that contains the properties that this Aspect Type resource defines.
- The properties block contains the following two sets of data:
  - The metadata of this Aspect Type resource.
  - The properties that this Aspect Type resource defines. These properties are not presented one by one. Instead, an Aspect Type resource displays the metadata of these properties in eight elements, all of which are prefixed with `attr_` as shown here:
    - Name of the property as shown in the `attr_name` element
    - Whether the property is a repeating property as shown in the `attr_repeating` element
    - Data type of the properties as shown in the `attr_type` element
    - Length of the properties as shown in the `attr_length` element
    - Label of the property as shown in the `attr_label` element
    - Whether the property is a hidden property as shown in the `attr_hidden` element
    - Whether the property is a non-null property as shown in the `attr_notnull` element
    - Whether the property is a read-only property as shown in the `attr_readonly` element

These eight elements contain the same number of items, which is also the number of properties this aspect type defines. The combination of items in the same position of these elements represents one property this aspect type defines. In the following sample, the combination of all first items indicates that this resource defines a string property named `annotation_name`. The property `annotation_name` is a repeating property with a maximum length of 225.

**Note:** When the value of an element that is prefixed with `attr_` is null, that element is not shown in the Response. That is why the following code sample only contains four elements.

## XML Response

```

<?xml version="1.0" encoding="UTF-8"?>
<aspect-type
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/dmc_aspect_type.xml"
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dmc_aspect_type">
  <properties>
    <group_name>docu</group_name>
    <acl_name>BOF_acl</acl_name>
    <world_permit>3</world_permit>
    <creation-date>2015-03-03T02:29:34.000+00:00</creation-date>
    <r_object_type>dmc_aspect_type</r_object_type>
    <min_dfc_version>6.0</min_dfc_version>
    <owner_name>Administrator</owner_name>
    <implementation_technology>Java</implementation_technology>
    <primary_class>com.documentum.smart.impl.aspect.AnnotatableAspect</primary_class>
    <owner_permit>7</owner_permit>
    <a_bof_version/>
    <a_interfaces xsi:nil="true"/>
    <r_object_id>0b000005800005b0</r_object_id>
    <r_modify_date>2015-03-03T02:30:13.000+00:00</r_modify_date>
    <group_permit>1</group_permit>
    <i_folder_id>
      <item>0b00000580000180</item>
    </i_folder_id>
    <r_folder_path>
      <item>/System/Modules/Aspect/my_aspect_1</item>
    </r_folder_path>
    <object_name>my_aspect_1</object_name>
    <target_object_type xsi:nil="true"/>
    <r_creator_name>Administrator</r_creator_name>
    <r_modifier>Administrator</r_modifier>
    <acl_domain>REPO_ADMIN</acl_domain>
    <a_module_type>Aspect</a_module_type>
    <attr_name>
      <item>annotation_name</item>
      <item>module_name</item>
      <item>annotation_metadata</item>
      <item>i_partition</item>
    </attr_name>
    <attr_repeating>
      <item>true</item>
      <item>true</item>
      <item>true</item>
      <item>false</item>
    </attr_repeating>
    <attr_type>
      <item>string</item>
      <item>string</item>
      <item>string</item>
      <item>integer</item>
    </attr_type>
    <attr_length>
      <item>255</item>
      <item>255</item>
      <item>1024</item>
      <item xsi:nil="true"/>
    </attr_length>
  </properties>
  <links>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/my_aspect_1"
      rel="self"/>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/types/dmi_03000005800001d3"
      rel="http://identifiers.emc.com/linkrel/type"/>
  </links>

```

```
</links>
</aspect-type>
```

## JSON Response

```
{
  "name" : "aspect-type",
  "type" : "dmc_aspect_type",
  "definition" : "http://localhost:8080/dctm-rest/repositories/REPO/types/dmc_aspect_type",
  "properties" : {
    "group_name" : "docu",
    "acl_name" : "BOF_acl",
    "world_permit" : 3,
    "creation-date" : "2015-03-03T02:29:34.000+00:00",
    "r_object_type" : "dmc_aspect_type",
    "min_dfc_version" : "6.0",
    "owner_name" : "Administrator",
    "implementation_technology" : "Java",
    "primary_class" : "com.documentum.smart.impl.aspect.AnnotatableAspect",
    "owner_permit" : 7,
    "a_bof_version" : "",
    "a_interfaces" : null,
    "r_object_id" : "0b000005800005b0",
    "r_modify_date" : "2015-03-03T02:30:13.000+00:00",
    "group_permit" : 1,
    "i_folder_id" : [
      "0b00000580000180"
    ],
    "r_folder_path" : [
      "/System/Modules/Aspect/my_aspect_1"
    ],
    "object_name" : "my_aspect_1",
    "target_object_type" : null,
    "r_creator_name" : "Administrator",
    "r_modifier" : "Administrator",
    "acl_domain" : "REPO_ADMIN",
    "a_module_type" : "Aspect",
    "attr_name" : [
      "annotation_name",
      "module_name",
      "annotation_metadata",
      "i_partition"
    ],
    "attr_repeating" : [
      true,
      true,
      true,
      false
    ],
    "attr_type" : [
      "string",
      "string",
      "string",
      "integer"
    ],
    "attr_length" : [
      255,
      255,
      1024,
      null
    ]
  },
  "links" : [
    {
      "rel" : "self",
```



```

    "href" : "http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/
              my_aspect_1"
  },
  {
    "rel" : "http://identifiers.emc.com/linkrel/type",
    "href" : "http://localhost:8080/dctm-rest/repositories/REPO/types/
              dmi_03000005800001d3"
  }
]
}

```

## Aspect Types

The Aspect Types resource represents a collection of Aspect Type resources.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the Aspect Types resource without the file extension	Aspect types	Server's current time	<a href="#">Aspect Type, page 84</a>	No

Entry ID	Entry Title	Entry Updated
URI of the Aspect Type	Aspect Type name	r_modify_date of the Aspect Type

## Link Relation

The following table lists general link relations in the Aspect Types resource.

Link Relation	Description	Resource Reference
self	This Aspect Types resource	<a href="#">Aspect Types, page 89</a>
first, last, next, previous	Pagination links	<a href="#">Aspect Types, page 89</a>

## Operations

The Aspect Types resource supports the following HTTP method.

Method	Description
GET	Get all aspect types in a repository

### Get Aspect Types

Get all aspect types in a repository.

#### HTTP Method

GET

#### Request Media Types

N/A

#### Request Query Parameters

This method supports the following common query parameters:

- inline
- sort
- page
- items-per-page
- include-total
- view
- filter
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

#### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

## Response Status

200 - Retrieved successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied  
404 - Object not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the Aspect Types collection.

- Properties that an Aspect Type resource defines, which are prefixed with `attr_` are not displayed in the response. Additionally, these properties do not take effect in the `view`, `filter`, and `sort` parameters.

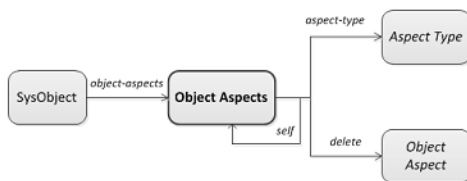
# Object Aspect(s)

## Object Aspects

The Object Aspects resource represents the aspects attached to a specified object in a repository. Using this resource, you can attach extra properties to certain objects.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



**See Also:** [SysObject](#), page 472, [Document](#), page 195, [Folder](#), page 209, [Cabinet](#), page 120, and [About the Diagram](#).

## Link Relation

The following table lists general link relations in the Object Aspects resource.

Link Relation	Description	Resource Reference
self	This Object Aspects resource	<a href="#">Object Aspects</a> , page 92
aspect-type [1]	Type definition of an aspect	<a href="#">Aspect Type</a> , page 84
delete [1]	Detach an aspect from the SysObject	<a href="#">Object Aspect</a> , page 98
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Object Aspects resource supports the following HTTP methods.

Method	Description
GET	Get all aspects attached to a SysObject
POST	Attach one or more aspects to a SysObject

## Get Attached Aspects

Get all aspects attached to a specified SysObject.

### HTTP Method

GET

### Server Accepted Request Media Types

None.

### Query Parameters

This method supports the following common query parameter:

- links

For more details about the query parameters, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more details about HTTP headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

None.

### Response Headers

- Content-Type

For more details about HTTP headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing)

## Response Status

200 - Information retrieved successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied  
404 - Object not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the aspects attached to the SysObject. The Response body mainly contains two parts:

- The first part is a set of `aspect` elements, each of which represents an aspect attached to the SysObject.
- The second part is the `links` block. Each attached aspect has two related links:
  - An `aspect-type` link pointing to the corresponding aspect type definition. For more information, see [Aspect Type](#)
  - A `delete` link enabling you to detach the aspect. For more information, see [Object Aspect](#).

## XML Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<object-aspects xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <aspect>my_aspect_1</aspect>
  <aspect>dm_checkin_control</aspect>
  <aspect>dm_version_behavior</aspect>
  <links>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580002523/aspects" rel="self"/>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580002523/aspects/my_aspect_1"
      rel="http://identifiers.emc.com/linkrel/delete" title="my_aspect_1"/>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/my_aspect_1"
      rel="http://identifiers.emc.com/linkrel/aspect-type" title="my_aspect_1"/>
  </link>
</object-aspects>
```

```

        href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/
        aspects/dm_checkin_control"
        rel="http://identifiers.emc.com/linkrel/delete" title="dm_checkin_control"/>
    <link
        href="http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/dm_checkin_control"
        rel="http://identifiers.emc.com/linkrel/aspect-type" title="dm_checkin_control"/>
    <link
        href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/
        aspects/dm_version_behavior"
        rel="http://identifiers.emc.com/linkrel/delete" title="dm_version_behavior"/>
    <link
        href="http://localhost:8080/dctm-rest/repositories/REPO/aspect-types
        /dm_version_behavior"
        rel="http://identifiers.emc.com/linkrel/aspect-type" title="dm_version_behavior"/>
</links>
</object-aspects>

```

## JSON Representation

```

{
  "aspects" : [
    "my_aspect_1",
    "dm_checkin_control",
    "dm_version_behavior"
  ],
  "links" : [
    {
      "rel" : "self",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/aspects/
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/delete",
      "title" : "my_aspect_1",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/
      aspects/my_aspect_1"
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/aspect-type",
      "title" : "my_aspect_1",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/my_aspect_1"
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/delete",
      "title" : "dm_checkin_control",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/
      aspects/dm_checkin_control"
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/aspect-type",
      "title" : "dm_checkin_control",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/dm_checkin_control"
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/delete",
      "title" : "dm_version_behavior",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/
      aspects/dm_version_behavior"
    },
    {
      "rel" : "http://identifiers.emc.com/linkrel/aspect-type",
      "title" : "dm_version_behavior",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/aspect-types/
      dm_version_behavior"
    }
  ]
}

```

## Attach Aspects to a SysObject

Attach one or more aspects to a specified SysObject.

### HTTP Method

POST

### Server Accepted Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

### Query Parameters

None.

### Request Headers

- Authorization
- Content-Type
- Accept

For more details about HTTP headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

To attach aspects to a SysObject, specify the aspect type names in the request body.

#### XML Representation

```
POST http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/aspects
Accept: application/vnd.emc.documentum+xml
Content-Type: application/vnd.emc.documentum+xml
<object-aspects xmlns="http://identifiers.emc.com/vocab/documentum"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <aspect>my_aspect_1</aspect>
  <aspect>dm_checkin_control</aspect>
  <aspect>dm_version_behavior</aspect>
</object-aspects>
```

#### JSON Representation

```
POST http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002523/aspects
Accept: application/vnd.emc.documentum+json
Content-Type: application/vnd.emc.documentum+json{
  "aspects":[
    "my_aspect_1", "dm_checkin_control", "dm_version_behavior"
  ]
}
```



```
}
```

## Response Headers

- Content-Type

For more details about HTTP headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing)

## Response Status

201 - Attached successfully  
 400 - Bad request; invalid property name or value  
 401 - Authentication failed  
 403 - Permission denied  
 404 - Object not found  
 406 - Not Acceptable  
 409 - State conflicted  
 415 - Unsupported media type  
 500 - Other unexpected server error

## Response Body

XML or JSON representation of all aspects attached to the SysObject.

## Object Properties Attached via Aspects

After you attach an aspect that has new properties defined to an object, the object will contain these properties in its representation. Names of the properties attached via aspects are presented in the following pattern:

*aspect\_name.property\_name.*

The attached aspect properties can be updated as other common object properties.

### Example 2-19. Object with Properties Attached via Aspects

```
{
  "properties" : {
```

```

    "r_object_id" : "0900000580003c83"
    "object_name" : "my doc",
    "r_object_type" : "dm_document",
    "my_aspect_1.annotation_name" : "anno",
    "my_aspect_1.module_name" : "mod",
    "my_aspect_1.annotation_metadata" : "meta",
    "my_aspect_1.i_partition" : 0,
  },
  "links" : [...]
}

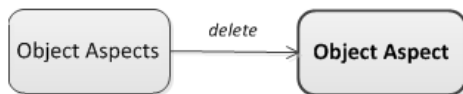
```

## Object Aspect

The Object Aspect resource is only used for detaching an aspect from a specified object.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Object Aspects, page 92](#) and [About the Diagram](#).

## Link Relation

None.

## Operations

The Object Aspect resource supports the following HTTP method.

Method	Description
DELETE	Detach an aspect from an object

## Detach an Aspect

Detach an aspect from an object.

## HTTP Method

DELETE

## Server Accepted Request Media Types

None.

## Query Parameters

None.

## Request Headers

- Authorization

For more details about HTTP headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

None.

### Example 2-20. Detaching an Aspect

```
DELETE http://localhost:8080/dctm-rest/repositories/REPO/objects/  
0900000580002523/aspects/my_aspect_1
```

## Response Headers

None.

## Supported Response Media Types

None.

## Response Status

- 204 - Detach the aspect successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied.
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful detach operation. The Response body contains no content.

## Batches

The Batches resource enables you to perform multiple operations using a single HTTP request. For example, you can create 10 cabinets, modify 5 folders, and delete 30 documents using one request. All resources except the Batches resource itself and the [Content Media](#) resource can be embedded in a batch request.

**Note:** The `rest.batch.operations.max.count` property in `rest-api-runtime.properties` determines the maximum number of operations allowed to be embedded in one batch request, which defaults to 1,000. To embed more operations in one batch request, specify a larger value for `rest.batch.operations.max.count`. However, the performance may degrade when the number of operations grows.

In addition to `rest.batch.operations.max.count`, how the system handles batch requests also depends on the following runtime properties in `rest-api-runtime.properties`.

Property	Description	Default
<code>rest.batch.running.tasks.max.count</code>	Specifies the maximum number of batch requests that can be executed simultaneously.  The system rejects new batch requests if the maximum number is reached.	50
<code>rest.batch.core.thread.pool.size</code>	Specifies the size of the thread pool used to execute background tasks.	5
<code>rest.batch.multipart.cache.max.size</code>	Specifies the maximum cache size in megabyte for a multipart batch request.  Setting the value of this property to 0 disables caching for the multipart batch requests.	20

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository](#), page 369 and [About the Diagram](#).

## Link Relations

The Batches resource does not contain any link relations.

## Operations

The Batches resource supports the following HTTP method.

Method	Description
POST	Create and execute a batch request.

## Create and Execute a Batch Request

A POST request to the Batches resource creates a Batch resource, which enables you to perform multiple operations in a single HTTP request. The Request contains the detail of each embedded operation. Additionally, you are able to control how the REST server processes the batch request (such as transactional vs. non-transactional and sequential vs. non-sequential) by specifying certain properties in the initial request.

### Supported HTTP Method

POST

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- multipart/related

### Request Query Parameters

N/A

### Request Headers

- Authorization

The Authorization header (user credential token) is applied to all operations within the batch request. Embedded operations in the batch request cannot set their own Authorization headers.

- Accept
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The Request body contains properties of the batch request, which controls how the REST server processes the Request, and properties of each operation embedded in the Request.

### Batch Request Properties

Property	Data type	Description
description	String	Description of the batch request.
transactional	Boolean	<p>Determines whether to process the batch request as a transaction.</p> <p><code>true</code>: All embedded operations are treated as one atomic unit, meaning that either all operations within a transaction are performed, or none of the operations are performed. A transactional batch request ensures that operations made by one process are not interrupted by another.</p> <p><code>false</code>: Embedded operations are not treated as one atomic unit. The batch operation may continue or terminate, depending on the <code>on-error</code> setting (if <code>sequential</code> is set to <code>true</code>), when one or more operations within the batch fail.</p> <p>Default value: <code>true</code></p>
sequential	Boolean	<p>Determines whether to process the batch request sequentially for non-transactional batch requests.</p> <p><code>true</code>: Operations in a batch are executed in the submission order.</p> <p><code>false</code>: The REST server determines whether to execute the operations in sequence or in parallel.</p> <p>Default value: <code>true</code></p>
on-error	String	<p>Determines whether to continue or terminate a batch operation when one or more of the embedded operations fail.</p> <p>This property works when <code>transactional</code> is set to <code>false</code> and <code>sequential</code> is set to <code>true</code>.</p> <p>Valid values are:</p> <ul style="list-style-type: none"><li>• <code>continue</code>: The REST server continues processing the batch operation when one or more of the embedded operations fail.</li></ul>

Property	Data type	Description
		<ul style="list-style-type: none"> <li><code>fail</code>: The REST server terminates the batch operation when one or more of the embedded operations fail.</li> </ul> <p>Default value: <code>fail</code></p>
<code>return-request</code>	Boolean	<p>Determines whether to return the Request of each operation in the response.</p> <p>Default value: <code>false</code></p>

### Operation Properties

Each batch request must have one operation at least. The maximum number of operations allowed to be embedded in one request is limited by the `rest.batch.operations.max.count` parameter.

Property	Data type	Description
<code>id *</code>	String	<p>Operation identifier.</p> <p>Each embedded operation must have its unique identifier in one batch request. REST services does not have constraints on how this value is produced. For example, you can use ordering numbers, UUIDs, or even random numbers.</p>
<code>description</code>	String	Description of the operation.
<code>request *</code>	String	<p>HTTP method and target URI of the operation.</p> <p>The URI can be either an absolute URI you get from a certain response or a relative URI, which starts with a slash (/).</p> <p>Example (absolute URI):</p> <pre>&lt;request method="POST" uri="http://localhost:8080/repositories/REPO/folders/0c00208080000107/objects"&gt;</pre> <p>Example (relative URI):</p> <pre>&lt;request method="POST" uri="/repositories/REPO/folders/0c00208080000107/objects"&gt;</pre> <p>The repository in the URI must be the same repository where the Batches resource is located.</p>

Property	Data type	Description
Content-Type header	String	MIME type of the Request body. Example: <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
Accept header	String	Acceptable media type for the response. Example: <header name="Accept" value="application/vnd.emc.documentum+xml"/>
Entity	CDATA /Escaped String	Request body of the operation.  For XML representation, the Request body is presented in a CDATA block or escaped.  <pre> &lt;entity&gt; &lt;![CDATA[ &lt;document type="dm_document"&gt; &lt;properties&gt; &lt;object_name&gt;readme&lt;/object_name&gt; &lt;title&gt;Readme for indexing&lt;/title&gt; &lt;/properties&gt; &lt;/document&gt; ]]&gt; &lt;/entity&gt; </pre> For JSON representation, the Request body is presented as a Sting.  <pre> { "entity" : "{ \"properties\": { \"object_name\": \"my test object\" } }" } </pre>
Required *		

## Request Samples

### Example 2-21. XML Representation

```

<?xml version="1.0" encoding="UTF-8"?>
<batch xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <description>batch for indexing</description>
  <transactional>false</transactional>
  <sequential>false</sequential>
  <on-error>continue</on-error>
  <return-request>true</return-request>
  <operations>
    <operation id="operation_1">
      <description>create a document operation</description>
      <request method="PUT" uri="<base URI>/objects">
        <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
        <header name="Accept" value="application/vnd.emc.documentum+xml"/>
        <entity>
          <![CDATA[
            <document type="dm_document">
              <properties>
                <object_name>readme</object_name>
                <title>Readme for indexing</title>

```



```

        </properties>
    </document>
  ]]>
</entity>
</request>
</operation>
...
<operations>
</batch>

```

### Example 2-22. JSON Representation

```

{
  "transactional" : true,
  "sequential" : false,
  "on-error" : "CONTINUE",
  "return-request" : true,
  "operations" : [ {
    "id" : "id-001",
    "description" : "create a document",
    "request" : {
      "method" : "POST",
      "uri" : "/<base URI>/objects",
      "headers" : [ {
        "name" : "Content-Type",
        "value" : "application/vnd.emc.documentum+json"
      } ],
      "entity" : "{\"properties\":{\"object_name\":\"my test object\"}}"
    }
  } ]
}

```

## Multipart Request

Operations embedded in a batch request may have binary contents attached, such as content creation, and content checkin. Documentum Platform REST Services leverages the XML-binary Optimized Packaging (XPO) protocol to support multipart operations in batches. As defined in XPO, a multipart batch request must have the `Multipart/Related` Content-type header.

Typically, a multipart batch request contains several portions as shown in the following pattern:

```

-- boundary
Content-Type: ...
Content-ID: ...
Content-disposition: ...
<batch>
  <operations>
    <operation id="id-1">
      <request>
        ...
        <entity>
          ...
        </entity>
        <attachment>
          <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
            href='cid:id-1-content' />
        </attachment>
      </request>
    </operation>

```

```

    <operation id="id-2">
      <request>
        ...
        <entity>
          ...
        </entity>
        <attachment>
          <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
            href='cid:id-2-content' />
        </attachment>
      </request>
    </operation>
  ...
</operations>
</batch>
-- boundary
Content-Type: ...
Content-ID: id-1-content
Content-disposition: ...
Attachment 1
-- boundary
Content-Type: ...
Content-ID: id-2-content
Content-disposition: ...
Attachment 2
-- boundary
...
-- boundary --

```

The first portion `batch` is identical to its counterpart in standard batch requests with the only addition of an extra `attachment` element in the `request` block of each embedded operation. The `attachment` element contains a content ID (`cid`) specifying the binary content to be attached which is defined in the subsequent portions of the batch request. The sequence of the portions holding binary contents must be identical to the sequence they are referenced in the `batch` portion.

### Multipart/Related Content-Type Header

The Multipart/Related Content-Type header contains the following parameters:

Parameters	Descriptions	Required or Not
boundary	Specifies the boundary used to separate each portion	Required
type	Specifies the media type of the batch request. Valid values are: <ul style="list-style-type: none"> <li>• <code>application/xop+xml</code> (for XML)</li> <li>• <code>application/jop+json</code> (for JSON)</li> </ul>	Required
start	Specified the portion to be processed first by Content ID. If not specified, the first portion is processed first. In a multipart batch request, the start portion must be a <code>batch</code> block.	Not required
start-info	The media type of the start portion. Valid values are: <ul style="list-style-type: none"> <li>• <code>application/vnd.emc.documentum+xml</code> (for XML)</li> <li>• <code>application/vnd.emc.documentum+json</code> (for JSON)</li> </ul>	Required

## Portion Header Set

A portion header set contains the following headers:

Headers	Descriptions	Required or Not
Content-Type	<p>For the start portion <code>batch</code>, this parameter specifies the media type of the batch request together with the media type of the start portion in the <code>type</code> property, for example:</p> <pre>Content-Type: application/xop+xml; type="application/vnd.emc.documentum+xml"</pre> <p>For portions holding binary contents, this parameter specifies the media type, for example:</p> <pre>Content-Type: text/plain</pre>	Required
Content-ID	Specifies the Content ID of the portion.	Required
Content-disposition	<p>This parameter must be set with the following pattern:</p> <pre>Content-disposition: form-data; name=Content ID</pre> <p>Example: Content-disposition: form-data; name=id-01-content</p>	Required

### Headers in Operation Blocks vs. Headers in Binary Content Blocks

Headers in an `operation` block describe the `entity` block next to it. Typically, the header set includes a `Content-Type` header indicating the media type of the content in the `entity` block and an `Accept` header indicating the expected media type of the response. A header set in a binary content block also contains a `Content-Type` header. This `Content-Type` header indicates media type of the binary content.

The following excerpt illustrates a multipart request that contains both an `entity` block and binary content:

```
...
<operation id="id-100">
  <request method="POST"
    uri="/repositories/REPO/folders/0c00208080000107/objects">
    <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
    <header name="Accept" value="application/vnd.emc.documentum+xml"/>
    <entity>
      <object><properties><object_name>my-test-doc</object_name>
      </properties></object>
    </entity>
    <attachment><xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
      href='cid:id-100-content'/></attachment>
    </request>
  </operation>
...

--frontier
Content-Type: text/plain
Content-ID: id-100-content
Content-disposition: form-data; name=id-100-content
```

This is the content of id-100

In a scenario where a multipart request does not include an entity block (for example, you upload binary content without updating any resource property), headers in operation blocks are used to describe the binary content. In the binary content block, all headers except Content-ID and Content-disposition are ignored. As a result, you must specify the media type of the binary content in the Content-Type header of the operation block.

In the following example, the media type of this individual request is text/plain and the REST server returns the response in application/vnd.emc.documentum+xml as specified in the operation block.

```
<operation id="id-100">
  <request method="POST"
    uri="/repositories/REPO/folders/0c00208080000107/objects">
    <header name="Content-Type" value="text/plain"/>
    <header name="Accept" value="application/vnd.emc.documentum+xml"/>
    <attachment><xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
      href='cid:id-100-content'/></attachment>
    </request>
  </operation>

--frontier
Content-Type: text/html
Accept: text/application/vnd.emc.documentum+json
Content-ID: id-100-content
Content-disposition: form-data; name=id-100-content
```

This is the content of id-100

In a Multipart/Related batch request, you are also allowed to embed requests that do not contain attachments. In this scenario, the headers in the operation block describe the entity block next to them and no binary content block pertains to the Request.

```
<operation id="id-100">
  <request method="POST"
    uri="/repositories/REPO/folders/0c00208080000107/objects">
    <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
    <header name="Accept" value="application/vnd.emc.documentum+xml"/>
    <entity>
      <object><properties><object_name>my-test-doc</object_name>
      </properties></object>
    </entity>
  </request>
</operation>
```

## Multipart Request Samples

### Example 2-23. Request in application/xop+xml

Request header:  
{Multipart/Related;boundary=frontier;  
type="application/xop+xml";  
start="batch";  
start-info="application/vnd.emc.documentum+xml"}

Payload:  
--frontier

```

Content-Type: application/xop+xml; type="application/vnd.emc.documentum+xml"
Content-ID: batch
Content-disposition: form-data; name=batch

<?xml version="1.0" encoding="UTF-8"?>
<batch>
  <transactional>true</transactional>
  <sequential>true</sequential>
  <operations>
    <operation id="id-100">
      <request method="POST"
        uri="/repositories/REPO/folders/0c00208080000107/objects">
        <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
        <header name="Accept" value="application/vnd.emc.documentum+xml"/>
        <entity>
          <object><properties><object_name>my-doc1</object_name>
          </properties></object>
        </entity>
        <attachment><xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:id-100-content'/></attachment>
        </request>
      </operation>
    <operation id="id-101">
      <request method="POST" uri="/repositories/REPO/folders/0c00208080000107/objects">
        <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
        <header name="Accept" value="application/vnd.emc.documentum+xml"/>
        <entity>
          <object><properties><object_name>my-doc2</object_name>
          </properties></object>
        </entity>
        <attachment><xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
          href='cid:id-101-content'/></attachment>
        </request>
      </operation>
    </operations>
  </batch>

--frontier
Content-Type: text/plain
Content-ID: id-100-content
Content-disposition: form-data; name=id-100-content
This is the content of id-100

--frontier
Content-Type: text/plain
Content-ID: id-101-content
Content-disposition: form-data; name=id-101-content
This is the content of id-101
--frontier--

```

#### Example 2-24. Request in application/jop+json

```

Request header:
{Multipart/Related;boundary=frontier;
type="application/jop+json";
start="batch";
start-info="application/vnd.emc.documentum+json"}

Payload:
--frontier
Content-Type: application/jop+json; type="application/vnd.emc.documentum+json"
Content-ID: batch
Content-disposition: form-data; name=batch

```

```
{
  "return-request" : true,
  "operations" :
  [
    {
      "id" : "id-100",
      "description" : "create object with content",
      "request" :
      {
        "method" : "POST",
        "uri" : "/repositories/REPO/folders/0c00208080000107/objects",
        "headers" :
        [
          {
            "name" : "Content-Type",
            "value" : "application/vnd.emc.documentum+json"
          },
          {
            "name" : "Accept",
            "value" : "application/vnd.emc.documentum+json"
          }
        ],
        "entity" : "{\"properties\":{\"object_name\":\"my test object\"}}",
        "attachment" : {
          "Include" : {
            "href" : "cid:id-100-content"
          }
        }
      }
    },
    {
      "id" : "id-101",
      "description" : "create object with content",
      "request" :
      {
        "method" : "POST",
        "uri" : "/repositories/REPO/folders/0c00208080000107/objects",
        "headers" :
        [
          {
            "name" : "Content-Type",
            "value" : "application/vnd.emc.documentum+json"
          },
          {
            "name" : "Accept",
            "value" : "application/vnd.emc.documentum+json"
          }
        ],
        "entity" : "{\"properties\":{\"object_name\":\"my test object\"}}",
        "attachment" : {
          "Include" : {
            "href" : "cid:id-101-content"
          }
        }
      }
    }
  ]
}

--frontier
Content-Type: text/plain
Content-ID: id-100-content
Content-disposition: form-data; name=id-100-content
i'm the content of id-100

--frontier
```

```
Content-Type: text/plain
Content-ID: id-101-content
Content-disposition: form-data; name=id-101-content
i'm the content of id-101
--frontier--
```

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 201 - Batch executed successfully
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed
- 415 - Unsupported Media Type
- 500 - Other unexpected server error

## Exceptions

Error Code	Description
E_ILLEGAL_TASK_OPERATIONS	The number of operations embedded in the batch request exceeds the threshold specified in <code>rest.batch.operations.max.count</code> .
E_BATCH_NO_OPERATION	No operation in the Request.
E_BATCH_NO_OPERATION_ID	One or more operations in the batch request do not contain an identifier.
E_BATCH_OPERATION_ID_DUPLICATED	The batch request contains duplicated operation identifiers.
E_BATCH_NO_OPERATION_REQUEST	One or more operations in the batch request do not have a request defined.
E_BATCH_NO_REQUEST_URI	One or more operations in the batch request do not have a URI defined.
E_BATCH_NO_REQUEST_METHOD	One or more operations in the batch request do not have a method defined.

Error Code	Description
E_METHOD_NOT_ALLOWED	One or more operations in the batch request do not have an invalid method defined.
E_BATCH_REQUEST_INVALID_URI	The Request URI is invalid.
E_BATCH_REQUEST_URI_ROOT	The relative URI does not start with “/”.
E_BATCH_PROHIBITION	There are operations on resources with the BatchProhibition annotation defined.
E_BATCH_TRANSACTION_PROHIBITION	There are resources with the TransactionProhibition annotation defined in a transactional batch request.

## Response Body

In this release, batch requests are executed in synchronous mode. The Response is not returned until the REST server completes the Request. The Response of a batch request includes:

- Batch properties. In addition to those described in [Batch Request Properties](#), the response also includes:
  - state, which indicates the state of the batch operation. In synchronous mode, the value of this property is always `Finished`.
  - sub-state, which indicates the sub-state of a finished batch operation. Valid values are:
    - `failed`: The REST server completes the batch operation. However, all the operations are failed. Generally, this value is returned for transactional batch requests.
    - `finished_with_error`: The REST server completes the batch operation, while the operation is partially failed. Generally, this value is returned for non-transactional batch requests.
  - owner, which indicates the user submitting the batch request.
  - submitted, which indicates the time point when the batch request is submitted.
  - started, which indicates the time point when the server starts processing the batch request.
  - finished, which indicates the time point when the batch operation completes.
- Each embedded request if `return-request` is `true`.
- The Response of each embedded operation.

### Example 2-25. Response Sample in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<batch xmlns="http://identifiers.emc.com/vocab/documentum"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <transactional>true</transactional>
  <sequential>false</sequential>
  <on-error>CONTINUE</on-error>
  <return-request>true</return-request>
  <state>FINISHED</state>
  <submitted>2014-07-03T16:37:32.147+08:00</submitted>
  <started>2014-07-03T16:37:32.147+08:00</started>
  <finished>2014-07-03T16:37:32.246+08:00</finished>
```



```

<owner>dmdadmin</owner>
<operations>
  <operation id="id-100">
    <description>get users</description>
    <state>FINISHED</state>
    <started>2014-07-03T16:37:32.147+08:00</started>
    <finished>2014-07-03T16:37:32.158+08:00</finished>
    <request method="GET" uri="/repositories/REPO/users?items-per-page=1">
      <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
      <header name="Accept" value="application/atom+xml"/>
    </request>
    <response status="200">
      <header name="Content-Type" value="application/atom+xml; charset=UTF-8"/>
      <entity><?xml version="1.0" encoding="UTF-8"?><feed xmlns="http://www.w3.org/2005/Atom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dm="http://identifiers.emc.com/vocab/documentum"><id>http://localhost:8080/dctm-rest/repositories/REPO/users</id><title>Users</title><author><name>EMC Documentum</name><uri>http://localhost:8080/dctm-rest/repositories/REPO/users</uri></author><dm:page>1</dm:page><dm:items-per-page>1</dm:items-per-page><link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1"/><link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1&page=2"/><link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1&page=1"/><entry><id>http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator</id><title>Administrator</title><updated>2014-01-21T12:24:16.000+08:00</updated><summary><content type="application/vnd.emc.documentum+xml" src="http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator"/><link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator"/></entry></feed>
    </entity>
  </response>
</operation>
  <operation id="id-1001">
    <description>get cabinets</description>
    <state>FINISHED</state>
    <started>2014-07-03T16:37:32.158+08:00</started>
    <finished>2014-07-03T16:37:32.244+08:00</finished>
    <request method="GET" uri="/repositories/REPO/cabinets?items-per-page=1">
      <header name="Content-Type" value="application/vnd.emc.documentum+xml"/>
      <header name="Accept" value="application/atom+xml"/>
    </request>
    <response status="200">
      <header name="Content-Type" value="application/atom+xml; charset=UTF-8"/>
      <entity><?xml version="1.0" encoding="UTF-8"?><feed xmlns="http://www.w3.org/2005/Atom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dm="http://identifiers.emc.com/vocab/documentum"><id>http://localhost:8080/dctm-rest/repositories/REPO/cabinets</id><title>Cabinets</title><updated>2014-07-03T16:37:32.238+08:00</updated><author><name>EMC Documentum</name><uri>http://localhost:8080/dctm-rest/repositories/REPO/cabinets</uri></author><dm:page>1</dm:page><dm:items-per-page>1</dm:items-per-page><link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/cabinets?items-per-page=1"/><link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/cabinets?items-per-page=1&page=2"/><link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/cabinets?items-per-page=1&page=1"/><entry><id>http://localhost:8080/dctm-rest/repositories/REPO/objects/0c0020808000dfc6</id><title>+ cabinet +</title><updated>2014-06-30T16:13:52.000+08:00</updated><summary>dm_cabinet 0c0020808000dfc6</summary><author><name>dmdadmin</name><uri>http://localhost:8080/dctm-rest/repositories/REPO/users/dmdadmin</uri></author><content type="application/vnd.emc.documentum+xml" src="http://localhost:8080/dctm-rest/repositories/REPO/objects/0c0020808000dfc6"/><link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0c0020808000dfc6"/></entry></feed>
    </entity>
  </response>
</operation>

```

```

        </response>
    </operation>
</operations>
</batch>

```

### Example 2-26. Response Sample in JSON

```

{
  "transactional" : true,
  "sequential" : false,
  "on-error" : "CONTINUE",
  "return-request" : true,
  "state" : "FINISHED",
  "submitted" : "2014-07-03T16:34:45.989+08:00",
  "started" : "2014-07-03T16:34:45.989+08:00",
  "finished" : "2014-07-03T16:34:46.107+08:00",
  "owner" : "dmadmin",
  "operations" : [ {
    "id" : "id-100",
    "description" : "get users",
    "state" : "FINISHED",
    "started" : "2014-07-03T16:34:45.989+08:00",
    "finished" : "2014-07-03T16:34:46.020+08:00",
    "request" : {
      "name" : "request",
      "method" : "GET",
      "uri" : "/repositories/REPO/users?items-per-page=1",
      "headers" : [ {
        "name" : "Content-Type",
        "value" : "application/vnd.emc.documentum+json"
      }, {
        "name" : "Accept",
        "value" : "application/vnd.emc.documentum+json"
      } ]
    }
  }, {
    "response" : {
      "name" : "response",
      "status" : 200,
      "headers" : [ {
        "name" : "Content-Type",
        "value" : "application/vnd.emc.documentum+json;charset=UTF-8"
      } ],
      "entity" : "{\n\"id\": \"http://localhost:8080/dctm-rest/repositories/REPO/users\n\", \"title\": \"Users\n\", \"author\": [\n{\n\"name\": \"EMC Documentum\n\"}\n], \"page\": 1,\n\"items-per-page\": 1,\n\"links\": [\n{\n\"rel\": \"self\n\", \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1\n\"}, {\n\"rel\": \"next\n\", \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1&page=2\n\"}, {\n\"rel\": \"first\n\", \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/users?items-per-page=1&page=1\n\"}\n], \"entries\": [\n{\n\"id\": \"http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator\n\", \"title\": \"Administrator\n\", \"updated\": \"2014-01-21T12:24:16.000+08:00\n\", \"summary\": \"\n\", \"content\": {\n\"content-type\": \"application/vnd.emc.documentum+json\n\", \"src\": \"http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator\n\"}, \"links\": [\n{\n\"rel\": \"edit\n\", \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator\n\"}\n]\n}\n]\n}\""
    }
  }, {
    "id" : "id-101",
    "description" : "get cabinets",
    "state" : "FINISHED",
    "started" : "2014-07-03T16:34:46.020+08:00",
    "finished" : "2014-07-03T16:34:46.105+08:00",

```

```

"request" : {
  "name" : "request",
  "method" : "GET",
  "uri" : "/repositories/REPO/cabinets?items-per-page=1",
  "headers" : [ {
    "name" : "Content-Type",
    "value" : "application/vnd.emc.documentum+json"
  }, {
    "name" : "Accept",
    "value" : "application/vnd.emc.documentum+json"
  } ]
},
"response" : {
  "name" : "response",
  "status" : 200,
  "headers" : [ {
    "name" : "Content-Type",
    "value" : "application/vnd.emc.documentum+json;charset=UTF-8"
  } ],
  "entity" : "{\n\"id\": \"http://localhost:8080/dctm-rest/repositories/REPO/cabinets\",
\n\"title\": \"Cabinets\", \n\"updated\": \"2014-07-03T16:34:46.103+08:00\", \n\"author\": [\n
{\n\"name\": \"EMC Documentum\"}], \n\"page\": 1, \n\"items-per-page\": 1, \n\"links\": [\n\"rel\"
: \"self\", \n\"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/cabinets?it
ems-per-page=1\", {\n\"rel\": \"next\", \n\"href\": \"http://localhost:8080/dctm-rest/rep
ositories/REPO/cabinets?items-per-page=1&page=2\"}, {\n\"rel\": \"first\", \n\"href\": \"h
ttp://localhost:8080/dctm-rest/repositories/REPO/cabinets?items-per-page=1&page=1\"
}], \n\"entries\": [\n\"id\": \"http://localhost:8080/dctm-rest/repositories/REPO/objects
/0c0020808000dfc6\", \n\"title\": \"\" + cabinet + \"\", \n\"updated\": \"2014-06-30T16
:13:52.000+08:00\", \n\"summary\": \"dm_cabinet 0c0020808000dfc6\", \n\"author\": [{\n\"name\"
: \"dmadmin\", \n\"uri\": \"http://localhost:8080/dctm-rest/repositories/REPO/users/dma
dmin\"}], \n\"content\": {\n\"content-type\": \"application/vnd.emc.documentum+json\", \n\"sr
c\": \"http://localhost:8080/dctm-rest/repositories/REPO/cabinets/0c0020808000dfc6\"}
, \n\"links\": [{\n\"rel\": \"edit\", \n\"href\": \"http://localhost:8080/dctm-rest/repositori
es/REPO/cabinets/0c0020808000dfc6\"}]]}"
}
} ]
}

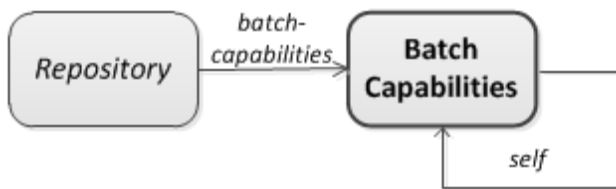
```

## Batch Capabilities

The Batch Capabilities resource indicates what batch features are implemented in Documentum Platform REST Services. This includes the supported batch properties, such as transactional/non-transactional, the list of batchable resources, and the list of non-batchable resources.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations in the Batch Capabilities resource.

Link Relation	Description	Resource Reference
self	The Batch Capabilities resource	<a href="#">Batch Capabilities, page 116</a>

## Operations

The Batch Capabilities resource supports the following HTTP method.

Method	Description
GET	Retrieve supported batch properties, the list of batchable resources, and the list of non-batchable resources.

### Get Batch Capabilities

Retrieves supported batch properties, the list of batchable resources, and the list of non-batchable resources.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
401 - Authentication failed  
415 - Unsupported Media Type  
500 - Other unexpected server error

## Response Body

XML or JSON representation that describes the supported batch features and provides the lists of batchable and non-batchable resources.

### Example 2-27. Response Sample in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<batch-capabilities xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <transactions>both</transactions>
  <sequence>both</sequence>
  <on-error>both</on-error>
  <batchable-resources>
    <batchable-resource>batch-capabilities</batchable-resource>
    <batchable-resource>cabinet</batchable-resource>
    <batchable-resource>cabinets</batchable-resource>
    <batchable-resource>checked-out-objects</batchable-resource>
    <batchable-resource>content</batchable-resource>
    <batchable-resource>contents</batchable-resource>
    <batchable-resource>current-user</batchable-resource>
    <batchable-resource>current-version</batchable-resource>
    <batchable-resource>default-folder</batchable-resource>
    <batchable-resource>document</batchable-resource>
    <batchable-resource>dql-query</batchable-resource>
    <batchable-resource>folder</batchable-resource>
    <batchable-resource>folder-child-links</batchable-resource>
    <batchable-resource>folder-child-objects</batchable-resource>
    <batchable-resource>folder-link</batchable-resource>
    <batchable-resource>folder-parent-links</batchable-resource>
    <batchable-resource>format</batchable-resource>
    <batchable-resource>formats</batchable-resource>
    <batchable-resource>group</batchable-resource>
    <batchable-resource>group-members</batchable-resource>
    <batchable-resource>groups</batchable-resource>
    <batchable-resource>home-document</batchable-resource>
    <batchable-resource>lock</batchable-resource>
    <batchable-resource>network-location</batchable-resource>
    <batchable-resource>network-locations</batchable-resource>
    <batchable-resource>object</batchable-resource>
    <batchable-resource>product-info</batchable-resource>
    <batchable-resource>relation</batchable-resource>
    <batchable-resource>relation-type</batchable-resource>
    <batchable-resource>relation-types</batchable-resource>
    <batchable-resource>relations</batchable-resource>
    <batchable-resource>repositories</batchable-resource>
    <batchable-resource>repository</batchable-resource>
    <batchable-resource>search</batchable-resource>
    <batchable-resource>type</batchable-resource>
    <batchable-resource>types</batchable-resource>
    <batchable-resource>user</batchable-resource>
    <batchable-resource>users</batchable-resource>
    <batchable-resource>versions</batchable-resource>
  </batchable-resources>
  <non-batchable-resources>
    <non-batchable-resource>batches</non-batchable-resource>
    <non-batchable-resource>content-media</non-batchable-resource>
  </non-batchable-resources>
  <links>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        batch-capabilities.xml" rel="self"/>
  </links>
```

```
</batch-capabilities>
```

### Example 2-28. Response Sample in JSON

```
{  "transactions" : "both",
   "sequence" : "both",
   "on-error" : "both",
   "batchable-resources" : [
     "batch-capabilities",
     "cabinet", "cabinets",
     "checked-out-objects",
     "content",
     "contents",
     "current-user",
     "current-version",
     "default-folder",
     "document",
     "dql-query",
     "folder",
     "folder-child-links",
     "folder-child-objects",
     "folder-link",
     "folder-parent-links",
     "format",
     "formats",
     "group",
     "group-members",
     "groups",
     "home-document",
     "lock",
     "network-location",
     "network-locations",
     "object",
     "product-info",
     "relation",
     "relation-type",
     "relation-types",
     "relations",
     "repositories",
     "repository",
     "search",
     "type",
     "types",
     "user",
     "users",
     "versions" ],
   "non-batchable-resources" : [
     "batches",
     "content-media" ],
   "links" : [ { "rel" : "self",
                 "href" : "http://localhost:8080/dctm-rest/repositories/
REPO/batch-capabilities" } ] }
```

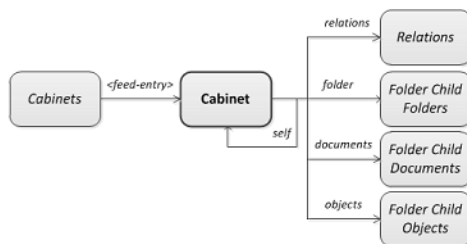
# Cabinet(s)

## Cabinet

The Cabinet resource represents a cabinet in the repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Cabinets, page 127](#) and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Cabinet resource.

Link Relation	Description	Resource Reference
self	Link to this cabinet.	<a href="#">Cabinet, page 120</a>
folders [1]	Collection of folders linked to this cabinet.	<a href="#">Folder Child Folders, page 233</a>
relations [1]	Collection of relations related to this cabinet.	<a href="#">Contents, page 164</a>
documents [1]	Collection of documents associated to this cabinet.	<a href="#">Folder Child Documents, page 216</a>
objects [1]	Collection of objects of this cabinet.	<a href="#">Folder Child Objects, page 248</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		



## Operations

The Cabinet resource supports the following HTTP methods.

Method	Description
GET	Retrieves properties, and other information of the Cabinet resource.
POST	Updates the properties for the Cabinet resource.
DELETE	Deletes the Cabinet resource from a repository.

### Get a Cabinet

Get properties, and other information of the Cabinet resource. Properties are returned as embedded elements in the response message body. Other information, such as relationships, is referenced from the link relations of the response message body.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
404 - Cabinet not found  
500 - Other unexpected server error

## Response Body

XML or JSON message that represents the cabinet.

## Update a Cabinet

Update the cabinet with given properties. If the client provides undefined properties in the Request body, the server throws an exception.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the Cabinet resource. Only properties that can be updated should be put in the message body.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Cabinet updated successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied; no permission to update the object or setting read-only properties is not allowed  
404 - Object not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the Cabinet resource. The successful response contains all properties of the updated cabinet.

## Delete a Cabinet

Deletes the Cabinet resource from the repository.

If the deletion fails, an exception is thrown and the Cabinet resource (including the cabinet tree and version history) is reverted to the original state.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Default value
del-non-empty	<p>Specifies whether this operation deletes a non-empty cabinet or not</p> <ul style="list-style-type: none"><li>• <code>true</code> - The cabinet and all its descendants are deleted.</li><li>• <code>false</code> - Only this cabinet is deleted. Deleting a non-empty cabinet results in an error.</li></ul> <p>(boolean)</p>	boolean	false
del-all-links	<p>Specifies whether a multi-linked descendant is deleted or unlinked from this specified cabinet.</p>	boolean	false

Variable	Description	Data type	Default value
	<ul style="list-style-type: none"> <li>• <code>true</code> - This operation deletes a multi-linked descendant along with all its folder links.</li> <li>• <code>false</code> - This operation only deletes a descendant's link to this cabinet. The descendant is still linked to other folders.</li> </ul> <p><b>Note:</b> Virtual documents are not supported.</p>		

### Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

N/A

### Response Media Types

N/A

### Response Status

- 204 - Success; no content is returned
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed

- 403 - Permission denied; no permission to delete the object
- 500 - Other unexpected server error

## Response Body

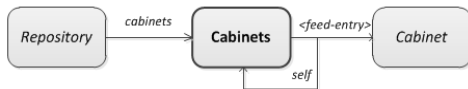
HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

## Cabinets

The Cabinets resource represents a collection of cabinets in a repository.

### Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

### Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the Cabinets resource without the file extension	List of cabinets	Server's current time	<a href="#">Cabinet, page 120</a>	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the cabinet	Cabinet name	Cabinet description	r_modify_date of the cabinet

### Link Relations

The following table lists general link relations in the Cabinets resource.

Link Relation	Description	Resource Reference
self	URI for cabinets collection feed	<a href="#">Cabinets, page 127</a>
first, last, next, previous	Pagination links	<a href="#">Cabinets, page 127</a>

### Operations

The Cabinets resource supports the following HTTP methods.

Method	Description
GET	Lists all available cabinets in the repository for the current user
POST	Create a new cabinet in the repository.

## Get Cabinets

List all available cabinets in the repository for the current user.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

The following parameter is specific to this operation.

Variable	Description	Data type	Default value
object-type	Specifies the exact sub type name to get from the collection of this type	string	null
thumbnail	<p>Specifies whether this operation deletes a non-empty folder or not.</p> <ul style="list-style-type: none"><li>• <code>true</code> - Return the thumbnail link for each entry in the collection resource.</li><li>• <code>false</code> - Do not return the thumbnail link for each entry in the collection resource.</li></ul>	boolean	false



This method supports the following common query parameters:

- inline
- page
- items-per-page
- include-total
- view
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

### Response Status

- 200 - Retrieved successfully
- 400 - Bad request; invalid property name or value

401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the cabinets collection.

- The body contains a list of cabinets (or subtypes of `dm_cabinet`).
- Each object may contain all properties of the cabinet, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order. The returned cabinets can be ordered by any non-repeating property.
- The total count is returned only when you explicitly set the `include-total` parameter to `true` in the Request. In this case, the link for last page is returned.
- Each cabinet object must contain links a specific to `dm_cabinet` type. See [Link Relations, page 472](#) in SysObject resource to view the links that are available to the `dm_cabinet` object type.

## Create a Cabinet

Create a new cabinet or a subtype of cabinet in the repository.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the cabinet to create.

The `r_object_type` property for the cabinets must be `dm_cabinet` or its subtype, otherwise, the Request is considered as a bad request and rejected. If this property is not specified in a client request, it is set to `dm_cabinet` by default.

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 201 - Cabinet created successfully
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 404 - No object found
- 409 - Conflict. The cabinet name already exists.
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the cabinet created.

# Checked Out Objects

The Checked Out Objects resource represents a collection of checked out objects in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository](#), page 369 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
URI of the checked out objects	Checked out objects	Server's current time	No

Entry	Entry ID	Entry Title	Entry Updated
Each entry represents a checked out object.	URI of the version	object_name of the object	r_modify_date for the object

## Link Relations

The following table lists link relations for the Checked Out Object resource.

Link Relation	Description	Resource Reference
self	This All Versions resource	<a href="#">Checked Out Objects</a> , page 133
first, last, next, previous	Pagination links	<a href="#">Checked Out Objects</a> , page 133

## Operations

The Checked Out Objects resource supports the following HTTP method.

Method	Description
GET	Retrieves a collection of checked out objects.

## Get Checked Out Objects

Retrieve a collection of checked out objects according to the query parameters specified in the Request.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- inline
- page
- items-per-page
- include-total
- links
- filter
- view

The *q* parameter is supported for full text searching with a subset of the Simple Search Language, however the parenthesis is not supported.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following query parameters are specific to this operation:

Variable	Description	Data type	Default value
checked-out-by	<p>Specifies the lock owner of the checked out objects with the following valid values:</p> <ul style="list-style-type: none"><li>• current-user</li><li>• specific-user</li><li>• any</li></ul> <p>Only checked out objects with the specified lock owner</p>	enum	current-user

Variable	Description	Data type	Default value
	are returned in the result.		
user-name	Specifies the user name of the lock owner.  Only works when the <i>checked-out-by</i> parameter is set to <i>specific-user</i>	string	null
thumbnail	Specifies whether this operation deletes a non-empty folder or not.  <ul style="list-style-type: none"> <li>• <code>true</code> - Return the thumbnail link for each entry in the collection resource.</li> <li>• <code>false</code> - Do not return the thumbnail link for each entry in the collection resource.</li> </ul>	boolean	false
include-all-versions	Specifies whether or not to list all versions of the child object's link.  <ul style="list-style-type: none"> <li>• <code>true</code> - Return all versions.</li> <li>• <code>false</code> - Only return the current version.</li> </ul>	boolean	false
q	Specifies the search criterion with a full-text expression in simple search language.  Parameter <code>q</code> must be encoded because it may contain non-English locale characters.  <b>Note:</b> International characters that are used in this query	String	No search criteria is used for the search.

Variable	Description	Data type	Default value
	parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.		

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error



## Response Body

XML or JSON representation of the checked out objects.

- Each object may contain all or a set of properties of the objects, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by object name.

# Comment(s)

## Comment

This resource allows you to work with comment objects.

### Link Relations

The following table lists link relations in the Comment resource.

Link Relation	Description	Resource Reference
self	This resource	<a href="#">Comment, page 138</a>
parent	The parent Comment	<a href="#">Comment, page 138</a>
<a href="http://identifiers.emc.com/linkrel/replies">http://identifiers.emc.com/linkrel/replies</a>	The replies to the Comment	<a href="#">Comment Replies, page 149</a>
<a href="http://identifiers.emc.com/linkrel/delete">http://identifiers.emc.com/linkrel/delete</a>	Delete the Comment	

### Operations

The Comment resource supports the following HTTP methods.

Method	Description
GET	Retrieves the Comment
DELETE	Deletes the Comment

### Get a Comment

Retrieves a comment.

### Supported HTTP Method

GET

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

### Request Query Parameters

- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

N/A

### Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

### Response Status

- 200 - Comment retrieved successfully
- 200, 201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-29. XML Response

```
<comment xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <object-id>0800000180143727</object-id>
  <comment-id>1324839</comment-id>
  <owner-name>John Doe</owner-name>
  <creation-date>2016-01-26T15:16:16.000+00:00</creation-date>
  <modified-date>2016-01-26T15:16:16.000+00:00</modified-date>
  <content-value>The spec looks good.&nbsp;
    Just two comments&lt;br&gt;&lt;ol&gt;&lt;li&gt;Please add a diagram to show the
    flow.&lt;/li&gt;&lt;li&gt;Please clarify the filter criteria.&lt;br&gt;&lt;/li&gt;
    &lt;/ol&gt;
  </content-value>
  <parent-id>0</parent-id>
  <title/>
  <can-delete>true</can-delete>
  <can-reply>true</can-reply>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
      objects/0900000180143717/comments/1324839"/>
    <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/
      objects/0900000180143717/comments"/>
    <link rel="http://identifiers.emc.com/linkrel/replies"
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324839/replies"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324839"/>
  </links>
</comment>
```

### Example 2-30. JSON Response

```
{
  "object-id": "0800000180143727",
  "comment-id": "1324839",
  "owner-name": "John Doe",
  "creation-date": "2016-01-26T15:16:16.000+00:00",
  "modified-date": "2016-01-26T15:16:16.000+00:00",
  "content-value": "The spec looks good.&nbsp;Just two comments&lt;br&gt;&lt;ol&gt;&lt;li&gt;
    Please add a diagram to show the flow.&lt;/font&gt;&lt;/font&gt;&lt;/font&gt;
    &lt;/li&gt;&lt;li&gt;Please clarify the filter criteria.&lt;br&gt;&lt;/li&gt;
    &lt;/ol&gt;",
  "parent-id": "0",
  "title": "",
  "can-delete": true,
  "can-reply": true,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324839"
    },
    {
      "rel": "parent",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/replies",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/"
    }
  ]
}
```

```
0900000180143717/comments/1324839/replies"
},
{
  "rel": "http://identifiers.emc.com/linkrel/delete",
  "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324839"
}
]
}
```

## Delete a Comment

This method allows the user who created the comment to delete the comment. The `delete` link for a comment is available only when a user can delete that comment.

### Supported HTTP Method

DELETE

### Request Media Types

N/A

### Request Query Parameters

N/A

### Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

N/A

## Response Media Types

N/A

## Response Status

204 No Content

## Response Body

N/A

# Comments

This resource allows you to work with a comment objects collection.

## Feed

Is feed? Yes.

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	Comments on object {objectId}	Server's current time	<a href="#">Comments, page 142</a>	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated	Entry Published
Comment resource URI	Comment's title	Comment { <i>commentId</i> } on object { <i>objectId</i> }	Comment's modification date	Comment's creation date

## Link Relations

The following table lists link relations in the Comments resource.

Link Relation	Description	Resource Reference
self	This resource	<a href="#">Comments, page 142</a>

## Operations

The Comments resource supports the following HTTP methods.

Method	Description
GET	Retrieves a list of root-level Comments
POST	Create a root-level Comment

### Get Root-level Comments

Retrieves the list of root level comments for the object.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

- inline
- page
- items-per-page
- include-total
- filter
- links
- sort

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 - Comments retrieved successfully
- 201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-31. XML Response

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
    comments</id>
  <title>Comments on object: 0900000180143717 under parent comment id: 0
</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2016-02-01T21:50:18.768+00:00</updated>
  <dm:page xmlns:dm="http://identifiers.emc.com/vocab/documentum">1</dm:page>
  <dm:items-per-page xmlns:dm="http://identifiers.emc.com/vocab/documentum">100
</dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
    09000001800d86ec/comments"/>
  <entry>
    <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0900000180143717/comments/1324839</id>
    <title></title>
    <author>
      <name>John Doe</name>
```



```

    <uri>http://localhost:8080/dctm-rest/repositories/REPO/users/John%2BDoe
</uri>
</author>
<summary>&lt;font color= '#363636' &gt;&lt;font size= '2' &gt;
&lt;font face= 'Arial' &gt;The spec looks good.& Just two comments&lt;br&gt;
&lt;/font&gt;&lt;/font&gt;&lt;/font&gt;&lt;ol&gt;&lt;li&gt;
    &lt;font color= '#363636' &gt;&lt;font size= '2' &gt;
    &lt;font face= 'Arial' &gt;Please add a diagram to show the flow.&lt;/font&gt;
    &lt;/font&gt;&lt;/font&gt;&lt;/li&gt;&lt;li&gt;&lt;font color= '#363636' &gt;
    &lt;font size= '2' &gt;&lt;font face= 'Arial' &gt;Please clarify the filter
        criteria.&lt;br&gt;&lt;/font&gt;&lt;/font&gt;&lt;/font&gt;&lt;/li&gt;
    &lt;/ol&gt;</summary>
<updated>2016-01-26T15:16:16.000+00:00</updated>
<published>2016-01-26T15:16:16.000+00:00</published>
<link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
objects/0900000180143717/comments/1324839"/>
<content type="application/xml"
src="http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324839"/>
</entry>
<entry>
    <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324841</id>
    <title></title>
    <author>
        <name>Jane Doe</name>
        <uri>http://localhost:8080/dctm-rest/repositories/REPO/users/
Jane%2BDoe</uri>
    </author>
    &lt;summary&gt;&lt;font face= 'Arial' &gt;&lt;font size= '2' &gt;
    &lt;font color= '#363636' &gt;Should we also support paging?&lt;br&gt;
    &lt;/font&gt;&lt;/font&gt;&lt;/font&gt;&lt;/summary&gt;
    <updated>2016-01-26T15:19:57.000+00:00</updated>
    <published>2016-01-26T15:19:57.000+00:00</published>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
objects/0900000180143717/comments/1324841"/>
    <content type="application/xml" src="http://localhost:8080/dctm-rest/
repositories/REPO/objects/0900000180143717/comments/1324841"/>
    </entry>
</feed>

```

### Example 2-32. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments",
  "title": "Comments on object: 0900000180143717 under parent comment id: 0",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2016-02-01T22:07:00.401+00:00",
  "page": 1,
  "items-per-page": 100,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments"
    }
  ],
  "entries": [
    {

```

```

    "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments/1324839",
    "title": "",
    "author": [
      {
        "name": "John Doe",
        "uri": "http://localhost:8080/dctm-rest/repositories/REPO/users/John%2BDoe"
      }
    ],
    "summary": "<font color= '#363636' ><font size= '2' ><font face= 'Arial' >
      The spec looks good. Just two comments<br></font></font></font>
        <ol><li><font color= '#363636' ><font size= '2' >
          <font face= 'Arial' >Please add a diagram to show the flow.</font>
          </font></font></li><li><font color= '#363636' >
            <font size= '2' >
              <font face= 'Arial' >Please clarify the filter criteria.<br>
              </font></font></li></ol>",
    "updated": "2016-01-26T15:16:16.000+00:00",
    "published": "2016-01-26T15:16:16.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000180143717/comments/1324839"
      }
    ],
    "content": {
      "type": "application/vnd.emc.documentum+json",
      "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324839"
    }
  },
  {
    "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments/1324841",
    "title": "",
    "author": [
      {
        "name": "Jane Doe",
        "uri": "http://localhost:8080/dctm-rest/repositories/REPO/users/Jane%2BDoe"
      }
    ],
    "summary": "<font face= 'Arial' ><font size= '2' >
      <font color= '#363636' >Should we also support paging?<br></font>
      </font></font>",
    "updated": "2016-01-26T15:19:57.000+00:00",
    "published": "2016-01-26T15:19:57.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000180143717/comments/1324841"
      }
    ],
    "content": {
      "type": "application/vnd.emc.documentum+json",
      "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324841"
    }
  }
]
}

```

## Create a Root-Level Comment

You can use this method to add a root-level comment. You must have `RELATE` permission on the object to add a comment. The content value of a comment can be either plain text or rich text. All content is sanitized before it is saved to the repository.

**Note:** There is a limitation in the Collaboration Service that requires you to have `WRITE` permission on the object in order to start a comment thread. In other words, a user requires `WRITE` permission to create the first comment, but only requires `RELATE` permission to add subsequent comments.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Content-Type
- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The content-value tag contains the rich text content of the comment. It's required for comment creation.

## Response Headers

N/A

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 201 - Created
- 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-33. XML Response

```
<comment xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <content-value>
    The spec looks good. Just two comments:&lt;br&gt;
    &lt;ol&gt;
      &lt;li&gt;Please add a diagram to show the flow.&lt;/li&gt;
      &lt;li&gt;Please clarify the filter criteria.&lt;br&gt;&lt;/li&gt;
    &lt;/ol&gt;
  </content-value>
</comment>
```

### Example 2-34. JSON Response

```
{
  "content-value": "The spec looks good. Just two comments:&lt;br&gt;
    &lt;ol&gt;&lt;li&gt;Please add a diagram to show the flow.&lt;/li&gt;
    &lt;li&gt;Please clarify the filter criteria.&lt;br&gt;&lt;/li&gt;
    &lt;/ol&gt;"
}
```

# Comment Replies

This resource allows you to work with a list of comment replies.

## Feed

Is feed? Yes.

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	Comments on object {objectId} in reply to comment {parentCommentId}	Server's current time	A reply comment to the object	Yes

Entry ID	Entry Title	Entry Summary	Entry Published	Entry Updated
Comment resource URI	Comment's title	Comment {commentId} on object {objectId} in reply to comment {parentCommentId}	Comment's creation date	Comment's modification date

## Link Relations

The following table lists link relations for the Comments resource.

Link Relation	Description	Resource Reference
self	This resource	<a href="#">Comment Replies, page 149</a>

## Operations

The Comments resource supports the following HTTP methods.

Method	Description
GET	Retrieves a list of replies for a comment
POST	Create a reply for a comment

## Get Replies

Retrieves the list of replies to a comment.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

- inline
- page
- items-per-page
- include-total
- filter
- links
- sort

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 - Replies retrieved successfully
- 201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-35. XML Response

```
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
    comments/1324839/replies</id>
  <title>Comments on object: 0900000180143717 under parent comment id: 1324839</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2016-02-12T19:47:24.307+00:00</updated>
  <dm:page xmlns:dm="http://identifiers.emc.com/vocab/documentum">1</dm:page>
  <dm:items-per-page xmlns:dm="http://identifiers.emc.com/vocab/documentum">100
  </dm:items-per-page>
  <linkrel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
    0900000180143717/comments/1324839/replies"/>
  <entry>
    <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments/1324840</id>
    <title/>
    <author>
      <name>John Doe</name>
      <uri>http://localhost:8080/dctm-rest/repositories/REPO/users/John%2BDoe</uri>
    </author>
    <summary>
      &lt;li;font face= 'Arial' &gt;
      &lt;li;font size= '2' &gt;
      &lt;li;font color= '#363636' &gt;Thanks for reviewing it.& I'll update the spec.
      &lt;li;br&gt;
      &lt;li;/font&gt;&lt;li;/font&gt;&lt;li;/font&gt;
    </summary>
    <updated>2016-01-26T15:19:21.000+00:00</updated>
    <published>2016-01-26T15:19:21.000+00:00</published>
    <linkrel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0900000180143717/comments/1324840"/>
    <contenttype="application/xml" src="http://localhost:8080/dctm-rest/repositories/
      REPO/objects/0900000180143717/comments/1324840"/>
  </entry>
</entry>
```

```

<id>http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
  comments/1324842</id>
<title/>
<author>
  <name>Jane Doe</name>
<uri>http://localhost:8080/dctm-rest/repositories/REPO/users/Jane%2BDoe</uri>
</author>
<summary>Do we need to expose this feature through REST api?&lt;br></summary>
<updated>2016-01-26T15:27:50.000+00:00</updated>
<published>2016-01-26T15:27:50.000+00:00</published>
<linkrel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
  0900000180143717/comments/1324842"/>
<contenttype="application/xml" src="http://localhost:8080/dctm-rest/repositories/
  REPO/objects/0900000180143717/comments/1324842"/>
</entry>
</feed>

```

### Example 2-36. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324839/replies",
  "title": "Comments on object: 0900000180143717 under parent comment id: 1324839",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2016-02-12T19:29:04.092+00:00",
  "page": 1,
  "items-per-page": 100,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
comments/1324839/replies"
    }
  ],
  "entries": [
    {
      "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324840",
      "title": "",
      "author": [
        {
          "name": "John Doe",
          "uri": "http://localhost:8080/dctm-rest/repositories/REPO/users/John%2BDoe"
        }
      ],
      "summary": "&lt;font face= 'Arial' &gt;&lt;font size= '2' &gt;
&lt;font color= '#363636' &gt;
        Thanks for reviewing it. I'll update the spec.&lt;br>
&lt;/font&gt;&lt;/font&gt;&lt;/font&gt;",
      "updated": "2016-01-26T15:19:21.000+00:00",
      "published": "2016-01-26T15:19:21.000+00:00",
      "links": [
        {
          "rel": "edit",
          "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000180143717/comments/1324840"
        }
      ],
      "content": {
        "type": "application/vnd.emc.documentum+json",

```



```

        "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000180143717/comments/1324840"
      },
    ],
    {
      "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000180143717/comments/1324842",
      "title": "",
      "author": [
        {
          "name": "Jane Doe",
          "uri": "http://localhost:8080/dctm-rest/repositories/REPO/users/Jane%2BDoe"
        }
      ],
      "summary": "Do we need to expose this feature through REST api?<br>",
      "updated": "2016-01-26T15:27:50.000+00:00",
      "published": "2016-01-26T15:27:50.000+00:00",
      "links": [
        { "rel": "edit", "href": "http://localhost:8080/dctm-rest/repositories/
          REPO/objects/0900000180143717/comments/1324842"... }
      ],
      "content": {
        "type": "application/vnd.emc.documentum+json",
        "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000180143717/comments/1324842"
      }
    }
  ]
}

```

## Create a Reply to a Comment

You can use this method to add a reply to a comment. You must have `RELATE` permission on the comment object in order to add a reply. The content value of a comment reply can be either plain text or rich text. All content is sanitized before it is saved to the repository.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

**comment-id:** The id of the comment

**parent-id:** The id of the parent comment

**content-value:** Contains the text content of the comment

**can-delete:** Indicates whether the user can delete this comment

**can-reply:** Indicates whether the user can reply to this comment

## Request Headers

- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The content-value tag contains the encoded text content of the reply. It's required for comment creation.

### Example 2-37. XML Request

```
<comment xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <content-value>
    Agree with John.<br><br><ol><li>Please add a diagram to show the
    flow.</li><li>Please clarify the filter criteria.<br><br></li>
    </ol>
  </content-value>
</comment>
```

### Example 2-38. JSON Request

```
{
  "content-value": "Agree with John.<br><br><ol><li>
    Please add a diagram to
    show the flow.</li><li>Please clarify the filter
    criteria.<br><br></li></ol>"
}
```

## Response Headers

- Location

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 201 - Created
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-39. XML Response

```
<comment xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <object-id>0800000180143727</object-id>
  <comment-id>1324839</comment-id>
  <owner-name>John Doe</owner-name>
  <creation-date>2016-01-26T15:16:16.000+00:00</creation-date>
  <modified-date>2016-01-26T15:16:16.000+00:00</modified-date>
  <content-value>Agree with John.<br><ol><li>Please add a diagram
    to show the flow.</li><li>Please clarify the filter criteria.<br>
    </li></ol>
  </content-value>
  <parent-id>1324830</parent-id>
  <title/>
  <can-delete>true</can-delete>
  <can-reply>true</can-reply>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
      objects/0900000180143717/comments/1324839"/>
    <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/
      objects/0900000180143717/comments"/>
    <link rel="http://identifiers.emc.com/linkrel/replies"
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
        comments/1324839/replies"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
        comments/1324839"/>
  </links>
</comment>
```

### Example 2-40. JSON Response

```
{
  "object-id": "0800000180143727",
  "comment-id": "1324839",
  "owner-name": "John Doe",
  "creation-date": "2016-01-26T15:16:16.000+00:00",
  "modified-date": "2016-01-26T15:16:16.000+00:00",
  "content-value": "Agree with John.<br><ol><li>Please add a diagram to
    show the flow.</li><li>Please clarify the filter criteria.
    <br></li></ol>",
  "parent-id": "1324830",
  "title": "",
  "can-delete": true,
  "can-reply": true,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
        comments/1324839"
```

```
{,
  {
    "rel": "parent",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments"
  },
  {
    "rel": "http://identifiers.emc.com/linkrel/replies",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments/1324839/replies"
  },
  {
    "rel": "http://identifiers.emc.com/linkrel/delete",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000180143717/
      comments/1324839"
  }
]
}
```

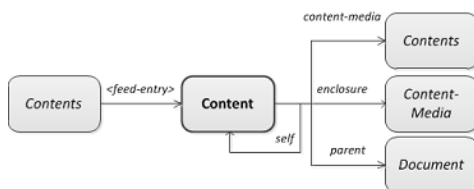
# Content(s)

## Content

The Content resource represents a content object in a repository. A content object contains both content properties and a link to the content stream.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Contents, page 164](#) and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Content resource.

Link Relation	Description	Resource Reference
self	Link to this content.	<a href="#">Content, page 157</a>
content-media [1] *	Link to the content binary stream for download.	Content URL from REST server when neither of the ACS server or BOCS server is available. ( <a href="#">Content Media, page 171</a> )  Distributed content URL when either of the ACS server or BOCS server is available.
enclosure [1] *	Link to content binary stream for download.	Content URL from REST server when neither of the ACS server or BOCS server is available. ( <a href="#">Content Media, page 171</a> )  Distributed content URL when either of the ACS server or BOCS server is available.

Link Relation	Description	Resource Reference
parent	Link to the associated object of this content.	<a href="#">Document, page 195</a>
<p>[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:</p> <p><code>http://identifiers.emc.com/linkrel/</code></p> <p>* Repeatable link relation</p>		

## Operations

The Content resource supports the following HTTP methods.

Method	Description
GET	Retrieves the metadata of the primary content or of the rendition content for an existing document object, optionally with content media links for content download.
DELETE	Deletes primary content and/or rendition content for a document object.

## Get Content

Retrieve the metadata of the primary content or of the rendition content for an existing document object, optionally with content media links for content download.

### Content URL return policy

By using the `media-url-policy` parameter, you can customize content URL return policy in this operation to control what URL to put in the `content-media` and `enclosure` link relations. Valid values for this parameter are:

Value	Description
DC-PREF *	<p>When you set <code>media-url-policy</code> to <code>DC-PREF</code>, this operation tries to retrieve a distributed content URL first. If no distributed content URL is available, a Content Media resource URL from the REST server is still acceptable. Details are as follows:</p> <ul style="list-style-type: none"> <li>If the BOCS server is available, and the network location is specified correctly, the BOCS content URL corresponding to the network location is returned.</li> </ul>

	<ul style="list-style-type: none"> <li>• If the BOCS server is available, but the specified network location is not correct, or there is no BOCS configured for this network location, or the BOCS server is unavailable, the ACS content URL is returned if any ACS server is available.</li> <li>• If there are multiple ACS URLs available, the REST server chooses the one with most proximity.</li> <li>• If no ACS or BOCS server is available, a Content Media resource URL is returned from the REST server.</li> </ul>
DC-ONLY	When you set <code>media-url-policy</code> to DC-ONLY, the REST server returns a distributed content URL (either from the ACS server or the BOCS server with the same logic as the DC-PREF option does. If there is no ACS server or BOCS server, the REST server returns the HTTP 400 Conflict error.
LOCAL	When you set <code>media-url-policy</code> to LOCAL, the REST server returns a Content Media resource URL from the REST server regardless of the availability of the ACS server and the BOCS server.
ALL	When you set <code>media-url-policy</code> to ALL, all available Content Media links are returned, each one pointing to a different content media server. It includes all available ACS URLs, all available BOCS URLs, and the local Content Media resource URL.
* Default	

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Value Range	Default value
<i>page</i>	Specifies the page number for the content imported.	integer	Any non-negative number.  Content Server defines the other constraints on this value.	0

Variable	Description	Data type	Value Range	Default value
<i>format</i>	Specifies the format for the content imported.	string	Any format registered in the Format table of the repository.	null
<i>modifier</i>	Specifies the page modifier for the content.	string	Any string value.	null
<i>media-url-policy</i>	Specifies the Content Media URL return policy. For detailed information, see <a href="#">Content URL return policy, page 158</a> .	enum	<ul style="list-style-type: none"> <li>• DC-PREF</li> <li>• DC-ONLY</li> <li>• LOCAL</li> <li>• ALL</li> </ul>	DC-PREF
<i>network-location</i>	<p>Specifies the network location identifier for BOCS content download.</p> <p>Used only when <i>media-url-policy</i> is set to DC-PREF or DC-ONLY.</p>	string	Any valid network location identifier corresponding to the <code>netloc_ident</code> property of the <code>dm_network_location_map</code> instance.	null

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

## Request Body

N/A



## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 204 - No Content
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied
- 500 - Other unexpected server error

## Response Body

XML or JSON message that represents the content metadata and a URL for content download.

## Delete Content

Delete primary content and/or rendition content for a document object. By default, when a primary content is deleted, the renditions on the same page are deleted at the same time. Renditions created by Documentum Platform REST Services are always deleted if the corresponding primary content is deleted. However, under some circumstances, renditions created via other approaches can remain when the primary content is deleted.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Value Range	Default Value
page	Specifies the page number for the content imported.	integer	Any non-negative number.  Content Server defines the other constraints on this value.	NA  This is a required parameter.
format	Specifies the format for the content imported.	string	Any format registered in the Format table of the repository.	NA  This is a required parameter.
modifier	Specifies the page modifier for the content.	string	Any string value.	null

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

**Response Status**

- 204 - Success; no content is returned
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed
- 403 - Permission denied; no permission to delete the object
- 500 - Other unexpected server error

**Response Body**

HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

## Contents

The Contents resource represents a collection of primary content objects and renditions of an object. Each content entry contains content properties and a link to the content binary stream.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [SysObject, page 472](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the content without the file extension.	Object name.	Last updated time of the object.	<a href="#">Content, page 157</a>	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the content	Content name	Content description	r_modify_date of the content

## Link Relations

The following table lists general link relations in the Contents resource.

Link Relation	Description	Resource Reference
self	Link to the contents.	<a href="#">Contents, page 164</a>
parent [1]	Link to the associated object or document of this content	<a href="#">Document, page 195</a>
first, last, next, previous	Pagination links.	<a href="#">Folder Child Objects, page 248</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:  <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Contents resource supports the following HTTP methods.

Method	Description
GET	Get a collection of content resources.
POST	Create a new primary content or rendition on a document.

### Get Content

Get a collection of all primary content, rendition content metadata, and URLs to the content binary stream for a document object, depending on the query parameter settings.

### Supported HTTP Method

GET

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

### Request Query Parameters

The following parameters are specific to this operation.

Variable	Description	Data type	Value Range	Default value
<i>media-url-policy</i>	Specifies the Content Media URL return policy. For detailed information, see <a href="#">Content URL return policy, page 158</a> .	enum	<ul style="list-style-type: none"><li>• DC-PREF</li><li>• DC-ONLY</li><li>• LOCAL</li><li>• ALL</li></ul>	null
<i>network-location</i>	Specifies the network location identifier for BOCS content download.	string	Any valid network location identifier corresponding to the netloc	null

Variable	Description	Data type	Value Range	Default value
	Used only when media-url-policy is set to DC-PREF or DC-ONLY.		_ident property of the dm_network_location_map instance.	

This operation also supports the following parameters:

- inline
- page
- items-per-page
- include-total

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

### Request Body

N/A

### Response Headers

- Content-Type

### Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

### Response Status

200 - Content retrieved successfully  
400 - Invalid syntax or missing a required value.  
401 - Invalid or missing authentication credentials  
500 - Other unexpected server error

## Response Body

XML or JSON message that represents the content, and a link to the content binary stream.

## Create Content

Create new primary content or renditions on a document.

## Supported HTTP Method

POST

## Request Media Types

Any MIME type registered in the Format table of the repository.

## Request Query Parameters

Variable	Description	Data type	Value Range	Default value
<i>primary</i>	Specifies whether the imported content is primary content or a rendition.	boolean	true - indicates the imported content is primary content.  false - indicates the imported content is a rendition.	true
<i>page</i>	Specifies the page number for the content imported.	integer	Any non-negative number.  Content Server defines the other constraints on this value.	0
<i>format</i>	Specifies the format for the content imported.	string	Any format registered in the Format table of the repository.	null

Variable	Description	Data type	Value Range	Default value
<i>overwrite</i>	Specifies whether or not to overwrite the existing content with the same content properties.	boolean	<p>true - Overwrite the existing primary content.</p> <p>false - Throw an exception when the primary content or rendition with the same content properties already exists.</p>	false
<i>modifier</i>	Specifies the page modifier for the content.	string	Any string value.	null
<i>content-length</i>	Specifies the byte count of the contents to be uploaded	string	An accurate byte count of the content to be uploaded	null
<i>content-charset</i>	<p>Used to sanitize content.</p> <p>This parameter tells the server how to parse the content for sanitizing. When the <i>content-charset</i> is not provided, the server tries to get metadata information from the content itself when it's in HTML format.</p> <p>When the metadata has a valid charset, then it is used to sanitize the content. The configuration default charset is used when the <i>content-charset</i> parameter and the metadata information are not found</p>	string	A valid charset	null



Variable	Description	Data type	Value Range	Default value
	<b>Note:</b> When the charset and the content do not match, the uploaded content may have an incorrectly encoded value.			

**Note:** When you import primary content, the `format` parameter is optional. However, when you specify this parameter, its value must be the same as the `format` of the primary content on page 0. When the page contains primary content, the `overwrite` parameter must be set to `true`.

**Note:** When storing content to Centera storage, the `content-length` query parameter must be provided. When providing the value for `content-length`, you must provide an accurate byte count of the content that you want to upload.

For other storage mediums, such as harddisk, the `content-length` parameter is not required and is optional. However, even when not required, when this parameter is provided, it must be accurate.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

Binary of the content.

## Response Headers

- Location
- Content-Type

## Response Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`
- `application/xml`
- `application/json`

## **Response Status**

201 - Created successfully  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
403 - Permission denied; no permission to delete the object  
406 - Not Acceptable  
409 - Conflict  
415 - Unsupported Media Type  
500 - Other unexpected server error

## **Response Body**

JSON or XML representation of the Content resource.

# Content Media

The Content Media resource represents the media of the primary content or rendition downloaded from the REST server, other than those from an ACS server or a BOCS server. This resource is typically used when no ACS server or BOCS server is available.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Content, page 157](#) and [About the Diagram](#).

## Link Relations

N/A

## Operations

The Content Media resource supports the following HTTP method.

Method	Description
GET	Retrieves a content media from the REST server.

## Get a Content Media from the REST Server

Retrieve a content media from the REST server when no ACS server or BOCS server is available.

### Supported HTTP Method

GET

### Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Value Range	Default value
<i>page</i>	Specifies the page number for the content imported.	integer	Any non-negative number.  Content Server defines the other constraints on this value.	0
<i>format</i>	Specifies the format for the content imported.	string	Any format registered in the Format table of the repository.	null
<i>modifier</i>	Specifies the page modifier for the Content Media.	string	Any string value.	null

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Authorization
- If-None-Match (If the REST client uses the ETag mechanism to enable client side web cache, the value of this header must be identical to the ETag value from the previous server response header.)

**Note:** When the client provides the If-None-Match header, the REST server compares the value of the header with the ETag of the current version of the resource. If these two values match, meaning that the resource has not been changed, the REST server sends back a short response with the HTTP 304 Not Modified status, which notifies the client to use the cached version.

## Request Body

N/A

## Response Headers

- Content-Type
- Content-Length
- ETag

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 304 - Not Modified
- 400 - Invalid syntax or missing a required value
- 403 - Permission denied
- 404 - Resource not found
- 500 - Other unexpected server error

## Response Body

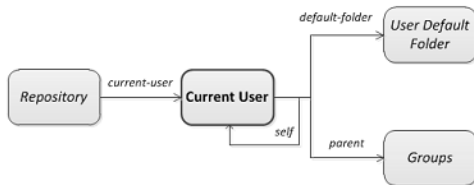
The Content Media downloaded from the REST server.

# Current User

The Current User resource represents the metadata of the user that is currently logged in to the repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Current User resource:

Link Relation	Description	Resource Reference
self	This User resource.	<a href="#">Current User, page 174</a>
default-folder [1]	Default folder (home cabinet) of the user.	<a href="#">User Default Folder, page 509</a>
parent	Groups to which the user belongs.	<a href="#">Groups, page 280</a>
edit	This user resource. Only a SysAdmin or SuperUser can find this link.	<a href="#">Current User, page 174</a>
delete	This user resource. Only a SysAdmin or SuperUser can find this link.	<a href="#">Current User, page 174</a>
permission-set	The permission set definition of the user.	<a href="#">User Permission Set, page 347</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:  <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

# Operations

The Current User resource supports the following HTTP method.

Method	Description
GET	Retrieves the metadata of the user that is currently logged in to the repository.

## Get a Current User

Retrieve the metadata of a current user.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameter:

- view

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

401 - Authentication failed

500 - Other unexpected server error

## Response Body

XML or JSON representation of the current login user.



# Current User Preference(s)

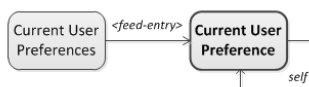
## Current User Preference

The Current User Preference resource represents the user preference settings of the current user. This resource provides the user that is currently logged in to the repository with the capability to manage the user preference setting. For example, you can use this resource to store client-related metadata.

The syntax and semantics of the user preference is determined by the client application that uses REST Services to store preference data. Documentum Platform REST Services introduces a preference resource model to store user preference content as plain text content. In addition to the user preference content, there are several properties to describe the preference metadata. User preference content characters must be escaped by the REST client.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Repository](#), page 369 and [About the Diagram](#).

## Link Relation

The following table lists link relations for the Current User Preference resource:

Link Relation	Description	Resource Reference
self	This User Preference resource.	<a href="#">Current User Preference</a> , page 177
edit	Edit this User Preference resource.	<a href="#">Current User Preference</a> , page 177
<a href="http://identifiers.emc.com/linkrel/delete">http://identifiers.emc.com/linkrel/delete</a>	Delete this User Preference resource.	<a href="#">Current User Preference</a> , page 177

## Operations

The Current User Preference resource supports the following HTTP methods:

Method	Description
GET	Get metadata, content, and other information from the preference instance
POST	Updates the attributes of the preference instance
DELETE	Deletes the preference instance

## Get Current User Preference Settings

Gets the attributes, content, and other information of the current user preference instance.

### Supported HTTP Method

GET

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

### Request Query Parameters

N/A

### Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Retrieved successfully  
 400 - Bad request  
 401 - Authentication failed  
 500 - Other unexpected server error

## Response Body

### Example 2-41. XML Response

```
<preference xmlns="http://identifiers.emc.com/vocab/documentum">
  <client>kw</client>
  <owner-name>dmdadmin</owner-name>
  <title>user preference for knowledge worker</title>
  <subject>user preference</subject>
  <keywords>
    <item>knowledge worker</item>
    <item>cn</item>
  </keywords>
  <creation-date>2015-11-10T05:40:40.000+00:00</creation-date>
  <r_modify_date>2015-11-10T05:49:52.000+00:00</r_modify_date>
  <preference-content>
    &lt;user-locale&gt;
      cn
    &lt;/user-locale&gt;
    &lt;timezone&gt;
      gmt+8
    &lt;/timezone&gt;
  </preference-content>
  <links>
    <link rel="self" href="/repositories/REPO/currentuser-preferences/kw"/>
    <link rel="edit" href="/repositories/REPO/currentuser-preferences/kw"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="/repositories/REPO/currentuser-preferences/kw"/>
  </links>
</preference>
```

### Example 2-42. JSON Response

```
{
  "name": "preference",
  "client": "kw",
  "owner-name": "dmdadmin",
  "title": "user preference for knowledge worker",
  "subject": "user preference",
  "keywords": ["knowledge worker", "cn"],
  "creation-date": "2015-11-10T05:40:40.000+00:00",
  "r_modify_date": "2015-11-10T05:49:52.000+00:00",
```

```
"preference-content": "{\"user-locale\":\"cn\", \"timezone\":\"gmt+8\"}"
"links": [
  {
    "rel": "self",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  },
  {
    "rel": "edit",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  },
  {
    "rel": "http://identifiers.emc.com/linkrel/delete",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  }
]
```

## Update User Preference Settings

Update the attributes for the preference instance. When the client provides undefined attributes in the Request body, the server throws an exception.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of preference settings of the current user. You can update the following properties:

- title
- subject
- keywords
- preference-content

The title, subject, and keywords properties have the same property definitions as those in the `dm_sysobject` type. There is no size limit on `preference-content`. However, the actual size affects the REST request and response payload.

You can place a CDATA block or escaped string in the `preference-content` property to store client-related metadata as shown in the following samples:

### Example 2-43. XML Representation

```
<preference>
  <properties>
    <title>Chrome Browser</title>
    <subject>The browser of Chrome</subject>
    <keywords>
      <item>Firefox</item>
      <item>Mozilla</item>
    </keywords>
    <preference-content><![CDATA[<user_name>Michael</user_name>]]></preference-content>
  </properties>
</preference>
```

### Example 2-44. JSON Representation

```
{
  "properties":{
    "title":"Chrome Browser",
    "subject":"The browser of Chrome",
    "keywords":["Firefox","Mozilla"],
    "preference-content":{"user_name":"Michael"}
  }
}
```

**Note:** The content in the `preference-content` element is treated as a plain text, and it is not parsed, it is only stored and retrieved as a string.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Preference settings updated successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
404 - User Preference not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the updated user preference settings.

### Example 2-45. XML Response

```
<?xml version='1.0' encoding='UTF-8' ?>
<preference>
  <properties>
    <client>client2</client>
    <owner_name>Administrator</owner_name>
    <title>the title of a preference</title>
    <subject>summary of this preference</subject>
    <keywords>
      <item>key1</item>
      <item>key2</item>
    </keywords>
    <creation-date>2015-09-07T05:47:00.000+00:00</creation-date>
    <r_modify_date>2015-09-07T06:06:39.000+00:00</r_modify_date>
    <preference-content>
      &lt;user_name&gt;Michael&lt;/user_name&gt;
    </preference-content>
  </properties>
  <links>
    <link rel="self" href="localhost:8080/dctm-rest/repositories/REPO/
currentuser-preferences/client2.xml" />
    <link rel="edit" href="localhost:8080/dctm-rest/repositories/REPO/
currentuser-preferences/client2.xml" />
    <link rel="http://identifiers.emc.com/linkrel/delete"
href="localhost:8080/dctm-rest/repositories/REPO/
currentuser-preferences/client2.xml" />
  </links>
```

```
</preference>
```

#### Example 2-46. JSON Response

```
{
  properties: {
    client: "client2",
    owner_name: "Administrator",
    title: "the title of a preference",
    subject: "summary of this preference",
    keywords: ["key1", "key2"],
    creation-date: "2015-09-07T05:47:00.000+00:00",
    r_modify_date: "2015-09-07T06:06:39.000+00:00",
    preference-content: "{\"user_name\":\"Michael\"}",
  }
  links:
  {
    rel: "self"
    href: "localhost:8080/dctm-rest/repositories/REPO/rentuser-preferences/client2"
  },
  {
    rel: "edit"
    href: "localhost:8080/dctm-rest/repositories/REPO/rentuser-preferences/client2"
  },
  {
    rel: "http://identifiers.emc.com/linkrel/delete"
    href: "localhost:8080/dctm-rest/repositories/REPO/rentuser-preferences/client2"
  }
};
}
```

## Delete User Preference Settings

Delete user preference settings.

### Supported HTTP Method

DELETE

### Request Media Types

N/A

### Request Query Parameters

N/A

## Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Delete successful
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Preference not found
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation.

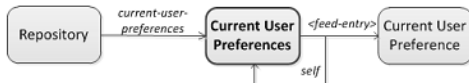


## Current User Preferences

The Preferences resource represents the collection of preferences metadata.

### Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Current User](#), page 174 and [About the Diagram](#).

### Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	Preferences	Server's current time	<a href="#">Current User Preference, page 177</a>	Yes

### Link Relation

The following table lists link relations for the Current User Preferences resource.

Link Relation	Description	Resource Reference
self	This object resource	<a href="#">Current User Preferences, page 185</a>
first, last, next, previous	Pagination link	<a href="#">Current User Preferences, page 185</a>

### Operations

The Current User Preferences resource supports the following HTTP methods.

Method	Description
GET	Gets the current user's preference collection.
POST	Creates a user preference for the current user.

## Get Current User Preferences Collection

Get the current user's preferences collection.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

- sort
- view
- inline
- page
- items-per-page
- include-total
- links
- filter

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Retrieved successfully  
 400 - Bad request  
 401 - Authentication failed  
 500 - Other unexpected server error

## Response Body

### Example 2-47. XML Response

```
<?xml version='1.0' encoding='UTF-8' ?>
<feed>
  <id>http://localhost:8080/dctm-rest/repositories/REPO/currentuser/preferences
  </id>
  <title>Preferences</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-08-17T04:36:37.195+00:00</updated>
  <dm:page>1</dm:page>
  <dm:items-per-page>100</dm:items-per-page>
  <link rel="self"
    href="http://localhost:8080/dctm-rest/repositories/REPO/currentuser/
      preferences.xml?inline=true" />
  <entry>
    <id>http://localhost:8080/dctm-rest/repositories/REPO/currentuser/
      preferences/clinet1
    </id>
    <title>the title of a peference</title>
    <summary>summary of this preference</summary>
    <updated>2015-08-17T04:36:37.197+00:00</updated>
    <published>2015-08-17T04:36:37.197+00:00</published>
    <link rel="edit"
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        currentuser/preferences/clinet1.xml" />
    <content>
      <dm:preference xmlns:dm="http://identifiers.emc.com/vocab/documentum">
        <dm:client>kw</dm:client>
        <dm:owner-name>dadmin</dm:owner-name>
        <dm:title>user preference for knowledge worker</dm:title>
        <dm:subject>user preference</dm:subject>
        <dm:keywords>
          <dm:item>knowledge worker</dm:item>
          <dm:item>cn</dm:item>
        </dm:keywords>
        <dm:creation-date>2015-11-10T05:40:40.000+00:00
        </dm:creation-date>
        <dm:r_modify_date>2015-11-10T05:49:52.000+00:00</dm:r_modify_date>
      </dm:preference>
    </content>
  </entry>
</feed>
```

```

        <dm:preference-content>
            <user-locale>cn</user-locale>
            <timezone>gmt+8</timezone>
        </dm:preference-content>
        <links>
            <link rel="self" href="/repositories/REPO/
            currentuser-preferences/kw"/>
            <link rel="edit" href="/repositories/REPO/
            currentuser-preferences/kw"/>
            <link rel="http://identifiers.emc.com/linkrel/delete"
            href="/repositories/REPO/currentuser-preferences/kw"/>
        </links>
    </dm:preference>
</content>
</entry>
</feed>

```

### Example 2-48. JSON Response

```

{
  id: "http://localhost:8080/dctm-rest/repositories/REPO/currentuser/preferences",
  title: "Preferences",
  author: [{name: "EMC Documentum"}],
  updated: "2015-08-17T04:40:48.970+00:00",
  page: 1,
  items-per-page: 100,
  links: [{rel: "self", href: "http://localhost:8080/dctm-rest/repositories/REPO/
    currentuser/preferences.json?inline=true"}],
  entries: [{
    id: "http://localhost:8080/dctm-rest/repositories/REPO/currentuser/preferences/
    client1",
    title: "the title of a preference",
    summary: "summary of this preference",
    updated: "2015-08-17T04:40:48.972+00:00",
    published: "2015-08-17T04:40:48.972+00:00",
    content: {
      {
        "name": "preference",
        "client": "kw",
        "owner-name": "dmadmin",
        "title": "user preference for knowledge worker",
        "subject": "user preference",
        "keywords": ["knowledge worker", "cn"],
        "creation-date": "2015-11-10T05:40:40.000+00:00",
        "r_modify_date": "2015-11-10T05:49:52.000+00:00",
        "preference-content": "{$\"user-locale\": \"cn\", \"timezone\": \"gmt+8\"}",
        "links": [
          {
            "rel": "self",
            "href": "/repositories/REPO/currentuser-preferences/kw"
          },
          {
            "rel": "edit",
            "href": "/repositories/REPO/currentuser-preferences/kw"
          },
          {
            "rel": "http://identifiers.emc.com/linkrel/delete",
            "href": "/repositories/REPO/currentuser-preferences/kw"
          }
        ]
      }
    }
  ]
}

```

## Create a User Preference

Create a user preference for the current user.

### Supported HTTP Method

POST

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

### Request Query Parameters

N/A

### Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

#### Example 2-49. XML Request

```
<preference xmlns="http://identifiers.emc.com/vocab/documentum">
  <client>kw</client>
  <title>user preference for knowledge worker</title>
  <subject>user preference</subject>
  <keywords>
    <item>knowledge worker</item>
    <item>cn</item>
  </keywords>
  <preference-content>
    &lt;user-locale&gt;cn&lt;/user-locale&gt;
    &lt;timezone&gt;gmt+8&lt;/timezone&gt;
  </preference-content>
</preference>
```

#### Example 2-50. JSON Request

```
{
  "client": "kw",
  "title": "user preference for knowledge worker",
  "subject": "user preference",
  "keywords": ["knowledge worker", "cn"],
  "preference-content": "&lt;user-locale&gt;cn&lt;/user-locale
                        &gt;&lt;timezone&gt;gmt+8&lt;/timezone&gt;
"
}
```

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

201 - Added successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
500 - Other unexpected server error

## Response Body

### Example 2-51. XML Response

```
<preference xmlns="http://identifiers.emc.com/vocab/documentum">
  <client>kw</client>
  <owner-name>dmadmin</owner-name>
  <title>user preference for knowledge worker</title>
  <subject>user preference</subject>
  <keywords>
    <item>knowledge worker</item>
    <item>cn</item>
  </keywords>
  <creation-date>2015-11-10T05:40:40.000+00:00</creation-date>
  <r_modify_date>2015-11-10T05:49:52.000+00:00</r_modify_date>
  <preference-content>
    &lt;user-locale&gt;cn&lt;/user-locale&gt;
    &lt;timezone&gt;gmt+8&lt;/timezone&gt;
  </preference-content>
  <links>
    <link rel="self" href="/repositories/REPO/currentuser-preferences/kw"/>
    <link rel="edit" href="/repositories/REPO/currentuser-preferences/kw"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="/repositories/REPO/currentuser-preferences/kw"/>
  </links>
</preference>
```

### Example 2-52. JSON Response

```
{
  "client": "kw",
  "owner-name": "dmadmin",
  "title": "user preference for knowledge worker",
  "subject": "user preference",
  "keywords": ["knowledge worker", "cn"],
```

```
"creation-date": "2015-11-10T05:40:40.000+00:00",
"r_modify_date": "2015-11-10T05:49:52.000+00:00",
"preference-content": "<user-locale>cn</user-locale>
                       <timezone>gmt+8</timezone>"
"links": [
  {
    "rel": "self",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  },
  {
    "rel": "edit",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  },
  {
    "rel": "http://identifiers.emc.com/linkrel/delete",
    "href": "/repositories/REPO/currentuser-preferences/kw"
  }
]
}
```

## Current Version

The Current Version resource represents the current version of the version tree of an object. A version tree has only one CURRENT version.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [SysObject, page 472](#) and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Current Version resource:

Link Relation	Description	Resource Reference
canonical [1]	Canonical URI of the current version of the object	<a href="#">SysObject, page 472</a> <a href="#">Document, page 195</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Current Version resource supports the following HTTP method.

Method	Description
GET	Retrieves the current version of an object.

## Get the Current Version

Retrieves the current version of a specified object from its version tree.



## Supported HTTP Method

GET

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

This method supports the following common query parameter:

- view

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 404 - Object not found
- 500 - Other unexpected server error

## Response Body

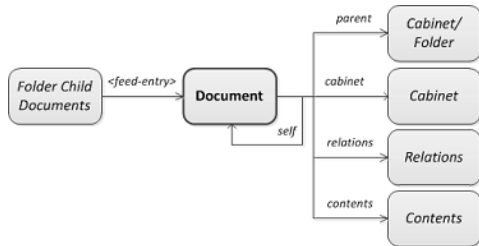
XML or JSON representation of the current version of the specified object.

# Document

The Document resource represents a document in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder Child Documents](#), page 216 and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Document resource:

Link Relation	Description	Resource Reference
self	This document.	<a href="#">Document</a> , page 195
parent	Parent folder (or cabinet) of this document.	<a href="#">Cabinet</a> , page 120 <a href="#">Folder</a> , page 209
cabinet [1]	Cabinet of this document.	<a href="#">Cabinet</a> , page 120
relations [1]	Collection of relations related to this document.	<a href="#">Relations</a> , page 357
contents	Collection of contents associated to this document.	<a href="#">Contents</a> , page 164
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:  <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Document resource supports the following HTTP methods.

Method	Description
GET	Retrieves properties, and other information of the document resource.
POST	Updates the properties for the document resource.
DELETE	Deletes the document resource from a repository.

## Get a Document

Get properties and other information of the document resource. Properties are returned as embedded elements in the response message body. Other information, such as relationships, is referenced from the link relations of the response message body.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

### Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 404 - Object not found
- 500 - Other unexpected server error

## Response Body

XML or JSON message that represents the document:

## Update a Document

Update the document with given properties. When the client provides undefined properties in the Request body, the server throws an exception.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the document resource. Only updatable properties can be put in the message body.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Document updated successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
404 - Object not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the document resource. The successful response contains all properties of the updated document.

## Delete a Document

Deletes the Document resource from the repository.

When the deletion fails, an exception is thrown and the Document resource is reverted to the original state.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Default value
del-version	<p>Delete options for multi-version document objects.</p> <ul style="list-style-type: none"><li>• <code>selected</code> - The version associated with the given object ID is removed.</li><li>• <code>unused</code> - (Default) The versions that do not have symbolic labels associated are removed.</li><li>• <code>all</code> - All revisions on the same version tree are removed.</li></ul>	string	all

## Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Success; no content is returned
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed
- 403 - Permission denied; no permission to delete the object
- 409 - State conflicted
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

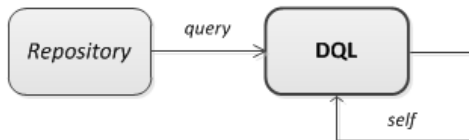


## DQL

The DQL resource represents a read-only DQL query.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
Feed URI	DQL Query Results	Server's current time	Yes

Entry	Entry ID	Entry Title	Entry Updated
Query result corresponding to one of the rows in the result set	Query resource URI with the item index in the current page	<p>Value of the first property of the query result.</p> <p>For example, if you perform the DQL statement <code>select r_object_id, object_name from dm_document,</code> the value of <code>r_object_id</code> is used as the title.</p>	<code>r_modify_date</code> of the query result if this property exists. Otherwise, an empty string.

## Link Relations

The following table lists link relations for the DQL collection resource.

Link Relation	Description	Resource Reference
self	This DQL resource	<a href="#">DQL, page 201</a>
first, last, next, previous	Pagination links.	<a href="#">DQL, page 201</a>

## Operations

The DQL resource supports the following HTTP methods.

Method	Description
GET	Performs a read-only DQL query with the <code>dql</code> query parameter.
POST	Performs a read-only DQL query with the <code>dql</code> query parameter or with the Request body encoded in <code>application/x-www-form-urlencoded</code> .

### Perform a read-only DQL query

Perform a read-only DQL query with the `dql` query parameter.

Using the GET operation to perform a query can result in a response with better cache performance. However, the web browser or client library may fail to handle the Request properly when the DQL query is very long (for example, over 2,000 characters) since the DQL statement is provided via the HTTP query parameter `dql`. To perform a query with a long DQL statement, use the POST operation. For more information, see [Perform a long read-only DQL query, page 205](#).

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

The following parameter is specific to this operation:

Variable	Description	Data type	Default value	Required or not
dql	The query parameter of the DQL resource	string	null  This parameter is required.	Yes
raw	Indicates whether to return the DQL results as raw property bags or with links generated.  When this parameter is set to false, the REST server generates link relations and/or including thumbnail links if possible.	boolean	false	No

**Note:** International characters that are used in query parameters must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.

Also, this operation supports the following common query parameter:

- page
- items-per-page

**Note:** This operation does not support the `include-total` query parameter. To obtain the total size of the result set, send another query by using the aggregation function `COUNT ( )` with the same conditions.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of result set of the DQL query.

XML representation

```
...
<entry>
  <id>http://localhost:8080/dctm-rest/repositories/acme01?dql=...</id>
  <title>090004d2800001d0</title>
  <updated>2013-06-18T20:17:39.331+08:00</updated>
  <content>
    <dm:query-result>
      <dm:properties>
        <dm:r_object_id>090004d2800001d0</dm:r_object_id>
        <dm:object_name>Default Signature Page Template</dm:object_name>
      </dm:properties>
    </dm:query-result>
  </content>
</entry>
...
```

JSON representation

```
...
{
  "id": "http://localhost:8080/dctm-rest/repositories/acme01?dql=...",
  "title": "090004d2800001d0",
  "updated": "2013-06-18T20:22:41.425+08:00",
  "content": {
    "name": "query-result",
```

```

        "properties": {
            "r_object_id": "090004d2800001d0",
            "object_name": "Default Signature Page Template"
        }
    },
    ...

```

- The query result is included in the `query-result` element.
- If the client tries to perform a non-select query, the server returns an HTTP 400 Bad Request status.
- The returned query result set only contains the resources that you have access to.
- Pagination is supported.

**Note:** The query result may contain link relations or thumbnail links. For more information, see *Generating Link Relations in DQL Results in the EMC Documentum Platform REST Services Development Guide*.

## Perform a long read-only DQL query

Perform a read-only DQL query with the `dql` query parameter or with the Request body encoded in `application/x-www-form-urlencoded`.

To perform a query with long DQL statement, use the POST operation with a request body in `application/x-www-form-urlencoded`.

## Supported HTTP Method

POST

## Request Media Types

- `application/x-www-form-urlencoded`

## Request Query Parameters

The following parameter is specific to this operation:

Variable	Description	Data type	Default value	Required or not
<i>dql</i>	Read-only DQL statement to perform	string	null  This parameter is required if the HTTP request does not contain a body.	No
<i>raw</i>	Indicates whether to return the DQL results as raw property bags or with links generated.  When this parameter is set to false, the REST server generates link relations and/or including thumbnail links if possible.	boolean	false	No

- page
- items-per-page

Also, this operation supports the following common query parameter:

**Note:** This operation does not support the `include-total` query parameter. To obtain the total size of the result set, send another query by using the aggregation function `COUNT()` with the same conditions.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

Query statement to perform in `application/x-www-form-urlencoded`. Alternatively, you can leave the Request body blank and use the `dql` parameter to hold the query statement. Use the Request body when the query statement is long.

**Note:** An error may occur if you specify the same query statement in both the Request body and the dql parameter.

### Example 2-53. Example request

```
Accept: application/vnd.emc.documentum+json
Content-Type: application/www-form-urlencoded
POST http://localhost:8080/dctm-rest/repositories/acme01
dql=select%20r_object_id,object_name%20from%20dm_document&items-per-page=100&page=1
```

## Response Headers

- Content-Type

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request
- 401 - Authentication failed
- 403 - Permission denied
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of result set of the DQL query.

XML representation

```
...
<entry>
  <id>http://localhost:8080/dctm-rest/repositories/acme01?dql=...</id>
  <title>090004d2800001d0</title>
  <updated>2013-06-18T20:17:39.331+08:00</updated>
  <content>
    <dm:query-result>
      <dm:properties>
        <dm:r_object_id>090004d2800001d0</dm:r_object_id>
        <dm:object_name>Default Signature Page Template</dm:object_name>
      </dm:properties>
    </dm:query-result>
```

```
</content>
</entry>
...
```

### JSON representation

```
...
{
  "id": "http://localhost:8080/dctm-rest/repositories/acme01?dql=...",
  "title": "090004d2800001d0",
  "updated": "2013-06-18T20:22:41.425+08:00",
  "content": {
    "name": "query-result",
    "properties": {
      "r_object_id": "090004d2800001d0",
      "object_name": "Default Signature Page Template"
    }
  }
},
...
```

- The query result is included in the `query-result` element.
- If the client tries to perform a non-select query, the server returns an HTTP 400 Bad Request status.
- The returned query result set only contains the resources that you have access to.
- Pagination is supported.

**Note:** The query result may contain link relations or thumbnail links. For more information, see *Generating Link Relations in DQL Results* in the *EMC Documentum Platform REST Services Development Guide*.

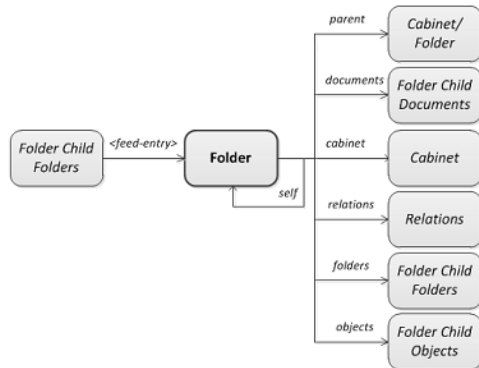


# Folder

The Folder resource represents a folder in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder Child Folders](#), page 233 and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Folder resource.

Link Relation	Description	Resource Reference
self	Link to this folder.	<a href="#">Folder</a> , page 209
parent	Link to the parent folder (or cabinet) of this folder.	<a href="#">Cabinet</a> , page 120 <a href="#">Folder</a> , page 209
relations [1]	Collection of relations related to this folder.	<a href="#">Relations</a> , page 357
cabinet [1]	Link to the cabinet of this folder.	<a href="#">Cabinet</a> , page 120
folders [1]	Collection of folders linked to this folder.	<a href="#">Folder Child Folders</a> , page 233
documents [1]	Collection of documents linked to this folder.	<a href="#">Folder Child Documents</a> , page 216

Link Relation	Description	Resource Reference
objects [1]	Collection of <i>SysObjects</i> linked to this folder.	<a href="#">Folder Child Objects, page 248</a>
<p>[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:</p> <p><a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a></p>		

## Operations

The Folder resource supports the following HTTP methods.

Method	Description
GET	Retrieves properties, and other information of the Folder resource.
POST	Updates the properties for the Folder resource.
DELETE	Deletes the Folder resource from a repository.

### Get a Folder

Get properties, and Other information, such as relationships, of the Folder resource. Properties are returned as embedded elements in the response message body. Other information, such as relationships, is referenced from the link relations of the response message body.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request; for example, bad property filter  
401 - Authentication failed  
404 - No object found  
500 - Other unexpected server error

## Response Body

XML or JSON message that represents the folder.

## Update a Folder

Update the folder with given properties. If the client provides undefined properties in the Request body, the server throws an exception.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the Folder resource. Only updatable properties can be put in the message body.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Update successful
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the folder. The successful response contains all properties of the updated folder.

## Delete a Folder

Deletes the Folder resource from the repository.

Folder children deletion is controlled by query parameters.

If the deletion fails, an exception is thrown and the Folder resource (including the folder tree and version history) is reverted to the original state.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data type	Default value
del-non-empty	Specifies whether this operation deletes a non-empty folder or not. <ul style="list-style-type: none"><li>• <code>true</code> - The folder and all its descendants are deleted.</li></ul>	boolean	false

Variable	Description	Data type	Default value
	<ul style="list-style-type: none"><li>• <code>false</code> - Only this folder is deleted. Deleting a non-empty folder results in an error.</li></ul>		
<code>del-all-links</code>	<p>Specifies whether a multi-linked descendant is deleted or unlinked from this specified folder.</p> <ul style="list-style-type: none"><li>• <code>true</code> - This operation deletes a multi-linked descendant along with all its folder links.</li><li>• <code>false</code> - This operation only deletes a descendant's link to this folder. The descendant is still linked to other folders.</li></ul> <p><b>Note:</b> Virtual documents are not supported.</p>	boolean	false

## Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Delete successfully
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed
- 403 - Permission denied; no permission to delete the object
- 409 - State conflicted
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

## Folder Child Documents

The Folder Child Documents resource represents the collection of all documents (an object of the `dm_document` type or its subtype) under a specified folder.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder](#), page 209 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the documents without the file extension	List of folder child documents	Server's current time	<a href="#">Document</a> , page 195	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the document	Document name	Document description	<code>r_modify_date</code> of the document

## Link Relations

The following table lists general link relations in the Folder Child Documents resource:

Link Relation	Description	Resource Reference
self	The URI for folder child objects collection feed.	<a href="#">Folder Child Objects</a> , page 248
first, last, next, previous	Pagination links.	<a href="#">Folder Child Objects</a> , page 248

## Operations

The Folder Child Objects resource supports the following HTTP methods:



Method	Description
GET	Retrieves a list all documents under the specified folder.
POST	Creates a new document under the specified folder.

## Get Child Documents under a Folder

Get a collection of child documents under the given folder. You can specify any folder types, including `dm_folder`, `dm_cabinet` or a custom type.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- `inline`
- `sort`

The default sort order is by creation date, descending (most recent to oldest).

- `page`
- `items-per-page`
- `include-total`
- `view`
- `links`
- `q`

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following parameters are specific to this operation:

Variable	Description	Data type	Default value
hide-shared-parent	Specifies whether to show the shared parent object of the lightweight object  <b>Note:</b> Enabling <code>hide-shared-parent</code> may decrease performance	boolean	false
include-all-versions	Specifies whether or not to list all versions of the child object's link.  <ul style="list-style-type: none"> <li>• <code>true</code> - Returns all versions.</li> <li>• <code>false</code> - Only returns the current version</li> </ul>	boolean	false
object-type	Specifies the exact sub type name to get from the collection of this type	string	null
thumbnail	Specifies whether this operation deletes a non-empty folder or not.  <ul style="list-style-type: none"> <li>• <code>true</code> - Return the thumbnail link for each entry in the collection resource.</li> <li>• <code>false</code> - Do not return the thumbnail link for each entry in the collection resource.</li> </ul>	boolean	false
q	Specifies the search criterion with a full-text expression in simple search language.  Parameter <code>q</code> must be encoded because it may contain non-English locale characters.	String	No search criteria is used for the search.

Variable	Description	Data type	Default value
	<b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.		

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

## Response Status

200 - Retrieved successfully  
 400 - Bad request; for example, bad property filter  
 401 - Authentication failed

403 - Permission denied; no permission to update the object or setting read-only properties is not allowed

404 - Document not found

500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of child objects under the specified folder.

- The body contains a list of document resources, each of which is filed under the specified folder.
- Each object may contain all properties of the object, depending on the setting of the query parameter `view`.
- Each object contains links specific to the type of the object. See [Link Relations, page 472](#) in SysObject resource, which provides information on what links are available for a certain type of object.
- The returned child objects collection only contains those that you have access to.

## Create a Document Under a Folder

Create a child document under the specified folder. You can set the properties when creating the object. The object can be the type of `dm_document` or its subtype. By default, the object type is `dm_document`.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

When storing content to Centera storage, the `content-length` query parameter must be provided. When providing the value for `content-length`, you must provide an accurate byte count of the content to be uploaded.

For other storage mediums, such as `harddisk`, the `content-length` parameter is not required and is optional. However, even when not required, if this parameter is provided, it must be accurate.

**Note:** When creating a document that contains no content, neither *content-length* or *content-charset* is used. These two variables are only used when your documents contain content.

Variable	Description	Data Type	Default Value
<i>content-length</i>	Specifies the byte count of the content to be uploaded	string	<p>null</p> <p><i>content-length</i></p> <p>When storing content to Centera storage, the <i>content-length</i> parameter must be provided. When providing the value for <i>content-length</i>, you must provide an accurate byte count of the content to be uploaded.</p> <p>For other storage mediums, such as harddisk, the <i>content-length</i> parameter is optional. However, even when not required, if this parameter is provided, it must be accurate.</p>
<i>content-charset</i>	<p>Used to sanitize content.</p> <p>This parameter tells the server how to parse the content for sanitizing. When the <i>content-charset</i> is not provided, the server tries to get metadata information from the content itself when it's in HTML format.</p> <p>When the metadata has a valid charset, then it is used to sanitize the content. The configuration</p>	string	null

Variable	Description	Data Type	Default Value
	default charset is used when the <i>content-charset</i> parameter and the metadata information are not found  <b>Note:</b> When the charset and the content do not match, the uploaded content may have an incorrectly encoded value.		

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the document to create.

- Object type of the new document is specified in the `r_object_type` property of the Request body.

The `r_object_type` property is optional. If this property is not specified in a client request, it is set to `dm_document` by default.

- Content Server validates the constraints and permissions on setting the other properties (including read-only properties).

### Example 2-54. Request Payload Sample in XML

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+xml
```

```
<object>
  <properties>
    <object_name>xyz</object_name>
  </properties>
</object>
```

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the first content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the second content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the third content part

```
--314159265358979--
```

### Example 2-55. Request Payload Sample in JSON

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+json
{
  "properties":{
    "object_name":"xyz"
  }
}
```

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the first content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the second content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the third content part

```
--314159265358979--
```

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Document created successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied; no permission to update the object or setting read-only properties is not allowed  
404 - No object found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the document created under the specified folder.

- Each object contains all properties of a document.
- Each object contains links specific to the document object type.

For more information on what kind of links are available for a certain type of object, see [Link Relations, page 472](#) in the SysObject resource.

## Import Document Metadata and Contents

Create a document and its contents under a specified folder in a single request.

## Supported HTTP Method

POST

## Request Media Types

- multipart/form-data
- multipart/mixed



## Request Query Parameters

Variable	Description	Data type	Value Range	Default value
<i>content-count</i>	Specifies how many pieces of content are submitted. The metadata part is excluded from the count.	integer	$\geq 0$	0
<i>all-primary</i>	Specifies whether to import all contents using the same type as the primary content for different pages (page 0, 1, 2...), or one primary content, and others as renditions for the first page, which is page 0	boolean	true or false	true
<i>format</i>	<p>Specifies the <code>dm_format</code> name for contents.</p> <p>When more than one type of contents are being imported, the <i>format</i> parameter can be used to define the format for each piece of content by using a comma separated string.</p> <p>For example,  <code>format=crtext,html,pub_html</code></p> <p>For more information, see <a href="#">format</a> below.</p>	string	Comma separated string	null

Variable	Description	Data type	Value Range	Default value
<i>modifier</i>	<p>Specifies the modifier of the contents. For the primary content, the modifier is ignored.</p> <p>For example, when you import with the <i>all-primary</i> parameter set to false, the value of the modifier can be: <code>modifier =, mod1, mod2, mod3</code>. That means the first (or primary) content has no modifier, and the second rendition's modifier is <code>mod1</code>, the third is <code>mod2</code> and so on.</p> <p>Setting the <i>all-primary</i> parameter to true , or when <code>content-count&lt;=1</code> causes all modifiers to be ignored.</p>	string	Comma separated string	null
<i>content-length</i>	<p>Specifies the byte count of the content to be uploaded. This parameter is optional.</p> <p>A comma separated list can be used to define multiple values for this parameter. For example:</p>	string	Comma separated string	null

Variable	Description	Data type	Value Range	Default value
	<p><code>content</code>  <code>-length</code>  <code>=1024,2048,,4096</code>  . The blank value shown here is 0.</p> <p>For more information, see <a href="#">content-length</a> below.</p>			
<code>content</code> <code>-charset</code>	<p>This parameter is used to sanitize content and it defines how the server parses the content while sanitizing. Multiple values for this parameter can be set using a comma separated list. For example:</p> <p><code>content</code>  <code>-charset=UTF-8,GBK,ISO-8859-1</code>.</p> <p>For more information, see <a href="#">content-charset</a> below.</p>	string	Comma separated string	null

**Note:** Here are some additional details about the variables listed above:

- *format*

The *format* list must match the content sequence. When you don't want to specify the format for a piece of content you can leave it blank in the comma separated list but you must still include the comma as a placeholder.

For example: `format=crtext,,html`

When the format for contents is not defined, the server uses the content media type to select the format.

When the *all-primary* parameter is *true*, all primary content types must have the same format. In this case, when you define more than one format, they must be the same

For example: `format=crtext,crtext,crtext`. You can also define one format for all of the primary content: `format=crtext`

- *content-charset*

When the metadata has a valid charset, then it is used to sanitize the content.

When the *content-charset* is not provided and the content is in HTML format, the server tries to get a value for the *content-charset* parameter from the metadata information of the content itself.

The configuration default charset is used when the *content-charset* parameter and the metadata information are not found

When the charset and the content do not match, the uploaded content may have an incorrectly encoded value.

- *content-length*

When the *all-primary* parameter is set to false, the first value of in the *content-length* parameter is used for the primary piece of content, the other values in the comma separated string are ignored.

When storing content to Centera storage, the *content-length* parameter is mandatory and must be provided.

For other storage mediums, such as harddisk, the *content-length* parameter is optional.

In all cases, including when the *content-length* parameter is not required, if a value for it is set, the value must be an accurate byte count of the contents to be uploaded.

- *content-count*

When you import more than one piece of contents, the REST server must know how many content pieces there are in total. The total number of content pieces cannot be obtained until the end of the multipart stream. Therefore, the REST client must specify this number in the value of the *content-count* parameter.

When the value set in the *content-count* parameter is greater than the actual number of content pieces, an exception is thrown because the system expects more content pieces than it has received.

When the value set in the *content-count* parameter is less than the actual number of content pieces, all content pieces that exist in addition to the number of pieces declared in the *content-count* parameter are ignored.

It is critical that the value provided by the *content-count* parameter be accurate.

## Request Headers

- Accept
- Authorization
- Content-Type (The Content-Type header must be either *multipart/form-data* or *multipart/mixed*, which notifies the server this request is a multipart request.)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The Request body is a multipart message that must conform to RFC 2046. A multipart/form-data or multipart/mixed message that contains a series of parts. This series must conform to the following rules:

- The series must contain one metadata part and the contents parts.
- The first part in the series must contain the metadata for the document to be created.

Content-Type for the first part must be one of the following values:

- application/vnd.emc.documentum+json (default)
- application/vnd.emc.documentum+xml

If the client sets Content-Type to an invalid value, the server uses application/vnd.emc.documentum+json to overwrite that value.

- The contents parts in the series must be the binary data of the primary or rendition contents. Content-Type for the other parts must be the MIME type for the contents.
- Each part in the series must contain a Content-Disposition header. The disposition type of the header must be form-data. The disposition must contain a parameter titled name that specifies the name of this part.

### Example 2-56. XML Request Payload for multipart/form-data or multipart/mixed

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+xml
```

```
<object>
  <properties>
    <object_name>xyz</object_name>
  </properties>
</object>
```

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the first content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the second content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the third content part

```
--314159265358979--
```

### Example 2-57. JSON Request Payload for multipart/form-data or multipart/mixed

```
--314159265358979
Content-Disposition: form-data; name=metadata
```

```
Content-Type: application/vnd.emc.documentum+json
{
  "properties":{
    "object_name":"xyz"
  }
}
```

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the first content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the second content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the third content part

```
--314159265358979--
```

### Example 2-58. How to use Multipart in a Custom Controller

```
@RequestMapping(method = RequestMethod.POST, consumes = { "multipart/form-data",
    "multipart/mixed" })
@ResponseStatus(HttpStatus.OK)
public void processMultipart(@RequestBody final Iterator<Part> parts) throws DfException,
MissingServletRequestParameterException, IOException {
    if(parts != null) {
        response.setContentType(MediaType.TEXT_PLAIN.toString());
        OutputStream out = response.getOutputStream();
        int p = 1;

        while(parts.hasNext()) {
            Part part = parts.next();
            for(String header : part.getHeaderNames()) {
                // Write the part header to the Response directly
                out.write((header + "=" + part.getHeader(header) + "\r\n" ).getBytes());
            }
            // Write the part content to the Response
            IOUtils.copy(part.getInputStream(), out);
            out.write("\r\n-----end-----\r\n".getBytes());
            out.flush();
        }
    }
}
```

The sample reads the multipart input and writes it back to the Response. Each part of the multipart input has headers, and the content stream. Each part can be individually read by using one of the standard `hasNext` and `next` Iterator methods.

All of the parts have to be read sequentially because they are all contained within one multipart stream and the REST server does not cache it.

**Note:** When more than one content part exists, the REST server must know how many content parts there are in total. However, the total number of content parts cannot be determined until the end of the multipart stream. Therefore, the REST client must provide the total number of content parts

to the REST server by setting a value for the *content-count* parameter. For more information about this parameter, see [content-count](#).

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 201 - Document created successfully
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the document created under the specified folder.

- Each object contains all properties of the object.
- Each object contains links specific to the type of the object. See [Link Relations, page 472](#) in SysObject resource, which provides information on what links are available for a certain type of object.

### Example 2-59. XML Response Payload for application/vnd.emc.documentum+xml or "application/xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns="http://identifiers.emc.com/vocab/documentum"
  definition="{typeResourceUri}"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="dm_document">
  <properties>
    <object_name>xyz</object_name>
```

```
<r_object_id>090020808001398e</r_object_id>
...
</properties>
<links>
  <link rel="self" href="{documentResourceUri}"/>
  ...
</links>
</document>
```

**Example 2-60. JSON Response payload for `application/vnd.emc.documentum+xml` or `application/xml`**

```
{
  "name": "document",
  "type": "dm_document",
  "definition": "<documentResourceUri>",
  "properties": {
    "object_name": "xyz",
    "r_object_id": "090020808001398e",
    ...
  }
  "links": [
    { "rel": "self", "href": "<documentResourceUri>" },
    ...
  ]
}
```



## Folder Child Folders

The Folder Child Folders resource represents the collection of all folders (an object of the `dm_folder` type or its subtype) under a specified folder.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder, page 209](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the folders.	List of folder child folders	Server's current time	<a href="#">Folder, page 209</a>	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the folder	Folder name	Folder description	<code>r_modify_date</code> of the folder

## Link Relations

The following table lists general link relations in the Folder Child Folders resource:

Link Relation	Description	Resource Reference
self	The URI for folder child objects collection feed.	<a href="#">Folder Child Folders, page 233</a>
first, last, next, previous	Pagination links.	<a href="#">Folder Child Folders, page 233</a>

## Operations

The folder child folders resource supports the following HTTP methods:

Method	Description
GET	Retrieves a list all child folders under the specified folder.
POST	Creates a new child folder under the specified folder.

## Get Child Folders under a Folder

Get the collection of child folders under the given folder. You can specify any folder types, including `dm_folder`, `dm_cabinet` or a custom type.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- `inline`

- `sort`

The default sort order is by creation date, descending (most recent to oldest).

- `page`

- `items-per-page`

- `include-total`

- `view`

- `links`

- `q`

- `filter`

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following parameters are specific to this operation:

Variable	Description	Data type	Default value
hide-shared-parent	Specifies whether to show the shared parent object of the lightweight object  <b>Note:</b> Enabling <code>hide-shared-parent</code> may decrease performance	boolean	false
object-type	Specifies the exact sub type name to get from the collection of this type	string	null
thumbnail	Specifies whether this operation deletes a non-empty folder or not. <ul style="list-style-type: none"><li>• <code>true</code> - Return the thumbnail link for each entry in the collection resource.</li><li>• <code>false</code> - Do not return the thumbnail link for each entry in the collection resource.</li></ul>	boolean	false

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request; for example, bad property filter  
401 - Authentication failed  
403 - Permission denied; no permission to update the object or setting read-only properties is not allowed  
404 - Folder not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of child objects under the specified folder.

- The body contains a list of child folder objects, each of which is filed under the specified folder.
- Each object may contain all properties of the folder, depending on the setting of the query parameter `view`.
- Each object contains links specific to the folder object type. See [Link Relations, page 472](#) in SysObject resource, which provides information on what kind of links are available for a certain type of object.
- The returned child objects collection only contains those that you have access to.

## Create a Child Folder under a Folder

Create a child folder under the specified folder. You can set the properties when creating the object. The object can be the type of `dm_folder` or its subtype other than `dm_cabinet` or its subtype. This is because `dm_cabinet` must be a top-level folder which cannot be filed under any other folder. By default, the object type is `dm_folder`.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the folder to create.

- The `object_name` property is required.
- Object type of the new document is specified in the `r_object_type` property of the Request body.

The `r_object_type` property is optional. If this property is not specified in a client request, it is set to `dm_folder` by default.

- Content Server validates the constraints and permissions on setting the other properties (including read-only properties).

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Folder created successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied; no permission to update the object or setting read-only properties is not allowed  
404 - No object found  
409 - Conflict. The folder name already exists  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the folder created under the specified folder. The Response contains all properties of the folder.

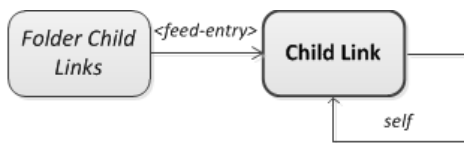
# Folder Child Link(s)

## Folder Child Link

The Folder Child Link resource represents the folder containment relationship between a parent folder and one of its child objects.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder Child Links, page 242](#) and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Child Link resource:

Link Relation	Description	Resource Reference
self	Link to this child link	<a href="#">Folder Child Link(s), page 239</a>

## Operations

The Child Link resource supports the following HTTP methods:

Method	Description
GET	Retrieves the folder link between the parent folder and its child object.
DELETE	Unlinks a given object from one of its parent folder.

## Retrieve a folder link

Retrieve the folder link between the child object and its parent folder.

## Supported HTTP Method

GET

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

400 - Invalid syntax or missing a required value



- 401 - Invalid or missing authentication credentials
- 403 - Permission denied.
- 404 - Object not found
- 409 - Conflict. For example, the object has already been checked out.
- 500 - Other unexpected server error

## **Response Body**

XML or JSON representation of the folder link between the object and its parent folder. The body contains the URI for the parent object and the URI for the child object. The body also contains the parent ID and child ID.

## **Unlink an Object from a Parent Folder**

Unlink an object from one of its parent folders.

### **Note:**

- To unlink an object from a primary link, you must have at least the Write permission and the Change Location permission on the object.
- To unlink an object from a secondary link, you must have at least the Browse permission and the Change Location permission on the object.
- If the repository is running under folder security, you must also have at least the Write permission on the folder or cabinet from which the object is being unlinked.
- Documents and folders must have at least one link to a folder or cabinet. Therefore, you cannot unlink an object from its only linked folder or cabinet.

## **Supported HTTP Method**

DELETE

## **Request Media Types**

N/A

## **Request Query Parameters**

N/A

## Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Unlink the object successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied.
- 404 - Object not found
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful unlink operation. The Response body contains no content.

## Folder Child Links

The Folder Child Links collection resource represents a collection of all folder containment relationships for a Folder. That is a set of links between a folder and its child objects.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder](#), page 209 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
URI of the feed	List of parent folders	r_modify_date of the object	Yes

Entry	Entry ID	Entry Title	Feed Updated
<a href="#">Folder Child Links</a> , page 242	URI of the folder link	Formatted string with both the child object ID and the parent object ID	r_modify_date of the child object.

## Link Relations

The following table lists general link relations in the Folder Child Links resource.

Link Relation	Description	Resource Reference
self	URI for Folder Child Links collection feed	<a href="#">Folder Child Links</a> , page 242
first, last, next, previous	Pagination links	<a href="#">Folder Child Links</a> , page 242

## Operations

The Folder Child Links resource supports the following HTTP methods.

Method	Description
GET	Retrieves a collection of child links for a given folder.
POST	Links an object to a given folder.

## Retrieve child links

Retrieve all child links for a given folder. That is a set of the links between the folder and all child objects under this folder.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

The following parameter is specific to this operation:

Variable	Description	Data type	Default value
include-all-versions	<p>Specifies whether or not to list all versions of the child object.</p> <ul style="list-style-type: none"><li>• <code>true</code> - Return all versions.</li><li>• <code>false</code> - Only return the current version.</li></ul>	boolean	false

This operation also supports the following common parameters:

- inline
- page
- items-per-page
- include-total
- sort
- filter
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of a collection child links for the given folder.

- The body contains a list of links between the given object and each of its parent folders.
- Each link contains the URI for the given folder and the URI for one of its child objects.
- Each entry contains the parent ID and the child ID.
- Each entry in the collection has a `self` link referring to the link resource.
- Pagination is supported.

## Link an Object to a Folder

Link a new object to a given folder or cabinet.

The first execution of link on an object defines the object's primary link, which is the place where the object is stored in the repository. Subsequent executions of link associate the object to other folders or cabinets. These links are called secondary links. This operation only allows you to create a new secondary link for an object. The existing folder links for this object do not change. For more

information about how to create primary link during the object creation, see [Folder Child Objects, page 248](#).

To create a secondary link, you must have Browse permission and the Change Location permission on the object. When the repository is running under folder security, you must also have the Write permission on the folder or cabinet to which the object is linked.

**Note:** You cannot link a cabinet to a folder.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the newly-created folder link between the given folder and the child object. The body contains the reference to the child object.

### Example 2-61. Example of the Request body

```
{
  "href": "http://localhost:8080/dctm-rest/repositories/myrepo/objects/0900000c80000cc4"
}
```

## Response Headers

- Location
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Link created successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
404 - No object found  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of newly-created link between the given folder and its new child object.

## Folder Child Objects

The Folder Child Objects resource represents the collection of all the SysObject resources under a specified folder.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder](#), page 209 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the objects.	List of folder child objects	Server's current time	<a href="#">SysObject</a> , page 472	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the object	Object name	Object description	r_modify_date of the object

## Link Relations

The following table lists general link relations in the Folder Child Objects resource.

Link Relation	Description	Resource Reference
self	The URI for folder child objects collection feed.	Folder child objects collection resource
first, last, next, previous	Pagination links.	Folder child objects collection resource

## Operations

The folder child objects resource supports the following HTTP methods.



Method	Description
GET	Retrieves a list all objects under the specified folder.
POST	Creates a new object under the specified folder.
POST	Imports the object metadata and primary content.
POST	Copies an object to the specified folder.

## Get Folder Child Objects

Get the collection of child objects under the given folder. You can specify any folder types, including `dm_folder`, `dm_cabinet` or a custom type.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- `inline`
- `sort`

The default sort order is by creation date, descending (most recent to oldest).

- `page`
- `items-per-page`
- `include-total`
- `view`
- `links`
- `filter`
- `q`

The `q` parameter is supported for full text searching with a subset of the Simple Search Language, however parentheses are not supported.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following parameters are specific to this operation:

Variable	Description	Data type	Default value
hide-shared-parent	Specifies whether to show the shared parent object of the lightweight object.  <b>Note:</b> Enabling <code>hide-shared-parent</code> may decrease performance.	boolean	false
include-all-versions	Specifies whether or not to list all versions of the child object's link. <ul style="list-style-type: none"><li>• <code>true</code> - Returns all versions.</li><li>• <code>false</code> - Only returns the current version</li></ul>	boolean	false
object-type	Specifies the exact sub type name to get from the collection of this type.	string	null
thumbnail	Specifies whether this operation deletes a non-empty folder or not. <ul style="list-style-type: none"><li>• <code>true</code> - Return the thumbnail link for each entry in the collection resource.</li><li>• <code>false</code> - Do not return the thumbnail link for each entry in the collection resource.</li></ul>	boolean	false
q	Specifies the search criteria with a full-text expression in simple search language.  Parameter <code>q</code> must be encoded because it may contain	String	No search criteria is used for the search.

Variable	Description	Data type	Default value
	non-English locale characters.  <b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.		

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request; for example, bad property filter

401 - Authentication failed

403 - Permission denied; no permission to update the object or setting read-only properties is not allowed

500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of child objects under the specified folder.

- The body contains a list of SysObject resources, each of which is filed under the specified folder.
- Each object may contain all properties of the object, depending on the setting of the query parameter `view`.
- Each object contains links specific to the type of the object. See [Link Relations, page 472](#) in SysObject resource, which provides information on what links are available for a certain type of object.
- The returned child objects collection only contains those that you have access to.

## Create an Object under a Folder

Create a child object under the specified folder. You can set the properties when creating the object. The object can be the type of `dm_sysobject` or its subtype. By default, the object type is `dm_sysobject`.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

When storing content to Centera storage, the `content-length` query parameter must be provided. When providing the value for `content-length`, you must provide an accurate byte count of the content to be uploaded. For other storage mediums, such as `harddisk`, the `content-length` parameter is not required and is optional. However, even when not required, if this parameter is provided, it must be accurate.

**Note:** When creating a document that contains no content, neither `content-length` or `content-charset` is used. These two variables are only used when your documents contain content.

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the object to create.

- The object type is specified in the `r_object_type` property of the Request body.  
The `r_object_type` property is optional. If this property is not specified in a client request, it is set to `dm_folder` by default.
- Content Server validates the `object_name` property in the Request. This property is mandatory for some types, for example, `dm_folder`.
- Content Server validates the constraints and permissions on setting the other properties (including read-only properties).

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`
- `application/xml`
- `application/json`

## Response Status

201 - Object created successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied; no permission to update the object or setting read-only properties is not allowed

404 - No object found

500 - Other unexpected server error

## Response Body

XML or JSON representation of the SysObject resource created under the specified folder.

- Each object contains all properties of the object.
- Each object contains links specific to the type of the object. See [Link Relations, page 472](#) in SysObject resource, which provides information on what links are available for a certain type of object.

## Import SysObject Metadata and Contents

Create a document and its contents under a specified folder in a single request.

## Supported HTTP Method

POST

## Request Media Types

- multipart/form-data
- multipart/mixed

## Request Query Parameters

All of the variables shown in the following table use the *POST* HTTP method.

Variable	Description	Data type	Value Range	Default value
<i>content-count</i>	Specifies how many pieces of content are submitted. The metadata part is excluded from the count.  For more information, see <a href="#">content-count</a>	integer	$\geq 0$	0

Variable	Description	Data type	Value Range	Default value
<i>all-primary</i>	Specifies whether to import all contents using the same type as the primary content for different pages (page 0, 1, 2...), or one primary content, and others as renditions for the first page, which is page 0	boolean	true or false	true
<i>format</i>	<p>Specifies the <code>dm_format</code> name for contents.</p> <p>When more than one type of contents are being imported, the <i>format</i> parameter can be used to define the format for each piece of content by using a comma separated string.</p> <p>For example,</p> <pre>format =crtext,html,pub_html</pre> <p>For more information, see <a href="#">format</a></p>	string	Comma separated string	null

Variable	Description	Data type	Value Range	Default value
<i>modifier</i>	<p>Specifies the modifier of the content. For the primary content, the modifier is ignored.</p> <p>For example, when you import with the <i>all-primary</i> parameter set to <i>false</i>, the value of <i>modifier</i> can be: <code>modifier =, mod1, mod2, mod3</code>. That means the first (or primary) contents has no modifier, and the second rendition's modifier is <code>mod1</code>, the third rendition's modifier is <code>mod2</code> and so on.</p> <p>For more information, see <a href="#">modifier</a></p>	string	Comma separated string	null
<i>content-length</i>	<p>Specifies the byte count of the content to be uploaded. This parameter is optional.</p> <p>A comma separated list can be used to define multiple values for this parameter. For example:  <code>content-length=1024,2048,,4096</code></p>	string	Comma separated string	null



Variable	Description	Data type	Value Range	Default value
	<p>. The blank value shown here is 0.</p> <p>For more information, see <a href="#">content-length</a></p>			
<code>content-charset</code>	<p>This parameter is used to sanitize content and it defines how the server parses the content while sanitizing. Multiple values for this parameter can be set using a comma separated list. For example:</p> <pre>content-charset=UTF-8,GBK,ISO-8859-1.</pre> <p>For more information, see <a href="#">content-charset</a></p>	string	Comma separated string	null

**Note:** Here are some additional details about the variables listed above:

- *format*

The *format* list must match the content sequence. When you don't want to specify the format for a piece of content you can leave it blank in the comma separated list but you must still include the comma as a placeholder.

For example: `format=crtext,,html`

When the format for contents is not defined, the server uses the content media type to select the format.

When the *all-primary* parameter is `true`, all primary content types must have the same format. In this case, when you define more than one format, they must be the same

For example: `format=crtext,crtext,crtext`. You can also define one format for all of the primary content: `format=crtext`

- *modifier*

Setting the *all-primary* parameter to `true`, or when `content-count<=1` causes all modifiers to be ignored.

- *content-charset*

When the metadata has a valid charset, then it is used to sanitize the content.

When the *content-charset* is not provided and the content is in HTML format, the server tries to get a value for the *content-charset* parameter from the metadata information of the content itself.

The configuration default charset is used when the *content-charset* parameter and the metadata information are not found

When the charset and the content do not match, the uploaded content may have an incorrectly encoded value.

The configuration default charset is used when the *content-charset* parameter and the metadata information are not found

- *content-length*

When the *all-primary* parameter is set to false, the first value of in the *content-length* parameter is used for the primary piece of content, the other values in the comma separated string are ignored.

When storing content to Centera storage, the *content-length* parameter is mandatory and must be provided.

For other storage mediums, such as harddisk, the *content-length* parameter is optional.

In all cases, including when the *content-length* parameter is not required, if a value for it is set, the value must be an accurate byte count of the contents to be uploaded.

When the *all-primary* parameter is set to false, the first value in the comma separated *content-length* string parameter is used for the primary piece of content, but the other values in the comma separated string are ignored.

- *content-count*

When you import more than one piece of contents, the REST server must know how many content pieces there are in total. The total number of content pieces cannot be obtained until the end of the multipart stream. Therefore, the client must specify this number in the value of the *content-count* parameter.

When the value set in the *content-count* parameter is greater than the actual number of content pieces, an exception is thrown because the system expects more content pieces.

When the value set in the *content-count* parameter is less than the actual number of content pieces, all content pieces that exist in addition to the number of pieces set in the *content-count* parameter are ignored.

The value provided by the *content-count* must be accurate.

## Request Headers

- Accept
- Authorization
- Content-Type (The Content-Type header must be either *multipart/form-data* or *multipart/mixed*, which notifies the server this request is a multipart request.)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The Request body is a multipart message that must conform to RFC 2046. A `multipart/form-data` or `multipart/mixed` message contains a series of parts. This series must conform to the following rules:

- The series must contain two parts.
- The first part in the series must contain the metadata for the document to be created.  
Content-Type for the first part must be one of the following values:
  - `application/vnd.emc.documentum+json` (default)
  - `application/vnd.emc.documentum+xml`
 When the REST client sets Content-Type to a value that is invalid, the Server uses `application/vnd.emc.documentum+json` to overwrite the invalid value.
- The second part in the series must be the binary data of the primary content. Content-Type for the second part must be the MIME type for the primary content.
- Each part in the series must contain a Content-Disposition Header. The Header's disposition type must be `form-data`. The disposition must contain a parameter titled `name`, which specifies the name of this part.

### Example 2-62. XML Request Payload for `multipart/form-data` or `multipart/mixed`

```
--314159265358979
Content-Disposition: form-data; name=metadata
Content-Type: application/vnd.emc.documentum+xml

<object>
  <properties>
    <object_name>xyz</object_name>
  </properties>
</object>

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the first content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the second content part

--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain

This is the third content part

--314159265358979--
```

### Example 2-63. JSON Request Payload for `multipart/form-data` or `multipart/mixed`

```
--314159265358979
Content-Disposition: form-data; name=metadata
```

```
Content-Type: application/vnd.emc.documentum+json
{
  "properties":{
    "object_name":"xyz"
  }
}
```

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the first content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the second content part

```
--314159265358979
Content-Disposition: form-data; name=binary
Content-Type: text/plain
```

This is the third content part

```
--314159265358979--
```

## Response Headers

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 201 - Object created successfully
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the SysObject resource created under the specified folder.

- Each object contains all properties of the object.
- Each object contains links specific to the type of the object. See [Link Relations, page 472](#) in SysObject resource, which provides information on what links are available for a certain type of object.

### Example 2-64. XML Response Payload for `application/vnd.emc.documentum+xml` or `application/xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns="http://identifiers.emc.com/vocab/documentum"
  definition="{typeResourceUri}"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="dm_document">
  <properties>
    <object_name>xyz</object_name>
    <r_object_id>090020808001398e</r_object_id>
    ...
  </properties>
  <links>
    <link rel="self" href="{documentResourceUri}"/>
    ...
  </links>
</document>
```

### Example 2-65. JSON Response payload for `application/vnd.emc.documentum+xml` or `application/xml`

```
{
  "name": "document",
  "type": "dm_document",
  "definition": "<documentResourceUri>",
  "properties": {
    "object_name": "xyz",
    "r_object_id": "090020808001398e",
    ...
  }
  "links": [
    { "rel": "self", "href": "<documentResourceUri>" },
    ...
  ]
}
```

## Copy an Object to a Folder

Copy an object to a folder. The type of the object must be `dm_sysobject` or its subtype. Properties of the object can be customized during the copy operation.

**Note:** When you copy a folder, whether and how to copy the objects under the folder depends on the variables set in the message body.

For more information, see [Request Body, page 262](#).

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The Request body contains the link to the object to copy, and the following variables.

Variable	Description	Data type	Default value
deep-copy	<p>Specifies whether or not to copy the objects under the folder:</p> <ul style="list-style-type: none"><li>• true - copy the objects under the folder</li><li>• false - copy the folder only.</li></ul>	boolean	true

Variable	Description	Data type	Default value
<code>replicate-vd-children</code>	<p>When the value of the <i>deep-copy</i> parameter is <code>true</code> and you copy a folder, copying a virtual document that is within that folder will cause the children of that virtual document to be copied as well.</p> <p>When the value of the <i>deep-copy</i> parameter is <code>false</code> and you copy a folder, the children of that virtual document are only referenced.</p>	boolean	<code>true</code>
<code>retain-acl</code>	<p>This parameter determines whether to apply a source object's ACL to the target object while copying an object.</p> <p>When this parameter is set to <code>true</code>:</p> <ul style="list-style-type: none"> <li>• When the source is an object or the source is a folder that has the <i>deep-copy</i> parameter set to <code>false</code>, this parameter causes the source ACL to be applied only to the newly copied object or folder.</li> <li>• When the source is a folder with the <i>deep-copy</i> parameter set to <code>true</code>, the ACL of all the source objects within that folder are applied to the</li> </ul>	boolean	<code>false</code>

Variable	Description	Data type	Default value
	<p>their related newly copied objects.</p> <p>When the <i>retain-acl</i> parameter is false, no ACL is applied to the target of the copy procedure.</p>		

**Example 2-66. JSON Request Payload**

```
{
  "href":"http://localhost/dctm-rest/repositories/acme/objects/xxxxx",
  "deep-copy":true,
  "replicate-vd-children":true,
  "retain-acl":true
}
```

**Example 2-67. XML Request Payload**

```
<dm:object href="http://localhost/dctm-rest/repositories/acme/objects/xxxxx"
  deep-copy="true" replicate-vd-children="true" retain-acl="true"/>
```

**Response Headers**

- Location
- Content-Length
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

**Response Media Types**

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

**Response Status**

- 201 - Object created successfully
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 404 - Object not found
- 500 - Other unexpected server error



## Response Body

XML or JSON representation of the object copied to this folder.

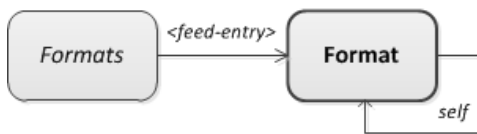
# Format(s)

## Format

The Format resource represents a `dm_format` instance stored in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Formats, page 269](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Format resource.

Link Relation	Description	Resource Reference
self	This format	<a href="#">Format, page 266</a>

## Operations

The Format resource supports the following HTTP method.

Method	Description
GET	Retrieves the information about a format.

## Get a Format

Retrieves the information about a format by a specified format name.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- view
- links
- sort
- inline
- filter
- page
- items-per-page
- include-total

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

- application/xml
- application/json

### **Response Status**

200 - Retrieved successfully  
401 - Authentication failed  
403 - Permission denied  
404 - Format not found  
500 - Other unexpected server error

### **Response Body**

XML or JSON representation of the format instance.

**Note:** The format name in the URL is encoded.

## Formats

The Formats resource represents a collection of `dm_format` instances stored in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Entry	Supports POST?
Feed URI	List of formats	<a href="#">Format, page 266</a>	No

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the format	Format name	Format description	<code>r_modify_date</code> of the format

## Link Relations

The following table lists link relations for the Formats resource.

Link Relation	Description	Resource Reference
self	This collection of formats.	<a href="#">Formats, page 269</a>
first, last, next, previous	Pagination links.	<a href="#">Formats, page 269</a>

## Operations

The Formats resource supports the following HTTP method.

Method	Description
GET	Retrieves the information about a collection of formats.

## Get Formats

Retrieve the information about a collection of formats.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- inline
- view
- items-per-page
- page
- include-total
- sort
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of formats.

- The body contains a list of the `dm_format` instances (or subtypes of `dm_format`).
- Each object may contain all or a set of properties of the format, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by format name.
- The format name in the URL is encoded.

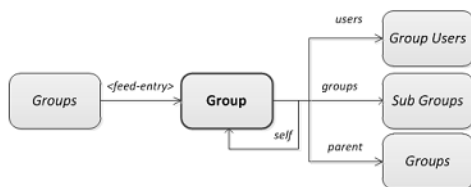
# Group(s)

## Group

The Group resource represents a group in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Groups, page 280](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Group resource.

Link Relation	Description	Resource Reference
self	This Group resource.	<a href="#">Group, page 272</a>
edit	This Group resource. Only sysAdmin, SuperUser or group owner can view this link	<a href="#">Group, page 272</a>
delete	This Group resource. Only sysAdmin, SuperUser or group owner can view this link	<a href="#">Group, page 272</a>
users [1]	Users that directly belong to this group.	<a href="#">Sub Groups, page 467</a>
groups [1]	Sub groups that directly belong to this group.	<a href="#">Sub Groups, page 467</a>
parent	Groups that this group belongs to.	<a href="#">Groups, page 280</a>

**Note:** [1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: `http://identifiers.emc.com/linkrel/`



## Operations

The Group resource supports the following HTTP methods:

Method	Description
GET	Retrieve a group instance
POST	Update a group
DELETE	Delete a group

### Get a Group

Retrieve all attributes of a group.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view
- links
- inline
- filter
- sort
- page
- items-per-page
- include-total

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
401 - Authentication failed  
404 - Group not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the group. The information about the parent groups and sub groups are presented in link relations instead of properties.

**Note:** The group name in the URL is encoded.

### Example 2-68. XML Response

```
<?xml version='1.0' encoding='UTF-8' ?>
<group xsi:type="dm_group" definition="http://localhost:8080/dctm-rest/repositories/
  REPO/types/dm_group.xml">
  <properties>
    <group_name>admingroup</group_name>
    <group_address />
    <users_names>
      <item>REPO_ADMIN</item>
      <item>Administrator</item>
    </users_names>
    <owner_name>Administrator</owner_name>
    <is_private>>false</is_private>
    <description />
    <globally_managed>>false</globally_managed>
```

```

<r_modify_date>2015-03-03T02:20:31.000+00:00</r_modify_date>
<alias_set_id>0000000000000000</alias_set_id>
<group_source />
<group_class>group</group_class>
<group_admin />
<r_has_events>false</r_has_events>
<is_dynamic>false</is_dynamic>
<is_dynamic_default>false</is_dynamic_default>
<group_global_unique_id>REPO:admingroup</group_global_unique_id>
<group_native_room_id>0000000000000000</group_native_room_id>
<group_directory_id>0000000000000000</group_directory_id>
<group_display_name>admingroup</group_display_name>
<is_protected>false</is_protected>
<is_module_only>false</is_module_only>
<i_is_replica>false</i_is_replica>
<i_vstamp>0</i_vstamp>
<r_object_id>l200000580000129</r_object_id>
</properties>
<links>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
    admingroup.xml" />
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
    admingroup.xml" />
  <link rel="delete" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
    admingroup.xml" />
  <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/
    groups.xml?group-name=admingroup" />
  <link rel="http://identifiers.emc.com/linkrel/groups"
    href="http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup/
    groups.xml" />
  <link rel="http://identifiers.emc.com/linkrel/users"
    href="http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup/
    users.xml" />
</links>
</group>

```

### Example 2-69. JSON Response

```

{
  "name": "group",
  "type": "dm_group",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_group",
  "properties": {
    "group_name": "admingroup",
    "group_address": "",
    "users_names": ["REPO_ADMIN",
      "Administrator"],
    "owner_name": "Administrator",
    "is_private": false,
    "description": "",
    "globally_managed": false,
    "r_modify_date": "2015-03-03T02:20:31.000+00:00",
    "alias_set_id": "0000000000000000",
    "group_source": "",
    "group_class": "group",
    "group_admin": "",
    "r_has_events": false,
    "is_dynamic": false,
    "is_dynamic_default": false,
    "group_global_unique_id": "REPO:admingroup",
    "group_native_room_id": "0000000000000000",
    "group_directory_id": "0000000000000000",
    "group_display_name": "admingroup",
    "is_protected": false,

```

```
    "is_module_only":false,
    "i_is_replica":false,
    "i_vstamp":0,
    "r_object_id":"1200000580000129"
  },
  "links":[
    {"rel":"self",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup"},
    {"rel":"edit",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup"},
    {"rel":"delete",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup"},
    {"rel":"parent",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups?
      group-name=admingroup"},
    {"rel":"http://identifiers.emc.com/linkrel/groups",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups/
      admingroup/groups"},
    {"rel":"http://identifiers.emc.com/linkrel/users",
     "href":"http://localhost:8080/dctm-rest/repositories/REPO/groups/
      admingroup/users"}
  ]
}
```

## Update a Group

Update the attributes of a Group. Updating a Group can only be done by the SysAdmin, SuperUser, or group owner.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-70. XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<dm:dm_group xsi:type="dm_group" definition="http://localhost:8080/dctm-rest/repositories/REPO/typ
  <dm:properties>
    <dm:owner_name>Administrator</dm:owner_name>
    .....
  </dm:properties>
</dm:dm_group>
```

### Example 2-71. JSON Request

```
{
  properties:{
    "owner_name":"Administrator",
    ...
  }
}
```

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Response Status

- 200 - Updated successfully
- 400 - Bad request
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Group not found
- 500 - Other unexpected server error

## Response Body

### Example 2-72. XML Response

```
<?xml version='1.0' encoding='UTF-8' ?>
<group xsi:type="dm_group" definition="http://localhost:8080/dctm-rest/repositories/
  REPO/types/dm_group.xml">
  <properties>
    <group_name>admingroup</group_name>
    <group_address />
    <users_names>
      <item>REPO_ADMIN</item>
      <item>Administrator</item>
    </users_names>
    <owner_name>Administrator</owner_name>
    <is_private>>false</is_private>
    <description />
    <globally_managed>>false</globally_managed>
    <r_modify_date>2015-03-03T02:20:31.000+00:00</r_modify_date>
    <alias_set_id>0000000000000000</alias_set_id>
    <group_source />
    <group_class>group</group_class>
    <group_admin />
    <r_has_events>>false</r_has_events>
    <is_dynamic>>false</is_dynamic>
    <is_dynamic_default>>false</is_dynamic_default>
    <group_global_unique_id>REPO:admingroup</group_global_unique_id>
    <group_native_room_id>0000000000000000</group_native_room_id>
    <group_directory_id>0000000000000000</group_directory_id>
    <group_display_name>admingroup</group_display_name>
    <is_protected>>false</is_protected>
    <is_module_only>>false</is_module_only>
    <i_is_replica>>false</i_is_replica>
    <i_vstamp>0</i_vstamp>
    <r_object_id>1200000580000129</r_object_id>
  </properties>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
      admingroup.xml" />
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
      admingroup.xml" />
    <link rel="delete" href="http://localhost:8080/dctm-rest/repositories/REPO/groups/
      admingroup.xml" />
    <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/
      groups.xml?group-name=admingroup" />
    <link rel="http://identifiers.emc.com/linkrel/groups"
      href="http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup/
      groups.xml" />
    <link rel="http://identifiers.emc.com/linkrel/users"
      href="http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup/
      users.xml" />
  </links>
</group>
```

### Example 2-73. JSON Response

```
{
  "name": "group",
  "type": "dm_group",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_group",
  "properties": {
    "group_name": "admingroup",
    "group_address": "",
    "users_names": ["REPO_ADMIN",
```

```

    "Administrator"],
    "owner_name": "Administrator",
    "is_private": false,
    "description": "",
    "globally_managed": false,
    "r_modify_date": "2015-03-03T02:20:31.000+00:00",
    "alias_set_id": "0000000000000000",
    "group_source": "",
    "group_class": "group",
    "group_admin": "",
    "r_has_events": false,
    "is_dynamic": false,
    "is_dynamic_default": false,
    "group_global_unique_id": "REPO:admingroup",
    "group_native_room_id": "0000000000000000",
    "group_directory_id": "0000000000000000",
    "group_display_name": "admingroup",
    "is_protected": false,
    "is_module_only": false,
    "i_is_replica": false,
    "i_vstamp": 0,
    "r_object_id": "1200000580000129"
  },
  "links": [
    { "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup" },
    { "rel": "edit",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup" },
    { "rel": "delete",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups/admingroup" },
    { "rel": "parent",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups?
        group-name=admingroup" },
    { "rel": "http://identifiers.emc.com/linkrel/groups",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups/
        admingroup/groups" },
    { "rel": "http://identifiers.emc.com/linkrel/users",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/groups/
        admingroup/users" }
  ]
}

```

## Delete a Group

Delete a group. Deleting a Group can only be done by the SysAdmin, SuperUser, or group owner.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

## Response Status

- 204 - Delete successful
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Group not found
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

N/A

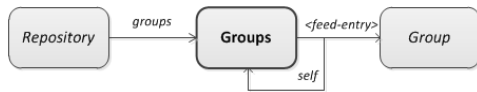
# Groups

The Groups resource represents a collection of group instances in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:





See Also: [Repository](#), page 369 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Entry	Supports POST?
Feed URI	List of groups	<a href="#">Group</a> , page 272	No

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the group	Group name	Group description	r_modify_date of the group

## Link Relations

The following table lists link relations for the Groups resource.

Link Relation	Description	Resource Reference
self	This collection of groups.	<a href="#">Groups</a> , page 280
first, last, next, previous	Pagination links.	<a href="#">Groups</a> , page 280

## Operations

The Groups resource supports the following HTTP method.

Method	Description
GET	Retrieves the information about a collection of groups.

## Get Groups

Retrieve the information about a collection of groups in a repository.

### Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- inline
- view
- items-per-page
- page
- include-total
- sort
- recursive
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

This method supports the following group-related query parameters:

Variable	Description	Data type	Default value
user-name	Only groups that contain a user (a <code>dm_user</code> instance) whose <code>user-name</code> property equals to the specified string are returned in the result.	string	null
group-name	Only groups that contain a sub group (a <code>dm_group</code> instance) whose <code>group_name</code> property equals to the specified string are returned in the result.	string	null
<b>Note:</b> You cannot use <i>username</i> and <i>groupname</i> simultaneously.			

**Note:** International characters that are used in the preceding query parameters must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of groups.

- The body contains a list of the `dm_group` instances (or subtypes of `dm_group`).
- Each object may contain all or a set of properties of the group, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by group name.
- The group name in the URL is encoded.

## Create a Group

Create a group in the repository. The SysAdmin, SuperUser or any other user with "Create group" permissions can create a Group.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The *group\_name* property is mandatory.

### Example 2-74. XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<dm_group xsi:type="dm_group" definition="http://localhost:8080/dctm-rest/repositories/
  REPO/types/dm_group">
  <properties>
    <group_name>group1</group_name>
    ...
  </properties>
</dm_group>
```

### Example 2-75. JSON Request

```
{
```

```
    "properties" : {  
      "group_name" : "group1",  
      ...  
    }  
  }  
}
```

## Response Headers

- Location

The URL of the Group that is created.

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json

## Response Status

201 - Created successfully

400 - Bad request

401 - Authentication failed

403 - Permission denied

500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

N/A

# Group User(s)

## Group User

The Group user resource represents the relationship between a group and one of its user members.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Group Users](#), page 288 and [About the Diagram](#).

## Link Relation

N/A

## Operations

The Group User resource supports the following HTTP method.

Method	Description
DELETE	Remove the relation between a group and it's user member

## Removing a User

Removes the relation between a group and it's user member.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept
- Content-Type
- Authentication

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-76. Removing a User

```
DELETE http://localhost:8080/dctm-rest/repositories/acme/groups/  
mydocument/users/myuser
```

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

## Response Status

- 204 - Delete successful
- 400 - Bad request, for example, bad attribute filter
- 401 - Authenticate failed
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation.

## Group Users

The Group Users resource represents a collection of users that belong to a specified group.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Group, page 272](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	List of users	Server's current time	<a href="#">User, page 495</a>	No

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the user	User name	User description	r_modify_date of the user

## Link Relations

The following table lists link relations for the Group Users resource.

Link Relation	Description	Resource Reference
self	This collection of users.	<a href="#">Group Users, page 288</a>
first, last, next, previous	Pagination links.	<a href="#">Group Users, page 288</a>

## Operations

The Group Users resource supports the following HTTP method.

Method	Description
GET	Retrieves the information about a collection of users.



## Get Group Users

Retrieve the information about a collection of users that belong to a given group.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- inline
- view
- items-per-page
- page
- include-total
- sort
- recursive
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
404 - Group not found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of users.

- The body contains a list of the `dm_user` instances (or subtypes of `dm_user`).
- Each object may contain all or a set of properties of the user, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by user name.

## Add a User Member

Add an existing group into this Group as a user member.

## Supported HTTP Method

POST

## Request Media Types

- application/atom+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

- href

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The *href* variable, which specifies the user resource, must be supplied. This variable is a `String` and has a `null` default value.

### Example 2-77. XML Request

```
<dm:user href="http://localhost/emc-rest/repositories/acme/users/xxxxx"/>
```

### Example 2-78. JSON Request

```
{
  "href": "http://localhost/emc-rest/repositories/acme/users/xxxxx"
}
```

## Response Headers

- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

### **Response Status**

201 - Created successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
404 - Group not found  
500 - Other unexpected server error

### **Response Body**

N/A

# Home Document

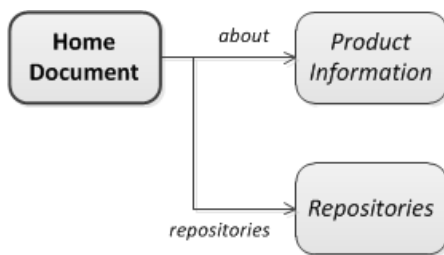
The Home Document collection resource provides an entry point for available resources. The design of Home Document adheres to the following RFC documents:

<http://tools.ietf.org/html/draft-wilde-home-xml-00> (XML representation)

<http://tools.ietf.org/html/draft-nottingham-json-home-02> (JSON representation)

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [About the Diagram](#)

## Link Relations

The following table lists link relations for the Home Document resource.

Link Relation	Description	Resource Reference
repositories [1]	Link to the available repositories.	<a href="#">Repositories, page 374</a>
about	Link to production information.	<a href="#">Product Information, page 351</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Home Document resource supports the following HTTP method.

Method	Description
GET	Retrieves the home document.

## Get Home Document

Retrieves the home document which provides an entry point for available resources. The URI of the Home Document resource is constant, for example `/services`. Your REST clients can use the `/services` URI to access the Documentum Platform REST services.

For example, when the context root of the REST server is deployed as `http://localhost:8080/dctm-rest`, then the Home Document's URI would be `http://localhost:8080/dctm-rest/services`.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept

This resource does not accept `application/vnd.emc.documentum+json` or `application/vnd.emc.documentum+xml`. Instead, it accepts `application/home+json` and `application/home+xml`.

**Note:** The Authorization header is not needed when a client tries to get this resource.

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type
- Content-Length

For more information about HTTP Headers, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/home+xml
- application/home+json

## Response Status

200 - Retrieved successfully  
 415 - Unsupported Media Type  
 500 - Other unexpected server error

## Response Body

XML or JSON representation of the home document.

### Example 2-79. XML Response

```
<resources>
  <resource rel="http://identifiers.emc.com/linkrel/repositories">
    <link href=
      "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories" />
    <hints>
      <allow>
        <i>GET</i>
      </allow>
      <representations>
        <i>application/xml</i>
        <i>application/json</i>
        <i>application/atom+xml</i>
        <i>application/vnd.emc.documentum+json</i>
      </representations>
    </hints>
  </resource>
  <resource rel="about">
    <link href= "http://core-rs-demo.lss.emc.com:8080/dctm-rest/product-info" />
    <hints>
      <allow>
        <i>GET</i>
      </allow>
      <representations>
        <i>application/xml</i>
        <i>application/json</i>
        <i>application/vnd.emc.documentum+xml</i>
        <i>application/vnd.emc.documentum+json</i>
      </representations>
    </hints>
  </resource>
```

</resources>

**Example 2-80. JSON Response**

```
{
  "resources": {
    "http://identifiers.emc.com/linkrel/repositories": {
      "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories.json",
      "hints": {
        "allow": [
          "GET"
        ],
        "representations": [
          "application/xml",
          "application/json",
          "application/atom+xml",
          "application/vnd.emc.documentum+json"
        ]
      }
    },
    "about": {
      "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/product-info.json",
      "hints": {
        "allow": [
          "GET"
        ],
        "representations": [
          "application/xml",
          "application/json",
          "application/vnd.emc.documentum+xml",
          "application/vnd.emc.documentum+json"
        ]
      }
    }
  }
}
```



# Lightweight Objects

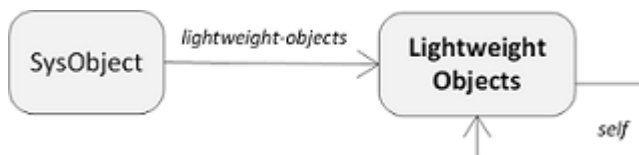
The Lightweight Objects resource represents the lightweight objects that share a shareable object.

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of lightweight objects.	Objects	Server's current time	<a href="#">SysObject, page 472</a>	Yes

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



## Link Relations

The following table lists link relations for the Lightweight Objects resource.

Link Relation	Description	Resource Reference
self	This Lightweight Objects resource	<a href="#">Lightweight Objects, page 297</a>
first, last, next, previous	Pagination links.	<a href="#">SysObject, page 472</a>

## Operations

The Lightweight Objects resource supports the following HTTP methods:

Method	Description
POST	Creates a new lightweight object that shares this shareable object
GET	Retrieves all lightweight objects that share this shareable object

## Create a Lightweight Object

Create a lightweight object and share this shareable object.

**Note:** When creating the lightweight object, if a lightweight object modifies the shared properties, then the lightweight object is materialized. This requires lightweight type is auto materialized.

One shared object can only be shared by one kind of lightweight type object at the same time.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-81. XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<object>
  <properties>
    <object_name>xyz</object_name>
    <r_object_type>lightweight type</r_object_type>
    ...
  </properties>
</object>
```

### Example 2-82. JSON Request

```
{
  "name": "object",
  "properties": {
```

```

        "object_name":"xyz",
        "r_object_type":"lightweight type",
        ...
    }
}

```

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Created successfully  
 400 - Bad request  
 401 - Authentication failed  
 403 - Permission denied  
 500 - Unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-83. XML Response

```

<?xml version="1.0" encoding="UTF-8"?>
<object>
  <properties>
    <object_name>xyz</object_name>
    <r_object_id>090020808001398e</r_object_id>
    ...
  </properties>
  <links>
    <link rel="self" href="{objectResourceUri}"/>
    ...
  </links>
</object>

```

### Example 2-84. JSON Response

```

{
  "name":"object",
  "type":"dm_sysobject",
  "definition":"","

```

```
"properties":{
  "object_name":"xyz",
  "r_object_id":"090020808001398e",
  ...
}
"links":[
  { "rel":"self", "href":"objectResourceUri"},
  ...
]
}
```

## Get the Lightweight Objects of a Shared Object

Get all the lightweight objects of the specified shared parent object.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

- inline
- items-per-page
- view
- include-total
- page
- sort
- filter
- links
- q

**Note:** The *view*, *sort*, and *filter* query parameters do not support the properties of the shared parent.

For example, an exception may be thrown when *view=i\_folder\_id&inline=true*.

The *q* parameter is supported for full text searching with a subset of the Simple Search Language, however parenthesis are not supported.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type
- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

201 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
404 - The shared object does not exist, is not shareable, or is not sharing  
500 - Unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-85. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/
    0800000580005381/lw-objects</id>
  <title>Objects</title>
```

```

<author>
  <name>EMC Documentum</name>
</author>
<updated>2015-12-28T07:31:31.795+00:00</updated>
<dm:page xmlns:dm="http://identifiers.emc.com/vocab/documentum">
  1</dm:page>
<dm:items-per-page xmlns:dm="http://identifiers.emc.com/vocab/documentum">
  100</dm:items-per-page>
<link href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
  0800000580005381/lw-objects?inline=true" rel="self"/>
<entry>
  <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/
  0800000580005d9a</id>
  <title>new lightweight 29</title>
  <author>
    <name>dave</name>
    <uri>http://localhost:8080/dctm-rest/repositories/REPO/users/
    dave</uri>
  </author>
  <summary>my_lightweight_object 0800000580005d9a</summary>
  <updated>2015-12-28T07:00:24.000+00:00</updated>
  <published>2015-12-28T07:00:24.000+00:00</published>
  <link href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
  0800000580005d9a" rel="edit"/>
  <content>
    <dm:object definition="http://localhost:8080/dctm-rest/repositories/REPO/
    types/my_lightweight_object"
    xmlns:dm="http://identifiers.emc.com/vocab/documentum"
    xsi:type="my_lightweight_object">
      <dm:properties>
        <dm:object_name>new lightweight 29</dm:object_name>
        <dm:i_sharing_parent>0800000580005381</dm:i_sharing_parent>
        <dm:r_object_type>my_lightweight_object</dm:r_object_type>
        <dm:r_page_cnt>0</dm:r_page_cnt>
        <dm:r_object_id>0800000580005d9a</dm:r_object_id>
        <dm:r_modify_date>2015-12-28T07:00:24.000+00:00
        </dm:r_modify_date>
        <dm:owner_name>dave</dm:owner_name>
        <dm:creation-date>2015-12-28T07:00:24.000+00:00
        </dm:creation-date>
      </dm:properties>
      <dm:links>
        <link href="http://localhost:8080/dctm-rest/repositories/REPO/
        objects/0800000580005d9a" rel="self"/>
        ...
      </dm:links>
    </dm:object>
  </content>
</entry>
</feed>

```

### Example 2-86. JSON Response

```

{
  "id" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0800000580005381/
  lw-objects",
  "title" : "Objects",
  "author" : [
    {
      "name" : "EMC Documentum"
    }
  ],
  "updated" : "2015-12-28T07:57:27.299+00:00",
  "page" : 1,

```

```

"items-per-page" : 100,
"links" : [
  {
    "rel" : "self",
    "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0800000580005381/lw-objects?inline=true"
  }
],
"entries" : [
  {
    "id" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0800000580005d9a",
    "title" : "new lightweight 29",
    "author" : [
      {
        "name" : "dave",
        "uri" : "http://localhost:8080/dctm-rest/repositories/REPO/users/dave"
      }
    ],
    "summary" : "my_lightweight_object 0800000580005d9a",
    "updated" : "2015-12-28T07:00:24.000+00:00",
    "published" : "2015-12-28T07:00:24.000+00:00",
    "links" : [
      {
        "rel" : "edit",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0800000580005d9a"
      }
    ],
    "content" : {
      "name" : "object",
      "type" : "my_lightweight_object",
      "definition" : "http://localhost:8080/dctm-rest/repositories/REPO/types/
        my_lightweight_object",
      "properties" : {
        "object_name" : "new lightweight 29",
        "i_sharing_parent" : "0800000580005381",
        "r_object_type" : "my_lightweight_object",
        "r_page_cnt" : 0,
        "r_object_id" : "0800000580005d9a",
        "r_modify_date" : "2015-12-28T07:00:24.000+00:00",
        "owner_name" : "dave",
        "creation-date" : "2015-12-28T07:00:24.000+00:00"
      },
      "links" : [
        {
          "rel" : "self",
          "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
            0800000580005d9a"
        },
        ...
      ]
    }
  }
]
}

```

# Lightweight Object Parent

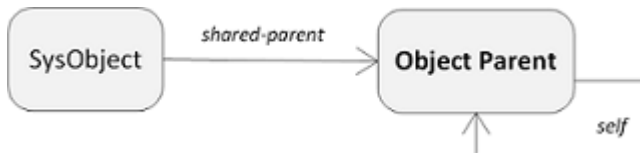
Lightweight object parent resource represents the parent sharable object of a lightweight object.

## Feed

Is feed? No.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



## Link Relations

The following table lists link relations for the Lightweight Object Parent resource:

Link Relation	Description	Resource Reference
self	This Lightweight Object Parent resource	<a href="#">Lightweight Object Parent, page 304</a>
This resource also has all of the same link relations as the <a href="#">SysObject, page 472</a> resource.		

## Operations

The Lightweight Object Parent resource supports the following HTTP methods:

Method	Description
GET	Retrieves the lightweight parent shareable object
POST	Reparent to another shareable object

## Get the Parent Shareable Object

Get the parent shareable object of a lightweight object.



## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

404 - The parent shared object does not exist, or the object is not a lightweight object

201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-87. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<object>
  <properties>
    <object_name>xyz</object_name>
    <r_object_id>090020808001398e</r_object_id>
    ...
  </properties>
  <links>
    <link rel="self" href="{objectResourceUri}"/>
    ...
  </links>
</object>
```

### Example 2-88. JSON Response

```
{
  "name": "object",
  "type": "shareable_type",
  "definition": "",
  "properties": {
    "object_name": "xyz",
    "r_object_id": "090020808001398e",
    ...
  }
  "links": [
    { "rel": "self", "href": "objectResourceUri" },
    ...
  ]
}
```

## Reparent a Lightweight Object

Reparent the lightweight object to another shareable object.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-89. XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<dm:object href="http://localhost/dctm-rest/repositories/acme/objects/xxxxx"/>
```

### Example 2-90. JSON Request

```
{
  "href": "http://localhost/dctm-rest/repositories/acme/objects/xxxxx"
}
```

## Response Headers

- Content-Type
- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Reparent successful

201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

The same as the [SysObject, page 472](#) resource.

### Example 2-91. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<object xmlns="http://identifiers.emc.com/vocab/documentum"
  definition="{typeResourceUri}"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="shareable_type">
  <properties>
    <object_name>xyz</object_name>
    <r_object_id>090020808001398e</r_object_id>
    ...
  </properties>
  <links>
    <link rel="self" href="{documentResourceUri}"/>
    ...
  </links>
</document>
```

### Example 2-92. JSON Response

```
{
  "name": "document",
  "type": "shareable_type",
  "definition": "<documentResourceUri>",
  "properties": {
    "object_name": "xyz",
    "r_object_id": "090020808001398e",
    ...
  }
  "links": [
    { "rel": "self", "href": "<documentResourceUri>" },
    ...
  ]
}
```

# Lightweight Object Materialization

Lightweight object materialization resource represents the materialization of a lightweight object.

## Feed

Is feed? No.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



## Link Relations

Link Relation	Description	Resource Reference
self	This Lightweight Object Parent resource	<a href="#">Lightweight Object Materialization, page 309</a>
The same relations as the sysobject		<a href="#">SysObject, page 472</a>

## Operations

The Lightweight Object Parent resource supports the following HTTP methods:

Method	Description
PUT	Materialize the lightweight object
DELETE	Dematerialize the lightweight object

## Materialize the Lightweight Object

Materializes an unmaterialized lightweight object.

## Supported HTTP Method

PUT

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Materialized successfully

201, 204, 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-93. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<object>
  <properties>
    <object_name>xyz</object_name>
    <r_object_id>090020808001398e</r_object_id>
    ...
  </properties>
  <links>
    <link rel="self" href="{objectResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/dematerialize" href=""/>
    <link rel="http://identifiers.emc.com/linkrel/parent" href=""/>
    ...
  </links>
</object>
```

### Example 2-94. JSON Response

```
{
  "name": "object",
  "type": "dm_sysobject",
  "definition": "",
  "properties": {
    "object_name": "xyz",
    "r_object_id": "090020808001398e",
    ...
  }
  "links": [
    { "rel": "self", "href": "objectResourceUri" },
    { "rel": "http://identifiers.emc.com/linkrel/dematerialize", "href": "" },
    { "rel": "http://identifiers.emc.com/linkrel/parent", "href": "" },
    ...
  ]
}
```

## Dematerialize a Lightweight Object

Dematerialize a materialized lightweight object.

### Supported HTTP Method

DELETE

### Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

## Response Status

204 - Dematerialize successful

404 - Resource not found

400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

N/A

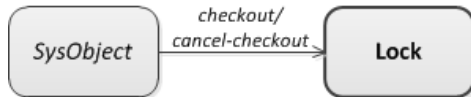


# Lock

The Lock resource represents a lock of the SysObject.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [SysObject, page 472](#) and [About the Diagram](#).

## Link Relations

The Lock resource is used as a link relation in the SysObject resource. See [SysObject, page 472](#).

## Operations

The Lock resource supports the following HTTP methods.

Method	Description
PUT	Checks out a SysObject.
DELETE	Cancels a check-out operation on a SysObject.

## Check out (lock) an object

Check out a SysObject.

### Supported HTTP Method

PUT

### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Object checked out successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied. For example, the object is an instance of `dm_folder` or `dm_cabinet` or their subtypes.  
404 - Object not found  
409 - Conflict. For example, the object has already been checked out.  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the object. A successful response contains all properties of the checked out object.

## Cancel the check-out operation (unlock) on an object.

Cancel the check-out operation on a SysObject.

## Supported HTTP Method

DELETE

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Accept
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

204 - Unlock the object successfully

400 - Invalid syntax or missing a required value

401 - Invalid or missing authentication credentials

403 - Permission denied. For example, the object is an instance of `dm_folder` or `dm_cabinet` or their subtype.

404 - Object not found

409 - Conflict. For example, the object has not been checked out.

500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful unlock operation. The Response body contains no content.

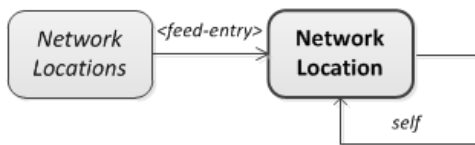
# Network Location(s)

## Network Location

The Network Location resource represents a network location (an instance of `dm_network_location_map`) stored in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Network Locations, page 320](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Network Location resource.

Link Relation	Description	Resource Reference
self	This network location resource.	<a href="#">Network Location, page 317</a>

## Operations

The Network Location resource supports the following HTTP method.

Method	Description
GET	Retrieves the metadata of a network location (an instance of <code>dm_network_location_map</code> ) in the repository.

## Get a Network Location

Retrieves the metadata of a network location (an instance of `dm_network_location_map`) specified by the identifier.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Network location not found
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the network location. The Response contains properties of a network location, depending on the setting of the `view` query parameter. By default, all properties are returned.

**Note:** The network location in the URL is encoded.

## Network Locations

The Network Locations collection resource represents the collection of network locations (instances of `dm_network_location_map`) in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
Feed URI	List of network locations	Server's current time	No

Entry	Entry ID	Entry Title	Entry Updated
<a href="#">Network Location, page 317</a>	Network location URI	Display name of the network location.	<code>r_modify_date</code> of the network location (the <code>dm_network_location_map</code> instance).

## Link Relations

The following table lists link relations for the Network locations collection resource.

Link Relation	Description	Resource Reference
self	This collection of network locations	<a href="#">Network Locations, page 320</a>
first, last, next, previous	Pagination links.	<a href="#">Network Locations, page 320</a>

## Operations

The Network locations collection resource supports the following HTTP method.



Method	Description
GET	Retrieves the metadata of a collection of network locations (instances of <code>dm_network_location_map</code> ) in the repository.

## Get Network Locations

Retrieves the metadata of a collection of network locations (instances of `dm_network_location_map`) in the repository.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- `sort`
- `view`
- `inline`
- `page`
- `items-per-page`
- `include-total`
- `filter`
- `links`

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- `Accept`
- `Authorization`

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of network locations.

- The body contains a list of network locations.
- Each object may contain all properties of the network location, depending on the setting of the query parameter `view`.
- The returned network locations collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by `netloc_ident`.
- The network location in the URL is encoded.

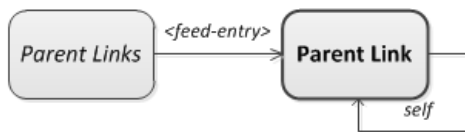
# Parent Link(s)

## Parent Link

The Parent Link resource represents the folder containment relationships between an object and its parent folder.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Parent Links, page 328](#) and [About the Diagram](#).

## Link Relations

The following table lists general link relations in the Parent Link resource.

Link Relation	Description	Resource Reference
self	This parent link.	<a href="#">Parent Link, page 323</a>

## Operations

The Parent Link resource supports the following HTTP methods.

Method	Description
GET	Retrieves the folder link between the child object and its parent folder.
PUT	Moves an object from a source folder into a destination folder.
DELETE	Unlinks an object from one of its parent folders.

## Retrieve a folder link

Retrieve the folder link between the child object and its parent folder.

## Supported HTTP Method

GET

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

400 - Invalid syntax or missing a required value

- 401 - Invalid or missing authentication credentials
- 403 - Permission denied.
- 404 - Object not found
- 409 - Conflict. For example, the object has already been checked out.
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the folder link between the object and its parent folder. The body contains the URI for the object and the URI for its parent folder. The body also contains the parent ID and child ID.

## Move an Object from one Folder to another

Move an object from a source folder to a destination folder.

### Note:

- If the object is a folder, this operation also moves the objects under the folder to the destination folder.
- Cabinet is not allowed to be moved.

## Supported HTTP Method

PUT

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

The following parameter is specific to this operation:

Variable	Description	Data type	Default value
<i>include-all-versions</i>	Specifies whether or not to move all versions of the object. <ul style="list-style-type: none"><li>• <code>true</code> - Return all versions.</li><li>• <code>false</code> - Only return the current version.</li></ul>	boolean	false

## Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of a reference to the object's new parent folder.

### Example 2-95. Example of the Request

```
PUT http://localhost:8080/emc-rest/repositories/myrepo/objects/0900000c
80000cc4/parent-links/0b0004d280002732
{
  "href":
  "http://localhost:8080/emc-rest/repositories/myrepo/folders/0b0004d28000ad36"
}
```

**Note:** The `href` property is the URI of the new destination folder.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied. For example, the object is an instance of `dm_folder` or `dm_cabinet` or their subtype.
- 404 - Object not found
- 409 - Conflict. For example, the object has been checked out.
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the folder link between the object and the destination folder. The Response contains the URI of the new destination folder.

## Unlink an Object from a Parent Folder

Unlink an object from one of its parent folders.

### Note:

- To unlink an object from a primary link, you must have at least the Write permission and the Change Location permission on the object.
- To unlink an object from a secondary link, you must have at least the Browse permission and the Change Location permission on the object.
- If the repository is running under folder security, you must also have at least the Write permission on the folder or cabinet from which the object is being unlinked.
- Documents and folders must have at least one link to a folder or cabinet. Therefore, you cannot unlink an object from its only linked folder or cabinet.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Unlink the object successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 404 - Object not found
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful unlink operation. The Response body contains no content.

# Parent Links

The Parent Links collection resource represents a collection of all folder containment relationships for an object. That is a set of links between an object and its parent folders.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [SysObject](#), [page 472](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
URI of the feed	Parent links	r_modify_date of the object	Yes



Entry	Entry ID	Entry Title	Feed Updated
<a href="#">Parent Link, page 323</a>	URI of the folder link	Formatted string with both the child object ID and the parent object ID	r_modify_date of the child object.

## Link Relations

The following table lists general link relations in the Parent Links resource.

Link Relation	Description	Resource Reference
self	URI for Parent Links collection feed	<a href="#">Parent Links, page 328</a>
first, last, next, previous	Pagination links	<a href="#">Parent Links, page 328</a>

## Operations

The Parent Links resource supports the following HTTP methods.

Method	Description
GET	Retrieves a collection of parent links for an object.
POST	Links an object to a new parent folder.

## Get Parent Links

Retrieve all parent links for a given object in a repository.

**Note:** For a cabinet, the parent links collection is empty because a cabinet does not have a parent folder.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- inline
- page
- items-per-page
- include-total

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of a collection parent links for the given object.

- The body contains a list of links between the given object and each of its parent folders.
- Each link contains the URI for the given object and the URI for one of its parent folders.
- Each entry contains the parent ID and the child ID.
- Each entry in the collection has a `self` link referring to the link resource.
- Pagination is supported.

## Link an Object to a New Folder

Link an object to a new folder or cabinet.

The first execution of `link` on an object defines the object's primary link, which is the place where the object is stored in the repository. Subsequent executions of `link` associate the object to other folders or cabinets. These links are called secondary links. This operation only allows you to create a new secondary link for an object. The existing folder links for this object do not change. For more information about how to create primary link during the object creation, see [Folder Child Objects, page 248](#).

To create a secondary link, you have at least the Browse permission and the Change Location permission on the object. If the repository is running under folder security, you must also have at least the Write permission on the folder or cabinet to which the object is linked.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

N/A

## Request Headers

- `Accept`
- `Authorization`

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of a reference to the object's new parent folder.

### Example 2-96. Example of the Request body

```
{
  "href":
    "http://localhost:8080/dctm-rest/repositories/myrepo/folders/0b0004d280002732"
}
```

## Response Headers

- Location
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

201 - Link created successfully  
400 - Bad request; invalid property name or value  
401 - Authentication failed  
403 - Permission denied  
404 - No object found  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of newly-created link between the given object and its new parent folder.

# Permissions

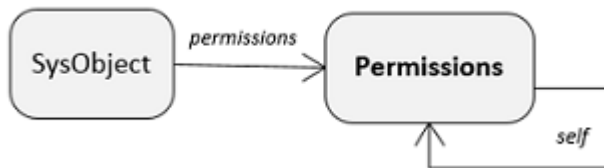
The permissions resource allows a specified user to get basic and extended permissions for a specific accessor on a sysobject. When no accessor is specified, the default accessor is the user that is currently logged into the system.

## Feed

Is feed? No.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



## Link Relations

The following table lists link relations for the Permissions resource.

Link Relation	Description	Resource Reference
self	This permissions resource	<a href="#">Permissions, page 333</a>

## Operations

Method	Description
GET	Get a specified accessor's basic and extended permissions on a certain object

## Get the Permission View Instance

Get the computed effective basic and extended permissions on a specified sysobject for a specified accessor. Permission attributes are returned as embedded elements in the Response message body.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

Variable	Description	Data Type	Value Range	Default Value
links	For more information, see <a href="#">Appendix B, REST Common Definition - URI Request Query Parameters</a>			
accessor	<p>A user name or group name to get permissions against for the sysobject.</p> <p>This parameter is not mandatory.</p> <p>The Server returns the current login user's permissions by default.</p> <p>The Server returns the system default permissions on this object when an unknown accessor (user) is provided by the client.</p>	string	N/A	The user that is currently logged in to the system

## Request Headers

- Authorization
- Accept
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#)

## Request Body

N/A

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#)

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Body

### Example 2-97. XML Request and Response

```
GET http://localhost:8080/dctm-rest/repositories/
    REPO/objects/0900000580002507/permissions?accessor=dave
Authorization: Basic ZG1hZG1pbjpwYXNzd29yZA==
Accept: application/vnd.emc.documentum+xml
```

Status Code 200 OK

Content-Type: application/vnd.emc.documentum+xml

```
<?xml version='1.0' encoding='UTF-8'?>
<permission
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" accessor="dave"
  basic-permission="Relate" extend-permissions="EXECUTE_PROC,CHANGE_LOCATION">
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
      objects/0900000580002507/permissions?accessor=dave"/>
  </links>
</permission>
```

**Example 2-98. JSON Request and Response**

```
GET http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002507/permissions?accessor=dave
Authorization: Basic ZG1hZG1pbjpwYXNzd29yZA==
Accept: application/vnd.emc.documentum+json
```

Status Code 200 OK

Content-Type: application/vnd.emc.documentum+json

```
{
  "accessor": "dave",
  "basic-permission": "Relate",
  "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION",
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580002507/permissions?accessor=dave"
    }
  ]
}
```



# Permission Set

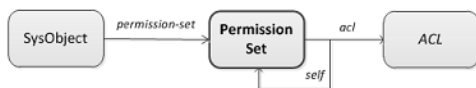
The permission set resource represents the permission set instance of a specified sysobject. Object level permission management, such as granting permissions or revoking permissions, is done using this resource's operations.

A permission set contains the following four sections that manage the access control of an object by using various aspects:

- **permitted:** This access level defines a set of users who are permitted to access the object.
- **restricted:** This access level defines a set of users who have restricted access to the object.
- **required group:** This access level requires that the user requesting access to the object must be a member of the group identified in the entry. If more than one group is defined in this section, the user requesting access to the object must be a member of all the groups defined.
- **required group set:** This access level requires that the user requesting access to the object be a member of at least one group in the set of groups defined.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [SysObject](#), page 472 and [About the Diagram](#).

## Feed

Is feed? No.

## Link Relation

The following table lists link relations for the Permission Set resource.

Link Relation	Description	Resource Reference
self	This Permission Set resource.	<a href="#">Permission Set</a> , page 337
edit	Edit this Permission Set resource.	<a href="#">Permission Set</a> , page 337
<a href="http://identifiers.emc.com/linkrel/acl">http://identifiers.emc.com/linkrel/acl</a>	Retrieve the ACL object associated to this permission set object.	<a href="#">ACL</a> , page 60

## Operations

The Permission Set resource supports the following HTTP methods.

Method	Description
GET	Get a permission set
PUT	Update a permission set

### Get the Permission Set Instance

Get the definition of the permission set of a specific object. The permission set returned may contain one or more sections of the complete permission set, which includes the `permitted`, `restricted`, `required-group`, and `required-group-set` sections. Sections that do not appear in the permission set returned are not set by that permission set.

#### HTTP Method

GET

#### Request Media Types

N/A

#### Request Query Parameters

This method supports the following Request query parameter:

- `links`

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

#### Request Headers

- `Accept`
- `Authorization`

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

#### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Permission set retrieved successfully  
 400 - Bad request  
 401 - Authentication failed  
 403 - Permission denied  
 404 - Permission set or SysObject not found  
 406 - Not Acceptable  
 409 - State conflicted  
 415 - Unsupported media type  
 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-99. XML Representation

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+xml

```
<?xml version='1.0' encoding='UTF-8'?>
<permission-set
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <permitted>
    <permission accessor="dm_world" basic-permission="Read"
  extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    <permission accessor="dm_owner" basic-permission="Delete"
  extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    .....
  </permitted>
  <restricted>
    <permission accessor="docu" basic-permission="Version"/>
    .....
  </restricted>
</required-group>
```

```
        <group>{group1}</group>
        <group>{group2}</group>
        .....
    </required-group>
    <required-group-set>
        <group>{group3}</group>
        <group>{group4}</group>
        .....
    </required-group-set>
    <links>
        <link rel="self" href="{permissionSetResourceUri}"/>
        <link rel="edit" href="{permissionSetResourceUri}"/>
        <link rel="http://identifiers.emc.com/linkrel/acl"
href="{aclResourceUri}"/>
    </links>
</permission-set>
```

**Example 2-100. JSON Representation**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+json

```
{
  "permitted": [
    {
      "accessor": "dm_world",
      "basic-permission": "Read",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
    },
    {
      "accessor": "dm_owner",
      "basic-permission": "Delete",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
    },
    .....
  ],
  "restricted": [
    {
      "accessor": "docu",
      "basic-permission": "Version"
    },
    .....
  ],
  "required-group": [
    "{group1}",
    "{group2}"
    .....
  ],
  "required-group-set": [
    "{group3}",
    "{group4}"
    .....
  ],
  "links": [
    {
      "rel": "self",
      "href": "{permissionSetResourceUri}"
    },
    {
      "rel": "edit",
      "href": "{permissionSetResourceUri}"
    },
  ],
}
```

```

    {
      "rel": "http://identifiers.emc.com/linkrel/acl",
      "href": "{aclResourceUri}"
    }
  ]
}

```

## Update the Permission Set Instance

The update operation on a permission set object enables you to do the following:

- grant or revoke a specific access level, or edit the existing access level for a user requesting access to the object (corresponding to add/delete/edit a user permission in the `permitted` attribute)
- add or remove a specific restricted level or edit the existing restricted level for a certain user that is requesting access to the object (corresponding to add/delete/edit a user permission in the `restricted` attribute)
- add or remove a membership constraint to or from the user requesting access to the object (corresponding to add/delete a group in `required-group` or `required-group-set` attribute)

When the object permission set is updated, a newly created ACL is assigned to the specified object.

Anyone who has `BROWSE` basic permissions and `CHANGE_PERMIT` extended permissions can update the basic permission. However, only the owner of the sysobject and the superuser can change basic and extended permissions. This means that because the PUT method must be used to update a permission set, only the owner of the sysobject and the superuser have sufficient permissions to update a permission set.

There are two built-in accessors: `dm_world` and `dm_owner`. Regardless of whether or not the REST client has specified any access level for `dm_world` or `dm_owner`, Content Server automatically grants users default permissions for the permission set.

The default basic and extended permission levels for these two users are defined in the `default_acl` property of the server configuration object.

When updating a permission set, you must provide a complete permission set object in the Request body, rather than specifying only the part you want to update. When any property is null or empty, the property's value will be reset to null.

A permission set object may contain a list of user permissions. A user permission is made up of an accessor, a basic-permission, extend-permissions, and application-permission. The following table shows permission set and user permission object attributes:

Attribute	Description	Valid Value	Default Value	Required
accessor	A user or group name	Any valid user or group name		Yes  This is mandatory for a user permission

Attribute	Description	Valid Value	Default Value	Required
basic-permission	The basic permissions for an accessor	One of the following: <ul style="list-style-type: none"> <li>• None</li> <li>• Browse</li> <li>• Read</li> <li>• Relate</li> <li>• Version</li> <li>• Write</li> <li>• Delete</li> </ul>	None	No
extend-permissions	The extended permissions for an accessor	One or more of the following: <ul style="list-style-type: none"> <li>• EXECUTE_PROC</li> <li>• CHANGE_LOCATION</li> <li>• CHANGE_PERMIT</li> <li>• CHANGE_STATE</li> <li>• DELETE_OBJECT</li> <li>• CHANGE_FOLDER_LINKS</li> </ul>	EXECUTE_PROC CHANGE_LOCATION	No
application-permission	Specifies a permission that is recognized by an application. An application permit is not recognized or enforced by Content Server. The application is responsible for	String	NULL	No

Attribute	Description	Valid Value	Default Value	Required
	enforcing the permission			
permitted	A list of granted user permissions	User permissions	dm_world and dm_owner user permissions	No
restricted	A list of restricted user permissions	User permissions	NULL	No
required-group	A list of group names	Any valid group name	NULL	No
required-group-set	A list of group names	Any valid group name	NULL	No

## HTTP Method

PUT

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-101. XML Request

```
<permission-set
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <permitted>
    <permission accessor="dm_world" basic-permission="Read"
  extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    <permission accessor="dm_owner" basic-permission="Write"
  extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    .....
  </permitted>
  <restricted>
    ...
  </restricted>
  <required-group>
    ...
  </required-group>
  <required-group-set>
    ...
  </required-group-set>
</permission-set>
```

### Example 2-102. JSON Request

```
{
  "permitted": [
    {
      "accessor": "dm_world",
      "basic-permission": "Read",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
    },
    {
      "accessor": "dm_owner",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION",
      "basic-permission": "Write"
    },
    .....
  ],
  "restricted": [
    .....
  ],
  "required-group": [
    .....
  ],
  "required-group-set": [
    .....
  ]
}
```



## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml  
application/xml
- application/vnd.emc.documentum+json  
application/json

## Response Status

- 200 - Permission set updated successfully
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Permission set not found
- 406 - Not Acceptable
- 409 - State conflicted
- 415 - Unsupported media type
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-103. XML Response

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+xml

```
<permission-set
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <permitted>
    <permission accessor="dm_world" basic-permission="Read"
      extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    <permission accessor="dm_owner" basic-permission="Write"
      extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
    .....
  </permitted>
  <restricted>
    ...
  </restricted>
  <required-group>
```

```
        ...
    </required-group>
    <required-group-set>
        ...
    </required-group-set>
    <links>
        <link rel="self" href="{permissionSetResourceUri}"/>
        <link rel="edit" href="{permissionSetResourceUri}"/>
        <link rel="http://identifiers.emc.com/linkrel/acl"
            href="{aclResourceUri}"/>
    </links>
</permission-set>
```

**Example 2-104. JSON Representation**

Status Code: 200 OK

Content-Type: application/vnd.emc.documentum+json

```
{
    "permitted": [
        {
            "accessor": "dm_world",
            "basic-permission": "Read",
            "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
        },
        {
            "accessor": "dm_owner",
            "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION",
            "basic-permission": "Write"
        },
        .....
    ],
    "restricted": [
        .....
    ],
    "required-group": [
        .....
    ],
    "required-group-set": [
        .....
    ],
    "links": [
        {
            "rel": "self",
            "href": "{permissionSetResourceUri}"
        },
        {
            "rel": "edit",
            "href": "{permissionSetResourceUri}"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/acl",
            "href": "{aclResourceUri}"
        }
    ]
}
```

# User Permission Set

The user permission set resource represents the associated permission set of a user.

Content Server uses the `default_acl` property of the server configuration object to specify which ACL the server uses as the default ACL. When creating a new object, if an ACL is not explicitly associated with the object you can use the following values:

- 1: The ACL associated with the object's primary folder
- 2: The ACL associated with the object's type
- 3: The ACL associated with the user who created the object
- 4: No default ACL is specified

The default is 3

When the `default_acl` property of the server configuration object has a value of 3, the creator's default permissions (or ACL) are used as the default ACL of the new object. This is the case when an ACL is not explicitly associated with the object.

When you have the associated permission set of a user, you can view the default permission set of all the sysobjects that were created by that user. You can also see any change of that user's permission set to determine how that change may effect the default permission set of any sysobjects that are created by that user in the future.

## Feed

Is feed? No.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [User](#), page 495 and [About the Diagram](#).

## Link Relation

The following table lists link relations for the User Permission Set resource.

Link Relation	Description	Resource Reference
self	This user permission set resource	<a href="#">User Permission Set, page 347</a>
<a href="http://identifiers.emc.com/linkrel/acl">http://identifiers.emc.com/linkrel/acl</a>	The current user's associated ACL object	<a href="#">ACL, page 60</a>

## Operations

The User Permission Set resource supports the following HTTP method.

Method	Description
GET	Get the permission set for a specific user

### Get the User Permission Set

Retrieve the definition of the associated permission set for a specific user. The user permission set has the same representation as the object's permission set. The permission set returned may contain one or more sections of the complete permission set, which includes the `permitted`, `restricted`, `required-group`, and `required-group-set` sections. Sections that do not appear in the permission set returned are not set by that permission set.

### HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

Parameter	Description	Data Type	Default Value
links	For more information, see <a href="#">Appendix B, REST Common Definition - URI Request Query Parameters</a> .		

## Request Headers

- Accept
- Authorization
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied  
404 - User or permission set object not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-105. XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<permission-set xmlns="http://identifiers.emc.com/vocab/documentum">
  <permitted>
    <permission accessor="dm_world" basic-permission="Read">
```

```
        extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
      <permission accessor="dm_owner" basic-permission="Delete"
        extend-permissions="EXECUTE_PROC,CHANGE_LOCATION"/>
        .....
    </permitted>
  <links>
    <link rel="self" href="{userPermissionSetResourceUri}"/>
    <link rel="http://identifiers.emc.com/linkrel/acl"
      href="{aclResourceUri}"/>
  </links>
</permission-set>
```

**Example 2-106. JSON Response**

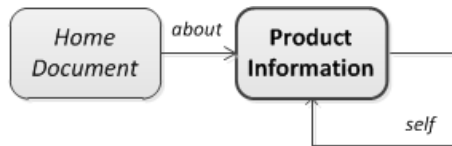
```
{
  "permitted": [
    {
      "accessor": "dm_world",
      "basic-permission": "Read",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
    },
    {
      "accessor": "dm_owner",
      "basic-permission": "Delete",
      "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
    },
    .....
  ],
  "links": [
    {
      "rel": "self",
      "href": "{userPermissionSetResourceUri}"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/acl",
      "href": "{aclResourceUri}"
    }
  ]
}
```

# Product Information

The Product Information resource provides general information about the product, including the product name, major version, minor version, and so on.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Home Document](#), page 293 and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Product Information resource.

Link Relation	Description	Resource Reference
self	Production information	<a href="#">Product Information</a> , page 351

## Operations

The Product Information resource supports the following HTTP method.

Method	Description
GET	Retrieves product information.

## Get Product Information

Retrieve product information about Documentum Platform REST Services, including product name, product version information. No authentication is required to perform this operation.

### Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept

**Note:** The Authorization header is not needed when a client tries to get this resource.

## Request Body

N/A

## Response Headers

- Content-Type
- Content-Length

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
415 - Unsupported Media Type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of product information.



**Example 2-107. XML Response**

```
<product-info name="documentum-rest-services-product-info">
  <properties>
    <product>Documentum Platform REST Services</product>
    <product_version>7.3.xxx.yyy</product_version>
    <major>7.3</major>
    <minor>xxx</minor>
    <build_number>yyy</build_number>
    <revision_number>zzz</revision_number>
  </properties>
  <links>
    <link rel="self" href=
      "http://core-rs-demo.lss.emc.com:8080/dctm-rest/product-info" />
  </links>
</product-info>
```

**Example 2-108. JSON Response**

```
{
  "name": "documentum-rest-services-product-info",
  "properties": {
    "product": "Documentum Platform REST Services",
    "product_version": "7.3.xxx.yyy",
    "major": "7.3",
    "minor": "xxx",
    "build_number": "yyy",
    "revision_number": "zzz"
  },
  "links": [
    {
      "rel": "self",
      "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/product-info.json"
    }
  ]
}
```

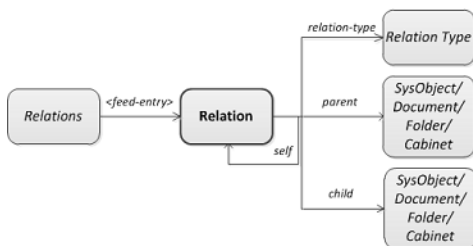
# Relation(s)

## Relation

The Relation resource represents a single relation instance in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Relations, page 357](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Relation resource.

Link Relation	Description	Resource Reference
self	This Relation resource	<a href="#">Relation, page 354</a>
relation-type [1]	Relation type definition	<a href="#">Relation Type, page 363</a>
parent	Reference to the parent object of this instance	<a href="#">SysObject, page 472</a> <a href="#">Document, page 195</a> <a href="#">Folder, page 209</a> <a href="#">Cabinet, page 120</a>
child	Reference to the child object of this instance	<a href="#">SysObject, page 472</a> <a href="#">Document, page 195</a> <a href="#">Folder, page 209</a> <a href="#">Cabinet, page 120</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Relation resource supports the following HTTP methods.

Method	Description
GET	Retrieves a specified relation.
DELETE	Deletes a specified relation.

### Get a Relation

Retrieve a specified relation instance.

### Supported HTTP Methods

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameter:

- view

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
401 - Authentication failed  
500 - Other unexpected server error

## Response Body

An XML or JSON representation of the relation.

## Delete a Relation

Delete a specified relation in the repository.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Delete successfully
- 401 - Authentication failed
- 404 - Resource not found

## Response Body

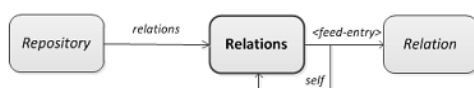
HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

# Relations

The Relations resource represents a collection of relations in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository](#), page 369 and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Entry	Supports POST?
Feed URI	List of relations	<a href="#">Relation, page 354</a>	Yes

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the relation	Relation name	Relation description	r_modify_date of the relation

## Link Relations

The following table lists link relations for the Relations resource.

Link Relation	Description	Resource Reference
self	This collection of relations.	<a href="#">Relations, page 357</a>
first, last, next, previous	Pagination links.	<a href="#">Relations, page 357</a>

## Operations

The Relations resource supports the following HTTP methods.

Method	Description
GET	Retrieves the information about a collection of relations.
POST	Create a new relation instance between two objects.

### Get Relations

Retrieve the information about a collection of relations.

#### Supported HTTP Method

GET

#### Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- inline
- view
- items-per-page
- page
- include-total
- sort
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

Also, this method supports the following relation-related query parameters:

Variable	Description	Data type	Default value
relation-name	Only relations whose name equals to the specified string are returned in the result.  Relation names are case sensitive.	string	null
related-object-id	Only relations that contain an object whose object ID equals to the specified string are returned in the result.	string	null
related-object-role	Indicates the role of the object specified by the object ID. (Only works when the <i>related-object-id</i> parameter is specified.)  Valid values are: <ul style="list-style-type: none"><li>• child</li><li>• parent</li><li>• any</li></ul>	string	any

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

400 - Bad request

401 - Authentication failed

## Response Body

XML or JSON representation of the collection of relations.

- The body contains a list of the `dm_relation` instances (or subtypes of `dm_relation`).
- Each object may contain all or a set of properties of the relation, depending on the setting of the query parameter `view`.
- Pagination is supported.
- By default, the results are listed in alphabetical order by relation name.



## Create a Relation

Create a relation between two objects in a repository.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the relation to create.

### XML representation

```
<?xml version="1.0" encoding="UTF-8"?>
<dm:relation xsi:type="dm_relation"
xmlns:dm="http://identifiers.emc.com/documentum"
xmldn:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dm:properties>
    <dm:relation_name>...</dm:relation_name>
    <dm:parent_id>...</dm:parent_id>
    <dm:child_id>...</dm:child_id>
    <dm:permanent_link>...</dm:permanent_link>
    <dm:order_no>...</dm:order_no>
  </dm:properties>
</dm:relation>
```

### JSON representation

```
{
  "type": "dm_relation",
  "properties":
```

```
{
  "relation_name": "...",
  "parent_id": "...",
  "child_id": "...",
  "permanent_link": ...,
  "order_no": ...
}
```

**Note:** To create a sub type of the relation instance, specify the `xsi:type` property in XML or the `type` property in JSON.

## Response Headers

- Content-Type
- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Relation created successfully  
400 - Bad request  
401 - Authentication failed  
403 - Permission denied

## Response Body

XML or JSON representation of the relations.

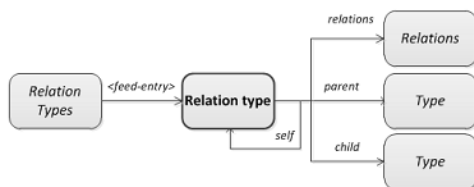
# Relation Type(s)

## Relation Type

The Relation Type resource represents a single instance of a relation type (`dm_relation_type` and its subtypes) in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Relation Types, page 366](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Relation Type resource.

Link Relation	Description	Resource Reference
self	This Relation resource.	<a href="#">Relation Type, page 363</a>
relations [1]	Relations that use this relation type	<a href="#">Relations, page 357</a>
[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:  <a href="http://identifiers.emc.com/linkrel/">http://identifiers.emc.com/linkrel/</a>		

## Operations

The Relation Type resource supports the following HTTP method.

Method	Description
GET	Retrieves a specified Relation Type resource.

## Get a Relation Type

Retrieve a specified Relation Type resource.

## Supported HTTP Method

GET

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

This method supports the following common query parameter:

- view

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

- application/xml
- application/json

**Response Status**

200 - Retrieved successfully  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
403 - Permission denied  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported Media Type  
500 - Other unexpected server error

**Response Body**

XML or JSON representation of the relation type.

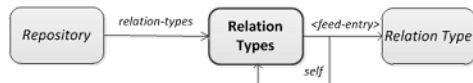
**Note:** The relation type name in the URL is encoded.

## Relation Types

The Relation Types resource represents a collection of relation type (`dm_relation_type` and its subtypes) instances in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the relation types	Relation Types	Server's current time	<a href="#">Relation Type, page 363</a>	Yes

Entry ID	Entry Summary	Entry Title	Entry Updated
URI of the relation type	Relation type description	Relation type name	<code>r_modify_date</code> of the relation type

## Link Relations

The following table lists link relations for the Relation Types resource.

Link Relation	Description	Resource Reference
self	This Relation Types resource.	<a href="#">Relation Types, page 366</a>
first, last, next, previous	Pagination links	<a href="#">Relation Types, page 366</a>

## Operations

The Relation Types resource supports the following HTTP method.

Method	Description
GET	Retrieves a collection of Relation Type instances.

## Get Relation Types

Retrieve a collection of Relation Type instances.

## Supported HTTP Method

GET

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

This method supports the following common query parameters:

- view
- links
- inline
- sort
- filter
- page
- items-per-page
- include-total

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid parameter value was provided  
401 - Authentication failed  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported Media Type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the relation types.

**Note:** The type name in the URL is encoded.

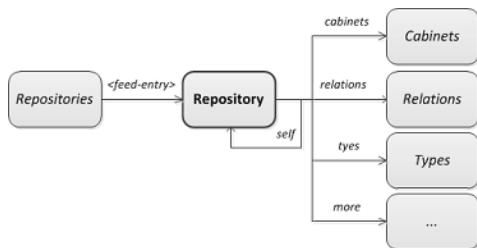


# Repository

The Repository resource provides information about a single repository. This resource requires authentication.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repositories, page 374](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations in the Repository resource.

Link Relation	Description	Resource Reference
self	Link to this repository.	<a href="#">Repository, page 369</a>
cabinets [1]	Link to the cabinets collection.	<a href="#">Cabinets, page 127</a>
relations [1]	Link to the Relations resource.	<a href="#">Relations, page 357</a>
relation-types [1]	Link to the relation types in this repository.	<a href="#">Relation Types, page 366</a>
users [1]	Link to the users in this repository.	<a href="#">Users, page 503</a>
current-user [1]	Link to the current login user of the repository.	<a href="#">Current User, page 174</a>
groups [1]	Link to the groups in this repository.	<a href="#">Groups, page 280</a>
formats [1]	Link to the formats in this repository.	<a href="#">Formats, page 269</a>
types [1]	Link to the Types resource	<a href="#">Types, page 488</a>
network-locations [1]	Link to the network locations of this repository.	<a href="#">Network Locations, page 320</a>

Link Relation	Description	Resource Reference
checked-out-objects [1]	Link to the Checked Out Objects resource	<a href="#">Checked Out Objects, page 133</a>
dql [1]	Link to the dql resource in this repository	<a href="#">DQL, page 201</a>
search [1]	Link to the search resource	<a href="#">Search, page 377</a>
aspect-types [1]	Link to aspect types resource	<a href="#">Aspect Types, page 89</a>
saved-searches [1]	Link to the saved-searches resource	<a href="#">Saved Searches, page 411</a>
search-templates [1]	Link to search templates resource	<a href="#">Saved Search Templates, page 402</a>
acls [1]	Link to ACLs in this repository resource	<a href="#">ACLs Collection, page 71</a>
current-user-preferences [1]	Link to the Current User Preferences resource	<a href="#">Current User Preferences, page 185</a>

[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string:

<http://identifiers.emc.com/linkrel/>

## Operations

The Repository resource supports the following HTTP method.

Method	Description
GET	Retrieves properties of the Repository resource.

## Get a Repository

Retrieve information about a single repository.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Response Status

200 - Retrieved successfully  
401 - Authentication failed  
404 - Repository not found  
406 - Not Acceptable  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the repository.

### Example 2-109. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<repository xmlns="http://identifiers.emc.com/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>1234</id>
  <name>acme01</name>
```

```

<server>
  <name>acme01</name>
  <host>CS70RC2_Main</host>
  <version xml:space="preserve">7.0.0000.0501 Win64.SQLServer</version>
  <docbroker>CS70RC2_Main</docbroker>
</server>
<links>
  <link rel="self" href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/
    repositories/acme01"/>
  <link rel="http://identifiers.emc.com/linkrel/users"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
      acme01/users"/>
  <link rel="http://identifiers.emc.com/linkrel/current-user"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
      acme01/currentuser"/>
  <link rel="http://identifiers.emc.com/linkrel/groups"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
      acme01/groups"/>
  <link rel="http://identifiers.emc.com/linkrel/cabinets"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      cabinets"/>
  <link rel="http://identifiers.emc.com/linkrel/formats"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      formats"/>
  <link rel="http://identifiers.emc.com/linkrel/network-locations"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      network-locations"/>
  <link rel="http://identifiers.emc.com/linkrel/relations"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      relations"/>
  <link rel="http://identifiers.emc.com/linkrel/relation-types"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      relation-types"/>
  <link rel="http://identifiers.emc.com/linkrel/checked-out-objects"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      checked-out-objects"/>
  <link rel="http://identifiers.emc.com/linkrel/types"
    href="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
      types"/>
  <link rel="http://identifiers.emc.com/linkrel/dql"
    hreftemplate="http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
      acme01{?dql,page,items-per-page}"/>
</links>
</repository>

```

### Example 2-110. JSON Response

```

{
  "id":1234,
  "name":"acme01",
  "servers":[
    {
      "name":"acme01",
      "host":"CS70RC2_Main",
      "version":"7.0.0000.0501 Win64.SQLServer",
      "docbroker":"CS70RC2_Main"
    }
  ],
  "links":[
    {
      "rel":"self",
      "href":"http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01"
    },
    {

```

```

        "rel": "http://identifiers.emc.com/linkrel/users",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            users"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/current-user",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            currentuser"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/groups",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            groups"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/cabinets",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            cabinets"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/formats",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            formats"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/network-locations",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            network-locations"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/relations",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            relations"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/relation-types",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            relation-types"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/checked-out-objects",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            checked-out-objects"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/types",
        "href": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/acme01/
            types"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/dql",
        "hreftemplate": "http://core-rs-demo.lss.emc.com:8080/dctm-rest/repositories/
            acme01/{?dql,page,items-per-page}"
    }
}
]
}

```

# Repositories

The Repositories resource represents a collection of available repositories (including those that are not running).

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Home Document, page 293](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the repositories without the file extension	List of repositories	Server's current time	<a href="#">Repository, page 369</a>	No

Entry ID	Entry Title	Entry Summary
URI of the repository without the file extension	Repository name	Repository description

## Link Relations

The following table lists general link relations in the Repositories resource.

Link Relation	Description	Resource Reference
self	Link to this Repositories resource.	<a href="#">Repositories, page 374</a>

## Operations

The Repositories resource supports the following HTTP method.

Method	Description
GET	Retrieves a collection of the Repository resources.

## Get Repositories

Retrieve a list of repositories with each repository's properties embedded in the corresponding entry.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- inline
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept

**Note:** The Authorization header is not needed when a client tries to get this resource.

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json (For compatible viewing)

## Response Status

200 - Retrieved successfully

500 - Other unexpected server error

## Response Body

XML or JSON representation of the repository.



## Search

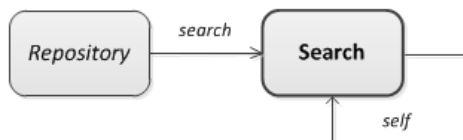
The Search resource enables you to search for objects in a repository according to the criteria that you specify. The search results can be navigated using facets.

For more information about facet searches, see the Facet Search section in the *EMC Documentum Platform REST Services Version 7.3 Development Guide*.

**Note:** The Documentum REST Search Service leverages the Documentum Foundation Classes (DFC) to perform search operations on the Content Server. The property `rest.search.dfc.request.timeout` in the `rest-api-runtime.properties` configuration file specifies a timeout (in milliseconds) for the execution of a DFC search request. A value of 0 indicates that there is no timeout for search request.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the Search resource	Search	Current time	A <a href="#">SysObject, page 472</a> or a subtype	Yes

## Link Relations

The following table lists general link relations in the Search resource.

Link Relation	Description	Resource Reference
self	The URI for Search results sysobjects feed	<a href="#">Search, page 377</a>
first, last, next, previous	Pagination links	Used for navigation when more than one page of search results are available
search	The URI template or actual URI for subsequent searches (for example: facet selection)	<a href="#">Search, page 377</a>

## Operations

The Search resource supports the following HTTP methods.

Method	Description
GET	Full-text search using URI parameters.
POST	Full-text search using a query document and URI parameters.

### Full-text Search Using URI Parameters

This operation enables you to perform a full-text search using simple search language along with URI parameters. The operation also supports basic facet navigation.

#### Supported HTTP Methods

GET

A GET request to perform a full-text search with URI parameters looks like the following:

```
http://localhost:8080/dctm-rest/repositories/acme01/search?q=search_criterion
&facet=property_name&facet-value-constraints=constraints
```

#### Request Media Types

N/A

#### Request Query Parameters

This operation supports the following URI query parameters:

- `include-total`

The `include-total` parameter in the Search service returns the total number of hit results.

- `inline`
- `items-per-page`
- `page`
- `sort`

The sort order is determined by the data type of the attribute. For example, when `r_content_size` is used with an attribute value of 123,3555 to sort search results, the data type is integer and REST sorts the search results by integer.

- `view`

**Note:**

- Pagination parameters (page, items-per-page, and include-total) only take effect on full text search results. As a result, pagination link relations do not appear when you navigate to the result set of a certain facet by using the `search` link relation in a `facet` section.

A facet is not calculated against the current page, but is instead calculated against the entire result set. For example, if there are 25 results returned for a keyword, the parameter `items-per-page` may have a value of 2. In this case, the Response contains only 2 results, even though the facets were calculated against 25 results.

- The `inline` parameter must be `true`, or the `view` parameter has no effect.
- Setting the `view` parameter to `:default` and to `:all` behave the same. Both settings return the following properties of an object:
  - `r_object_id`
  - `object_name`
  - `r_object_type`
  - `r_modify_date`
  - `r_modifier`
  - `summary`

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

The following query parameters are specific to this operation:

Variable	Description	Data type	Default value
q	<p>Specifies the search criterion with a full-text expression in simple search language.</p> <p>Parameter <code>q</code> must be encoded because it may contain non-English locale characters.</p> <p><b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.</p>	String	No search criteria is used for the search.

Variable	Description	Data type	Default value
locations	<p>Specifies a list of folder paths separated by comma (,) to which the query is scoped.</p> <p>Parameter locations must be encoded because it may contain non-English locale characters.</p> <p><b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.</p>	String	No location scope constraints are used for the search.
collections	<p>Specifies a list of collections separated by comma (,). to which the query is scoped. The collections must exist in xPlore.</p>	String	No collection constraints are used for the search.
facet	Specifies the property to be used as facet	String	No property is used for the search.
facet-value-constraints	<p>Specifies a property constraint expression. For example, when facet is set to <code>r_object_type</code> and facet-value-constraints is set to <code>dm_object</code>, only <code>dm_object</code> instances are returned.</p> <p>You can use boolean operators <code>+</code> for AND and <code> </code> for OR in a property constraint expression to define more complex constraints. For example, the expression <code>dm_object dm</code></p>	String	No property constraints are used for the search.

Variable	Description	Data type	Default value
	<p><code>_folder</code> returns both <code>dm_object</code> and <code>dm_folder</code> instances.</p> <p>Parameter facet -value-constraints must be encoded because it may contain non-English locale characters.</p> <p><b>Note:</b> International characters that are used in this query parameter must be sent with URL encoded by the UTF-8 charset. Otherwise, the result may be incorrect.</p>		
timezone	<p>Indicates the time zone used to compute date facets.</p> <p>Valid time zones include (case-insensitive):</p> <ul style="list-style-type: none"> <li>• Abbreviation, such as GMT and UTC</li> <li>• Full names, such as America/Los_Angeles</li> <li>• Custom time zones, such as GMT-8:00</li> </ul>	String	GMT
object-type	Specifies an object type. Only instances of the specified type or its sub types are returned in the search result	String	<code>dm_sysobject</code>

## Request Headers

- Accept
- Content-Type
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type
- Content-Length

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/json
- application/xml

## Response Status

200 - Retrieved successfully  
400 - Bad request; invalid attribute name or value  
401 - Authentication failed  
403 - Permission denied  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the search results with facets.

- The body must contain a list of sysobjects (or subtypes of `dm_sysobject`), each of which may or may not contain all attributes of a sysobject, depending on the setting of the `view` and `inline` query parameters.
- The returned sysobjects collection only contains objects for which the user has access.
- Pagination is supported.
- Default sort order is the ranking returned by the search engine and it can be overridden with any sysobject attribute.
- Total count can be returned only when it is explicitly specified in the `Request` parameter, along with it the link for last page can be returned.

**Example 2-111. XML Feed Response**

```

<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">
  ...
  <!--The URI of the GET request sent to the REST server-->
  <link rel="self"
    href="http://localhost:8080/dctm-rest/repositories/acme01/search?
    q=test&facet=r_object_type"/>
  <!--A URI template that lists all available query parameters you can use-->
  <link rel="search"
    hreftemplate="http://localhost:8080/dctm-rest/repositories/acme01/search
    {?q,collections,locations,facet,inline,page,items-per-page,include-total,
    sort,view,facet-value-constraints}/>
  <!--Search results are listed in the following entries-->
  <entry>
    ...
  </entry>
  ...
  ...
  <entry>
    ...
  </entry>
  <!--Facet information-->
  <dm:facet>
    <!--The property used as the facet-->
    <dm:facet-id>r_object_type</dm:facet-id>
    <dm:facet-label>Type</dm:facet-label>
    <!--Data of the results belonging to this fact-->
    <dm:facet-value>
      <dm:facet-value-id>san_biz</dm:facet-value-id>
      <dm:facet-value-count>13</dm:facet-value-count>
      <dm:facet-id>r_object_type</dm:facet-id>
      <dm:facet-value-constraint>san_biz</dm:facet-value-constraint>
      <!--A link that navigate you to the corresponding results of this fact-->
      <link rel="search"
        href="http://localhost:8080/dctm-rest/repositories/acme01/search?
        q=test&facet=r_object_type&fact-vale-constraints=san_biz"/>
    </dm:facet-value>
    <dm:facet-value>
      <dm:facet-value-id>dm_document</dm:facet-value-id>
      <dm:facet-value-count>12</dm:facet-value-count>
      <dm:facet-id>r_object_type</dm:facet-id>
      <dm:facet-value-constraint>dm_document</dm:facet-value-constraint>
      <link rel="search"
        href="http://localhost:8080/dctm-rest/repositories/acme01/search?
        q=test&facet=r_object_type&fact-vale-constraints=dm_document"/>
    </dm:facet-value>
  </dm:facet>
</feed>

```

**Example 2-112. JSON Response**

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/acme01/search",
  "updated": "2014-04-30T14:27:49.643+08:00",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],

```

```
"page": 1,
"items-per-page": 3,
"links": [
  {
    "rel": "self",
    "href": "http://localhost:8080/dctm-rest/repositories/acme01/search.json?q=test
      &items-per-page=3&inline=false&include_total=true"
  },
  {
    "rel": "next",
    "href": "http://localhost:8080/dctm-rest/repositories/acme01/search.json?q=test
      &items-per-page=3&inline=false&include_total=true&page=2"
  },
  {
    "rel": "first",
    "href": "http://localhost:8080/dctm-rest/repositories/acme01/search.json?q=test
      &items-per-page=3&inline=false&include_total=true&page=1"
  },
  {
    "rel": "search",
    "hreftemplate": "http://localhost:8080/dctm-rest/repositories/acme01/
      search.json{?q,collections,locations,facet,inline,page,items-per-page,
      include-total,sort,view}"
  }
],
"entries": [
  {
    "id": "090004d280005117",
    "title": "Native UCF Functional Test Plan.doc",
    "updated": "2013-05-07T15:56:45.000+08:00",
    "summary": "Native UCF Functional Test Plan\t\t09/13/2010 EMC | Documentum...
    Project Native UCF Functional Test Plan Document.../Updated some tests Functional
    Test Plan Template Version: 1.2... _Toc408805185 _Toc262725323 _Toc272245663 1.
    Test Plan Identifier",
    "author": [
      {
        "name": "dmadmin"
      }
    ],
    "content": {
      "content-type": "application/json",
      "src": ""
    },
    "links": [
      {
        "rel": "self",
        "href": "http://localhost:8080/dctm-rest/repositories/acme01/objects/
          090004d280005117.json"
      }
    ],
    "score": "1.0",
    "terms": [
      "Test",
      "Tested",
      "tests",
      "test",
      "testing",
      "Testing",
      "tested"
    ]
  },
  {
    "id": "090004d280009f72",
    "title": "TESTED",
    "updated": "2013-09-10T15:33:43.000+08:00",
```



```

    "author": [
      {
        "name": "dmadmin"
      }
    ],
    "content": {
      "content-type": "application/json",
      "src": ""
    },
    "links": [
      {
        "rel": "self",
        "href": "http://localhost:8080/dctm-rest/repositories/acme01/objects/090004d280009f72.json"
      }
    ],
    "score": "0.4935832917690277",
    "terms": [
      "test"
    ]
  },
  {
    "id": "090004d2800072ae",
    "title": "AAA BACKFLOW TESTING LLC",
    "updated": "2013-09-10T15:25:15.000+08:00",
    "author": [
      {
        "name": "dmadmin"
      }
    ],
    "content": {
      "content-type": "application/json",
      "src": ""
    },
    "links": [
      {
        "rel": "self",
        "href": "http://localhost:8080/dctm-rest/repositories/acme01/objects/090004d2800072ae.json"
      }
    ],
    "score": "0.48992398381233215",
    "terms": [
      "test"
    ]
  }
]
}

```

## Full-text Search Using Query Document and URI Parameters

Retrieves a list of sysobjects or sysobject subtypes that match the search criteria, which means that you can perform a search on metadata.

### Supported HTTP Methods

POST

### Request Media Types

- application/vnd.emc.documentum+json
- application/vnd.emc.documentum+xml

### Request Query Parameters

The following URI parameters are supported:

- include-total
- page
- items-per-page
- inline
- facet-id-constraints

Contains the key value pairs for each facet id and its constraint. The id should comply with the facet id in the AQL.

For example:

- You can use `id1` as the facet id for the `r_object_type` property
- You can use `id2` as the facet id for the `owner_name` property

`facet-id-constraints=id1=dm_document,id2=Administrator`

**Note:** Using any other Search URI parameters not shown above with AQL causes an error message to be shown in the Documentum Platform REST Services system.

### Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the query document.

### Example 2-113. XML Request

```
POST http://localhost:8080/dctm-rest/repositories/acme01/search.xml
Content-Type: application/vnd.emc.documentum
```

```
<search>
<types>
<type>dm_sysobject</type>
</types>
<expression-set>
<fulltext fuzzy="true">dmadmin</fulltext>
</expression-set>
<columns>
<column>object_name</column>
<column>r_object_id</column>
</columns>
<facets>
<facet id="r_modify_date">
<attributes><attribute>r_modify_date</attribute></attributes>
</facet>
</facets>
</search>
```

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Headers

- Content-Type
- Content-Length

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request; invalid attribute name or value
- 401 - Authentication failed
- 403 - Permission denied
- 404 - No object found
- 500 - Other unexpected server error

## Response Body

Same response body as for the Get method: see [Response Body, page 382](#).

### Example 2-114. XML Request

POST <http://localhost:8080/emc-rest/repositories/acme01/search>  
Content-Type: application/vnd.emc.documentum+xml

```
<search all-versions="true" max-results-for-facets="10"
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <types>
    <type>dm_document</type>
  </types>
  <columns>
    <column>r_object_type</column>
    <column>object_name</column>
  </columns>
  <sorts>
    <sort ascii="false" ascending="true" lang="fr">r_content_size</sort>
    <sort ascii="true" ascending="false" lang="zh">object_name</sort>
  </sorts>
  <collections>
    <collection>collection1</collection>
    <collection>collection2</collection>
  </collections>
  <locations>
    <id-location descendent="true">
      <id>0c00000580001914</id>
    </id-location>
    <path-location descendent="false">
      <path>/dmadmin</path>
    </path-location>
  </locations>
  <facet-definitions>
    <facet-definition id="id1" group-by="alpharange" max-values="8">
      <attributes>
        <attribute>r_object_type</attribute>
      </attributes>
      <sort>FREQUENCY</sort>
      <properties>
        <property name="range">a:e, f:z</property>
      </properties>
    </facet-definition>
    <facet-definition id="id2" group-by="relativeDate" max-values="9">
      <attributes>
        <attribute>r_modify_date</attribute>
      </attributes>
      <sort>VALUE_ASCENDING</sort>
    </facet-definition>
  </facet-definitions>
  <expression-set>
    <expressions>
      <property name="object_name" operator="CONTAINS" exact-match="true" repeating="true"
        case-sensitive="false" fuzzy="true">rest
      </property>
      <fulltext fuzzy="true">rest</fulltext>
      <fulltext fuzzy="false">test</fulltext>
      <property-range name="r_content_size" operator="BETWEEN" repeating="false">
        <from>0</from>
        <to>10000</to>
      </property-range>
      <property-list name="r_object_type" operator="IN" repeating="false">
```

```

        <values>
          <value>dm_document</value>
          <value>dm_folder</value>
        </values>
      </property-list>
    <expression-set>
      <expressions>
        <fulltext fuzzy="false">test</fulltext>
      </expressions>
    </expression-set>
    <relative-date name="r_modify_date"
time-unit="MONTH" operator="GREATER_THAN" >2</relative-date>
  </expressions>
</expression-set>
</search>

```

### Example 2-115. JSON Request

POST http://localhost:8080/emc-rest/repositories/acme01/search  
Content-Type: application/vnd.emc.documentum+json

```

{
  "types": ["dm_document"],
  "columns": [
    "r_object_type",
    "object_name"
  ],
  "collections": [
    "collection1",
    "collection2"
  ],
  "locations": [
    {
      "location-type": "id-location",
      "id": "0c00000580001914",
      "descendent": true
    },
    {
      "location-type": "path-location",
      "path": "/dmadmin",
      "descendent": false
    }
  ],
  "sorts": [
    {
      "property": "r_content_size",
      "ascending": true,
      "lang": "fr",
      "ascii": false
    },
    {
      "property": "object_name",
      "ascending": false,
      "lang": "zh",
      "ascii": true
    }
  ],
  "facet-definitions": [
    {
      "id": "id1",
      "attributes": ["r_object_type"],
      "group-by": "alpharange",
      "sort": "FREQUENCY",
      "properties": {

```

```
        "range": "a:e, f:z",
      },
      "max-values": 8
    },
    {
      "id": "id2",
      "attributes": ["r_modify_date"],
      "group-by": "r_modify_date",
      "sort": "VALUE_ASCENDING",
      "max-values": 9
    }
  ],
  "expression-set": {
    "expression-type": "expression-set",
    "expressions": [
      {
        "expression-type": "property",
        "name": "object_name",
        "operator": "CONTAINS",
        "value": "rest",
        "exact-match": true,
        "repeated": true,
        "case-sensitive": false,
        "fuzzy": true
      },
      {
        "expression-type": "fulltext",
        "value": "rest",
        "fuzzy": true
      },
      {
        "expression-type": "fulltext",
        "value": "test",
        "fuzzy": false
      },
      {
        "expression-type": "property-range",
        "name": "r_content_size",
        "operator": "BETWEEN",
        "from": "0",
        "to": "10000",
        "repeating": false
      },
      {
        "expression-type": "property-list",
        "name": "r_object_type",
        "operator": "IN",
        "values": [
          "dm_document",
          "dm_folder"
        ],
        "repeating": false
      },
      {
        "expression-type": "expression-set",
        "expressions": [
          {
            "expression-type": "fulltext",
            "value": "test",
            "fuzzy": false
          }
        ]
      }
    ]
  },
  {
    {
```

```

        "expression-type": "relative-date",
        "name": "r_modify_date",
        "value": 2,
        "time-unit": "MONTH",
        "operator": "EQUAL"
    }
}
}
}

```

### Example 2-116. Response Headers

Status Code: 200 OK  
 Content-Type: application/vnd.emc.documentum+json;charset=UTF-8  
 Date: Tue, 19 Feb 2013 10:15:21 GMTServer: Apache-Coyote/1.1  
 Transfer-Encoding: chunked

### Example 2-117. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/acme01/search",
  "updated": "2014-04-30T14:27:49.643+08:00",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "page": 1,
  "items-per-page": 3,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?items-per-page=3&inline=false&include_total=true"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?items-per-page=3&inline=false&include_total=true&page=2"
    },
    {
      "rel": "first",
      "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?items-per-page=3&inline=false&include_total=true&page=1"
    }
  ],
  "entries": [
    ...
    ...
  ]
}

```

## Search with Facets

Faceted searching allows you to view a collection of search results information by applying one or more filters. Faceted searching, also known as Faceted Navigation, uses a classification system to divide information into dimensions, called facets, allowing the classifications to be used in many ways.

For more information, see [Faceted Search](#).

REST services supports faceted navigation in two ways:

- URI parameter (facet) in GET request
- Abstract Query Language (AQL) in the POST body

### Facets Using Query Parameters

A URI facet parameter can be used to specify one attribute as the facet. By default, the attributes of SysObject that are facet-enabled depends on the xPlore configuration. The configuration for xPlore must be modified and re-indexed to facet new attributes or add additional facet values to existing attributes.

Attribute	DFC Data Type	Default Grouped By
r_modifier	DF_STRING	string
keywords	DF_STRING	string
r_modify_date	DF_TIME	relativeDate
r_full_content_size	DF_DOUBLE	string
a_application_type	DF_STRING	string
r_object_type	DF_STRING	string
owner_name	DF_STRING	string

The strategy used to group facets depends on the data type of the attribute. Additional grouping strategies can be achieved using AQL language in a POST statement.

For more information, see [Facet Using Query Document](#), page 394.

Another URI parameter called *facet-value-constraints* is used to define an expression that outlines the constraints that are used in the parameter facet. This parameter allows you to navigate a result set under the constraints that your expression defines for the attribute in the parameter facet. Facets used in a feed are returned with the full text search results. You can view the facets in the following response sample:

```
"facets": [
  {
    "facet-id": "facet_r_object_type",
    "facet-value": [
      {
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/REPO/
search?facet=r_object_type
&facet-value-constraints=dm_document",
          "rel": "search"
        }
      }
    ]
  }
]
```



```

        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "dm_document",
        "facet-value-count": 116,
        "facet-value-id": "dm_document"
    },
    {
        "link": {
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search?facet=r_object_type&facet-value-constraints=dm_folder",
            "rel": "search"
        },
        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "dm_folder",
        "facet-value-count": 68,
        "facet-value-id": "dm_folder"
    },
    {
        "link": {
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search?facet=r_object_type&facet-value-constraints=dm_cabinet",
            "rel": "search"
        },
        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "dm_cabinet",
        "facet-value-count": 25,
        "facet-value-id": "dm_cabinet"
    },
    {
        "link": {
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search?facet=r_object_type&facet-value-constraints=dm_sysobject",
            "rel": "search"
        },
        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "dm_sysobject",
        "facet-value-count": 12,
        "facet-value-id": "dm_sysobject"
    },
    {
        "link": {
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search?facet=r_object_type&facet-value-constraints=rest_core",
            "rel": "search"
        },
        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "rest_core",
        "facet-value-count": 8,
        "facet-value-id": "rest_core"
    },
    {
        "link": {
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search?facet=r_object_type&facet-value-constraints=dm_network_location_map",
            "rel": "search"
        },
        "facet-id": "facet_r_object_type",
        "facet-value-constraint": "dm_network_location_map",
        "facet-value-count": 2,
        "facet-value-id": "dm_network_location_map"
    }
    ],
    "facet-label": "Type"
}
]

```

## Facet Using Query Document

Facet navigation using [AQL](#) allows you to harness the full power of facets. There are three kinds of facet requests:

- Single facet
- Multiple facet
- Nested facet

**Note:** From an implementation perspective, the facet constraints are defined by the underlying grouping strategy. You can determine facets by the grouping strategies used. The grouping strategies available for a given attribute depends on the data type of that attribute. For example, it is not meaningful to use a *day* or *relativeDate* grouping strategy for a string type attribute, such as *object\_name*.

### Grouping Strategies

Strategy	Description	Options (properties)
string		<ul style="list-style-type: none"> <li>• <code>nullValueFacet</code> (optional): if this property is set, null values will be counted as a facet. The facet will be named using the value of the property. By default, the null values are ignored when computing facets.</li> <li>• <code>keepDuplicateValues</code> (optional): true/false. Default: false. For a document, attributes with repeating values may have duplicate values. By default, duplicates are removed. If this property is set to true, duplicates entries are counted. Since: xPlore 1.1 P03 and 1.2</li> <li>• <code>caseInsensitive</code> (optional): true/false. Default: false. If this property is set to true, facet computation ignore case. For example, if a results set has a document with a value 'emc' and another 'EMC', two facets will be returned,</li> </ul>

Strategy	Description	Options (properties)
		emc(1) and EMC(1). Since: xPlore 1.1 P03 and 1.2
range	<p>Applies to numeric values.</p> <p>Range grouping requires the use of the 'range' property, which defines the ranges to use in grouping.</p> <p>Example:</p> <pre>0:1000,1000:10000,10000:</pre>	<p>range (required). This property defines a list of 'buckets' to use for grouping. Each bucket is separated by comma (','),. A bucket is defined by a lower and upper bounds separated by colon (':'). Open ended bucket can be defined by omitting one of the bounds but keeping the comma separator. Overlapping buckets are not supported and may result in inconsistent results.</p> <p>For 'range':</p> <ul style="list-style-type: none"> <li>The boundaries themselves are parsed as Double</li> </ul> <p>Example:</p> <pre>'0:10,10:100.89,100:'</pre> <ul style="list-style-type: none"> <li>The lower-bound is inclusive. The upper-bound is exclusive. For example, for a facet value corresponding to '0:1000', the constraint should be (attribute &gt;= 0 and attribute &lt;1000).</li> </ul>
alphanrange	<p>Applies to string values (since xPlore 1.3). Range grouping requires the use of the 'range' property, which defines the ranges to use for grouping.</p> <p>Example:</p> <pre>'a:m,n:r,s:z'</pre> <p>. This grouping strategy uses Unicode order, not language dependent order.</p> <p>In practice, it is only useful for ASCII characters.</p>	<p>The range that is mentioned in the range strategy above is also required for the alphanrange strategy. In addition to this, the following also apply to the alphanrange strategy:</p> <ul style="list-style-type: none"> <li>The boundaries are parsed as one character.</li> </ul> <p>Example:</p> <pre>'a:m,n:r,s:z'</pre> <ul style="list-style-type: none"> <li>This grouping strategy is case-insensitive. Special care must be taken when adding a constraint based on such</li> </ul>

Strategy	Description	Options (properties)
		<p>a facet value because the comparison operators on string (&lt;, &lt;=, &gt;=, &gt;) are case sensitive. As a consequence, one has to add 2 constraints: one on the lowercase range and one on the uppercase range.</p> <p>The upper-bound and lower-bound are returned as a property of the facet value. The boundaries are inclusive. As a consequence, the constraint on the upper-bound should add 1 to the actual character.</p> <p>Example: for a facet value corresponding to 'a-m', the constraint should be (attribute &gt;= 'a' and attribute &lt; 'n') or (attribute &gt;= 'A' and attribute &lt; 'N')</p> <p>For more information, see <a href="#">range</a> above</p>
<p>date strategy including:</p> <ul style="list-style-type: none"> <li>• day</li> <li>• week</li> <li>• month</li> <li>• quarter (since xPlore v1.3)</li> <li>• year</li> <li>• relativeDate</li> </ul>	<p>The facet values returned are the start date of the range, using the UTC time zone</p> <p>For example, for January 2011, the facet value is 2011-01-01T00:00:00 (the time is adjusted when the 'timezone' property is specified).</p>	<ul style="list-style-type: none"> <li>• <code>timezone</code> (optional): a timezone code. Default: Z (UTC). The timezone to be used to compute date grouping.</li> <li>• <code>returnUTC</code> (optional) - since xPlore 1.3: a boolean. Default: 'true' in REST. When <code>timezone</code> is specified and <code>returnUTC</code> is false, the dates in the facets are returned formatted in the specified timezone. The same timezone is used to compute the date range boundaries and to format the facets.</li> </ul> <p>When <code>returnUTC</code> is set to 'true', the date in the facet is formatted in UTC. In this</p>

Strategy	Description	Options (properties)
		<p>case, the timezone is only be used for computation, but not for formatting.</p> <ul style="list-style-type: none"> <li>• <code>skipEmptyValues</code> (optional): A Boolean value whose default is 'true'.</li> </ul> <p>When <code>skipEmptyValues</code> is set to 'true', only the facet values with a count strictly greater than zero are returned.</p> <p>When <code>skipEmptyValues</code> is set to 'false', the query returns all the facet values between the oldest non-zero facet value and the most recent non-zero facet value.</p> <p><b>Note:</b> This property only works for date type with grouping strategy day, month, year, week or quarter.</p>

### Max for Facet Group

The `max-values` property is the attribute of the facet definition that defines the greatest number of groups that each facet can return.

### Facet Ordering

- **FREQUENCY**  
Ordered by count, descending
- **NONE**  
Keep the order returned by the facet handler
- **VALUE\_ASCENDING**  
Ordered by facet value ID (ascending). The sort is alphanumeric. We assume that even if the data type of the facet is date or numeric, the values are encoded in such way that they are alphanumerically sortable.
- **VALUE\_DESCENDING**  
Ordered by facet value ID (descending). The sort is alphanumeric. We assume that even if the data type of the facet is date or numeric, the values are encoded in such way that they are alphanumerically sortable.

**Example 2-118. JSON Single Facet**

To navigate the facet results, you can POST the original request body to the link in the facet value. The following JSON sample uses a Request and a Response to demonstrate how to use a single facet:

**Request**

```
{
  ...
  ...
  "columns": ["owner_name"],
  "facet-definitions": [
    {
      "id": "id1",
      "attributes": ["r_object_type"],
      "sort": "FREQUENCY"
    }
  ]
}
```

**Response**

In the Response, the URL in the facet value has the *facet-id-constraints* parameter. This parameter specifies the facet constraint `id1=dm_folder`:

```
"facets": [
  {
    "facet-id": "id1",
    "facet-label": "Type",
    "facet-value": [
      {
        "facet-id": "id1",
        "facet-value-constraint": "dm_document",
        "facet-value-count": 116,
        "facet-value-id": "dm_document",
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?facet-id-constraints=id1%3Ddm_document",
          "rel": "search"
        }
      },
      {
        "facet-id": "id1",
        "facet-value-constraint": "dm_folder",
        "facet-value-count": 68,
        "facet-value-id": "dm_folder",
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?facet-id-constraints=id1%3Ddm_folder",
          "rel": "search"
        }
      }
    ]
  }
]
```

**Example 2-119. XML Multiple Facets**

This code sample shows how to use two separate facets against `r_object_type` and `owner_name`. The facets results have two facet aggregations respectively.

**XML Request**

```
<search all-versions="true">
  <facet-definitions>
    <facet-definition id="id1" group-by="alpharange" max-values="8">
      <attributes>
```

```

    <attribute>r_object_type</attribute>
  </attributes>
  <properties>
    <property name="range">a:r, s:z</property>
  </properties>
  <sort>FREQUENCY</sort>
</facet-definition>
<facet-definition>
  <attributes>
    <attribute>owner_name</attribute>
  </attributes>
</facet-definition>
</facet-definitions>
</search>

```

## JSON Response

```

"facets": [
  {
    "facet-id": "id1",
    "facet-label": "Type",
    "facet-value": [
      {
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?facet-id-constraints=id1%3Da:r",
          "rel": "search"
        },
        "facet-id": "id1",
        "facet-value-constraint": "a:r",
        "facet-value-count": 116,
        "facet-value-id": "a:r"
      },
      {
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?&facet-id-constraints=id1%3Ds:z",
          "rel": "search"
        },
        "facet-id": "id1",
        "facet-value-constraint": "s:z",
        "facet-value-count": 68,
        "facet-value-id": "s:z"
      }
    ]
  },
  {
    "facet-id": "facet_owner_name",
    "facet-label": "Owner Name",
    "facet-value": [
      {
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?facet-id-constraints=facet_owner_name%3Ddave",
          "rel": "search"
        },
        "facet-id": "facet_owner_name",
        "facet-value-constraint": "dave",
        "facet-value-count": 126,
        "facet-value-id": "dave"
      },
      {
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution?facet-id-constraints=facet_owner_name%3Ddadmin",

```

```

        "rel": "search"
      },
      "facet-id": "facet_owner_name",
      "facet-value-constraint": "dmadmin",
      "facet-value-count": 138,
      "facet-value-id": "dmadmin"
    }
  ]
}
]

```

### Example 2-120. JSON Nested Facet

This code sample shows a nested facet definition, where `r_object_type` points to `owner_name` in a tree like structure. For each `r_object_type`, the grouping by `owner_name` goes a step further. Also, the facet values are hierarchical. The first nested facet value has constraint `id1=dm_document` and `id2=dmadmin`:

### JSON Request

```

{
  "columns":["owner_name"],
  "facet-definitions": [
    {
      "id": "id1",
      "attributes": ["r_object_type"],
      "sort": "FREQUENCY",
      "facet-definition": {
        "id": "id2",
        "attributes": ["owner_name"],
        "sort": "FREQUENCY"
      }
    }
  ]
}

```

### JSON Response

```

{"facets": [
  {
    "facet-id": "id1",
    "facet-label": "Type",
    "facet-value": [
      {
        "facet-value-constraint": "dm_document",
        "facet-value-count": 116,
        "facet-value-id": "dm_document",
        "facet-id": "id1",
        "link": {
          "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution&facet-id-constraints=id1%3Ddm_document",
          "rel": "search"
        },
        "facet-value": [
          {
            "link": {
              "href": "http://localhost:8080/dctm-rest/repositories/acme01/searches/0c10304585007313/execution&facet-id-constraints=id1%3Ddm_document,id2%3Ddave",
              "rel": "search"
            },
            "facet-id": "id2",
            "facet-value-constraint": "dave",
            "facet-value-count": 86,
            "facet-value-id": "dave"
          }
        ]
      }
    ]
  }
]

```



$$] \}$$

# Saved Search(es)

## Saved Search

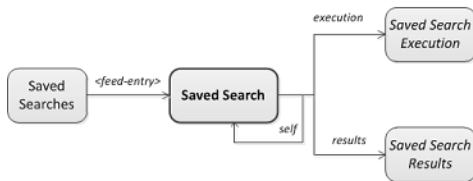
You can define search criteria and save the search. After your search has been saved, you can load, edit the search criteria, or execute the saved search and view the saved results of the search.

## Feed

Is feed? No.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [About the Diagram](#).

For more information, see [Saved Searches](#), page 411.

## Link Relations

Link Relation	Description	Resource Reference
self	Save the search	<a href="#">Saved Search</a> , page 402
edit	Edit the saved search	<a href="#">Saved Search</a> , page 402
<a href="http://identifiers.emc.com/linkrel/delete">http://identifiers.emc.com/linkrel/delete</a>	Delete the saved search	<a href="#">Saved Search</a> , page 402
<a href="http://identifiers.emc.com/linkrel/search-execution">http://identifiers.emc.com/linkrel/search-execution</a>	Execute the saved search	<a href="#">Saved Search Execution</a> , page 420
<a href="http://identifiers.emc.com/linkrel/saved-search-results">http://identifiers.emc.com/linkrel/saved-search-results</a>	View or work with the saved search results	<a href="#">Saved Search Results</a> , page 426
<a href="http://identifiers.emc.com/linkrel/as-search-template">http://identifiers.emc.com/linkrel/as-search-template</a>	Save the saved search as a template	<a href="#">Saved Search</a> , page 402

## Supported HTTP Methods

Method	Description
GET	Retrieves saved search
POST	Updates the saved search
DELETE	Removes the saved search

## Operations

### Get a Saved Search

Gets a saved search from a repository.

#### Supported HTTP Method

GET

#### Request Media Types

N/A

#### Request Query Parameters

Variable	Description	Data Type	Value Range	Default Value
links	For more information, see <a href="#">Appendix B, REST Common Definition - URI Request Query Parameters</a> .			

#### Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

#### Request Body

N/A

#### Response Headers

N/A

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 Retrieved successfully

204, 400, 401, 403, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Body

### Example 2-121. XML Response

```
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_smart_list"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/
    dm_smart_list.xml">
  <properties>
    <object_name>search iig</object_name>
    <title>this is saved search</title>
    <creation-date>2016-03-10T01:38:44.000+00:00</creation-date>
    <r_modify_date>2016-03-10T01:38:44.000+00:00</r_modify_date>
    <r_modifier>Administrator</r_modifier>
    <owner_name>Administrator</owner_name>
    <r_is_public>true</r_is_public>
    <selected_sources>
      <item>REPO</item>
    </selected_sources>
    <has_results>>false</has_results>
    <results_count>-1</results_count>
    <query_type>query_builder</query_type>
    <r_object_id>080000018000157f</r_object_id>
  </properties>
  <query-document>&lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;search
    xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false"
    include-hidden-objects="false"&gt;&lt;repositories&gt;&lt;repository
      &gt;REPO&lt;/repository&gt;&lt;/repositories&gt;&lt;types&gt;&lt;type
      &gt;dm_document&lt;/type&gt;&lt;/types&gt;
      &lt;expression-set operator="AND"&gt;&lt;expressions&gt;
        &lt;fulltext fuzzy="true"&gt;iig&lt;/fulltext&gt;&lt;/expressions&gt;
        &lt;/expression-set&gt;&lt;columns&gt;&lt;column&gt;r_object_type
          &lt;/column&gt;&lt;column&gt;object_name&lt;/column&gt;
          &lt;column&gt;summary&lt;/column&gt;&lt;/columns&gt;&lt;/search&gt;
    </query-document>
  </links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
      saved-searches/080000018000157f.xml"/>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
      saved-searches/080000018000157f.xml"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
```

```

    href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    080000018000157f.xml"/>
    <link rel="http://identifiers.emc.com/linkrel/saved-search-results"
    href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    080000018000157f/results.xml"/>
    <link rel="http://identifiers.emc.com/linkrel/search-execution"
    href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    080000018000157f/execution.xml"/>
    <link rel="http://identifiers.emc.com/linkrel/as-search-template"
    href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates.xml"/>
  </links>
</saved-search>

```

### Example 2-122. JSON Response

```

{
  "name": "saved-search",
  "type": "dm_smart_list",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_smart_list",
  "properties": {
    "object_name": "search iig",
    "title": "this is saved search",
    "creation-date": "2016-03-10T01:38:44.000+00:00",
    "r_modify_date": "2016-03-10T01:38:44.000+00:00",
    "r_modifier": "Administrator",
    "owner_name": "Administrator",
    "r_is_public": true,
    "selected_sources": [
      "REPO"
    ],
    "has_results": false,
    "results_count": -1,
    "query_type": "query_builder",
    "r_object_id": "080000018000157f"
  },
  "query-document": "{ \"all-versions\":false, \"include-hidden-objects\":false,
  \"repositories\":[\"REPO\"], \"types\":[\"dm_document\"], \"expression-set\":
  { \"expression-type\": \"expression-set\", \"operator\": \"AND\", \"expressions\":
  [{ \"expression-type\": \"fulltext\", \"value\": \"iig\", \"fuzzy\":true}] },
  \"columns\":[\"r_object_type\", \"object_name\", \"summary\"]",
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      080000018000157f"
    },
    {
      "rel": "edit",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      080000018000157f"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/delete",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      080000018000157f"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/saved-search-results",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      080000018000157f/results"
    },
    {
      "rel": "http://identifiers.emc.com/linkrel/search-execution",

```

```
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/080000018000157f/execution"
      },
      {
        "rel": "http://identifiers.emc.com/linkrel/as-search-template",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates"
      }
    ]
  }
}
```

## Update a Saved Search

You can use this resource to update a saved search. You can update the `object_name`, `title`, and `r_is_public` attributes. You can also update a `query-document` that contains an AQL. However, the AQL must be escaped.

For more information about the Abstract Query Language, see [AQL](#).

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

## Request Headers

- Authorization, Accept, Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers..](#)

## Request Body

Here is a code sample that shows you a saved search, which has properties and an embedded query document in AQL. As you can see, the AQL statements in the `<query-document>` element must all be escaped as shown below.

### Example 2-123. XML Request

```
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <properties>
    <object_name>dave public</object_name>
```

```

        <title>this is saved search</title>
        <r_is_public>true</r_is_public>
    </properties>
    <query-document>
    <search xmlns="http://identifiers.emc.com/vocab/documentum"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <types>
    <type>dm_sysobject</type>
    </types>
        <expression-set>
            <expressions>
                <property-list name="owner_name" operator="IN">
                    <values>
                        <value>dmadmin</value>
                    </values>
                </property-list>
            </expressions>
        </expression-set>
    <facet-definitions>
        <facet-definition id="id-1">
            <attributes>
                <attribute>r_object_type</attribute>
            </attributes>
        </facet-definition>
    </facet-definitions>
    </search>
    </query-document>
</saved-search>

```

#### Example 2-124. JSON Request

```

{
  "properties": {
    "object_name": "dave public",
    "title": "this is saved search",
    "r_is_public": true,
  },
  "query-document": "
  {
    \"types\": [\"dm_document\"],
    \"expression-set\": {
      \"expression-type\": \"expression-set\",
      \"expressions\": [
        {
          \"expression-type\": \"property\",
          \"name\": \"myprop\",
          \"operator\": \"CONTAINS\",
          \"value\": \"myvalue\"
        }
      ]
    }
  }
  "
}

```

#### Response Status

- 200 - OK
- 400 - Bad request; invalid attribute name or value
- 401 - Authentication failed
- 403 - Forbidden, unauthorized to perform the Request

- 404 - No object found
- 406 Not acceptable, invalid media type in request header
- 409 Conflict, creation or update request could not be completed
- 415 Unsupported media type
- 500 - Internal server error or unexpected server condition

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Body

For more information, see [Response Headers, page 403](#)

### Example 2-125. XML Request

```
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <properties>
    <object_name>public search</object_name>
    <title>this is saved search</title>
    <r_is_public>true</r_is_public>
  </properties>
  <query-document>
    &lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;search
      xmlns="http://identifiers.emc.com/vocab/documentum"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false"
      include-hidden-objects="false"&gt;&lt;&lt;repositories&gt;&lt;repository&gt;REPO
        &lt;/repository&gt;&lt;/repositories&gt;&lt;types&gt;&lt;type&gt;dm_document&lt;/type&gt;
        &lt;/types&gt;&lt;expression-set operator="AND"&gt;&lt;expressions&gt;&lt;
          fulltext fuzzy="true"&gt;iig&lt;/fulltext&gt;&lt;/expressions&gt;&lt;/expression-set&gt;
          &lt;columns&gt;&lt;column&gt;r_object_type&lt;/column&gt;&lt;column&gt;object_name
            &lt;/column&gt;&lt;column&gt;summary&lt;/column&gt;&lt;/columns&gt;&lt;/search&gt;
        </query-document>
  </saved-search>
```

### Example 2-126. XML Response

```
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dm_smart_list"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/
    dm_smart_list">
  <properties>
    <object_name>public search</object_name>
    <title>this is saved search</title>
    <creation-date>2016-02-03T01:47:42.000+00:00</creation-date>
    <r_modify_date>2016-02-03T01:50:01.000+00:00</r_modify_date>
    <r_modifier>dmadmin</r_modifier>
    <owner_name>dmadmin</owner_name>
    <r_is_public>true</r_is_public>
    <selected_sources>
      <item>REPO</item>
    </selected_sources>
    <has_results>>false</has_results>
    <results_count>-1</results_count>
    <query_type>query_builder</query_type>
```



```

    <r_object_id>0800000180010d42</r_object_id>
  </properties>
  <query-document>
    &lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;search
    xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false"
      include-hidden-objects="false"&gt;&lt;repositories&gt;&lt;repository&gt;
        REPO&lt;/repository&gt;&lt;/repositories&gt;&lt;types&gt;&lt;type&gt;dm_document
        &lt;/type&gt;&lt;/types&gt;&lt;expression-set operator="AND"&gt;&lt;expressions&gt;
        &lt;fulltext fuzzy="true"&gt;iig&lt;/fulltext&gt;&lt;/expressions&gt;
        &lt;/expression-set&gt;&lt;columns&gt;&lt;column&gt;r_object_type&lt;/column&gt;
        &lt;column&gt;object_name&lt;/column&gt;&lt;column&gt;summary&lt;/column&gt;
        &lt;/columns&gt;&lt;/search&gt;
    </query-document>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
      saved-searches/0800000180010d42"/>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
      saved-searches/0800000180010d42"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        saved-searches/0800000180010d42"/>
    <link rel="http://identifiers.emc.com/linkrel/saved-search-results"
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        saved-searches/0800000180010d42/results"/>
    <link rel="http://identifiers.emc.com/linkrel/search-execution"
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        saved-searches/0800000180010d42/execution"/>
    <link rel="http://identifiers.emc.com/linkrel/as-search-template"
      href="http://localhost:8080/dctm-rest/repositories/REPO/search-templates"/>
  </links>
</saved-search>

```

### Example 2-127. JSON Request

POST http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42  
 Content-Type: application/vnd.emc.documentum+json

```

{
  "properties": {
    "object_name": "public search",
    "title": "this is saved search",
    "r_is_public": true
  },
  "query-document": "{ \"all-versions\":false, \"include-hidden-objects\":false, \"
repositories\":[\"REPO\"], \"types\":[\"dm_document\"],
  \"expression-set\":{ \"expression-type\": \"expression-set\", \"operator\": \"AND\",
  \"expressions\":[{ \"expression-type\": \"fulltext\", \"value\": \"iig\",
  \"fuzzy\":true}], \"columns\":[\"r_object_type\", \"object_name\", \"summary\"]"
}

```

### Example 2-128. JSON Response

```

{
  "name": "saved-search",
  "type": "dm_smart_list",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_smart_list",
  "properties": {
    "object_name": "public search",
    "title": "this is saved search",
    "creation-date": "2016-02-03T01:47:42.000+00:00",
    "r_modify_date": "2016-02-03T01:48:53.000+00:00",
    "r_modifier": "dmadmin",
    "owner_name": "dmadmin",

```

```
    "r_is_public": true,
    "selected_sources": [
      "REPO"
    ],
    "has_results": false,
    "results_count": -1,
    "query_type": "query_builder",
    "r_object_id": "0800000180010d42"
  },
  "query-document": "{ \"all-versions\": false, \"include-hidden-objects\": false,
  \"repositories\": [ \"REPO\" ], \"types\": [ \"dm_document\" ],
  \"expression-set\": { \"expression-type\": \"expression-set\", \"operator\": \"AND\",
  \"expressions\": [ { \"expression-type\": \"fulltext\", \"value\": \"iig\",
  \"fuzzy\": true } ] }, \"columns\": [ \"r_object_type\", \"object_name\", \"summary\" ],
  \"links\": [
    {
      \"rel\": \"self\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42\"
    },
    {
      \"rel\": \"edit\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42\"
    },
    {
      \"rel\": \"http://identifiers.emc.com/linkrel/delete\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42\"
    },
    {
      \"rel\": \"http://identifiers.emc.com/linkrel/saved-search-results\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42/results\"
    },
    {
      \"rel\": \"http://identifiers.emc.com/linkrel/search-execution\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/0800000180010d42/execution\"
    },
    {
      \"rel\": \"http://identifiers.emc.com/linkrel/as-search-template\",
      \"href\": \"http://localhost:8080/dctm-rest/repositories/REPO/search-templates\"
    }
  ]
}
```

## Delete a Saved Search

You can use this resource to delete a saved search.

## Supported HTTP Methods

DELETE

## Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers..](#)

## Request Body

N/A

## Response Status

204 Delete was successful

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes.](#)

# Saved Searches

This resource retrieves saved searches that are visible to the logged in user. The visible searches include all of the user's saved searches and all shared searches that are available to the user.

## Feed

Feed id	Feed title	Feed Updated	Support Post?
<Feed URI of the Saved Searches resource without file extension>	Saved Searches	Current time	Yes

Entry id	Entry title	Entry Updated
<Entry URI of the Saved Search>	Saved Search name	Modified date

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



**See Also:** [Repository, page 369](#) and [About the Diagram.](#)

## Link Relations

Link Relation	Description	Resource Reference
self	Saved Searches resource	<a href="#">Saved Searches, page 411</a>
first, last, next, previous	Pagination links	Used for navigation when more than one page of search results are available

## Supported HTTP Methods

Method	Description
GET	Retrieves the visible saved searches
POST	Creates a saved search

## Operations

### Get a Saved Searches Collection

Gets the saved searches collection from the repository.

#### Request Method

- GET

#### Request Query Parameters

Variable	Description	Data Type	Value Range	Default Value
inline, items-per-page, total, page, sort, filter, links	For more information, see <a href="#">Appendix B, REST Common Definition - URI Request Query Parameters</a> .			

**Note:** The Saved Searches resource ignores the `view` parameter.

**Note:** In the feed, the `query-document` is not shown even when `inline=true` has been set.

#### Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500, ...
- For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Headers

- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Type

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Example

### Example 2-129. XML Request

```
GET http://localhost/dctm-rest/repositories/REPO/saved-searches?items-per-page=1
Accept: application/atom+xml
```

### Example 2-130. XML Response

```
Status Code: 200 OK
Content-Type: application/atom+xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/saved-searches</id>
  <title>Saved Searches</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-07-30T02:56:38.870+00:00</updated>
```

```

<dm:page
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
</dm:page>
<dm:items-per-page
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">100
</dm:items-per-page>
<dm:total
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
</dm:total>
<link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
  saved-searches.xml?include-total=true&inline=false&view=has_results
  &sort=object_name%20asc&filter=starts-with(object_name,%27dave%27)"/>
<entry>
  <id>http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c518000472c</id>
  <title>dave public</title>
  <author>
    <name>dave</name>
  </author>
  <summary>this is saved search</summary>
  <updated>2015-07-30T02:41:25.000+00:00</updated>
  <published>2015-07-30T02:41:25.000+00:00</published>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c518000472c.xml"/>
  <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/REPO/
    saved-searches/08024c518000472c.xml"/>
</entry>
</feed>

```

### Example 2-131. JSON Request

GET http://localhost/dctm-rest/repositories/REPO/saved-searches?items-per-page=1  
 Accept: application/vnd.emc.documentum+json

### Example 2-132. JSON Response

Status Code: 200 OK  
 Content-Type: application/vnd.emc.documentum+json;charset=UTF-8

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches",
  "title": "Saved Searches",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-07-30T02:57:34.222+00:00",
  "page": 1,
  "items-per-page": 100,
  "total": 1,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches?
        include-total=true&inline=false&view=has_results&
        sort=object_name%20asc&filter=starts-with(object_name,%27dave%27) "
    }
  ],
  "entries": [
    {
      "id": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
        08024c518000472c",
      "title": "dave public",

```

```

    "author": [
      {
        "name": "dave"
      }
    ],
    "summary": "this is saved search",
    "updated": "2015-07-30T02:41:25.000+00:00",
    "published": "2015-07-30T02:41:25.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000472c"
      }
    ],
    "content": {
      "type": "application/vnd.emc.documentum+json",
      "src": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000472c"
    }
  }
]
}

```

## Create a Saved Search

For a saved search, there are four basic attributes: name, r\_object\_name, title, r\_is\_public, and query-document. All the other attributes are populated by the system. The name and query-document parameters are mandatory to create a saved search.

## Supported HTTP Method

- POST

## Request Media Type

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

- N/A

## Request Headers

- Authorization
- Accept
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

Here is a code sample that shows you a saved search, which has properties and an embedded query document in AQL. The AQL statements in the <query-document> element must be escaped as shown below.

For more information about the Abstract Query Language, see [AQL](#).

### Example 2-133. XML Request

```
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <properties>
    <object_name>dave public</object_name>
    <title>this is saved search</title>
    <r_is_public>true</r_is_public>
  </properties>
  <query-document>

    &lt;search xmlns="http://identifiers.emc.com/vocab/documentum"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"&gt;
    &lt;types&gt;
    &lt;type&gt;dm_sysobject&lt;/type&gt;
    &lt;/types&gt;
    &lt;expression-set&gt;
      &lt;expressions&gt;
        &lt;property-list name="owner_name" operator="IN"&gt;
          &lt;values&gt;
            &lt;value&gt;dmadmin&lt;/value&gt;
          &lt;/values&gt;
        &lt;/property-list&gt;
      &lt;/expressions&gt;
    &lt;/expression-set&gt;
    &lt;facet-definitions&gt;
      &lt;facet-definition id="id-1"&gt;
        &lt;attributes&gt;
          &lt;attribute&gt;r_object_type&lt;/attribute&gt;
        &lt;/attributes&gt;
      &lt;/facet-definition&gt;
    &lt;/facet-definitions&gt;
    &lt;/search&gt;

  </query-document>
</saved-search>
```

### Example 2-134. JSON Request

```
{
  "properties":{
    "object_name": "dmadmin private",
    "title": "this is a facet search",
    "r_is_public": false
  },
  "query-document": "
  {
    \"types\": [\"dm_document\"],
    \"expression-set\": {
      \"expression-type\": \"expression-set\",
      \"expressions\": [
        {
          \"expression-type\": \"property\",
          \"name\": \"myprop\",
```



```

        \"operator\": \"CONTAINS\",
        \"value\": \"myvalue\"
      }
    ]
  }
}

```

## Response Headers

- Location (The URI of the newly created search)
- Content-Type
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Type

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 201 Created successfully
- 400, 401, 403, 404, 406, 409, 415, 500, ...

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body



### Caution:

- In AQL, there is an element named `location` that element must contain a valid path location. A search with an invalid location can be successfully created. Attempting to load a search that contains an invalid location will cause DFC to throw an exception. When this is the case, the REST layer will use the value `UNREACHABLE LOCATION PATH` for the invalid path and no search results can be returned.
- Location must include source information. If you save to a location without using any source information, the system will add a default source (current repository) as the source information. This behavior is consistent with executing a query document.

### Example 2-135. JSON Response

```

{
  "properties": {

```

```

        "object_name": "dmadmin private",
        "title": "this is a facet search",
        "creation-date": "2015-07-30T03:06:59.000+00:00",
        "r_modify_date": "2015-07-30T03:06:59.000+00:00",
        "r_modifier": "dave",
        "owner_name": "dave",
        "r_is_public": false,
        "selected_sources": [
            "REPO"
        ],
        "has_results": false,
        "results_count": -1,
        "query_type": "query_builder",
        "r_object_id": "08024c518000501f"
    },
    "query-document": "{\\"all-versions\\":false,\\"facet-definitions\\":[{\\"id\\":\\"id-1\\",
\\"attributes\\":[{\\"r_object_type\\"},\\"group-by\\":\\"string\\",\\"sort\\":\\"FREQUENCY\\",
\\"properties\\":[{\\"skipEmptyValues\\":\\"true\\"},\\"max-values\\":8}],
\\"repositories\\":[\\"REPO\\"],\\"types\\":[\\"dm_sysobject\\"],
\\"expression-set\\":{
\\"expression-type\\":\\"expression-set\\",\\"operator\\":\\"AND\\",
\\"expressions\\":[{\\"expression-type\\":\\"property-list\\",\\"name\\":\\"owner_name\\",
\\"operator\\":\\"IN\\",\\"values\\":[\\"dmadmin\\"],\\"repeating\\":false}]
}
}]",
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000501f"
        },
        {
            "rel": "edit",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000501f"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/delete",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000501f"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/saved-search-results",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000501f/results"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/search-execution",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c518000501f/execution"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/as-search-template",
            "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates"
        }
    ]
}
}

```

### Example 2-136. XML Response

```

<?xml version='1.0' encoding='UTF-8'?>
<saved-search
  xmlns="http://identifiers.emc.com/vocab/documentum"

```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<properties>
  <object_name>dave public</object_name>
  <title>this is saved search</title>
  <creation-date>2015-07-30T03:09:01.000+00:00</creation-date>
  <r_modify_date>2015-07-30T03:09:01.000+00:00</r_modify_date>
  <r_modifier>dave</r_modifier>
  <owner_name>dave</owner_name>
  <r_is_public>true</r_is_public>
  <selected_sources>
    <item>REPO</item>
  </selected_sources>
  <has_results>>false</has_results>
  <results_count>-1</results_count>
  <query_type>query_builder</query_type>
  <r_object_id>08024c5180005021</r_object_id>
</properties>
<query-document>
  &lt;?xml version='1.0' encoding='UTF-8'?&gt;
  &lt;search xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    all-versions="false"&gt;
    &lt;repositories&gt;&lt;repository&gt;REPO
    &lt;/repository&gt;&lt;/repositories&gt;&lt;types&
    &gt;&lt;type&gt;dm_document&lt;/type&gt;
    &lt;/types&gt;&lt;expression-set operator="AND"&gt;&
    &lt;expressions&gt;
    &lt;fulltext fuzzy="true"&gt;rest&lt;/fulltext&gt;&
    &lt;fulltext fuzzy="false"&gt;test
    &lt;/fulltext&gt;&lt;/expressions&gt;&
    &lt;/expression-set&gt;&lt;columns&gt;
    &lt;column&gt;r_object_type&lt;/column&gt;&
    &lt;column&gt;object_name&lt;/column&gt;
    &lt;column&gt;summary&lt;/column&gt;&lt;/columns&
    &gt;&lt;/search&gt;
  </query-document>
<links>
  <link rel="self"
href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c518000501f.xml"
></link>
  <link rel="edit"
href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c518000501f.xml"
></link>
  <link rel="http://identifiers.emc.com/linkrel/delete"
href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c518000501f.xml"
></link>
  <link rel="http://identifiers.emc.com/linkrel/saved-search-results"
href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c518000501f/results.xml"
></link>
  <link rel="http://identifiers.emc.com/linkrel/search-execution"
href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c518000501f/execution.xml"
></link>
  <link rel="http://identifiers.emc.com/linkrel/as-search-template"
href="http://localhost:8080/dctm-rest/repositories/REPO/search-templates.xml"
></link>
</links>
</saved-search>

```

## Saved Search Execution

The Saved Search Execution resource enables you to run a specified saved search and retrieve real-time results.

### Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Saved Search](#), page 402 and [About the Diagram](#).

### Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the Saved Search Execution resource	Saved Search Execution	Server's current time	Search result	No

Entry ID	Entry Name	Entry Updated
URI of a result	Result name	r_modify_date of a result

### Link Relations

The following table lists general link relations in the Saved Search Execution resource.

Link Relation	Description	Resource Reference
self	This Saved Search Execution resource.	<a href="#">Saved Search Execution</a> , page 420
first, last, next, previous	Pagination links.	Used for navigation when more than one page of search results are available

### Operations

The Saved Search Execution resource supports the following HTTP method.

Method	Description
GET	Retrieve real-time results of a saved search.

## Execute a Saved Search

Execute a saved search and retrieve real-time results.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

**facet-id-constraints:** specifies one or more pairs of facet id and its constraints in the following pattern: *facet\_id\_1%3Dconstraints\_1, facet\_id\_2%3Dconstraints\_2 ... facet\_id\_n%3Dconstraints\_n*

**Note:** International characters and symbols that are used in this query parameter must be sent URL encoded with the UTF-8 charset. Otherwise, the result may be incorrect.

Here's an example to show you how *facet-id-constraints* works. In this example, the facet query is against the attribute *r\_full\_content\_size* and the grouping strategy used is *range*. There are two ranges specified in the AQL: [0, 100000) and [10000, 20000):

```
<search xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <expression-set>
    <expressions>
      <property-list name="owner_name" operator="IN">
        <values>
          <value>dmadmin</value>
        </values>
      </property-list>
    </expressions>
  </expression-set>
  <facet-definitions>
    <facet-definition group-by="range">
      <attributes>
        <attribute>r_full_content_size</attribute>
      </attributes>
      <properties>
        <property name="range">0:10000,10000:20000</property>
      </properties>
    </facet-definition>
  </facet-definitions>
</search>
```

The search service returns the search results and the facet results. We can see that there are two groups returned for [0, 100000) and [10000, 20000) respectively. Submitting the original AQL to the URL of each group navigates to the results of group.

```
<dm:facet
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">
  <dm:facet-id>facet_r_full_content_size</dm:facet-id>
  <dm:facet-label>Full Content Size</dm:facet-label>
  <dm:facet-value>
    <dm:facet-id>facet_r_full_content_size</dm:facet-id>
    <dm:facet-value-id>0:10000</dm:facet-value-id>
    <dm:facet-value-count>36</dm:facet-value-count>
    <dm:facet-value-constraint>0/10000</dm:facet-value-constraint>
    <link rel="search" href="http://localhost:8080/dctm-rest/repositories/REPO/
      search.xml?facet-id-constraints=facet_r_full_content_size%3D0/10000"/>
  </dm:facet-value>
  <dm:facet-value>
    <dm:facet-id>facet_r_full_content_size</dm:facet-id>
    <dm:facet-value-id>10000:20000</dm:facet-value-id>
    <dm:facet-value-count>10</dm:facet-value-count>
    <dm:facet-value-constraint>10000/20000</dm:facet-value-constraint>
    <link rel="search" href="http://localhost:8080/dctm-rest/repositories/REPO/
      search.xml?facet-id-constraints=facet_r_full_content_size%3D10000/20000"/>
  </dm:facet-value>
</dm:facet>
```

This method supports the following common query parameters:

- links
- inline
- include-total
- page
- iters-per-page

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

## Request Body

N/A.

## Response Headers

- Content-Type

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Results retrieved successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied
- 404 - Saved search not found
- 406 - Not Acceptable
- 409 - State conflicted
- 415 - Unsupported media type
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the real-time results of the saved search.

- The body contains a list of search results.
- The returned result collection only contains those that the REST client has access to.
- Pagination is supported.

### Example 2-137. XML Response

```
<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/saved-searches
    /08024c5180005288/execution</id>
  <title>Search results</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-08-04T07:19:05.885+00:00</updated>
  <dm:page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:items-per-page>
```

```

<link rel="self" href="http://localhost:8080/dctm-rest/repositories/
REPO/saved-searches/08024c5180005288/execution.xml?items-per-page=1"/>
<link rel="next" href="http://localhost:8080/dctm-rest/repositories/
REPO/saved-searches/08024c5180005288/execution.xml?items-per-page=1&page=2"/>
<link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/
saved-searches/08024c5180005288/execution.xml?items-per-page=1&page=1"/>
<link rel="search" hreftemplate="http://localhost:8080/dctm-rest/
repositories/REPO/search.xml{?collections, facet, include-total, inline,
items-per-page, locations, object-type, page, q, sort, timezone, view}"/>
<entry>
  <id>09024c518000488c</id>
  <title>ADFS and Shib 0.81.docx</title>
  <author>
    <name>UnitTestUser</name>
  </author>
  <summary>Federated Access 18 Step 3: ...</summary>
  <updated>2015-08-04T06:50:23.000+00:00</updated>
  <link rel="edit"
href="http://localhost:8080/dctm-rest/repositories/REPO/objects/09024c518000488c.xml"/>
  <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/REPO
/objects/09024c518000488c.xml"/>
  <relevance:score
    xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">1.0
  </relevance:score>
  <dm:terms
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:term>Test</dm:term>
    <dm:term>test</dm:term>
    <dm:term>testing</dm:term>
    <dm:term>rest</dm:term>
    <dm:term>Testing</dm:term>
    <dm:term>tested</dm:term>
  </dm:terms>
</entry>
...
<entry>...</entry>
</feed>

```

### Example 2-138. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c5180005288/execution",
  "title": "Search results",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-08-04T07:17:59.804+00:00",
  "page": 1,
  "items-per-page": 1,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c5180005288/execution?items-per-page=1"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
08024c5180005288/execution?items-per-page=1&page=2"
    },
    {

```



```

        "rel": "first",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c5180005288/execution?items-per-page=1&page=1"
    },
    {
        "rel": "search",
        "hreftemplate": "http://localhost:8080/dctm-rest/repositories/REPO/search{?collections,facet,include-total,inline,items-per-page,locations,object-type,page,q,sort,timezone,view}"
    }
],
"entries": [
    {
        "id": "09024c518000488c",
        "title": "ADFS and Shib 0.81.docx",
        "author": [
            {
                "name": "UnitTestUser"
            }
        ],
        "summary": "Federated Access\t18 Step 3: ...",
        "updated": "2015-08-04T06:50:23.000+00:00",
        "links": [
            {
                "rel": "edit",
                "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/09024c518000488c"
            }
        ],
        "content": {
            "type": "application/vnd.emc.documentum+json",
            "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/09024c518000488c"
        },
        "score": "1.0",
        "terms": [
            "Test",
            "test",
            "testing",
            "rest",
            "Testing",
            "tested"
        ]
    }
    ...
    {...}
]
}

```

## Saved Search Results

The Saved Search Results resource allows you to take the results of a search and save it in Content Server. Saving search results enables you to retrieve search results without having to re-execute the search.

This resource returns saved results, instead of real-time results.

**Note:** Facet results cannot be saved

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



For more information, see [Saved Search, page 402](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
URI of the Saved Search Results resource	Saved Search Results	Time when the results are saved.	Search result	No

Entry ID	Entry Name	Entry Updated
URI of a result	Result name	r_modify_date of a result

## Link Relations

The following table lists general link relations in the Saved Search Results resource.

Link Relation	Description	Resource Reference
self	This Saved Search Results resource.	<a href="#">Saved Search Results, page 426</a>
first, last, next, previous	Pagination links.	Used for navigation when more than one page of search results are available

## Operations

The Saved Search Results resource supports the following HTTP methods.

Method	Description
PUT	Enable or refresh results of a saved search.
GET	Retrieve results of a saved search.
DELETE	Disable saved results.

### Enable and Refresh Saved Results

By default, a newly created Saved Search resource does not save results. This operation enables any previously disabled saved results. This operation causes the value of the `has_results` attribute of a saved search to change from `false` to `true`.

When `has_results` is already `true`, which means that results saving is already enabled, this operation can be used to refresh the saved results and prevent users from seeing any out-of-date search results.

### Supported HTTP Method

PUT

### Request Media Types

N/A

### Request Query Parameters

The following parameter is specific to this method:

Parameter	Description	Data Type	Range	Default
<code>max-results-count</code>	The maximum number of results to for a saved search	integer	<ul style="list-style-type: none"><li>-1 and any non-negative integer.</li><li>-1: Apply the DFC configuration</li><li>non-negative integer: the smaller of <code>max-results-count</code> and DFC configuration takes effect</li></ul>	1000

**Note:** When -1 is used as a value for the *max-results-count* parameter, each results saving Request may require more time to retrieve as many as possible (limited by DFC config) matching objects. Using a default value of 1000 (10 pages by default) improves performance in most cases.

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization (optional)

## Request Body

A PUT request is used to turn on saved results.

### Example 2-139. Request Minimal Payload for `application/vnd.emc.documentum+xml` or `application/xml` Media Types

```
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c5180004332/saved-results</id>
  <title>Search results</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-08-04T02:54:54.399+00:00</updated>
  <dm:page xmlns:dm="http://identifiers.emc.com/vocab/documentum">1</dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1</dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
    saved-searches/08024c5180004332/saved-results.xml?items-per-page=1"/>
  <link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/
    saved-searches/08024c5180004332/saved-results.xml?items-per-page=1&page=2"/>
  <link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/
    saved-searches/08024c5180004332/saved-results.xml?items-per-page=1&page=1"/>
  <link rel="search" hreftemplate="http://localhost:8080/dctm-rest/repositories/REPO/
    search.xml{?collections, facet, include-total, inline, items-per-page, locations,
      object-ttype, page, q, sort, timezone, view}"/>
  <entry>
    <id>09024c5180004344</id>
    <title>ADFS and Shib 0.81.docx</title>
    <summary>Federated Access 18 Step 3: Test AD FS 2.0 as the Identity... Step 5:
      Test Shibboleth as the Identity Provider and AD FS 2.0 as...
      following sections. AD FS 2.0 The test deployment that is created in the...
      domain controller and the federation server in test deployments</summary>
    <updated>2015-08-04T03:00:22.950+00:00</updated>
    <link rel="edit"
      href="http://localhost:8080/dctm-rest/repositories/REPO/
        objects/09024c5180004344.xml"/>
    <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/
      REPO/objects/09024c5180004344.xml"/>
    <relevance:score
      xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">1.0
    </relevance:score>
    <dm:terms
      xmlns:dm="http://identifiers.emc.com/vocab/documentum">
```

```

        <dm:term>Test</dm:term>
        <dm:term>test</dm:term>
        <dm:term>testing</dm:term>
        <dm:term>rest</dm:term>
        <dm:term>Testing</dm:term>
        <dm:term>tested</dm:term>
    </dm:terms>
</entry>
</feed>

```

**Example 2-140. Request Minimal Payload for application/vnd.emc.documentum+json or application/json Media Types**

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c5180004332/saved-results",
  "title": "Search results",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-08-04T02:54:54.399+00:00",
  "page": 1,
  "items-per-page": 1,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
        08024c5180004332/saved-results?items-per-page=1"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
        08024c5180004332/saved-results?items-per-page=1&page=2"
    },
    {
      "rel": "first",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
        08024c5180004332/saved-results?items-per-page=1
&page=1"
    },
    {
      "rel": "search",
      "hreftemplate": "http://localhost:8080/dctm-rest/repositories/REPO/
        search{?collections, facet, include-total, inline, items-per-page, locations, object-type, page,
    }
  ],
  "entries": [
    {
      "id": "09024c5180004344",
      "title": "ADFS and Shib 0.81.docx",
      "author": [
        null
      ],
      "summary": "Federated Access\t18 Step 3: Test AD FS 2.0 as the Identity... Step 5: Test Shibbol
      "updated": "2015-08-04T02:55:22.883+00:00",
      "links": [
        {
          "rel": "edit",
          "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/09024c5180004344"
        }
      ]
    }
  ]
}

```

```
],
"content": {
  "type": "application/vnd.emc.documentum+json",
  "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/09024c5180004344"
},
  "score": "1.0",
  "terms": [
    "Test",
    "test",
    "testing",
    "rest",
    "Testing"]
]
}
```

## Response Headers

- Content-Type

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Results retrieved successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied  
404 - Saved search not found  
406 - Not Acceptable  
409 - State conflicted  
415 - Unsupported media type  
500 - Other unexpected server error

## Response Body

XML or JSON representation of saved results of the search. See the [response body](#) of the Saved Search Execution for representation samples.

- This method sets the `has_results` property of a saved search to `true`.
- The `results_count` represents the count of saved results whether or not `max-results-count` has been reached.

## Get Saved Results

Get saved results of a search.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- links
- inline
- include-total
- page
- items-per-page

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization (optional)

## Request Body

N/A

## Response Headers

- Content-Type

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Results retrieved successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied
- 404 - Saved results not enabled
- 406 - Not Acceptable
- 409 - State conflicted
- 415 - Unsupported media type
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of saved results of the search. See the [response body](#) of the Saved Search Execution for representation samples.

- The body contains a list of search results.
- The returned result collection only contains those that the REST client has access to.
- Pagination is supported.

## Disable Saved Results

Disable saved results. After this operation:

- The `has_results` property of the saved search resource is set to false.
- The `results_count` property of the saved search resource is set to -1.

## Supported HTTP Method

DELETE



**Request Media Types**

N/A

**Request Query Parameters**

N/A

**Request Headers**

- Accept
- Authorization (optional)

**Request Body**

N/A

**Response Headers**

N/A

**Response Media Types**

N/A

**Response Status**

- 204 - Saved results disabled successfully
- 400 - Invalid syntax or missing a required value
- 401 - Invalid or missing authentication credentials
- 403 - Permission denied
- 406 - Not Acceptable
- 409 - State conflicted
- 415 - Unsupported media type
- 500 - Other unexpected server error

**Response Body**

After you disable saved results, the system returns `204 no content` with no content in the response body.

# Saved Search Template(s)

## Saved Search Template

There are two ways to create a search template. You can create a search template using a generalized set of search criteria or you can use the existing criteria of a saved search to create a search template.

A search template allows you to share a search with others users. Users who receive the search template can customize it for their own specific use, and can also give it to other users who can customize it further.

In the search template resource, AQL expressions that contain variables have the boolean attribute `template` set to `true`. Variables are set to a predicate value that can be updated at a later time.

An expression containing variables generally follows this pattern:

```
<sample_expression attr_1=v_1 attr_2=v2 ... template="true">predicate_value</sample_expression>
```

For example:

```
<property-list name="r_object_type" operator="IN" repeating="false" template="true">
  <values>
    <value>dm_document</value>
  </values>
</property-list>
```

The following table lists all AQL expressions that support variables:

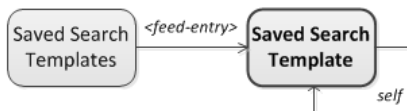
Expression Type	Variable	Sample (variable predicates highlighted in bold)
<a href="#">fulltext</a>	Fulltext search criterion	<code>&lt;fulltext fuzzy="false" template="true"&gt;key word&lt;/fulltext&gt;</code>
<a href="#">property</a>	Property value <b>Note:</b> The search operator cannot be <code>IS_NULL</code> or <code>IS_NOT_NULL</code> .	<code>&lt;property name="object_name" operator="CONTAINS" exact-match="false" repeating="false" case-sensitive="false" fuzzy="false" template="true"&gt;myobj&lt;/property&gt;</code>
<a href="#">property list</a>	Value list	<code>&lt;property-list name="r_object_type" operator="IN" repeating="false" template="true"&gt;     &lt;values&gt;       &lt;value&gt;dm_document&lt;/value&gt;       &lt;value&gt;dm_folder&lt;/value&gt;     &lt;/values&gt;   &lt;/property-list&gt;</code>

Expression Type	Variable	Sample (variable predicates highlighted in bold)
<a href="#">relative date</a>	Relative value and time unit	<relative-date name="r_modify_date" time-unit=" <b>MONTH</b> " operator=" <b>GREATER_THAN</b> " repeating="false" template="true">-3</relative-date>
<a href="#">property range</a>	From and To values	<property-range name="r_modify_date" operator="BETWEEN" repeating="false" template="true"> <from>2010-10-13T07:33:13.009+00:00</from> <to>2014-11-11T01:35:17.089+00:00</to> </property-range>

A search template differs from a saved search in that a search template may contain variables in its search criteria while a saved search cannot.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [About the Diagram](#).

For more information, see [Saved Search Templates](#), page 442.

## Feed

Is feed? No.

## Link Relations

The following table lists the link relations for the Search Template resource.

Link Relation	Description	Resource Reference
self	This Search Template resource.	<a href="#">Saved Search Template</a> , page 434
<a href="http://identifiers.emc.com/linkrel/delete">http://identifiers.emc.com/linkrel/delete</a>	This Search Template resource.	<a href="#">Saved Search Template</a> , page 434
<a href="http://identifiers.emc.com/linkrel/search-execution">http://identifiers.emc.com/linkrel/search-execution</a>	Execute the search template	<a href="#">Search Template Execution</a> , page 458

## Supported HTTP Methods

The Search Template resource supports the following HTTP methods.

Method	Description
GET	Get a search template.
DELETE	Delete a search template.

## Operations

### Get a Search Template

Get a search template.

#### Supported HTTP Method

GET

#### Request Media Types

N/A

#### Request Query Parameters

For more information about the query parameters, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

#### Request Headers

- Accept
- Authorization (optional)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

#### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Body

XML or JSON representation of the search template. Search criteria containing variables are presented in the `query-document-template` block where all variables have the `template` attribute set to `true`. Moreover, the `external-variables` block is appended to `query-document-template`, which contains a number of `expression_type-var` sections describing all variables in the template.

```
<external-variables>
  ...
  <fulltext-variable>
    <id>/expression[0]</id>
    <expression-type>fulltext</expression-type>
    <data-type>string</data-type>
    <variable-value>key word</variable-value>
  </fulltext-variable>
  <property-variable>
    <id>/expression[1]/expression[0]</id>
    <expression-type>property</expression-type>
    <property-name>object_name</property-name>
    <data-type>string</data-type>
    <operator>CONTAINS</operator>
    <variable-value>attr1</variable-value>
  </property-variable>
  ...
</external-variables>
```

An `expression_type` variable section contains the following elements:

Element	Description
id	xPath-like identifier of the expression where the variable is located. For example, /expression[1]/expression[0] indicates the first child expression of the second expression under the root.
expression-type	Type of the expression where the variable is located.
property-name	Name of the property that is used as a variable. Not valid in fulltext expressions.
data-type	Data type of the property that is used as a variable. Not valid in fulltext expressions.
operator	Operator of the expression where the variable is located. Not valid in fulltext and relative date expressions. The GREATER_THAN operator can be used for relative date expressions.
variable-value	Predicate variable value.

**Example 2-141. XML Response**

```
<?xml version="1.0" encoding="UTF-8"?>
<saved-search-template
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dmc_search_template"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/
    dmc_search_template.xml">
  <properties>
    <object_name>test</object_name>
    <subject>Search template Description for unit tests.</subject>
    <creation-date>2015-09-23T01:45:31.000+00:00</creation-date>
    <r_modify_date>2015-09-23T01:45:31.000+00:00</r_modify_date>
    <owner_name>dmadmin</owner_name>
    <r_is_public>true</r_is_public>
    <selected_sources>
      <item>REPO</item>
    </selected_sources>
    <r_object_id>0900000180010d1f</r_object_id>
  </properties>
</query-document-template>&lt;?xml version='1.0' encoding='UTF-8'?>
  &lt;search
    xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false">
    &lt;repositories>&lt;repository>REPO&lt;/repository>
    &lt;/repositories>&lt;types>
    &lt;type>dm_sysobject&lt;/type>&lt;/types>
    &lt;expression-set operator="AND">
    &lt;expressions>&lt;fulltext fuzzy="false" template="true">key word
    &lt;/fulltext>
    &lt;expression-set operator="AND">&lt;expressions>
    &lt;property name="object_name" operator="CONTAINS" exact-match="false"
    repeating="false"
    case-sensitive="false" fuzzy="false" template="true">attr1&lt;/property>
    &lt;/expressions>
    &lt;/expression-set>&lt;property-range name="r_modify_date"
    operator="BETWEEN"
    repeating="false" template="true">&lt;from>1973-01-26T01:45:31 000+00:00
    &lt;/from>
    &lt;to>2006-01-30T01:12:31 000+00:00&lt;/to>
    &lt;/property-range>&lt;relative-date
    name="r_modify_date" time-unit="MONTH" operator="GREATER_THAN" repeating="false"
```

```

        template="true">-3</relative-date></expressions>
        </expression-set></search>
</query-document-template>
<external-variables>
  <fulltext-variable>
    <id>/expression[0]</id>
    <expression-type>fulltext</expression-type>
    <data-type>string</data-type>
    <variable-value>key word</variable-value>
  </fulltext-var>
  <property-variable>
    <id>/expression[1]/expression[0]</id>
    <expression-type>property</expression-type>
    <property-name>object_name</property-name>
    <data-type>string</data-type>
    <operator>CONTAINS</operator>
    <variable-value>attr1</variable-value>
  </property-variable>
  <property-variable>
    <id>/expression[2]/from</id>
    <expression-type>property-range</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <operator>GREATER_EQUAL</operator>
    <variable-value>1973-01-26T01:45:31 000+00:00</variable-value>
  </property-variable>
  <property-variable>
    <id>/expression[2]/to</id>
    <expression-type>property-range</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <operator>LESS_EQUAL</operator>
    <variable-value>2006-01-30T01:12:31 000+00:00</variable-value>
  </property-variable>
  <relative-date-variable>
    <id>/expression[3]</id>
    <expression-type>relative-date</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <variable-value>-3:MONTH</variable-value>
  </relative-date-variable>
</external-variables>
<links>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
search-templates/08024c5180004333/execution.xml"/>
  <link rel="http://identifiers.emc.com/linkrel/search-execution"
    href="http://localhost:8080/dctm-rest/repositories/REPO/
search-templates/08024c5180004333.xml"/>
  <link rel="http://identifiers.emc.com/linkrel/delete"
    href="http://localhost:8080/dctm-rest/
repositories/REPO/search-templates/08024c5180004333.xml"/>
</links>
</saved-search-template>

```

#### Example 2-142. JSON Response

```

{
  "name": "saved-search-template",
  "type": "dmc_search_template",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/
dmc_search_template",
  "properties": {
    "object_name": "test",
    "subject": "Search template Description for unit tests.",

```

```

    "creation-date": "2015-09-23T01:45:31.000+00:00",
    "r_modify_date": "2015-09-23T01:45:31.000+00:00",
    "owner_name": "dmadmin",
    "r_is_public": true,
    "selected_sources": [
        "REPO"
    ],
    "r_object_id": "0900000180010d1f"
},
"query-document-template": "{ \"all-versions\": false, \"repositories\": [ \"REPO\" ],
 \"types\": [ \"dm_sysobject\" ], \"expression-set\": { \"expression-type\": \"expression-set\",
 \"operator\": \"AND\", \"expressions\": [ { \"expression-type\": \"fulltext\", \"value
\": \"key word\", \"fuzzy\": false, \"template\": \"true\" }, { \"expression-type\": \"expression-set\",
 \"operator\": \"AND\", \"expressions\": [ { \"expression-type\": \"property\", \"name\":
 \"object_name\", \"operator\": \"CONTAINS\", \"value\": \"attr1\", \"exact-match\": false,
 \"repeating\": false, \"case-sensitive\": false, \"fuzzy\": false, \"template\": \"true\" } ] },
 { \"expression-type\": \"property-range\", \"name\": \"r_modify_date\", \"operator\":
 \"BETWEEN\", \"from\": \"1973-01-26T01:45:31 000+00:00\",
 \"to\": \"2006-01-30T01:12:31 000+00:00\",
 \"repeating\": false, \"template\": \"true\" }, { \"expression-type\": \"relative-date\",
 \"name\": \"r_modify_date\", \"value\": -3, \"time-unit\": \"MONTH\", \"operator\":
 \"GREATER_THAN\", \"repeating\": false, \"template\": \"true\" } ] } }",
"external-variables": [
    {
        "id": "/expression[0]",
        "expression-type": "fulltext",
        "data-type": "string",
        "variable-value": "key word"
    },
    {
        "id": "/expression[1]/expression[0]",
        "expression-type": "property",
        "property-name": "object_name",
        "data-type": "string",
        "operator": "CONTAINS",
        "variable-value": "attr1"
    },
    {
        "id": "/expression[2]/from",
        "expression-type": "property-range",
        "property-name": "r_modify_date",
        "data-type": "datetime",
        "operator": "GREATER_EQUAL",
        "variable-value": "1973-01-26T01:45:31 000+00:00"
    },
    {
        "id": "/expression[2]/to",
        "expression-type": "property-range",
        "property-name": "r_modify_date",
        "data-type": "datetime",
        "operator": "LESS_EQUAL",
        "variable-value": "2006-01-30T01:12:31 000+00:00"
    },
    {
        "id": "/expression[3]",
        "expression-type": "relative-date",
        "property-name": "r_modify_date",
        "data-type": "datetime",
        "variable-value": "-3:MONTH"
    }
],
"links": [
    {
        "rel": "self",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates/"
    }
]

```



```
        "08024c5180004333"
      },
      {
        "rel": "http://identifiers.emc.com/linkrel/search-execution",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-template/08024c5180004333/execution"
      },
      {
        "rel": "http://identifiers.emc.com/linkrel/delete",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates/08024c5180004333"
      }
    ]
  }
}
```

## Delete a Search Template

Delete a search template.

### HTTP Method

DELETE

### Request Media Types

N/A

### Request Query Parameters

N/A

### Request Headers

- Authorization (optional)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

N/A

## Response Media Types

N/A

## Response Status

204 - Delete the search template successfully  
400 - Invalid syntax or missing a required value  
401 - Invalid or missing authentication credentials  
403 - Permission denied  
404 - Saved search not found  
409 - Conflict  
500 - Other unexpected server error

## Response Body

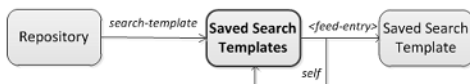
HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

# Saved Search Templates

This resource is a collection of search templates. Users can get list of search templates or create a new search template with this resource.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [About the Diagram](#).

For more information, see [Saved Search Template](#), page 434.

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
The URI of the Search Templates resource	Search templates	Server's current time	Yes

Entry	Entry ID	Entry Title	Entry Updated
<a href="#">Saved Search Template, page 434</a>	Search template URI	Display name of the search template.	r_modify_date of the object

## Link Relations

Link Relation	Description	Resource Reference
self	this object	<a href="#">Saved Search Templates, page 442</a>
first, last, next, previous	Pagination links	Used for navigation when more than one page of search results are available

## Supported HTTP Methods

Method	Description
GET	Retrieves a visible search template collection
POST	Creates a new search template

## Operations

### Create a Search Template

#### Request Method

POST

#### Request Query Parameters

N/A

#### Request Headers

- Authorization (optional)
- Accept
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Body

### Example 2-143. XML Request Body

Here is a code sample that shows you a search template that has properties and an embedded query document in AQL. The AQL statements in the <query-document> element must be escaped as shown below.

For more information about Abstract Query Language, see [AQL](#).

```
<search-template
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="dmc_search_template"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/
types/dmc_search_template.xml">
  <properties>
    <object_name>wangchen</object_name>
    <subject>Search template Description for unit tests.</subject>
    <r_is_public>true</r_is_public>
  </properties>
  <query-document-template>
    &lt;search xmlns="http://identifiers.emc.com/vocab/documentum"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false"&gt;
      &lt;repositories&gt;
        &lt;repository&gt;REPO&lt;/repository&gt;
      &lt;/repositories&gt;
      &lt;types&gt;
        &lt;type&gt;dm_sysobject&lt;/type&gt;
      &lt;/types&gt;
      &lt;expression-set operator="AND"&gt;
        &lt;expressions&gt;
          &lt;fulltext fuzzy="false" template="true"&gt;key word&lt;/fulltext&gt;
          &lt;expression-set operator="AND"&gt;
            &lt;expressions&gt;
              &lt;property name="object_name" operator="CONTAINS"
                exact-match="false" repeating="false" case-sensitive="false"
                fuzzy="false" template="true"&gt;attr1&lt;/property&gt;
            &lt;/expressions&gt;
          &lt;/expression-set&gt;
          &lt;property-range name="r_modify_date" operator="BETWEEN" repeating="false"
            template="true"&gt;
            &lt;from&gt;1973-01-26T03:09:01:000+00:00&lt;/from&gt;
            &lt;to&gt;2015-07-30T035:10:04.000+00:00&lt;/to&gt;
          &lt;/property-range&gt;
          &lt;relative-date name="r_modify_date" time-unit="MONTH"
            operator="GREATER_THAN" repeating="false" template="true"&gt;-3&lt;/relative-date&gt;
        &lt;/expressions&gt;
      &lt;/expression-set&gt;
    &lt;/search&gt;
  </query-document-template>
</search-template>
```

### Example 2-144. JSON Request Body

```
{
```

```

    "properties": {
      "object_name": "wangchen",
      "subject": "Search template Description for unit tests.",
      "r_is_public": true
    },
    "query-document-template": "    {
      \"all-versions\": false,
      \"repositories\": [
        \"REPO\"
      ],
      \"types\": [
        \"dm_sysobject\"
      ],
      \"expression-set\": {
        \"expression-type\": \"expression-set\",
        \"operator\": \"AND\",
        \"expressions\": [
          {
            \"expression-type\": \"fulltext\",
            \"value\": \"key word\",
            \"fuzzy\": false,
            \"template\": \"true\"
          },
          {
            \"expression-type\": \"expression-set\",
            \"operator\": \"AND\",
            \"expressions\": [
              {
                \"expression-type\": \"property\",
                \"name\": \"object_name\",
                \"operator\": \"CONTAINS\",
                \"value\": \"attr1\",
                \"exact-match\": false,
                \"repeating\": false,
                \"case-sensitive\": false,
                \"fuzzy\": false,
                \"template\": \"true\"
              }
            ]
          },
          {
            \"expression-type\": \"property-range\",
            \"name\": \"r_modify_date\",
            \"operator\": \"BETWEEN\",
            \"from\": \"1973-07-30T03:09:01.000+00:00\",
            \"to\": \"2015-07-30T03:10:04.000+00:00\",
            \"repeating\": false,
            \"template\": \"true\"
          },
          {
            \"expression-type\": \"relative-date\",
            \"name\": \"r_modify_date\",
            \"value\": -3,
            \"time-unit\": \"MONTH\",
            \"operator\": \"GREATER_THAN\",
            \"repeating\": false,
            \"template\": \"true\"
          }
        ]
      }
    }
  }
}

```

## Response Status

- 201 - Created successfully
- 400 - Bad request; invalid attribute name or value
- 401 - Authentication failed
- 403 - Forbidden, unauthorized to perform the Request
- 404 - No object found
- 406 - Not acceptable, invalid media type in request header
- 409 - Conflict, creation or update request could not be completed
- 415 - Unsupported media type
- 500 - Internal server error or unexpected server condition

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Headers

- Content-Type
- Location: The URI for the newly added search template
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#)

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

### Example 2-145. XML Response

The value of the `relative-date` variable has the pattern `number:time-unit` to specify the relative date against the current time.

```
<?xml version="1.0" encoding="UTF-8"?>
<search-template
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dmc_search_template"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types/dmc_search_template.xml"
  >
  <properties>
    <object_name>wangchen</object_name>
    <subject>Search template Description for unit tests.</subject>
    <creation-date>2015-09-23T01:45:31.000+00:00</creation-date>
    <r_modify_date>2015-09-23T01:45:31.000+00:00</r_modify_date>
    <owner_name>dmdadmin</owner_name>
    <r_is_public>true</r_is_public>
    <selected_sources>
      <item>REPO</item>
    </selected_sources>
    <r_object_id>0900000180010d1f</r_object_id>
  </properties>
  <query-document-template>&lt;?xml version='1.0' encoding='UTF-8'?&gt;&lt;search
    xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false"&gt;&lt;repository
      &lt;repository&gt;REPO&lt;/repository&gt;&lt;/repositories&gt;&lt;types&gt;&lt;type&gt;
```

```

</types><<expression-set operator="AND"><<expressions><<fulltext fuzzy="false" t
<<expression-set operator="AND"><<expressions><<property name="object_name" operator="
exact-match="false" repeating="false" case-sensitive="false" fuzzy="false" template="true">attr
</query-document-template>
<external-variables>
  <fulltext-variable>
    <id>/expression[0]</id>
    <expression-type>fulltext</expression-type>
    <data-type>string</data-type>
    <variable-value>key word</variable-value>
  </fulltext-variable>
  <property-variable>
    <id>/expression[1]/expression[0]</id>
    <expression-type>property</expression-type>
    <property-name>object_name</property-name>
    <data-type>string</data-type>
    <operator>CONTAINS</operator>
    <variable-value>attr1</variable-value>
  </property-variable>
  <property-variable>
    <id>/expression[2]/from</id>
    <expression-type>property-range</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <operator>GREATER_EQUAL</operator>
    <variable-value>2015-09-23T01:45:31.000+00:00</variable-value>
  </property-variable>
  <property-variable>
    <id>/expression[2]/to</id>
    <expression-type>property-range</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <operator>LESS_EQUAL</operator>
    <variable-value>2015-09-23T01:45:31.000+00:00</variable-value>
  </property-variable>
  <relative-date-variable>
    <id>/expression[3]</id>
    <expression-type>relative-date</expression-type>
    <property-name>r_modify_date</property-name>
    <data-type>datetime</data-type>
    <variable-value>-3:MONTH</variable-value>
  </relative-date-variable>
</external-variables>
<links>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates/08024c5180004333.xml"/>
  <link rel="delete" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates/08024c5180004333/execution.xml"/>
  <link rel="execution"
    href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates/08024c5180004333.xml"/>
</links>
</search-template>

```

#### Example 2-146. JSON Response

```

{
  "name": "search-template",
  "type": "dmc_search_template",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dmc_search_template",
  "properties": {
    "object_name": "wangchen",
    "subject": "Search template Description for unit tests.",
    "creation-date": "2015-09-23T01:45:31.000+00:00",

```

```

    "r_modify_date": "2015-09-23T01:45:31.000+00:00",
    "owner_name": "dmadmin",
    "r_is_public": true,
    "selected_sources": [
        "REPO"
    ],
    "r_object_id": "0900000180010d1f"
},
"query-document-template": "{ \"all-versions\": false, \"repositories\": [ \"REPO\" ],
\"types\": [ \"dm_sysobject\" ], \"expression-set\": { \"expression-type\": \"expression-set\",
\"operator\": \"AND\", \"expressions\": [
    { \"expression-type\": \"fulltext\",
      \"value\": \"key word\", \"fuzzy\": false, \"template\": \"true\" },
    { \"expression-type\": \"expression-set\", \"operator\": \"AND\",
      \"expressions\": [ { \"expression-type\": \"property\", \"name\":
        \"object_name\", \"operator\": \"CONTAINS\", \"value\": \"attr1\",
        \"exact-match\": false, \"repeating\": false, \"case-sensitive\": false,
        \"fuzzy\": false, \"template\": \"true\" } ] },
    { \"expression-type\": \"property-range\", \"name\": \"r_modify_date\",
      \"operator\": \"BETWEEN\", \"from\": \"1973-09-23T01:45:31.000+00:00\",
      \"to\": \"2015-09-23T08:45:31.000+00:00\", \"repeating\": false, \"template\": \"true\" },
    { \"expression-type\": \"relative-date\", \"name\": \"r_modify_date\", \"value\": -3,
      \"time-unit\": \"MONTH\", \"operator\": \"GREATER_THAN\", \"repeating\": false,
      \"template\": \"true\" }
    ] } }\",
    \"external-variables\": [
        {
            \"id\": \"/expression[0]\",
            \"expression-type\": \"fulltext\",
            \"data-type\": \"string\",
            \"variable-value\": \"key word\"
        },
        {
            \"id\": \"/expression[1]/expression[0]\",
            \"expression-type\": \"property\",
            \"property-name\": \"object_name\",
            \"data-type\": \"string\",
            \"operator\": \"CONTAINS\",
            \"variable-value\": \"attr1\"
        },
        {
            \"id\": \"/expression[2]/from\",
            \"expression-type\": \"property-range\",
            \"property-name\": \"r_modify_date\",
            \"data-type\": \"datetime\",
            \"operator\": \"GREATER_EQUAL\",
            \"variable-value\": \"2015-09-23T01:45:31.000+00:00\"
        },
        {
            \"id\": \"/expression[2]/to\",
            \"expression-type\": \"property-range\",
            \"property-name\": \"r_modify_date\",
            \"data-type\": \"datetime\",
            \"operator\": \"LESS_EQUAL\",
            \"variable-value\": \"2015-09-23T01:45:31.000+00:00\"
        },
        {
            \"id\": \"/expression[3]\",
            \"expression-type\": \"relative-date\",
            \"property-name\": \"r_modify_date\",
            \"data-type\": \"datetime\",
            \"variable-value\": \"-3:MONTH\"
        }
    ],
    \"links\": [

```



```

    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
              search-templates/08024c5180004333"
    },
    {
      "rel": "delete",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
              search-template/08024c5180004333/execution"
    },
    {
      "rel": "execution",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
              search-templates/08024c5180004333"
    }
  ]
}

```

## Saved Search as Search Template

This operation creates a new search template from an existing saved search. Each call to this method creates a new search template. Any expressions that are in the saved search that can be templates will become templates. When a property expression has the *IS\_NULL* or *IS\_NOT\_NULL* operator it does not become a template. This is because there is no value in the property expression that can be used as a variable.

## HTTP Method

POST

## Request Media Types

application/vnd.emc.documentum+xml

application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The three attributes of the Search Template that can be specified by users include the following:

- `object_name`
- `subject`
- `r_is_public`

You can set the `object_name` attribute, however, when a value for the `object_name` attribute is not specified in the Request body, its value is set from the saved search used to create the new template. The values for all of the other attributes are set using the values from the source saved search used to create the new template. The `search-reference` attribute specifies the URL of the source saved search.

### Example 2-147. XML Request Payload

```
<search-template
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" search-reference="http://localhost:8080/dc
  <properties>
    <object_name>new template</object_name>
    <subject>Search template created from saved search.</subject>
    <r_is_public>true</r_is_public>
  </properties>
</search-template>
```

### Example 2-148. JSON Request Payload

```
{
  "properties": {
    "object_name": "new template",
    "subject": "Search template created from saved search.",
    "r_is_public": true
  },
  "search-reference": "http://localhost:8080/dctm-rest/repositories/REPO/
    saved-searches/0800000580008fd3"
}
```

## Response Headers

- `Content-Type`  
For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).
- `Location` The URI of the newly created search template.

## Response Media Types

- `application/vnd.emc.documentum+xml`
- `application/xml`

- application/vnd.emc.documentum+json
- application/json

## Response Status

- 201 Created successfully
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Body

### Example 2-149. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<search-template
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="dmc_search_template"
  definition="http://localhost:8080/dctm-rest/repositories/REPO/types
    /dmc_search_template.xml">
  <properties>
    <object_name>wangchen</object_name>
    <subject>Search template Description for unit tests.</subject>
    <r_creation_date>2015-09-23T01:45:31.000+00:00</r_creation_date>
    <r_modify_date>2015-09-23T01:45:31.000+00:00</r_modify_date>
    <owner_name>dmdadmin</owner_name>
    <r_is_public>true</r_is_public>
    <selected_sources>
      <item>REPO</item>
    </selected_sources>
    <r_object_id>0900000180010d1f</r_object_id>
  </properties>
  <query-document-template><?xml version='1.0' encoding='UTF-8'?><search
    xmlns="http://identifiers.emc.com/vocab/documentum"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" all-versions="false">
    <repositories><repository>REPO</repository>
    </repositories><types><type>dm_sysobject</type>
    </types><expression-set operator="AND"><expressions>
    <fulltext fuzzy="false" template="true">key word</fulltext>
    <expression-set operator="AND"><expressions><property
    name="object_name" operator="CONTAINS" exact-match="false" repeating="false"
    case-sensitive="false" fuzzy="false" template="true">attr1</property>
    </expressions></expression-set><property-range name="r_modify_date"
    operator="BETWEEN" repeating="false" template="true">
    <from>01/26/1973 00:00:00</from><to>11/30/2006 00:00:00
    </to></property-range><relative-date name="r_modify_date"
    time-unit="MONTH" operator="GREATER_THAN" repeating="false" template="true">
    -3</relative-date></expressions></expression-set>
    </search>
  </query-document-template>
  <external-variables>
    <fulltext-variable>
      <id>/expression[0]</id>
      <expression-type>fulltext</expression-type>
      <data-type>string</data-type>
      <variable-value>key word</variable-value>
    </fulltext-variable>
    <property-variable>
```

```

        <id>/expression[1]/expression[0]</id>
        <expression-type>property</expression-type>
        <property-name>object_name</property-name>
        <data-type>string</data-type>
        <operator>CONTAINS</operator>
        <variable-value>attr1</variable-value>
    </property-variable>
    <property-variable>
        <id>/expression[2]/from</id>
        <expression-type>property-range</expression-type>
        <property-name>r_modify_date</property-name>
        <data-type>datetime</data-type>
        <operator>GREATER_EQUAL</operator>
        <variable-value>01/26/1973 00:00:00</variable-value>
    </property-variable>
    <property-variable>
        <id>/expression[2]/to</id>
        <expression-type>property-range</expression-type>
        <property-name>r_modify_date</property-name>
        <data-type>datetime</data-type>
        <operator>LESS_EQUAL</operator>
        <variable-value>11/30/2006 00:00:00</variable-value>
    </property-variable>
    <relative-date-variable>
        <id>/expression[3]</id>
        <expression-type>relative-date</expression-type>
        <property-name>r_modify_date</property-name>
        <data-type>datetime</data-type>
        <variable-value>-3:MONTH</variable-value>
    </relative-date-variable>
</external-variables>
<links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
    /search-templates/08024c5180004333.xml"/>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates/08024c5180004333.xml"/>
    <link rel="http://identifiers.emc.com/linkrel/delete"
    href="http://localhost:8080/dctm-rest/repositories/REPO/search-templates/
    08024c5180004333.xml"/>
</links>
</search-template>

```

### Example 2-150. JSON Response

```

{
  "name": "search-template",
  "type": "dmc_search_template",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/
  dmc_search_template",
  "properties": {
    "object_name": "wangchen",
    "subject": "Search template Description for unit tests.",
    "r_creation_date": "2015-09-23T01:45:31.000+00:00",
    "r_modify_date": "2015-09-23T01:45:31.000+00:00",
    "owner_name": "dmadmin",
    "r_is_public": true,
    "selected_sources": [
      "REPO"
    ],
    "r_object_id": "0900000180010d1f"
  },
  "query-document-template": "{ \"all-versions\": false, \"repositories\": [ \"REPO\" ],
  \"types\": [ \"dm_sysobject\" ], \"expression-set\": { \"name\": \"expression-set\",
  \"operator\": \"AND\", \"expressions\": [ { \"expression-type\": \"fulltext\",

```

```

\value\":"key word",\fuzzy\:false,\template\":"true\"},
{"name\":"expression-set",\operator\":"AND",\expressions\":[
  {\expression-type\":"property",\name\":"object_name",
   \operator\":"CONTAINS",\value\":"attr1",\exact-match\:false,
   \repeating\:false,\case-sensitive\:false,\fuzzy\:false,
   \template\":"true\"}}],
{"expression-type\":"property-range",\name\":"r_modify_date",
 \operator\":"BETWEEN",\from\":"01/26/1973 00:00:00",
 \to\":"11/30/2006 00:00:00",\repeating\:false,\template\":"true\"},
{"expression-type\":"relative-date",\name\":"r_modify_date",
 \value\":"-3",\time-unit\":"MONTH",\operator\":"GREATER_THAN",
 \repeating\:false,\template\":"true\"}}}],
"external-variables": [
  {
    "id": "/expression[0]",
    "expression-type": "fulltext",
    "data-type": "string",
    "variable-value": "key word"
  },
  {
    "id": "/expression[1]/expression[0]",
    "expression-type": "property",
    "property-name": "object_name",
    "data-type": "string",
    "operator": "CONTAINS",
    "variable-value": "attr1"
  },
  {
    "id": "/expression[2]/from",
    "expression-type": "property-range",
    "property-name": "r_modify_date",
    "data-type": "datetime",
    "operator": "GREATER_EQUAL",
    "variable-value": "01/26/1973 00:00:00"
  },
  {
    "id": "/expression[2]/to",
    "expression-type": "property-range",
    "property-name": "r_modify_date",
    "data-type": "datetime",
    "operator": "LESS_EQUAL",
    "variable-value": "11/30/2006 00:00:00"
  },
  {
    "id": "/expression[3]",
    "expression-type": "relative-date",
    "property-name": "r_modify_date",
    "data-type": "datetime",
    "variable-value": "-3:MONTH"
  }
],
"links": [
  {
    "rel": "self",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates/08024c5180004333"
  },
  {
    "rel": "edit",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-template/08024c5180004333"
  },
  {
    "rel": "http://identifiers.emc.com/linkrel/delete",
    "href": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates/

```

```
{
  "id": "08024c5180004333"
}
```

## Get Search Templates

Get a collection of search templates.

### HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- inline
- items-per-page
- page
- sort
- filter
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization (optional)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 Retrieved successfully
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#)

## Response Body

### Example 2-151. XML Response

```
<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/search-templates
  </id>
  <title>Search Templates</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-09-23T02:02:59.525+00:00</updated>
  <dm:page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates.xml?inline=true&items-per-page=1"/>
  <link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates.xml?inline=true&items-per-page=1&page=2"/>
  <link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/
    search-templates.xml?inline=true&items-per-page=1&page=1"/>
  <entry>
    <id>http://localhost:8080/dctm-rest/repositories/REPO/search-templates/
      0900000180010d15</id>
    <title>wangchen</title>
    <author>
      <name>dmadmin</name>
    </author>
```

```

<summary>Search template Description for unit tests.</summary>
<updated>2015-09-23T02:02:59.526+00:00</updated>
<published>2015-09-23T01:25:57.000+00:00</published>
<link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
  search-templates/0900000180010d15.xml"/>
<content>
  <dm:search-template
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:properties>
      <dm:object_name>wangchen</dm:object_name>
      <dm:r_object_id>0900000180010d15</dm:r_object_id>
      <dm:r_is_public>1</dm:r_is_public>
      <dm:r_modify_date>2015-09-23T01:25:57.000+00:00
      </dm:r_modify_date>
      <dm:creation-date>2015-09-23T01:25:57.000+00:00
      </dm:creation-date>
      <dm:owner_name>dmadmin</dm:owner_name>
      <dm:selected_sources>
        <dm:item>REPO</dm:item>
      </dm:selected_sources>
      <dm:subject>Search template Description for unit tests.
      </dm:subject>
    </dm:properties>
    <dm:links>
      <link rel="self" href="http://localhost:8080/dctm-rest/
        repositories/REPO/search-templates/0900000180010d15.xml"/>
    </dm:links>
  </dm:search-template>
</content>
</entry>
</feed>

```

### Example 2-152. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/search-templates",
  "title": "Search Templates",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-09-23T02:02:17.107+00:00",
  "page": 1,
  "items-per-page": 1,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
        search-templates?inline=true&items-per-page=1"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
        search-templates?inline=true&items-per-page=1&page=2"
    },
    {
      "rel": "first",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/
        search-templates?inline=true&items-per-page=1&page=1"
    }
  ],
  "entries": [
    {

```



```

    "id": "http://localhost:8080/dctm-rest/repositories/REPO/
      search-templates/0900000180010d15",
    "title": "wangchen",
    "author": [
      {
        "name": "dmadmin"
      }
    ],
    "summary": "Search template Description for unit tests.",
    "updated": "2015-09-23T02:02:17.108+00:00",
    "published": "2015-09-23T01:25:57.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/
          search-templates/0900000180010d15"
      }
    ],
    "content": {
      "name": "search-template",
      "properties": {
        "object_name": "wangchen",
        "r_object_id": "0900000180010d15",
        "r_is_public": 1,
        "r_modify_date": "2015-09-23T01:25:57.000+00:00",
        "creation-date": "2015-09-23T01:25:57.000+00:00",
        "owner_name": "dmadmin",
        "selected_sources": [
          "REPO"
        ],
        "subject": "Search template Description for unit tests."
      },
      "links": [
        {
          "rel": "self",
          "href": "http://localhost:8080/dctm-rest/repositories/REPO/
            search-templates/0900000180010d15"
        }
      ]
    }
  }
]
}

```

## Search Template Execution

This resource fills in the variables and allows you to retrieve real time search results from a search template.

### Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [About the Diagram](#).

For more information, see [Saved Search Templates](#), page 442.

### Feed

Is feed? Yes

### Link Relations

Link Relation	Description	Resource Reference
self	This object resource	This resource
first, last, next, previous	Pagination links	Used for navigation when more than one page of search results are available

### Supported HTTP Methods

Method	Description
POST	Retrieves the results of an instantiated search template

## Operations

### Get Search Results of an Instantiated Search Template

This operation is similar to execute as an AQL query document. The AQL used to execute this operation comes from the Content Server and user inputs.

#### HTTP Method

POST

#### Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

#### Request Query Parameters

Variable	Description	Data Type	Value Range	Default Value
links, inline, include-total, page, items-per-page	For more information, see <a href="#">Appendix B, REST Common Definition - URI Request Query Parameters</a> .			
facet-id-constraints	<p>Contains the key value pairs for each facet id and its constraint.</p> <p>For example:</p> <ul style="list-style-type: none"><li>• You can use <code>id1</code> as the facet id for the <code>r_object_type</code> property</li><li>• You can use <code>id2</code> as the facet id for the <code>owner_name</code> property</li></ul> <p><code>facet-id-constraints=id1=dm_document,id2=Administrator</code></p>			

#### Request Headers

- Authorization
- Accept
- For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

- Only the `external-variables` parameter will have an effect when used in the Request body. The system will ignore the `query-document-template` and `properties` parameters are ignored when included in the Request.
- In each variable, `id` and `variable-value` are mandatory (for JSON, `variable-type` is required). Users can skip other elements; but if other elements are used in the Request body, they must be correct.
- The date type should comply with this pattern `yyyy-MM-dd'T'HH:mm:ss.SSSZ`, and as specified in [ISO 8601](#).

### Example 2-153. XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<search-template xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://identifiers.emc.com/vocab/documentum http://identifiers.emc.com/vocab/documentum.xsd">
  <external-variables>
    <fulltext-variable>
      <id>/expression[0]</id>
      <variable-value>key word</variable-value>
    </fulltext-variable>
    <property-list-variable>
      <id>/expression[1]</id>
      <expression-type>property-list</expression-type>
      <property-name>object_name</property-name>
      <data-type>string</data-type>
      <operator>IN</operator>
      <variable-values>
        <item>v1</item>
        <item>v2</item>
      </variable-values>
    </property-list-variable>
    <property-variable>
      <id>/expression[2]/expression[0]</id>
      <expression-type>property</expression-type>
      <property-name>object_name</property-name>
      <data-type>string</data-type>
      <operator>CONTAINS</operator>
      <variable-value>attr1</variable-value>
    </property-variable>
    <property-variable>
      <id>/expression[3]/from</id>
      <variable-value>2014-10-13T07:33:13.009+00:00</variable-value>
    </property-variable>
    <property-variable>
      <id>/expression[3]/to</id>
      <expression-type>property-range</expression-type>
      <property-name>r_modify_date</property-name>
      <data-type>datetime</data-type>
      <operator>LESS_EQUAL</operator>
      <variable-value>2014-11-13T07:33:13.009+00:00</variable-value>
    </property-variable>
    <relative-date-variable>
      <id>/expression[4]</id>
      <expression-type>relative-date</expression-type>
      <property-name>r_modify_date</property-name>
      <data-type>datetime</data-type>
      <variable-value>-3:MONTH</variable-value>
      <operator>GREATER_THAN</operator>
    </relative-date-variable>
  </external-variables>
</search-template>
```

```
</search-template>
```

#### Example 2-154. JSON Request

```
{
  "external-variables": [
    {
      "variable-type": "fulltext-variable",
      "id": "/expression[0]",
      "variable-value": "key word"
    },
    {
      "variable-type": "property-list-variable",
      "id": "/expression[1]",
      "expression-type": "property-list",
      "property-name": "object_name",
      "data-type": "string",
      "operator": "IN",
      "variable-values": [
        "v1",
        "v2"
      ]
    },
    {
      "variable-type": "property-variable",
      "id": "/expression[2]/expression[0]",
      "variable-value": "attr1"
    },
    {
      "variable-type": "property-variable",
      "id": "/expression[3]/from",
      "expression-type": "property-range",
      "property-name": "r_modify_date",
      "data-type": "datetime",
      "operator": "GREATER_EQUAL",
      "variable-value": "2014-10-13T07:33:13.009+00:00"
    },
    {
      "variable-type": "property-variable",
      "id": "/expression[3]/to",
      "expression-type": "property-range",
      "property-name": "r_modify_date",
      "data-type": "datetime",
      "operator": "LESS_EQUAL",
      "variable-value": "2014-11-13T07:33:13.009+00:00"
    },
    {
      "variable-type": "relative-date-variable",
      "id": "/expression[4]",
      "expression-type": "relative-date",
      "property-name": "r_modify_date",
      "data-type": "datetime",
      "variable-value": "3:MONTH",
      "operator": "GREATER_THAN"
    }
  ]
}
```

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

- 200 - Retrieved results successfully
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-155. XML Response

```
<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c5180005288/
    execution</id>
  <title>Search results</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-08-04T07:19:05.885+00:00</updated>
  <dm:page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c5180005288/execution.xml?items-per-page=1"/>
  <link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c5180005288/execution.xml?items-per-page=1&page=2"/>
  <link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
    08024c5180005288/execution.xml?items-per-page=1&page=1"/>
  <link rel="search" hreftemplate="http://localhost:8080/dctm-rest/repositories/REPO/
    search.xml{?collections, facet, include-total, inline, items-per-page, locations, object-type,
    page, q, sort, timezone, view}"/>
  <entry>
    <id>09024c518000488c</id>
    <title>ADFS and Shib 0.81.docx</title>
    <author>
```

```

    <name>UnitTestUser</name>
  </author>
  <summary>Federated Access 18 Step 3: Test AD FS 2.0 as the Identity...
    Step 5: Test Shibboleth as the Identity Provider and AD FS 2.0 as...
    following sections. AD FS 2.0 The test deployment that is created in the...
    domain controller and the federation server in test deployments</summary>
  <updated>2015-08-04T06:50:23.000+00:00</updated>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
    09024c518000488c.xml"/>
  <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/REPO/
    objects/09024c518000488c.xml"/>
  <relevance:score
    xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">1.0
  </relevance:score>
  <dm:terms
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:term>Test</dm:term>
    <dm:term>test</dm:term>
    <dm:term>testing</dm:term>
    <dm:term>rest</dm:term>
    <dm:term>Testing</dm:term>
    <dm:term>tested</dm:term>
  </dm:terms>
</entry>
</feed>

```

### Example 2-156. JSON Response

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/08024c5180005288/
  execution",
  "title": "Search results",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-08-04T07:17:59.804+00:00",
  "page": 1,
  "items-per-page": 1,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      08024c5180005288/execution?items-per-page=1"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      08024c5180005288/execution?items-per-page=1&page=2"
    },
    {
      "rel": "first",
      "href": "http://localhost:8080/dctm-rest/repositories/REPO/saved-searches/
      08024c5180005288/execution?items-per-page=1&page=1"
    },
    {
      "rel": "search",
      "hreftemplate": "http://localhost:8080/dctm-rest/repositories/REPO/
      search{?collections, facet, include-total, inline, items-per-page, locations, object-type,
      page, q, sort, timezone, view}"
    }
  ],
}

```

```
"entries": [
  {
    "id": "09024c518000488c",
    "title": "ADFS and Shib 0.81.docx",
    "author": [
      {
        "name": "UnitTestUser"
      }
    ],
    "summary": "Federated Access\tl8 Step 3: Test AD FS 2.0 as the Identity...
Step 5: Test Shibboleth as the Identity Provider and AD FS 2.0 as...
following sections. AD FS 2.0 The test deployment that is created in the...
domain controller and the federation server in test deployments",
    "updated": "2015-08-04T06:50:23.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
09024c518000488c"
      }
    ],
    "content": {
      "type": "application/vnd.emc.documentum+json",
      "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/
09024c518000488c"
    },
    "score": "1.0",
    "terms": [
      "Test",
      "test",
      "testing",
      "rest",
      "Testing",
      "tested"
    ]
  }
]
```



# Sub-Group(s)

## Sub Group

The Sub Group resource represents the relationship between a group and one of its sub groups.

## Resource Relationships

The following diagram illustrates how this resource is related with other resources:



See Also: [Sub Groups, page 467](#) and [About the Diagram](#).

## Link Relation

N/A

## Operations

The Sub Group resource supports the following HTTP method.

Method	Description
DELETE	Remove a sub group from a group

## Remove a Sub Group

Removes the relation between a group and it's sub group.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

N/A

## Request Headers

- Accept
- Content-Type
- Authentication

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

### Example 2-157. Removing a User

```
DELETE http://localhost:8080/dctm-rest/repositories/REPO/groups/  
mygroup/groups/my subgroup
```

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

## Response Status

- 204 - Delete successful
- 400 - Bad request, for example, bad attribute filter
- 401 - Authenticate failed
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation.

## Sub Groups

The Sub Groups resource represents a collection of sub groups that directly belong to a specified group.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Group, page 272](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	List of groups	Server's current time	<a href="#">Group, page 272</a>	No

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the group	Group name	Group description	r_modify_date of the group

## Link Relations

The following table lists link relations for the Sub Groups resource.

Link Relation	Description	Resource Reference
self	This collection of groups.	<a href="#">Groups, page 280</a>
first, last, next, previous	Pagination links.	<a href="#">Groups, page 280</a>

## Operations

The Sub Groups resource supports the following HTTP method.

Method	Description
GET	Retrieves the information about a collection of groups.

## Add a Sub Group

Add an existing group into this group as a sub group

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Content-Type
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

Variable	Description	Data Type	Value Range	Default Value
href	The specified group resource URI.  <b>Note:</b> The <i>href</i> variable is mandatory.	String	N/A	null

### Example 2-158. XML Request

```
<dm:group href="http://localhost:8080/emc-rest/
  repositories/REPO/groups/xxxxx"/>
```

### Example 2-159. JSON Request

```
{
  "href": "http://localhost:8080/emc-rest/repositories/REPO/groups/xxxxx"
```

```
}
```

## Response Headers

- Location

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

N/A

## Response Status

- 201 - Add successful
- 400 - Bad request, for example, bad attribute filter
- 401 - Authenticate failed
- 403 - Permission denied
- 404 - Group not found
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

N/A

## Get Sub Groups

Retrieve the information about a collection of sub groups for a given parent group.

## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- inline
- view
- items-per-page
- page
- include-total
- sort
- recursive
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization (optional)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request
- 401 - Authentication failed
- 404 - Group not found
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of groups.

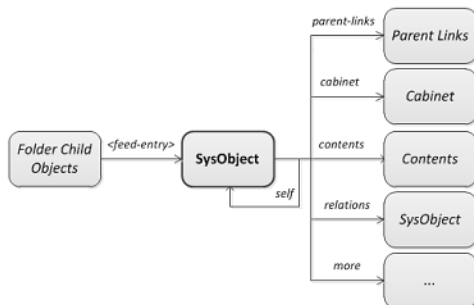
- The body contains a list of the `dm_group` instances (or subtypes of `dm_group`).
- Each object may contain all or a set of properties of the group, depending on the setting of the query parameter `view`.
- The returned child objects collection only contains those that you have access to.
- Pagination is supported.
- By default, the results are listed in alphabetical order by group name.

# SysObject

The SysObject resource represents a SysObject instance in a repository, such as a document or a folder.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Folder Child Objects](#), page 248 and [About the Diagram](#).

## Link Relations

### Any SysObject Link Relations

Link Relation	Description	Resource Reference
self	This SysObject resource.	<a href="#">SysObject</a> , page 472
parent-links [1]	Parent folder (or cabinet) of this SysObject resource.	<a href="#">Parent Links</a> , page 328
edit [1]	Edit this object.	<a href="#">SysObject</a> , page 472
delete [1]	Delete this object.	<a href="#">SysObject</a> , page 472
relations [1]	Collection of relations that is related to this SysObject resource.	<a href="#">SysObject</a> , page 472
comments [1]	The comments associated to this object	<a href="#">Comments</a> , page 142
virtual-document-nodes [1]	The virtual document nodes when the sysobject is a virtual document	<a href="#">Virtual Document Nodes</a> , page 516
object-aspects [1]	Attached aspect of the sysobject	
permission-set [1]	Permission set of the sysobject	<a href="#">Permission Set</a> , page 337
permissions [1]	Specified user's permissions on a given sysobject	<a href="#">Permissions</a> , page 333



### Folder Specific Link Relations

Link Relation	Description	Resource Reference
folders [1]	Collection of child folders that are linked to this SysObject resource.	<a href="#">Folder Child Folders, page 233</a>
documents [1]	Collection of documents that are linked to this SysObject resource.	<a href="#">Folder Child Documents, page 216</a>
objects [1]	Collection of SysObject resources that are linked to this SysObject resource.	<a href="#">Folder Child Objects, page 248</a>
child-links [1]	Collection of folder children.	<a href="#">Folder Child Objects, page 248</a>

### Any Non-Folder SysObject Link Relations

Link Relation	Description	Resource Reference
contents	Collection of relations that is related to this SysObject resource.	<a href="#">Contents, page 164</a>
primary-content [1]	Points to object primary content metadata	<a href="#">Content, page 157</a>

### Versioning Link for Any Non-Folder SysObject

Link Relation	Description	Resource Reference
checkout [1]	Check out this object.	<a href="#">Lock, page 313</a>
checkin-next-major [1]	Check in this object with the next major version policy.	<a href="#">All Versions, page 44</a>
checkin-next-minor [1]	Check in this object with the next minor version policy.	<a href="#">All Versions, page 44</a>
checkin-branch [1]	Check in this object with the branch version policy.	<a href="#">All Versions, page 44</a>
cancel-checkout [1]	Cancel the check-out operation on this object.	<a href="#">Lock, page 313</a>
version-history	Collection of versions of this document.	<a href="#">All Versions, page 44</a>
current-version [1]	Current version of the object.	<a href="#">Current Version, page 192</a>
predecessor-version	Predecessor version of the object.	<a href="#">SysObject, page 472</a>

### Lightweight Object Specific Link Relations

Link Relation	Description	Resource Reference
shared-parent [1]	Shared parent of the sysobject if it is a lightweight child object.	<a href="#">Lightweight Object Parent, page 304</a>
lightweight-objects [1]	Primary content metadata.	<a href="#">Lightweight Objects, page 297</a>
materialize [1]	Materialize the lightweight object following the link relation.	<a href="#">Lightweight Object Materialization, page 309</a>
dematerialize [1]	Dematerialize the lightweight object following the link relation.	<a href="#">Lightweight Object Materialization, page 309</a>

[1] The above link relations are defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: `http://identifiers.emc.com/linkrel/`



#### Caution:

- The versioning links are not displayed for the SysObject Collection
- The links `checkin-next-major`, `checkin-next-minor`, and `checkin-branch` are dynamically checked

These links are displayed only when their related version policy is allowed

### Permission-enabled Link Relations

Link Relation	Basic Permission	Extended Permission	Object State	User Privilege	Version Policy
self	browse				
contents	read				
primary-content	read				
edit	write		not locked	lock owner if the object is locked	
delete	delete	delete			
child-links	browse				
parent-links	browse				
checkout	version		not locked		
checkin-next-major	version		checked out	lock owner	allowed next major
checkin-next-minor	version		checked out	lock owner	allowed next minor
checkin-branch	version		checked out	lock owner	allowed branch

Link Relation	Basic Permission	Extended Permission	Object State	User Privilege	Version Policy
cancel-checkout	version		checked out	lock owner, object owner, or super user	
version-history	browse				
current-version	browse				
predecessor-version	browse				

A single SysObject resource differs from a collection feed of the SysObject resource in the following aspects:

- A single SysObject contains versioning links while a collection does not
- Documentum Platform REST Services does not perform any permission check on a collection
  - The contents links are shown in a collection without permission check.
  - When an object is not locked in a collection, the `edit` and `delete` links are displayed without permission check, meaning that even when a user is not the lock owner or does not have the corresponding permission, the `edit` and `delete` links are accessible
  - When an object is locked, there's no `delete` link in either the single or collection resource
  - When an object is locked in a collection, and a user is the lock owner, the `edit` link is displayed without permission check, meaning that even when the user does not have the write permission, the `edit` link is accessible
  - When a single resource is locked and the user is the lock owner, a check to determine if the user has `write` permissions is done.
  - When a collection resource is locked and the user is the lock owner, the `edit` link is always displayed, without any permission checking



**Caution:** The `checkin-*` links depend on the `dfc` object and version policy, which must get objects one by one.

The `browse`, `read`, `delete`, and `browse...` permission dependant ACL, can be retrieved by the `dfc` or `dfc acl` object.

The `cancel-checkout` link requires user privileges.

A simple performance test has determined that retrieving the `dfc` object, calculating the ACL and version policy, is up to 6 times slower than retrieving the collection using DQL with attributes only. This latency could be worse when using WAN because of the numerous RPC calls that are necessary.

Avoid checking permissions when working with a collection. When enabling a link requires the use of other objects in addition to the current object, then the permissions are not checked. For example, when working with a parent and cabinet link relation. The parent folder and cabinet do not perform any checks to verify access.

## Operations

The SysObject resource supports the following HTTP methods.

Method	Description
GET	Retrieves properties, and other information of the SysObject resource
POST	Updates the properties for the SysObject resource
DELETE	Deletes the SysObject resource from a repository

### Get an Object

Retrieve properties, and other information of the SysObject resource. Properties are returned as embedded elements in the response message body. Other information, such as relationships, versions, and contents, is referenced via link relations in the response message body.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Accept
- Authorization (optional)

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully

400 - Bad request; for example, the specified `r_object_id` matches an object that is not a `dm_sysobject`

401 - Authentication failed

404 - No object found with the specified `r_object_id`

500 - Other unexpected server error

## Response Body

XML or JSON representation of the object.

## Delete an Object

Delete the SysObject resource from the repository.

For a folder object, folder children deletions are controlled by query parameters. If a deletion fails, an exception is thrown and the SysObject resource (including the folder tree and version history) is reverted to the original state.

For a document object, the version policy is controlled by query parameters.

## Supported HTTP Method

DELETE

## Request Media Types

N/A

## Request Query Parameters

The following query parameters are specific to this operation:

Variable	Description	Data type	Default value
del-non-empty	<p>Specifies whether or not this operation deletes a non-empty folder.</p> <ul style="list-style-type: none"><li>• <code>true</code> - The folder and all its descendants are deleted.</li><li>• <code>false</code> - Only this folder is deleted. Deleting a non-empty folder results in an error.</li></ul>	boolean	false
del-all-links	<p>Specifies whether a multi-linked descendant is deleted or unlinked from this specified folder.</p> <ul style="list-style-type: none"><li>• <code>true</code> - This operation deletes a multi-linked descendant along with all its folder links.</li><li>• <code>false</code> - This operation only deletes a descendant's link to this folder. The descendant is still linked to other folders.</li></ul>	boolean	false

Variable	Description	Data type	Default value
	<b>Note:</b> Virtual documents are not supported.		
del-version	Deletion options for multi-version document objects. <ul style="list-style-type: none"> <li>selected - The version associated with the given object ID is removed.</li> <li>unused - (Default) The versions that do not have symbolic labels associated are removed.</li> <li>all - All revisions on the same version tree are removed.</li> </ul>	string	all

This operation also supports the following common query parameter:

- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

N/A

## Response Media Types

N/A

## Response Status

- 204 - Success; no content is returned
- 400 - Bad request; invalid parameter value was provided
- 401 - Authentication failed
- 403 - Permission denied; no permission to delete the object
- 409 - State conflicted, for example, document is locked
- 500 - Other unexpected server error

## Response Body

HTTP 204 No Content status upon a successful delete operation. The Response body contains no content.

If the resource type is not `dm_sysobject` or its subtype, the error code 400 with the invalid parameter is returned.

## Update an Object

Update a SysObject resource in the repository with given properties.

## Supported HTTP Method

POST

## Request Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type



For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

XML or JSON representation of the object and properties. Only updatable properties can be put in the message body. Permission update is excluded.

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Update successful
- 400 - Bad request; invalid property name or value
- 401 - Authentication failed
- 403 - Permission denied; no permission to update the object or setting read-only properties is not allowed
- 404 - Object not found
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the object. A successful response contains all properties of the updated object. Permission set is not returned.

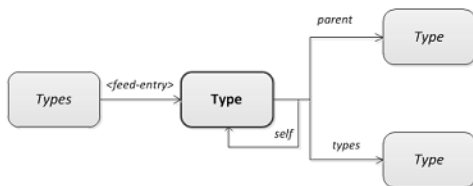
# Type(s)

## Type

The Type resource represents a single type object.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Types, page 488](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the Type resource.

Link Relation	Description	Resource Reference
self	This Type resource.	<a href="#">Type, page 482</a>
types [1]	Sub types of this Type object.	<a href="#">Type, page 482</a>
parent	Link to the associated Type object.	<a href="#">Type, page 482</a>
lightweight-types [1]	The lightweight types of current shareable Type.	<a href="#">Types, page 488</a>
parent-shareable-type [1]	The parent shareable Type of current lightweight Type.	<a href="#">Type, page 482</a>
assist-values [1]	The Value Assistance of the Type.	<a href="#">Value Assistance, page 512</a>

[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: `http://identifiers.emc.com/linkrel/`

## Operations

The Type resource supports the following HTTP method.

Method	Description
GET	Retrieves a Type object in the repository

## Get a Type

Retrieves the metadata and content properties of the current type object.

### HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

The following query parameters are specific to this operation.

Variable	Description	Data type	Value Range	Default value
inherited	<p>Specifies whether or not to return inherited properties from the parent type.</p> <ul style="list-style-type: none"><li>• <code>true</code> - Returns inherited properties from the parent type.</li><li>• <code>false</code> - Only returns properties that directly belong to this type object.</li></ul>	boolean	true, false	true

Variable	Description	Data type	Value Range	Default value
locale	Specifies the locale of the Request.  <b>Note:</b> If the corresponding label of the specified locale cannot be found, the label is null.	string	Valid locale	null
include-value -assist	Specifies whether the response includes the definition information of value assist	boolean	true, false	false

When the relative label value of the specified locale is invalid, locale is not returned. Nevertheless, all other attributes are returned.

## Request Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request

- 401 - Authentication failed
- 404 - Type not found
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Body

### Example 2-160. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<type
  xmlns="http://identifiers.emc.com/vocab/documentum"
  xmlns:dm="http://identifiers.emc.com/vocab/documentum"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  name="dm_document" label="Document" category="standard"
  parent="http://localhost:8080/dctm-rest/repositories/REPO/types/dm_sysobject.xml">
  <properties>
    <property name="object_name" repeating="false" type="string" length="255"
      label="Name" hidden="false" required="false" notnull="false"
      readonly="false" searchable="true">
    <dependencies>
      <item>attr1</item>
      <item>attr2</item>
    </dependencies>
  </property>
  <property name="r_object_type" repeating="false" type="string" length="32"
    label="Type" hidden="false" required="false" notnull="false" readonly="true"
    searchable="true">
    <dependencies>
      <item>attr1</item>
      <item>attr2</item>
    </dependencies>
    <value-assist>
      <fixed-list condition="condition1" allow_user_values="true">
        <value>value1</value>
        <value>value2</value>
        <value>value3</value>
      </fixed-list>
      <query condition="condition2" allow_user_values="false"
        allow_caching="true" query-attribute="object_name">
        select object_name from dm_sysobject</query>
      <fixed-list "allow_user_values"="false">
        <value>value4</value>
        <value>value5</value>
        <value>value6</value>
      </fixed-list>
    </value-assist>
  </property>
  <property name="title" repeating="false" type="string" length="400"
    label="Title" hidden="false" required="false" notnull="false"
    readonly="false" searchable="true"/>
  ...
  ...
  <property name="i_partition" repeating="false" type="integer"
    label="Partition Number" hidden="true" required="false" notnull="false"
    readonly="true" searchable="true"/>
  <property name="i_is_replica" repeating="false" type="boolean"
    label="Is Replica" hidden="true" required="false" notnull="false"
    readonly="true" searchable="true"/>
  <property name="i_vstamp" repeating="false" type="integer" label="Version Stamp"
```

```

        hidden="true" required="false" notnull="false" readonly="true"
        searchable="false"/>
    </properties>
    <links>
        <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/
        types/dm_document.xml"/>
        <link rel="http://identifiers.emc.com/linkrel/types"
        href="http://localhost:8080/dctm-rest/repositories/REPO/types.xml?
        parent-type=dm_document"/>
        <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/types/dm_sysobje
    </links>
        <link rel="http://identifiers.emc.com/linkrel/assist-values"
        href="http://localhost:8080/dctm-rest/repositories/REPO/types/dm_document/
        assist-values"/>
    </links>
</type>

```

### Example 2-161. JSON Response

```

{
  "name": "dm_document",
  "label": "Document",
  "category": "standard",
  "parent": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_sysobject",
  "properties": [
    {
      "name": "object_name",
      "repeating": false,
      "type": "string",
      "length": 255,
      "label": "Name",
      "hidden": false,
      "required": false,
      "notnull": false,
      "readonly": false,
      "searchable": true,
      "dependencies": ["attr1", "attr2"],
      "value-assistance": [
        {
          "type": "fixed-list",
          "condition": "condition1",
          "allow-user-values": true,
          "values": [
            "value1", "value2", "value3"
          ]
        },
        {
          "type": "query",
          "condition": "condition2",
          "allow-user-values": false,
          "allow-caching": true,
          "query-attribute": "object_name",
          "query-expression": "select object_name from dm_sysobject"
        },
        {
          "type": "fixed-list",
          "allow-user-values": false,
          "values": [
            "value4", "value5", "value6"
          ]
        }
      ]
    },
    {
      "name": "object_name",
      "repeating": false,
      "type": "string",
      "length": 255,
      "label": "Name",
      "hidden": false,
      "required": false,
      "notnull": false,
      "readonly": false,
      "searchable": true,
      "dependencies": ["attr1", "attr2"],
      "value-assistance": [
        {
          "type": "fixed-list",
          "condition": "condition1",
          "allow-user-values": true,
          "values": [
            "value1", "value2", "value3"
          ]
        },
        {
          "type": "query",
          "condition": "condition2",
          "allow-user-values": false,
          "allow-caching": true,
          "query-attribute": "object_name",
          "query-expression": "select object_name from dm_sysobject"
        },
        {
          "type": "fixed-list",
          "allow-user-values": false,
          "values": [
            "value4", "value5", "value6"
          ]
        }
      ]
    }
  ]
}

```

```

        "name": "r_object_type",
        "repeating": false,
        "type": "string",
        "length": 32,
        "label": "Type",
        "hidden": false,
        "required": false,
        "notnull": false,
        "readonly": true,
        "searchable": true
    },
    {
        "name": "title",
        "repeating": false,
        "type": "string",
        "length": 400,
        "label": "Title",
        "hidden": false,
        "required": false,
        "notnull": false,
        "readonly": false,
        "searchable": true
    },
    ...
    ...
    {
        "name": "i_is_replica",
        "repeating": false,
        "type": "boolean",
        "label": "Is Replica",
        "hidden": true,
        "required": false,
        "notnull": false,
        "readonly": true,
        "searchable": true
    },
    {
        "name": "i_vstamp",
        "repeating": false,
        "type": "integer",
        "label": "Version Stamp",
        "hidden": true,
        "required": false,
        "notnull": false,
        "readonly": true,
        "searchable": false
    }
],
"links": [
    {
        "rel": "self",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_document"
    },
    {
        "rel": "http://identifiers.emc.com/linkrel/types",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/types?parent-type=dm_document"
    },
    {
        "rel": "parent",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_sysobject"
    },
    {

```

```

        "rel": "http://identifiers.emc.com/linkrel/assist-values",
        "href": "http://localhost:8080/dctm-rest/repositories/REPO/types/
                  dm_document/assist-values"
      }
    ]
  }

```

## Types

The Types collection resource represents the collection of type objects in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Supports POST?
Feed URI	Types collection	Server's current time	No

Entry	Entry ID	Entry Title	Entry Updated
<a href="#">Type, page 482</a>	Type URI	Display name of the type.	r_modify_date of the object

## Link Relations

The following table lists link relations for the Types collection resource.

Link Relation	Description	Resource Reference
self	This collection of type objects	<a href="#">Types, page 488</a>
first, last, next, previous	Pagination links.	<a href="#">Types, page 488</a>

## Operations

The Types collection resource supports the following HTTP method.



Method	Description
GET	Retrieves the metadata of a collection of types in the repository.

## Get Types

Retrieve the metadata of a collection of types in the repository.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following types-related query parameters:

Variable	Description	Data type	Default value
parent-type	<p>Gets direct sub types or all sub types of the specified type.</p> <p>You can combine this variable with <code>recursive</code> to indicate whether or not to return all sub types or just the direct sub types of the specified type.</p>	string	null
dml-view	<p>Specifies the elements to return.</p> <ul style="list-style-type: none"><li>• <code>all</code>: All content will be returned. This includes:<ul style="list-style-type: none"><li>— all properties of the type:</li></ul></li></ul>	string	basic

Variable	Description	Data type	Default value
	<p>name, label, and parent</p> <ul style="list-style-type: none"> <li>— all properties of the properties in the type: <ul style="list-style-type: none"> <li>name, label, type, length, repeating, hidden, notnull, readonly, required, default, and searchable</li> </ul> </li> <li>• basic: Only basic dml is returned. This includes: <ul style="list-style-type: none"> <li>— basic properties of the type: <ul style="list-style-type: none"> <li>name, and parent</li> </ul> </li> <li>— basic properties of the properties in the type: <ul style="list-style-type: none"> <li>name, type, length, and repeating</li> </ul> </li> </ul> </li> <li>• none: Only a minimum set of the content is returned. This includes the name, and the parent of the type. All information about the properties in the type is not returned.</li> </ul>		

Variable	Description	Data type	Default value
inherited	Specifies whether or not to include inherited properties from parent type. This parameter does not work when <code>dml-view</code> is set to <code>none</code> .	boolean	true
locale	Specifies the locale of the Request. This parameter only works when <code>dml-view</code> is set to <code>all</code> .	string	null

Also, this method supports the following common query parameters:

- sort
- inline
- page
- items-per-page
- include-total
- recursive
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
400 - Bad request  
401 - Authentication failed  
404 - Types cannot be found  
500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of type objects.

**Note:** The type name in the URL is encoded.

### Example 2-162. Types in XML

```
<feed>
  <id>http://localhost:8080/dctm-rest/repositories/REPO/types</id>
  <title>Types</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2014-12-16T06:45:43.571+00:00</updated>
  <dm:page>1</dm:page>
  <dm:items-per-page>100</dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/types" />
  <link rel="next" href="http://localhost:8080/dctm-rest/repositories/REPO/
types?items-per-page=100&page=2" />
  <link rel="first" href="http://localhost:8080/dctm-rest/repositories/REPO/
types?items-per-page=100&page=1" />
  <entry>
    <id>
      http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl
    </id>
    <title>dm_acl</title>
    <author>
      <name>REPO</name>
      <uri>
        http://localhost:8080/dctm-rest/repositories/REPO/users/REPO
      </uri>
    </author>
    <summary>031e848280000101</summary>
    <updated>2014-12-16T06:45:43.571+00:00</updated>
    <published>2014-12-16T06:45:43.571+00:00</published>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
types/dm_acl" />
    <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/
REPO/types/dm_acl" />
  </entry>
</feed>
```

```

</entry>
<entry>
  <id>
    http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config
  </id>
  <title>dm_acs_config</title>
  <author>
    <name>Administrator</name>
    <uri>
      http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator
    </uri>
  </author>
  <summary>031e8482800001cb</summary>
  <updated>2014-12-16T06:45:43.571+00:00</updated>
  <published>2014-12-16T06:45:43.571+00:00</published>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/REPO/
types/dm_acs_config" />
  <content type="application/xml" src="http://localhost:8080/dctm-rest/repositories/
REPO/types/dm_acs_config" />
</entry>
...
</feed>

```

### Example 2-163. Types in JSON

```

{
  id: "http://localhost:8080/dctm-rest/repositories/REPO/types"
  title: "Types"
  author:
    {
      name: "EMC Documentum"
    }
  updated: "2014-12-16T06:40:50.231+00:00"
  page: 1
  items-per-page: 100
  links:
    {
      rel: "self"
      href: "http://localhost:8080/dctm-rest/repositories/REPO/types"
    }
    {
      rel: "next"
      href:
        http://localhost:8080/dctm-rest/repositories/REPO/types?items-per-page=100&page=2"
    }
    {
      rel: "first"
      href:
        http://localhost:8080/dctm-rest/repositories/REPO/types?items-per-page=100&page=1"
    }
  entries:
    {
      id: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl"
      title: "dm_acl"
      author:
        {
          name: "REPO"
          uri: "http://localhost:8080/dctm-rest/repositories/REPO/users/REPO"
        }
      summary: "031e848280000101"
      updated: "2014-12-16T06:40:50.232+00:00"
      published: "2014-12-16T06:40:50.232+00:00"
      links:

```

```
{
  rel: "edit"
  href: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl"
}
content: {
  type: "application/json"
  src: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl"
}
}
{
  id: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config"
  title: "dm_acs_config"
  author:
    {
      name: "Administrator"
      uri: "http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator"
    }
  summary: "031e8482800001cb"
  updated: "2014-12-16T06:40:50.232+00:00"
  published: "2014-12-16T06:40:50.232+00:00"
  links:
    {
      rel: "edit"
      href: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config"
    }
  content: {
    type: "application/json"
    src: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config"
  }
}
...
}
```

**Note:** In release 7.1, the property indicating the MIME type was named as `content-type`. This name is deprecated and replaced with `type`.

#### Example 2-164. Representation in 7.1

```
content: {
  content-type: "application/json"
  src: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config"
}
```

#### Example 2-165. Current Representation

```
content: {
  type: "application/json"
  src: "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acs_config"
}
```

You can set the `rest.api.compatibility.version` property in `rest-api-runtime.properties` to 7.1 for backward compatibility.

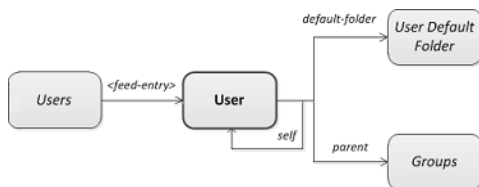
# User(s)

## User

The User resource represents the metadata of a user in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Users, page 503](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the User resource.

Link Relation	Description	Resource Reference
self	This User resource.	<a href="#">User, page 495</a>
default-folder [1]	Default folder (home cabinet) of the user.	<a href="#">User Default Folder, page 509</a>
parent	Groups that the user belongs to.	<a href="#">Groups, page 280</a>
edit	This User resource. Only SysAdmin or SuperUser can view this link	<a href="#">User, page 495</a>
delete [1]	This User resource. Only SysAdmin or SuperUser can view this link	<a href="#">User, page 495</a>
permission-set [1]	The permission set definition for the User	<a href="#">Permission Set, page 337</a>

[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: <http://identifiers.emc.com/linkrel/>

## Operations

The User resource supports the following HTTP method.

Method	Description
GET	Retrieves the metadata of a User

### Get a User

Retrieve the metadata of a user.

### Supported HTTP Method

GET

### Request Media Types

N/A

### Request Query Parameters

This method supports the following common query parameters:

- view

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

### Request Headers

- Authorization
- Accept

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

### Request Body

N/A

### Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).



## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Get successful  
 400 - Bad request  
 401 - Authentication failed  
 404 - User not found  
 500 - Other unexpected server error

## Response Body

**Note:** The password attribute is not displayed.

### Example 2-166. XML Response

```
<user xsi:type="dm_user" definition="http://localhost:8080/dctm-rest/repositories/REPO/
types/dm_user.xml">
  <properties>
    <user_name>Administrator</user_name>
    <user_os_name>Administrator</user_os_name>
    <user_address>JO@EMC.COM</user_address>
    <user_group_name>docu</user_group_name>
    <user_privileges>8</user_privileges>
    <owner_def_permit>7</owner_def_permit>
    <world_def_permit>3</world_def_permit>
    <group_def_permit>5</group_def_permit>
    <default_folder>/Administrator</default_folder>
    <r_is_group>false</r_is_group>
    <user_db_name>Administrator</user_db_name>
    <description />
    <acl_domain>Administrator</acl_domain>
    <acl_name>dm_4500000580000101</acl_name>
    <user_os_domain>restcs72ga</user_os_domain>
    <home_docbase />
    <user_state>0</user_state>
    <client_capability>8</client_capability>
    <globally_managed>false</globally_managed>
    <r_modify_date>2015-03-05T07:55:35.000+00:00</r_modify_date>
    <user_delegation />
    <workflow_disabled>false</workflow_disabled>
    <alias_set_id>0000000000000000</alias_set_id>
    <user_source>inline password</user_source>
    <user_ldap_dn />
    <user_xprivileges>0</user_xprivileges>
    <r_has_events>false</r_has_events>
    <failed_auth_attempt>-1</failed_auth_attempt>
    <user_admin />
    <user_global_unique_id>restcs72ga:Administrator</user_global_unique_id>
    <user_login_name>Administrator</user_login_name>
    <user_login_domain>restcs72ga</user_login_domain>
```

```

    <user_initials />
    <user_web_page />
    <first_failed_auth_utc_time xsi:nil="true" />
    <last_login_utc_time>2015-03-02T18:15:33.000+00:00</last_login_utc_time>
    <deactivated_utc_time xsi:nil="true" />
    <deactivated_ip_addr />
    <restricted_folder_ids xsi:nil="true" />
    <root_log_dir />
    <i_is_replica>false</i_is_replica>
    <i_vstamp>4</i_vstamp>
    <r_object_id>1100000580000102</r_object_id>
  </properties>
  <links>
    <link rel="self" href="http://localhost:8080/dctm-rest/repositories/REPO/users/
      Administrator" />
    <link rel="parent" href="http://localhost:8080/dctm-rest/repositories/REPO/
      groups.xml?user-name=Administrator" />
    <link rel="http://identifiers.emc.com/linkrel/default-folder"
      href="http://localhost:8080/dctm-rest/repositories/REPO/users/
      Administrator/home.xml" />
  </links>
</user>

```

### Example 2-167. JSON Response

```

{
  "name": "user",
  "type": "dm_user",
  "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_user.json",
  "properties": {
    "user_name": "Administrator",
    "user_os_name": "Administrator",
    "user_address": "JO@EMC.COM",
    "user_group_name": "docu",
    "user_privileges": 8,
    "owner_def_permit": 7,
    "world_def_permit": 3,
    "group_def_permit": 5,
    "default_folder": "/Administrator",
    "r_is_group": false,
    "user_db_name": "Administrator",
    "description": "",
    "acl_domain": "Administrator",
    "acl_name": "dm_4500000580000101",
    "user_os_domain": "restcs72ga",
    "home_docbase": "",
    "user_state": 0,
    "client_capability": 8,
    "globally_managed": false,
    "r_modify_date": "2015-03-05T07:55:35.000+00:00",
    "user_delegation": "",
    "workflow_disabled": false,
    "alias_set_id": "0000000000000000",
    "user_source": "inline password",
    "user_ldap_dn": "",
    "user_xprivileges": 0,
    "r_has_events": false,
    "failed_auth_attempt": -1,
    "user_admin": "",
    "user_global_unique_id": "restcs72ga:Administrator",
    "user_login_name": "Administrator",
    "user_login_domain": "restcs72ga",
    "user_initials": "",
    "user_web_page": ""
  }
}

```

```

        "first_failed_auth_utc_time":null,
        "last_login_utc_time":"2015-03-02T18:15:33.000+00:00",
        "deactivated_utc_time":null,
        "deactivated_ip_addr":"",
        "restricted_folder_ids":null,
        "root_log_dir":"",
        "i_is_replica":false,
        "i_vstamp":4,
        "r_object_id":"1100000580000102"
    },
    "links":[{"rel":"self",
"href":"http://localhost:8080/dctm-rest/repositories/REPO/users/Administrator.json"},
{"rel":"parent",
"href":"http://localhost:8080/dctm-rest/repositories/REPO/groups.json?
user-name=Administrator"},
{"rel":"http://identifiers.emc.com/linkrel/default-folder",
"href":"http://localhost:8080/dctm-rest/repositories/REPO/users/
Administrator/home.json"}]
}

```

## Update a User

Updates the attributes of a User.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/xml
- application/vnd.emc.documentum+json
- application/json

## Request Query Parameters

N/A

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

The following read-only properties cannot be updated:

- i\_vstamp
- i\_is\_replica
- r\_has\_events
- r\_modify\_date
- r\_is\_group

### Example 2-168. XML Request

```
<user xsi:type="dm_user" definition="http://localhost:8080/dctm-rest/repositories/
  REPO/types/dm_user.xml">
  <properties>
    <user_address>JO@EMC.COM</user_address>
    <user_group_name>docu</user_group_name>
    <user_privileges>8</user_privileges>
    .....
  </properties>
</user>
```

### Example 2-169. JSON Request

```
{
  "properties":{
    "user_address":"JO@EMC.COM",
    "user_group_name":"docu",
    "user_privileges":8,
    ...
  }
}
```

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Response Status

200 - Update successful  
400 - Bad request  
401 - Authentication failed  
403 - Authentication failed  
404 - User not found

## 500 - Other unexpected server error

### Response Body

**Note:** The password attribute is not displayed.

#### Example 2-170. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<dm:dm_user xmlns:dm="http://ns.emc.com/documentum"
  xmlns="http://www.w3.org/2005/Atom">
  <dm:attributes>
    <dm:r_object_id>1100000180000101</dm:r_object_id>
    <dm:user_name>acme</dm:user_name>
    <dm:user_os_name>acme</dm:user_os_name>
    <dm:user_address>acme</dm:user_address>
    <dm:user_group_name>docu</dm:user_group_name>
    <dm:user_privileges>16</dm:user_privileges>
    <dm:owner_def_permit>7</dm:owner_def_permit>
    <dm:world_def_permit>3</dm:world_def_permit>
    <dm:group_def_permit>5</dm:group_def_permit>
    <dm:default_folder href="http://localhost:8080/emc-rest/repositories/acme/users
      /61636d65/home.xml"/>/acme</dm:default_folder>
    <dm:r_is_group>false</dm:r_is_group>
    <dm:user_db_name>acme</dm:user_db_name>
    <dm:acl_domain>acme</dm:acl_domain>
    <dm:acl_name>dm_4500000180000100</dm:acl_name>
    <dm:user_os_domain>2k8vm</dm:user_os_domain>
    <dm:user_state>0</dm:user_state>
    <dm:client_capability>8</dm:client_capability>
    <dm:globally_managed>false</dm:globally_managed>
    <dm:r_modify_date>2012-08-31T22:42:42.000+0800</dm:r_modify_date>
    <dm:workflow_disabled>false</dm:workflow_disabled>
    <dm:alias_set_id>0000000000000000</dm:alias_set_id>
    <dm:user_xprivileges>0</dm:user_xprivileges>
    <dm:r_has_events>false</dm:r_has_events>
    <dm:failed_auth_attempt>0</dm:failed_auth_attempt>
    <dm:user_global_unique_id>2k8vm</dm:user_global_unique_id>
    <dm:user_login_name>acme</dm:user_login_name>
    <dm:user_login_domain>2k8vm</dm:user_login_domain>
    <dm:last_login_utc_time>2012-08-31T14:42:42.000+0800
    </dm:last_login_utc_time>
    <dm:i_is_replica>false</dm:i_is_replica>
    <dm:i_vstamp>0</dm:i_vstamp>
  </dm:attributes>
  <link rel="self"
    href="http://localhost:8080/emc-rest/repositories/acme/users/61636d65.xml"/>
  <link rel="edit"
    href="http://localhost:8080/emc-rest/repositories/acme/users/61636d65.xml"/>
  <link rel="delete"
    href="http://localhost:8080/emc-rest/repositories/acme/users/61636d65.xml"/>
  <link rel="parent"
    href="http://localhost:8080/emc-rest/repositories/acme/groups.xml?
    username=61636d65"/>
  <link rel="http://names.emc.com/link/documentum/relations/default-folder"
    href="http://localhost:8080/emc-rest/repositories/acme/users/61636d65/home.xml"/>
</dm:dm_user>
```

#### Example 2-171. JSON Response

```
{
  "attributes" : {
    "r_object_id" : "1100000180000101",
```

```
    "user_name" : "acme",
    "user_os_name" : "acme",
    "user_address" : "acme",
    "user_group_name" : "docu",
    "user_privileges" : 16,
    "owner_def_permit" : 7,
    "world_def_permit" : 3,
    "group_def_permit" : 5,
    "default_folder" : "/acme",
    "r_is_group" : false,
    "user_db_name" : "acme",
    "acl_domain" : "acme",
    "acl_name" : "dm_4500000180000100",
    "user_os_domain" : "2k8vm",
    "user_state" : 0,
    "client_capability" : 8,
    "globally_managed" : false,
    "r_modify_date" : "2012-08-31T22:42:42.000+0800",
    "workflow_disabled" : false,
    "alias_set_id" : "0000000000000000",
    "user_xprivileges" : 0,
    "r_has_events" : false,
    "failed_auth_attempt" : 0,
    "user_global_unique_id" : "2k8vm:acme",
    "user_login_name" : "acme",
    "user_login_domain" : "2k8vm",
    "last_login_utc_time" : "2012-08-31T14:42:42.000+0800",
    "i_is_replica" : false,
    "i_vstamp" : 0
  },
  "links" : [ {
    "rel" : "self",
    "href" : "http://localhost:8080/emc-rest/repositories/acme/users/61636d65.json"
  }, {
    "rel" : "edit",
    "href" : "http://localhost:8080/emc-rest/repositories/acme/users/61636d65.json"
  }, {
    "rel" : "delete",
    "href" : "http://localhost:8080/emc-rest/repositories/acme/users/61636d65.json"
  }, {
    "rel" : "parent",
    "href" : "http://localhost:8080/emc-rest/repositories/acme/groups.json?username=61636d65"
  }, {
    "rel" : "http://names.emc.com/link/documentum/relations/default-folder",
    "href" : "http://localhost:8080/emc-rest/repositories/acme/users/61636d65/home.json"
  } ]
}
```

## Delete a User

Deletes a User. Only the SysAdmin or SuperUser can perform this operation. When a user has been deleted, the deletion may have an impact on the user's authentication.

## Supported HTTP Method

DELETE

**Request Media Types**

N/A

**Request Query Parameters**

N/A

**Request Headers**

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

**Request Body**

N/A

**Response Headers**

N/A

**Response Media Types****Response Status**

- 200 - Delete successful
- 400 - Bad request
- 401 - Authentication failed
- 403 - Permission denied
- 404 - User not found
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

**Response Body**

N/A

## Users

The Users resource represents the metadata of a collection of users in a repository.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Feed ID	Feed Title	Feed Updated	Entry	Supports POST?
Feed URI	List of users	Server's current time	<a href="#">User, page 495</a>	No

Entry ID	Entry Title	Entry Summary	Entry Updated
URI of the user	User name	User description	<code>r_modify_date</code> of the user

## Link Relations

The following table lists link relations for the Users resource.

Link Relation	Description	Resource Reference
self	This collection of users.	<a href="#">Users, page 503</a>
first, last, next, previous	Pagination links.	<a href="#">Users, page 503</a>

## Operations

The Users resource supports the following HTTP method.

Method	Description
GET	Retrieves the metadata of a collection of users.

## Get Users

Retrieve the metadata of a collection of users with the specified filters.



## Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- sort
- view
- inline
- page
- items-per-page
- include-total
- links
- filter

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

- 200 - Retrieved successfully
- 400 - Bad request
- 401 - Authentication failed
- 500 - Other unexpected server error

## Response Body

XML or JSON representation of the collection of users.

**Note:** The user name in the URL is encoded.

## Create a User

Create a user in a repository. The Sysadmin, SuperUser, or any other users that have the Create Group privilege can create a user.

## Supported HTTP Method

POST

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Request Query Parameters

N/A

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

- The `user_name` property is mandatory
- The `user_login_name` or `user_os_name` is mandatory

### Example 2-172. Request Body in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<dm:dm_user xmlns:dm="http://ns.emc.com/documentum" xmlns="http://www.w3.org/2005/Atom">
  <dm:properties>
    <dm:user_name>acme</dm:user_name>
    <dm:user_login_name>acme</dm:user_login_name>
    // ...
  </dm:properties>
</dm:dm_user>
```

### Example 2-173. Request Body in JSON

```
{
  "properties" : {
    "user_name" : "acme",
    "user_login_name" : "acme"
    // ...
  }
}
```

## Response Headers

- Location - The URL of the newly created user.

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- `application/vnd.emc.documentum+xml`
- `application/vnd.emc.documentum+json`

## Response Status

201 - Created successfully

400 - Bad request such as providing invalid attributes

401 - Authentication failed

403 - Permission denied

500 - Other unexpected server error

For more information, see

## Response Body

N/A

# User Default Folder

The User Default Folder resource represents the default folder (typically, the home cabinet) for a given user.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources.



See Also: [User, page 495](#) and [About the Diagram](#).

## Link Relations

The following table lists link relations for the User Default Folder resource.

Link Relation	Description	Resource Reference
self	This default folder.	<a href="#">User Default Folder, page 509</a>
canonical	Canonical URI for the default folder resource.	<a href="#">Cabinet, page 120</a> <a href="#">Folder, page 209</a>

## Operations

The default folder resource supports the following HTTP method.

Method	Description
GET	Retrieves the default folder for a given user.

### Get a User Default Folder

Retrieve information about the default folder for a given user. Typically, the default folder is a cabinet (`dm_cabinet`) and is called "home cabinet." The default folder can also be an instance of `dm_folder` or its subtypes.

### Supported HTTP Method

GET

## Request Media Types

N/A

## Request Query Parameters

This method supports the following common query parameters:

- view
- links

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Accept
- Authorization

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Body

N/A

## Response Headers

- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml
- application/json

## Response Status

200 - Retrieved successfully  
401 - Authentication failed  
403 - Permission denied  
404 - User not found

500 - Other unexpected server error

## Response Body

XML or JSON representation of the user's default folder.

**Note:** The user name in the URL is encoded.

# Value Assistance

The Value Assistance resource represents the value assistances of a Type.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:

See Also: [Repository, page 369](#) and [About the Diagram](#).

## Feed

Is feed? No.

## Link Relations

Link Relation	Description	Resource Reference
self	This document object resource	<a href="#">Value Assistance, page 512</a>

## Supported HTTP Methods

The Value Assistance resource supports the following HTTP method.

Method	Description
POST	Retrieves the Value Assistance and dependency values of the Type

## Operations

### Get the Value Assistance of the Type

Get the Value Assistance and dependency properties of the specified Type



## Request URI Query

The following query parameters are specific to this operation:

Variable	Description	Data Type	Default Value
included-properties	The property names to be included in the Response.  The property names are comma separated. For example, <code>included-properties=attr1,attr2,attr3</code>	string	All properties belonging to the Type

## Request Headers

- Authorization
- Accept
- Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

## Request Body

### Example 2-174. XML Request

The input available for one property of a certain type may depend on the value of other properties. When you want to get the assist values for a property, you must provide the dependent values.

Resources return *multi-properties* assist values, therefore you must provide all the dependent values the assist values require.

You can find every property's dependencies information in the type resource, within the <dependencies> element under the <property> element. Here's a code sample that illustrates this point:

```
<assist-value-request>
  <!-- Includes all of the dependency values for input -->
  <properties>
    <!-- Some assist values may be dependent upon the value of attr1,
         so this element provides the value of attr1.-->
    <attr1>value1</attr1>
    <!-- When the dependency value is a list, input as an array is supported,
         as shown here in the attr2 element-->
    <attr2>
      <item>XXX</item>
```

```
        <item>YYY</item>
      </attr2>
      <attr3>4</attr3>
    </properties>
  </assist-value-request>
```

**Example 2-175. JSON Request**

```
{
  "properties":
  {
    "attr1": "value1",
    "attr2": ["XXX", "YY"],
    "attr3": 3
  }
}
```

**Response Status**

- 200 - Get successful
- 400 - Bad request
- 401 - Authentication failed
- 403 - Permission denied
- 404 - Types cannot be found
- 500 - Other unexpected server error

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

**Response Headers**

Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

**Response Media Types**

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

**Response Body****Example 2-176. XML Response**

```
<?xml version='1.0' encoding='UTF-8'?>
<assist-values>
  <properties>
    <attr1 allow-user-values="false">
      <value label="One">1</value>
      <value label="Two">2</value>
    </attr1>
    <attr2 allow-user-values="false">
      <value label="Three">3</value>
      <value label="Four">4</value>
```

```
        </attr2>
    </properties>
    <links>
        <link rel="self" href="XXX"/>
    </links>
</assist-values>
```

**Example 2-177. JSON Response**

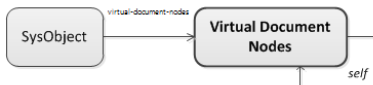
```
{  "properties": {
    "attr1": {
        "allow-user-values": false,
        "values": [
            {"value": 1, "label": "One"},
            {"value": 2, "label": "Two"},
        ]
    },
    "attr2": {
        "allow-user-values": true,
        "values": [
            {"value": 3, "label": "Three"},
            {"value": 4, "label": "Four"},
        ]
    }
  },
  "links": [
    {"rel": "yyy", "href": "xxx"}
  ]
}
```

# Virtual Document Nodes

The Virtual Document nodes resource represents the virtual document hierarchy of the flatten feed.

## Resource Relationships

The following diagram illustrates how this resource is related to other resources:



For more information, see [SysObject](#), page 472 and [About the Diagram](#).

## Feed

Is feed? Yes.

Feed ID	Feed Title	Entry	Supports POST?
Feed URI	Virtual Document Nodes	Link to a node or a sub tree	No

Entry ID	Entry Title	Entry Updated
Object ID of the object related to the node	Name of the object related to the node	<code>r_modify_date</code> of the object related to the node

## Link Relations

The following table lists link relations for the Virtual Document Nodes resource.

Link Relation	Description	Resource Reference
self	This virtual document nodes resource	<a href="#">Virtual Document Nodes, page 516</a>
first, last, next, and previous	Pagination links to additional virtual document nodes	Used for navigation when more than one page of results are available

## Supported HTTP Methods

The Virtual Document Nodes resource supports the following HTTP method.

Method	Description
GET	Gets the virtual document hierarchy

## Operations

### Get a Virtual Document Hierarchy

Get the virtual document hierarchy of a specified object. The object can be a virtual document or an assembly object.

#### Request URI Query

The following query parameters are specific to this operation:

Variable	Description	Data Type	Value Range	Default Value
binding-label	The symbolic version label to use when resolving late binding nodes.	string	n/a	null
follow-assembly	Determines whether the server selects a component's descendants using the containment objects or a snapshot associated with the component. The Server selects components from the snapshot when the follow-assembly property is set to true and a snapshot is available.	boolean	true, false	false
include-broken	Specifies whether to display nodes with broken bindings using the CURRENT label.	boolean	true, false	false
depth	Determines the level of depth that the virtual document hierarchy will be returned. Setting a value of -1 will return the whole virtual directory tree.	int	-1 or $\geq 0$	1
vdm-number	The node's number in the virtual document tree. For example 1.2.3	string	n/a	null

This method also supports the following common query parameters:

- links
- inline
- page
- items-per-page

For more information, see [Appendix B, REST Common Definition - URI Request Query Parameters](#).

## Request Headers

- Authorization
- Accept

## Request Media Types

N/A

## Request Body

N/A

## Response Status

- 200 Operation was successful
- 400, 401, 403, 404, 406, 409, 415, 500

For more information, see [Appendix C, REST Common Definition - HTTP Status Codes](#).

## Response Headers

Content-Type

For more information, see [Appendix D, REST Common Definition - HTTP Headers](#).

## Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json

## Response Body

XML or JSON representation of the virtual document.

- The body contains a list of nodes in the virtual document.
- vdm-number is presented in the `summary` property of each node except the root.
- The collection of nodes returned only include those nodes that the REST client has access to.
- Pagination is supported.

### Example 2-178. XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <id>http://localhost:8080/dctm-rest/repositories/REPO/objects/
    0900000580004876/vd-nodes</id>
  <title>Virtual Document Nodes</title>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2016-02-17T07:54:32.322+00:00</updated>
  <dm:page xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:page>
  <dm:items-per-page xmlns:dm="http://identifiers.emc.com/vocab/documentum">100
  </dm:items-per-page>
  <link
    href="http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004876/
      vd-nodes?inline=true" rel="self"/>
  <entry>
    <id>0900000580004876</id>
    <title>grand</title>
    <updated>2016-02-17T07:46:44.000+00:00</updated>
    <published>2016-02-17T07:46:44.000+00:00</published>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004876/vd-nodes?inline=true" rel="edit"/>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004876" rel="alternate"/>
    <content>
      <dm:virtual-document-node xmlns:dm="http://identifiers.emc.com/vocab/
        documentum">
        <dm:are-children-compound>false</dm:are-children-compound>
        <dm:available-versions>
          <dm:item>CURRENT</dm:item>
          <dm:item>1.0</dm:item>
        </dm:available-versions>
        <dm:binding>CURRENT</dm:binding>
        <dm:can-be-removed>true</dm:can-be-removed>
        <dm:can-be-restructured>true</dm:can-be-restructured>
        <dm:child-count>2</dm:child-count>
        <dm:chronicle-id>0900000580004876</dm:chronicle-id>
        <dm:copy-behavior>0</dm:copy-behavior>
        <dm:follow-assembly>false</dm:follow-assembly>
        <dm:node-id>0000000000000000</dm:node-id>
        <dm:is-binding-broken>false</dm:is-binding-broken>
        <dm:is-compound>false</dm:is-compound>
        <dm:is-from-assembly>false</dm:is-from-assembly>
        <dm:is-structurally-modified>false
          </dm:is-structurally-modified>
        <dm:is-virtual-document>true</dm:is-virtual-document>
      </dm:virtual-document-node>
    </content>
  </entry>
</feed>
```

```

    <dm:late-binding-value>CURRENT</dm:late-binding-value>
    <dm:override-late-binding>true</dm:override-late-binding>
    <dm:selected-object-name>grand</dm:selected-object-name>
    <dm:selected-object-id>0900000580004876
  </dm:selected-object-id>
  <dm:links>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004876/vd-nodes?inline=true"
      rel="http://identifiers.emc.com/linkrel/virtual-document-nodes"/>
    <link
      href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004876" rel="alternate"/>
  </dm:links>
</dm:virtual-document-node>
</content>
</entry>
<entry>
  <id>0900000580004877</id>
  <title>parent-0</title>
  <summary>1</summary>
  <updated>2016-02-17T07:46:44.000+00:00</updated>
  <published>2016-02-17T07:46:44.000+00:00</published>
  <link
    href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0900000580004876/vd-nodes?inline=true&object-id=0900000580004877&
      vdm-number=1" rel="edit"/>
  <link
    href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0900000580004877" rel="alternate"/>
  <content>
    <dm:virtual-document-node xmlns:dm="http://identifiers.emc.com/vocab/
      documentum">
      <dm:are-children-compound>false</dm:are-children-compound>
      <dm:available-versions>
        <dm:item>CURRENT</dm:item>
        <dm:item>1.0</dm:item>
      </dm:available-versions>
      <dm:binding/>
      <dm:can-be-removed>true</dm:can-be-removed>
      <dm:can-be-restructured>true</dm:can-be-restructured>
      <dm:child-count>5</dm:child-count>
      <dm:chronicle-id>0900000580004877</dm:chronicle-id>
      <dm:copy-behavior>0</dm:copy-behavior>
      <dm:follow-assembly>false</dm:follow-assembly>
      <dm:node-id>0000000000000001</dm:node-id>
      <dm:is-binding-broken>false</dm:is-binding-broken>
      <dm:is-compound>false</dm:is-compound>
      <dm:is-from-assembly>false</dm:is-from-assembly>
      <dm:is-structurally-modified>false</dm:is-structurally-modified>
      <dm:is-virtual-document>true</dm:is-virtual-document>
      <dm:late-binding-value>CURRENT</dm:late-binding-value>
      <dm:override-late-binding>true</dm:override-late-binding>
      <dm:vdm-number>1</dm:vdm-number>
      <dm:selected-object-name>parent-0</dm:selected-object-name>
      <dm:selected-object-id>0900000580004877</dm:selected-object-id>
      <dm:links>
        <link
          href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
            0900000580004876/vd-nodes?inline=true&object-id=0900000580004877
            &vdm-number=1"
          rel="http://identifiers.emc.com/linkrel/virtual-document-nodes"/>
        <link
          href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
            0900000580004877" rel="alternate"/>
      </dm:links>
    </dm:virtual-document-node>
  </content>
</entry>

```



```

        </dm:links>
    </dm:virtual-document-node>
</content>
</entry>
<entry>
    <id>090000058000487d</id>
    <title>parent-1</title>
    <summary>2</summary>
    <updated>2016-02-17T07:46:44.000+00:00</updated>
    <published>2016-02-17T07:46:44.000+00:00</published>
    <link
        href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
            0900000580004876/vd-nodes?inline=true&object-id=090000058000487d&
            vdm-number=2" rel="edit"/>
    <link
        href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
            090000058000487d" rel="alternate"/>
    <content>
        <dm:virtual-document-node
            xmlns:dm="http://identifiers.emc.com/vocab/documentum">
            <dm:are-children-compound>false</dm:are-children-compound>
            <dm:available-versions>
                <dm:item>CURRENT</dm:item>
                <dm:item>1.0</dm:item>
            </dm:available-versions>
            <dm:binding/>
            <dm:can-be-removed>true</dm:can-be-removed>
            <dm:can-be-restructured>true</dm:can-be-restructured>
            <dm:child-count>5</dm:child-count>
            <dm:chronicle-id>090000058000487d</dm:chronicle-id>
            <dm:copy-behavior>0</dm:copy-behavior>
            <dm:follow-assembly>false</dm:follow-assembly>
            <dm:node-id>0000000000000002</dm:node-id>
            <dm:is-binding-broken>false</dm:is-binding-broken>
            <dm:is-compound>false</dm:is-compound>
            <dm:is-from-assembly>false</dm:is-from-assembly>
            <dm:is-structurally-modified>false</dm:is-structurally-modified>
            <dm:is-virtual-document>true</dm:is-virtual-document>
            <dm:late-binding-value>CURRENT</dm:late-binding-value>
            <dm:override-late-binding>true</dm:override-late-binding>
            <dm:vdm-number>2</dm:vdm-number>
            <dm:selected-object-name>parent-1</dm:selected-object-name>
            <dm:selected-object-id>090000058000487d</dm:selected-object-id>
            <dm:links>
                <link
                    href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
                        0900000580004876/vd-nodes?inline=true&
                        object-id=090000058000487d&vdm-number=2"
                    rel="http://identifiers.emc.com/linkrel/virtual-document-nodes"/>
                <link
                    href="http://localhost:8080/dctm-rest/repositories/REPO/objects/
                        090000058000487d" rel="alternate"/>
            </dm:links>
        </dm:virtual-document-node>
    </content>
</entry>
</feed>

```

### Example 2-179. JSON Response

```

{
  "id" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004876/
    vd-nodes",
  "title" : "Virtual Document Nodes",

```

```
"author" : [
  {
    "name" : "EMC Documentum"
  }
],
"updated" : "2016-02-17T07:53:50.326+00:00",
"page" : 1,
"items-per-page" : 100,
"links" : [
  {
    "rel" : "self",
    "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
      0900000580004876/vd-nodes?inline=true"
  }
],
"entries" : [
  {
    "id" : "0900000580004876",
    "title" : "grand",
    "updated" : "2016-02-17T07:46:44.000+00:00",
    "published" : "2016-02-17T07:46:44.000+00:00",
    "links" : [
      {
        "rel" : "edit",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000580004876/vd-nodes?inline=true"
      },
      {
        "rel" : "alternate",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000580004876"
      }
    ]
  },
  {
    "content" : {
      "name" : "virtual-document-node",
      "are-children-compound" : false,
      "available-versions" : [
        "CURRENT",
        "1.0"
      ],
      "binding" : "CURRENT",
      "can-be-removed" : true,
      "can-be-restructured" : true,
      "child-count" : 2,
      "chronicle-id" : "0900000580004876",
      "copy-behavior" : 0,
      "follow-assembly" : false,
      "node-id" : "0000000000000000",
      "is-binding-broken" : false,
      "is-compound" : false,
      "is-from-assembly" : false,
      "is-structurally-modified" : false,
      "is-virtual-document" : true,
      "late-binding-value" : "CURRENT",
      "override-late-binding" : true,
      "selected-object-name" : "grand",
      "selected-object-id" : "0900000580004876",
      "links" : [
        {
          "rel" : "http://identifiers.emc.com/linkrel/virtual-document-nodes",
          "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
            0900000580004876/vd-nodes?inline=true"
        },
        {
          "rel" : "alternate",
```

```

        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000580004876"
      }
    ]
  }
},
{
  "id" : "0900000580004877",
  "title" : "parent-0",
  "summary" : "1",
  "updated" : "2016-02-17T07:46:44.000+00:00",
  "published" : "2016-02-17T07:46:44.000+00:00",
  "links" : [
    {
      "rel" : "edit",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004876/vd-nodes?inline=true&object-id=0900000580004877&
        vdm-number=1"
    },
    {
      "rel" : "alternate",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
        0900000580004877"
    }
  ],
  "content" : {
    "name" : "virtual-document-node",
    "are-children-compound" : false,
    "available-versions" : [
      "CURRENT",
      "1.0"
    ],
    "binding" : "",
    "can-be-removed" : true,
    "can-be-restructured" : true,
    "child-count" : 5,
    "chronicle-id" : "0900000580004877",
    "copy-behavior" : 0,
    "follow-assembly" : false,
    "node-id" : "0000000000000001",
    "is-binding-broken" : false,
    "is-compound" : false,
    "is-from-assembly" : false,
    "is-structurally-modified" : false,
    "is-virtual-document" : true,
    "late-binding-value" : "CURRENT",
    "override-late-binding" : true,
    "vdm-number" : "1",
    "selected-object-name" : "parent-0",
    "selected-object-id" : "0900000580004877",
    "links" : [
      {
        "rel" : "http://identifiers.emc.com/linkrel/virtual-document-nodes",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000580004876/vd-nodes?inline=true&
          object-id=0900000580004877&vdm-number=1"
      },
      {
        "rel" : "alternate",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
          0900000580004877"
      }
    ]
  }
},
},

```

```
{
  "id" : "090000058000487d",
  "title" : "parent-1",
  "summary" : "2",
  "updated" : "2016-02-17T07:46:44.000+00:00",
  "published" : "2016-02-17T07:46:44.000+00:00",
  "links" : [
    {
      "rel" : "edit",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000580004876/vd-nodes?inline=true&
object-id=090000058000487d&vdm-number=2"
    },
    {
      "rel" : "alternate",
      "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
090000058000487d"
    }
  ],
  "content" : {
    "name" : "virtual-document-node",
    "are-children-compound" : false,
    "available-versions" : [
      "CURRENT",
      "1.0"
    ],
    "binding" : "",
    "can-be-removed" : true,
    "can-be-restructured" : true,
    "child-count" : 5,
    "chronicle-id" : "090000058000487d",
    "copy-behavior" : 0,
    "follow-assembly" : false,
    "node-id" : "0000000000000002",
    "is-binding-broken" : false,
    "is-compound" : false,
    "is-from-assembly" : false,
    "is-structurally-modified" : false,
    "is-virtual-document" : true,
    "late-binding-value" : "CURRENT",
    "override-late-binding" : true,
    "vdm-number" : "2",
    "selected-object-name" : "parent-1",
    "selected-object-id" : "090000058000487d",
    "links" : [
      {
        "rel" : "http://identifiers.emc.com/linkrel/virtual-document-nodes",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
0900000580004876/vd-nodes?inline=true&
object-id=090000058000487d&vdm-number=2"
      },
      {
        "rel" : "alternate",
        "href" : "http://localhost:8080/dctm-rest/repositories/REPO/objects/
090000058000487d"
      }
    ]
  }
}
]
```

}



# Appendix A

## Link Relations

Documentum Platform REST Services uses the following link relations.

**Table 1. Public Link Relations**

Link Relation	Description	Specification
about	Returns product information	<a href="http://tools.ietf.org/html/rfc6903">http://tools.ietf.org/html/rfc6903</a>
canonical	Designates an Internationalized Resource Identifier (IRI) as preferred over resources with duplicative content.	<a href="http://tools.ietf.org/html/rfc6596">http://tools.ietf.org/html/rfc6596</a>
child	Points to a hierarchical child, or subobject, of the current object.	<a href="http://www.w3.org/MarkUp/draft-ietf-html-relrev-00.txt">http://www.w3.org/MarkUp/draft-ietf-html-relrev-00.txt</a>
contents	Points to the contents metadata feed for a content object.	<a href="http://www.w3.org/TR/1999/REC-html401-19991224">http://www.w3.org/TR/1999/REC-html401-19991224</a>
edit	Points to a resource that can be used to edit the link's context.	<a href="http://tools.ietf.org/html/rfc5023">http://tools.ietf.org/html/rfc5023</a>
enclosure	Identifies a related resource that is potentially large and might require special handling.	<a href="http://bitworking.org/projects/atom/rfc5023.html#new-link-relation">http://bitworking.org/projects/atom/rfc5023.html#new-link-relation</a>
first	An IRI that refers to the furthest preceding resource in a series of resources.	<a href="http://tools.ietf.org/html/rfc5005">http://tools.ietf.org/html/rfc5005</a>
icon	Points to the icon for an entry, a page or a site.	<a href="http://www.w3.org/TR/html5/links.html#rel-icon">http://www.w3.org/TR/html5/links.html#rel-icon</a>
last	An IRI that refers to the furthest following resource in a series of resources.	<a href="http://tools.ietf.org/html/rfc5005">http://tools.ietf.org/html/rfc5005</a>
next	Indicates that the link's context is a part of a series, and that the next in the series is the link target.	<a href="http://tools.ietf.org/html/rfc5005">http://tools.ietf.org/html/rfc5005</a>

Link Relation	Description	Specification
parent	Refers to one of the following: <ul style="list-style-type: none"> <li>• parent type of a type</li> <li>• parent folder of a sysobject</li> <li>• parent document of a document</li> <li>• parent object of a relation</li> <li>• parent group of a group, a role, or a user.</li> </ul>	<a href="http://www.w3.org/MarkUp/draft-ietf-html-relrev-00.txt">http://www.w3.org/MarkUp/draft-ietf-html-relrev-00.txt</a>
predecessor-version	Points to a resource containing the predecessor version in the version history.	<a href="http://tools.ietf.org/html/rfc5829">http://tools.ietf.org/html/rfc5829</a>
previous	Points to the previous resource in an ordered series of resources.	<a href="http://tools.ietf.org/html/rfc5005">http://tools.ietf.org/html/rfc5005</a>
related	Points to the feed for a sysobject related to specified relation type(s).	<a href="http://tools.ietf.org/html/rfc4287">http://tools.ietf.org/html/rfc4287</a>
self	Conveys an identifier for the link's context.	<a href="http://www.ietf.org/rfc/rfc4287.txt">http://www.ietf.org/rfc/rfc4287.txt</a>
version-history	Points to a resource containing the version history for the context.	<a href="http://tools.ietf.org/html/rfc5829">http://tools.ietf.org/html/rfc5829</a>

Table 2. Link Relations Introduced in Documentum Platform REST Services

Link Relation	Description
acl	Points to the ACL resource for a specific permission set object
acls	Points to the ACLs feed under a repository
as-search-template	Points to the save as search template behavior for a saved search
associations	Points to the associated Sysobjects feed for a specific ACL object
assist-values	Points to the type assist values resource to obtain the value assistance of a type
aspect-types	Points to the aspect types resource to show all aspect types of a repository
aspect-type	Points to the single aspect type resource
batches	Points to the batches resource under a repository.



batch-capabilities	Points to the batch capabilities resource under a repository.
cabinets	Points to the cabinets feed under a repository.
cancel-checkout	Points to the cancel-checkout behavior for a versionable object.
checkin-branch	Points to the checkin as branch version behavior for an object that is checked out.
checkin-next-major	Points to the checkin as next major version behavior for an object that is checked out.
checkin-next-minor	Points to the checkin as next minor version behavior for an object that is checked out.
checkout	Points to the checkout behavior for a versionable object .
child-links	Points to the child links feed resource of a folder.
content-media	Indicates that the content can be downloaded.
comments	Points to top level comments for a sysobject.
current-user	Points to the current login user resource under a repository.
current-user-preferences	Points to the current user preferences resource to manipulate the user preference
current-version	Points to the current version resource for a versionable object.
default-folder	Points to the default folder resource for a user.
delete	Points to the delete behavior of an object.
dematerialize	link to dematerialize a lightweight object
documents	Points to the documents feed under a repository.
folders	Points to the folders feed under a repository.
format	Points to the format resource for a content resource.
formats	Points to the formats feed under a repository.
groups	Points to the groups feed under a repository or direct member groups under a group.
lightweight-types	Points to the types resource which contains the collection of children lightweight types of a shareable type
lightweight-objects	Points to the collection of lightweight objects shares one shareable object
logout	Points to the log out action for single sign on user.
materialize	link to materialize a lightweight object

objects	lists folder contents.
object-aspects	Points to the object aspects resource to get attached aspects of an object, or attaches an aspect to the object
parent-links	Points to the parent links resource of a non-cabinet sysobject.
parent-shareable-type	Points to the parent shareable type of a lightweight type
permissions	Points to the permissions resource for a Sysobject
permission-set	Points to the permission set resource for a Sysobject or a user
primary-content	Points to the primary content resource for a content sysobject type.
relate	Indicates that the object can be related to other objects.
relation-type	Points to the relation type resource for a relation instance.
relation-types	Points to the relation types feed under a repository.
relations	Points to the relations feed under a repository, relations feed for a specific relation type, or relations feed related to a specified object.
replies	Points to replied comments for a parent comment.
repositories	Points to the repositories feed from the docbrokers.
saved-searches	Points to the saved searches resource under a repository
search-templates	Points to the search templates resource under a repository
search-execution	Points to the execution behavior for a saved search or search template
saved-search-results	Points to the results resource of a saved search
shared-parent	Points the parent shareable object of a lightweight object
types	Points to the data dictionary types feed under a repository.
users	Points to the users feed under a repository, or direct member users under a group.

virtual-document-nodes	Points to the virtual document nodes resource to fetch all nodes of an virtual document
The fully qualified Documentum link relation path is prefixed with the following string: <code>http://identifiers.emc.com/linkrel/</code>	



## Appendix B

# REST Common Definition - URI Request Query Parameters

Table 3. URI Request Query Parameters

Parameter Name	Description	Data Type	Value Range	Default Value
inline	Whether or not to show content (the object instance) in the atom entry for a collection	boolean	true, false  When set to <code>true</code> it returns the object instance embedded into the entry's content element  When <code>false</code> it does not return an object instance	false  <b>Note:</b> Although the default value is <code>false</code> , we recommend that you try setting this parameter to <code>true</code> in your development environment to get the entire content of resources and explore the complete data structure.  In your production environment, you can set this parameter to <code>false</code> for better performance.

Parameter Name	Description	Data Type	Value Range	Default Value
items-per-page	<p>The number of entries per page</p> <p>When the result feed is paged, an element <code>&lt;dm:items-per-page&gt;xx&lt;/dm:items-per-page&gt;</code> is displayed under the feed root</p>	integer	<p><math>\geq 1</math></p>	100
page	<p>The number of the page to be served</p> <p>When the result feed is paged, an element <code>&lt;dm:page&gt;xx&lt;/dm:page&gt;</code> is displayed under the feed root</p>	integer	<p><math>\geq 1</math></p> <p>When set to true, an element <code>&lt;dm:total&gt;xx&lt;/dm:total&gt;</code> is displayed under the feed root</p>	<p>1</p> <p>For example when <code>item_per_page=200</code> &amp; <code>page=2</code></p> <p>Returns items number 201 to 400</p>
include-total	Calculates the total number of feed items instead of returning them all in one page	boolean	<p>true, false</p> <p>When true the total number of feed items are calculated by the server and returned</p> <p>When false no calculation of the total count is done</p>	false
sort	Sorting for entries in a collection result	string	<p>Sort consists of multiple sort specifications, separated by comma (',') character.</p> <p>Each sort specification consists of an attribute to be sorted and its sort</p>	Varies by resource and each resources has its own default order

Parameter Name	Description	Data Type	Value Range	Default Value
			<p>order, separated by ' '; character.</p> <p>Sort order can be either "DESC" or "ASC", case insensitive</p> <p>Sort order is optional. When not specified, the default sort order is "ASC".</p> <p>Optionally sort can be specified with non-repeating attributes. For example: sort=r_modify_date desc,object_id asc,title</p> <p>When an attribute with an invalid name is specified, an error is thrown</p>	
view	Properties to return	<p>string</p> <p>For more information, see the Property View section of the <i>EMC® Documentum® Platform REST Services Development Guide</i>.</p>	<p>e.g. ?view=[view name]?(.column)*</p> <p>The <i>view-name</i> and <i>columns*</i> parameters must be mutually exclusive.</p> <p>Names of predefined views start with a colon ":" character.</p> <p>The following <i>view-name</i> values are defined:</p> <ul style="list-style-type: none"> <li>• all</li> <li>• default</li> </ul>	default

Parameter Name	Description	Data Type	Value Range	Default Value
			When no <i>view-name</i> is specified, the names of properties or predefined views are returned as a comma separated list.	
links	Whether or not the link relations are returned for this object representation	boolean	true, false  A value of false means that no links that are too expensive to calculate are returned. Only easy to navigate links are returned	true
recursive	When get children of an object, whether include all indirect children recursively, e.g. children's children	boolean	true, false	false
dql	a DQL query	DQL expression	N/A	N/A
filter	A filter expression in a subset of XPath  For more information, see the Filter Expression section of the <i>EMC® Documentum® Platform REST Services Development Guide</i> .	N/A	N/A	no filtering
q	Specifies full text search criterion when sending	string	String that follows the simple	N/A



Parameter Name	Description	Data Type	Value Range	Default Value
	<p>a GET request to the <a href="#">Search</a> resource or certain collection resources</p> <p>For more information, see the Full Text Query in Collection Resources section of the <i>EMC® Documentum® Platform REST Services Development Guide</i>.</p>		search language syntax	

**Note:** Please see the individual resource sections to see how each of the above parameters are used, as this may vary by resource.

**Note:** When the *view* parameter is used on a single object resource that supports this parameter (such as the Cabinet resource, Document resource, etc.), the *view* attribute names must match the definition of the object data type. A Bad request (400) error is returned if any unknown attributes for the object data type are specified in the *view* parameter.

When the *view* parameter is used on a collection based resource that supports this parameter (such as the Cabinets resource, Folder Child Documents resource, etc.), a comma separated list of *view* attributes can contain both base type attribute names and arbitrary extended attribute names.

For example, let's assume that a folder X has a number of documents in it with the object type *dm\_document* and document sub types *doc\_ext1*, *doc\_ext2*, and so on. When a REST client retrieves the folder children, it can specify the *view* attributes from both *dm\_document* and its sub types.

Here is a code sample that shows you the Request:

```
// Get a folder child documents feed.
// In this sample, attributes 'r_object_id' and 'object_name' come from dm_document.
// 'doc_ext1_attr' comes from doc_ext1, and 'doc_ext2_attr' comes from dmc_ext2.
GET /documents?inline=true&
    view=r_object_id,object_name,doc_ext1_attr,doc_ext2_attr HTTP/1.1
```

Please also note the following *view* usage on a collection based resource:

- Aspect type attributes can also be put into the *view* when the aspect type is attachable to the collection base type, such as `. dm_document` is in the above sample.
- Collection items return the specified attributes only when the attributes exist in those objects.
- Unknown attributes for collection items are ignored.

# REST Common Definition - HTTP Status Codes

Table 4.

HTTP Status	Description
200 OK	The Request has succeeded. The information returned with the response is dependent on the method used in the Request, for example: GET an entity corresponding to the requested resource is sent in the response; PUT an entity describing or containing the modified resource
201 Created	The Request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the entity of the response, with the most specific URI for the resource given by a Location header field. The entity format is specified by the media type given in the Content-Type header field
204 No Content	The Server fulfilled the Request, but does not need to return a response message body
304 Not Modified	The Request performs a GET on the resource, but the resource has not been modified since last GET
3xx Redirections	Future use
400 Bad Request	The Request could not be understood by the server due to malformed syntax, missing or invalid information (such as validation error on an input field, a missing required value, and so on). The client SHOULD NOT repeat the Request without modifications
401 Unauthorized	The authentication credentials included with this request are missing or invalid. The Response MUST include a WWW-Authenticate header field containing a challenge applicable to the requested resource
403 Forbidden	The Server recognized your credentials, but you do not possess authorization to perform this request, and the Request SHOULD NOT be repeated
404 Not Found	The Request specified a URI of a resource that does not exist

HTTP Status	Description
405 Method Not Allowed	The HTTP verb specified in the Request (DELETE, GET, HEAD, POST, PUT) is not supported for this request URI
406 Not Acceptable	The resource identified by this request is not capable of generating a representation corresponding to one of the media types in the Accept header of the Request
409 Conflict	A creation or update request could not be completed, because it would cause a conflict in the current state of the resources supported by the server (for example, an attempt to create a new resource with a unique identifier already assigned to some existing resource)
415 Unsupported Media Type	The Server is refusing to service the Request because the entity of the Request is in a format not supported by the requested resource for the requested method
500 Internal Server Error	The Server encountered an unexpected condition which prevented it from fulfilling the Request

## Appendix D

# REST Common Definition - HTTP Headers

HTTP Header	Description	In Request or Response	Value Range
Authorization	The authorization header for authentication	Request	<p>BASE64 encoded HTTP basic authentication header with encoded credential.</p> <p>For example:</p> <pre>Authorization: Basic QWxhZGRpbjpvcGVu IHNlc2FtZQ==</pre> <p>Or, Kerberos authentication header with encoded credential.</p> <p>For example:</p> <pre>Authorization: Negotiate YIIZG1hZG1p bjpwYXNzd29yZ...</pre> <p><b>Note:</b> Kerberos constrained delegation is supported.</p>
Accept	Acceptable media type for the response	Request	See below
Content-Type	Media type of the Request or response body	Request Response	See below

HTTP Header	Description	In Request or Response	Value Range
Content-Length	The length of the Request body in octets (8-bit bytes)	Request Response	Non-negative number
Location	URI of the newly created resource	Response	URI
ETag	ETag value generated by REST server	Response	String value
If-None-Match	Checks whether the resource has changed	Request	ETag value in previous server's response
Last-Modified	Last modified time of the resource	Response	HTTP-Date
If-Modified-Since	Checks whether the resource has changed since last modified time	Request	Last-Modified value in the previous server's response
Set-Cookie	Set an HTTP cookie	Response	Used by client token
WWW-Authenticate	Indicates the authentication scheme that should be used to access the requested entity	Response	Used by http basic authentication and http Kerberos to negotiate authentication

**Note:** The Accept request-header field can be used to specify certain media types which are acceptable for the response. Accept headers can be used to indicate that the Request is specifically limited to a small set of desired types, as in the case of a request for an in-line image.

The asterisk "\*" character is used to group media types into ranges, with "\*/\*" indicating all media types and "type/\*" indicating all subtypes of that type. The media-range MAY include media type parameters that are applicable to that range.

Each media-range can be followed by one or more `accept-params`, beginning with the "q" parameter for indicating a relative quality factor. The first "q" parameter (if any) separates the media-range parameter(s) from the `accept-params`. Quality factors allow the user or user agent to indicate the relative degree of preference for that media-range, using the `qvalue` scale from 0 to 1. The default value is `q=1`.

**Example D-1. Sample**

```
Accept: audio/*; q=0.2, audio/basic
```

The above statement is interpreted as "I prefer audio/basic, but send me any audio type if it is the best available after an 80% reduction in quality."

**Note:** The Content-Type entity-header field indicates the media type of the entity-body sent to the recipient, or in the case of the HEAD method, the media type that would have been sent had the Request been a GET.

**Example D-2. Sample**

```
Content-Type = "Content-Type" ":" media-type
```

Specify a value for media-type above. Here is an example of a field:

```
Content-Type: text/html; charset=ISO-8859-4
```





## AQL

The Abstract Query Language (AQL) is an implementation-independent search description language. The language hides the actual query language used for Documentum components (DQL, xQuery) or external sources. In REST services, the Search resource utilizes AQL to represent its query document.

### Example E-1. Query document in AQL — XML

```
<?xml version='1.0' encoding='UTF-8'?>
<search all-versions="true" max-results-for-facets="10"
xmlns="http://identifiers.emc.com/vocab/documentum"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <types>
    <type>dm_document</type>
  </types>
  <columns>
    <column>r_object_type</column>
    <column>object_name</column>
  </columns>
  <sorts>
    <sort ascii="false" ascending="true" lang="fr">r_content_size</sort>
    <sort ascii="true" ascending="false" lang="zh">object_name</sort>
  </sorts>
  <collections>
    <collection>collection1</collection>
    <collection>collection2</collection>
  </collections>
  <locations>
    <id-location descendent="true">
      <id>0c00000580001914</id>
    </id-location>
    <path-location descendent="false">
      <path>/dmadmin</path>
    </path-location>
  </locations>
  <facet-definitions>
    <facet-definition id="id1" group-by="alpharange" max-values="8">
      <attributes>
        <attribute>r_object_type</attribute>
      </attributes>
      <sort>FREQUENCY</sort>
      <properties>
        <property name="range">a:e, f:z</property>
      </properties>
    </facet-definition>
    <facet-definition id="id2" group-by="relativeDate" max-values="9">
      <attributes>
        <attribute>r_modify_date</attribute>
      </attributes>
      <sort>VALUE_ASCENDING</sort>
    </facet-definition>
  </facet-definitions>
</search>
```

```
        </facet-definition>
    </facet-definitions>
    <expression-set>
        <expressions>
            <property name="object_name" operator="CONTAINS" exact-match="true"
                repeating="true" fuzzy="true">rest
            </property>
            <fulltext fuzzy="true">rest</fulltext>
            <fulltext fuzzy="false">test</fulltext>
            <property-range name="r_content_size" operator="BETWEEN" repeating="false">
                <from>0</from>
                <to>10000</to>
            </property-range>
            <property-list name="r_object_type" operator="IN" repeating="false">
                <values>
                    <value>dm_document</value>
                    <value>dm_folder</value>
                </values>
            </property-list>
            <expression-set>
                <expressions>
                    <fulltext fuzzy="false">test</fulltext>
                </expressions>
            </expression-set>
            <relative-date name="r_modify_date" time-unit="MONTH"
                operator="GREATER THAN" >-2</relative-date>
        </expressions>
    </expression-set>
</search>
```

#### Example E-2. Query document in AQL — JSON

```
{
  "all-versions": true,
  "include-hidden-objects": false,
  "max-results-for-facets": 10,
  "types": [
    "dm_document"
  ],
  "columns": [
    "r_object_type",
    "object_name"
  ],
  "sorts": [
    {
      "property": "r_content_size",
      "ascending": true,
      "lang": "fr",
      "ascii": false
    },
    {
      "property": "object_name",
      "ascending": false,
      "ascii": true
    }
  ],
  "collections": [
    "collection1",
    "collection2"
  ],
  "locations": [
    {
      "location-type": "id-location",
```

```

        "id": "0c00000580001914",
        "repository": "REPO",
        "descendent": true
    },
    {
        "location-type": "path-location",
        "path": "/dmadmin",
        "repository": "REPO",
        "descendent": false
    }
],
"facet-definitions": [
    {
        "id": "id1",
        "attributes": [
            "r_object_type"
        ],
        "group-by": "alpharange",
        "sort": "FREQUENCY",
        "properties": {
            "skipEmptyValues": "true",
            "range": "a:e, f:z"
        },
        "max-values": 8
    },
    {
        "id": "id2",
        "attributes": [
            "r_modify_date"
        ],
        "group-by": "relativeDate",
        "sort": "VALUE_ASCENDING",
        "properties": {
            "returnUTC": "true",
            "skipEmptyValues": "true"
        },
        "max-values": 9
    }
],
"repositories": [
    "REPO"
],
"expression-set": {
    "expression-type": "expression-set",
    "operator": "AND",
    "expressions": [
        {
            "expression-type": "property",
            "name": "object_name",
            "operator": "CONTAINS",
            "value": "rest",
            "exact-match": true,
            "repeating": true,
            "case-sensitive": true,
            "fuzzy": true
        },
        {
            "expression-type": "fulltext",
            "value": "rest",
            "fuzzy": true
        },
        {
            "expression-type": "fulltext",
            "value": "test",
            "fuzzy": false
        }
    ]
}

```

```
    },
    {
      "expression-type": "property-range",
      "name": "r_content_size",
      "operator": "BETWEEN",
      "from": "0",
      "to": "10000",
      "repeating": false
    },
    {
      "expression-type": "property-list",
      "name": "r_object_type",
      "operator": "IN",
      "values": [
        "dm_document",
        "dm_folder"
      ],
      "repeating": false
    },
    {
      "expression-type": "expression-set",
      "operator": "AND",
      "expressions": [
        {
          "expression-type": "fulltext",
          "value": "test",
          "fuzzy": false
        }
      ]
    },
    {
      "expression-type": "relative-date",
      "name": "r_modify_date",
      "value": -2,
      "time-unit": "MONTH",
      "operator": "GREATER_THAN",
      "repeating": false
    }
  ]
}
```

The following table introduces elements in AQL.

Element	Description
all-versions	Indicates whether or not all versions of the objects are expected to be included in the query results.
include-hidden-objects	Indicates whether or not the hidden objects should be included in the query results.  Its default value is false.
types	Defines the type of the objects to search.  You cannot define multiple types.  The URL parameter <code>object-type</code> cannot be used in a search operation with AQL.

Element	Description											
columns	<p>Defines the properties of the object to display in the result set.</p> <p>This element works when the URL parameter <code>inline</code> is set to <code>true</code>.</p> <p>The URL parameter <code>view</code> cannot be used in a search operation with AQL.</p>											
collections	<p>Defines the collections to search.</p> <p>The URL parameter <code>collections</code> cannot be used in a search operation with AQL.</p>											
locations	<p>Defines the folders to search.</p> <p>You can specify locations in the following two ways:</p> <table><tr><th>Location Type</th><th>Option</th><th>Example</th></tr><tr><td rowspan="2">id location</td><td>id: The ID of the folder to search</td><td rowspan="2"><pre>&lt;id-location descendent="true"&gt; &lt;id&gt;0c00000580001914&lt; /id&gt; &lt;/id-location&gt;</pre></td></tr><tr><td>descendent: Whether to search the sub-folders or not  Default value: false</td></tr><tr><td rowspan="2">path location</td><td>path: The path of the folder to search</td><td rowspan="2"><pre>&lt;path-location descendent="false"&gt; &lt;path&gt;/dmadmin&lt;/path&gt; &lt;/path-location&gt;</pre></td></tr><tr><td>descendent: Whether to search the sub-folders or not  Default value: false</td></tr></table> <p>The URL parameter <code>locations</code> cannot be used in a search operation with AQL.</p>	Location Type	Option	Example	id location	id: The ID of the folder to search	<pre>&lt;id-location descendent="true"&gt; &lt;id&gt;0c00000580001914&lt; /id&gt; &lt;/id-location&gt;</pre>	descendent: Whether to search the sub-folders or not  Default value: false	path location	path: The path of the folder to search	<pre>&lt;path-location descendent="false"&gt; &lt;path&gt;/dmadmin&lt;/path&gt; &lt;/path-location&gt;</pre>	descendent: Whether to search the sub-folders or not  Default value: false
Location Type	Option	Example										
id location	id: The ID of the folder to search	<pre>&lt;id-location descendent="true"&gt; &lt;id&gt;0c00000580001914&lt; /id&gt; &lt;/id-location&gt;</pre>										
	descendent: Whether to search the sub-folders or not  Default value: false											
path location	path: The path of the folder to search	<pre>&lt;path-location descendent="false"&gt; &lt;path&gt;/dmadmin&lt;/path&gt; &lt;/path-location&gt;</pre>										
	descendent: Whether to search the sub-folders or not  Default value: false											
sorts	<p>Defines how the system sorts results.</p> <p>REST services uses two strategies, ASCII sort and locale sort. The following describes the details:</p> <table><tr><th>Option</th><th>Description</th></tr><tr><td>property</td><td>Property to sort against</td></tr><tr><td>ascending</td><td>Whether or not to sort the results in ascending order  Default value: true</td></tr><tr><td>ascii</td><td>Whether or not to sort items using the ASCII table.  Default value: true</td></tr></table>	Option	Description	property	Property to sort against	ascending	Whether or not to sort the results in ascending order  Default value: true	ascii	Whether or not to sort items using the ASCII table.  Default value: true			
Option	Description											
property	Property to sort against											
ascending	Whether or not to sort the results in ascending order  Default value: true											
ascii	Whether or not to sort items using the ASCII table.  Default value: true											

Element	Description	
		When this value is set to false, you must specify a locale using <code>lang</code> .
	<code>lang</code>	Specifies the locale that the system uses to sort the results. The supported locale depends on the xPlore configuration.
	You can specify multiple sort items. In this case, the system sorts results by the first sort item first, then by the second one, and so on so forth. For example, with the <code>sorts</code> element defined as follows, the system sorts results by <code>r_content_size</code> first, and then by <code>object_name</code> for objects with the same <code>r_content_size</code> .	
<code>expression-set</code>	<p>Defines the search criteria by using a set of expressions. Expression-sets can be nested and associated with each other using logical AND (default) or logical OR. The following five expressions are supported:</p> <ul style="list-style-type: none"> <li>• fulltext expression</li> <li>• property expression</li> <li>• property list expression</li> <li>• property range expression</li> <li>• relative date expression</li> </ul>	
<code>facet-definitions</code>	<p>List of facet definitions. See <a href="#">Facet in AQL</a>.</p> <p>The URL parameters <code>facet</code> and <code>facet-value-constraints</code> cannot be used in a search operation with AQL.</p>	
<code>max-results-for-facets</code>	Sets the maximum number of results to return.	

## Expressions in AQL

The `expression-set` element defines the search criteria by using a set of expressions. Expressions can be nested and associated with each other using logical AND (default) or logical OR.

### Example E-3. XML

```
<expression-set>
  <expressions>
    <property name="object_name" operator="CONTAINS" exact-match="true"
      repeating="true" fuzzy="true">rest
    </property>
    <fulltext fuzzy="true">rest</fulltext>
    <fulltext fuzzy="false">test</fulltext>
    <property-range name="r_content_size" operator="BETWEEN" repeating="false">
```

```

        <from>0</from>
        <to>10000</to>
    </property-range>
    <property-list name="r_object_type" operator="IN" repeating="false">
        <values>
            <value>dm_document</value>
            <value>dm_folder</value>
        </values>
    </property-list>
    <expression-set>
        <expressions>
            <fulltext case-sensitive="false" fuzzy="false">test</fulltext>
        </expressions>
    </expression-set>
    <relative-date name="r_modify_date" time-unit="MONTH"
        operator="GREATER_THAN" >-2</relative-date>
    </expressions>
</expression-set>

```

#### Example E-4. JSON

```

"expression-set": {
  "expression-type": "expression-set",
  "operator": "AND",
  "expressions": [
    {
      "expression-type": "property",
      "name": "object_name",
      "operator": "CONTAINS",
      "value": "rest",
      "exact-match": true,
      "repeating": true,
      "case-sensitive": true,
      "fuzzy": true
    },
    {
      "expression-type": "fulltext",
      "value": "rest",
      "fuzzy": true
    },
    {
      "expression-type": "fulltext",
      "value": "test",
      "fuzzy": false
    },
    {
      "expression-type": "property-range",
      "name": "r_content_size",
      "operator": "BETWEEN",
      "from": "0",
      "to": "10000",
      "repeating": false
    },
    {
      "expression-type": "property-list",
      "name": "r_object_type",
      "operator": "IN",
      "values": [
        "dm_document",
        "dm_folder"
      ],
      "repeating": false
    }
  ]
}

```

```
{
  "expression-type": "expression-set",
  "operator": "AND",
  "expressions": [
    {
      "expression-type": "fulltext",
      "value": "test",
      "fuzzy": false
    }
  ]
},
{
  "expression-type": "relative-date",
  "name": "r_modify_date",
  "value": -2,
  "time-unit": "MONTH",
  "operator": "GREATER_THAN",
  "repeating": false
}
]
```

## Fulltext Expression

A `fulltext` expression is similar to the URL parameter `q` in that it follows the simple search language syntax. Moreover, `fulltext` expressions support the follow two options:

- `fuzzy`: a boolean indicating whether to or not to enable fuzzy search, which defaults to `true`

In the following snippet, the first row matches `RESTful services`, but not `RESTFULL` or `RESTfol` while the second row matches all of these terms.

### Example E-5. Fulltext Expression

```
<fulltext fuzzy="false">RESTful</fulltext>
<fulltext fuzzy="true">RESTful</fulltext>
```

## Property Expression

A property expression defines property value constraints by using a variety of logical operators.

Option	Description
name	Property name
value	Property value
operator	Supports the following logical operators: <code>EQUAL</code> , <code>NOT_EQUAL</code> , <code>GREATER_THAN</code> , <code>LESS_THAN</code> , <code>GREATER_EQUAL</code> , <code>LESS_EQUAL</code> , <code>BEGINS_WITH</code> , <code>CONTAINS</code> , <code>DOES_NOT_CONTAIN</code> , <code>ENDS_WITH</code> , <code>IS_NULL</code> , <code>IS_NOT_NULL</code>



Option	Description
fuzzy	Whether or not to enable fuzzy search for the following operators: SEARCH_OP_DOES_NOT_CONTAIN, SEARCH_OP_CONTAINS, SEARCH_OP_EQUAL  Default value: false
exact-match	Whether or not to evaluate an EQUAL or NOT_EQUALS constraint.  When the constraint is evaluated, the system performs no stemming, stop words removal, thesaurus search, or wildcard matching.  Default value: false
repeating	Determines whether or not DFC should generate DQL instead of xQuery when the type of a repeating property is neither DF_ID nor DF_STRING.  Default value: false
case-sensitive	Indicates whether the search operation is case sensitive. Default value: false

The following property expression returns true when the object\_name of an object is rest.

#### Example E-6. Property Expression in XML

```
<property name="object_name" operator="EQUAL" exact-match="true"
  repeating="true" case-sensitive="true" fuzzy="false">rest
</property>
```

#### Example E-7. Property Expression in JSON

```
{
  "expression-type": "property",
  "name": "object_name",
  "operator": "EQUAL",
  "value": "rest",
  "exact-match": true,
  "repeating": true,
  "case-sensitive": true,
  "fuzzy": false
}
```

## Property List Expression

A property expression defines property value constraints by using a list of discrete values. The following table describes the details:

Option	Description
name	Property name

Option	Description
operator	Supports the following logical operators: <ul style="list-style-type: none"> <li>IN: returns true if the value of the property is in the value list.</li> <li>NOT IN: returns true if the value of the property is not in the value list.</li> </ul>
values	List of discrete property values
repeating	Whether the property is repeating

The following expression returns true if the `r_object_type` is either `dm_document` or `dm_folder`.

#### Example E-8. Property List Expression in XML

```
<property-list name="r_object_type" operator="IN" repeating="false">
  <values>
    <value>dm_document</value>
    <value>dm_folder</value>
  </values>
</property-list>
```

#### Example E-9. Property List Expression in JSON

```
{
  "expression-type": "property-list",
  "name": "r_object_type",
  "operator": "IN",
  "values": [
    "dm_document",
    "dm_folder"
  ],
  "repeating": false
}
```

## Property Range Expression

A property expression defines property value constraints by using a value rang. The following table describes the details:

Option	Description
name	Property name
operator	Supports the logical BETWEEN operator. Returns true when the value of the property is in the range specified by <code>from</code> and <code>to</code> attributes.
from	Starting value
to	Ending value
repeating	Whether the property is repeating

The following expression returns true if the `r_content_size` is smaller than or equal to 1,000.

#### Example E-10. Property Range Expression in XML

```
<property-range name="r_content_size" operator="BETWEEN" repeating="false">
  <from>0</from>
  <to>10000</to>
```

```
</property-range>
```

#### Example E-11. Property Range Expression in JSON

```
{
  "expression-type": "property-range",
  "name": "r_content_size",
  "operator": "BETWEEN",
  "from": "0",
  "to": "10000",
  "repeating": false
}
```

## Relative Date Expression

A relative date expression defines constraints on date properties using a relative date. A relative date shows the time difference to the current date, with a minus sign (-) indicating a certain amount of time before the current date, and a plus sign, which can be omitted, indicating a certain amount of time after the current date. The following table describes the details:

Option	Description
name	Time related property name, such as <code>r_modify_date</code>
time unit	Supports these time units: <code>MINUTE</code> , <code>DAY</code> , <code>MONTH</code> , <code>YEAR</code>
value	Number of the time unit
operator	Supports these operators: <code>EQUAL</code> , <code>GREATER_THAN</code> , <code>LESS_THAN</code> , <code>GREATER_EQUAL</code> , <code>LESS_EQUAL</code>

For example, the following expression returns `true` when `r_modify_date` is greater than the current date minus 10 months:

#### Example E-12. Relative Date Expression in XML

```
<relative-date name="r_modify_date" operator="GREATER_THAN"
time-unit="MONTH">-10</relative-date>
```

#### Example E-13. Relative Date Expression in JSON

```
{
  "expression-type": "relative-date",
  "name": "r_modify_date",
  "value": -10,
  "time-unit": "MONTH",
  "operator": "GREATER_THAN",
  "repeating": false
}
```

## Expressions as Templates

Expressions in AQL can also be used to define search templates. When the attribute `"template"` is `true`, specific elements in the expressions are activated as variables.

The following table lists all AQL expressions that support variables:

Expression Type	Variable	Sample (variable predicates highlighted in bold)
<a href="#">fulltext</a>	Fulltext search criterion	<code>&lt;fulltext fuzzy="false" template="true"&gt;<b>key word</b>&lt;/fulltext&gt;</code>
<a href="#">property</a>	Property value <b>Note:</b> The search operator cannot be IS_NULL or IS_NOT_NULL.	<code>&lt;property name="object_name" operator="CONTAINS" exact-match="false" repeating="false" case-sensitive="false" fuzzy="false" template="true"&gt;<b>myobj</b>&lt;/property&gt;</code>
<a href="#">property list</a>	Value list	<code>&lt;property-list name="r_object_type" operator="IN" repeating="false" template="true"&gt; &lt;values&gt; &lt;value&gt;<b>dm_document</b>&lt;/value&gt; &lt;value&gt;<b>dm_folder</b>&lt;/value&gt; &lt;/values&gt; &lt;/property-list&gt;</code>
<a href="#">relative date</a>	Relative value and time unit	<code>&lt;relative-date name="r_modify_date" time-unit="MONTH" operator="GREATER_THAN" repeating="false" template="true"&gt;-3&lt;/relative-date&gt;</code>
<a href="#">property range</a>	From and To values	<code>&lt;property-range name="r_modify_date" operator="BETWEEN" repeating="false" template="true"&gt; &lt;from&gt;2010-10-13T07:33:13.009+00:00&lt;/from&gt; &lt;to&gt;2014-11-11T01:35:17.089+00:00&lt;/to&gt; &lt;/property-range&gt;</code>

## Facet in AQL

Using AQL, a facet definition becomes more powerful. You can define multiple facets in one query document. These facets can also be nested. Furthermore, facet grouping strategy and facet ordering can both be configured in AQL.

## Grouping Strategy

A grouping strategy defines how to group the results or a facet search. Facet definitions in AQL support the following grouping strategies:

- string
- range

- alphanrange
- date

#### Example E-14. String Grouping Strategy in XML

```
<facet-definition id="id1" group-by="string">
  <attributes>
    <attribute>r_object_type</attribute>
  </attributes>
</facet-definition>
```

#### Example E-15. String Grouping Strategy in JSON

```
{
  "id": "id1",
  "attributes": [
    "r_object_type"
  ],
  "group-by": "alphanrange",
}
```

The string grouping strategy is commonly used when a string property is set as the facet, for example, `r_object_type`. This grouping strategy has the following options:

- `nullValueFacet` (optional): indicates whether or not to take null values into account when computing the facet grouping, which defaults to false.
- `keepDuplicateValues` (optional): indicates whether or not to take duplicated values for repeating properties into account when computing the facet grouping, which defaults to false. This option works for xPlore 1.1 P03 and later versions.
- `caseInsensitive` (optional): indicates whether or not to take letter case into account when computing the facet grouping, which defaults to false. This option works for xPlore 1.1 P03 and later versions.

#### Example E-16. Range Grouping Strategy in XML

```
<facet-definition id="id1" group-by="range">
  <attributes>
    <attribute>r_content_size</attribute>
  </attributes>
  <properties>
    <property name="range">0:1000,1000:10000,10000:</property>
  </properties>
  <sort>FREQUENCY</sort>
</facet-definition>
```

#### Example E-17. Range Grouping Strategy in JSON

```
{
  "id": "id1",
  "attributes": [
    "r_content_size"
  ],
  "group-by": "range",
  "sort": "FREQUENCY",
  "properties": {
    "range": "0:1000,1000:10000,10000:"
  }
}
```

```
}
```

The `range` grouping strategy is commonly used when a numeric property is set as the facet, for example, `r_content_size`. This grouping strategy has the following option:

- `range` (required): Specifies a list of buckets for grouping. Each bucket is separated by a comma (`,`). A bucket is composed of a lower and upper bounds separated by a colon (`:`). You can create open ended buckets by omitting one of the bounds but keeping the comma separator. Overlapping buckets are not supported and may result in inconsistent results.

**Note:**

- The lower-bound is inclusive while the upper-bound is exclusive.
- The boundaries are parsed as `Double`. Example: `0:10,10:100.89,100:`

y

**Example E-18. Alphanrange Grouping Strategy in XML**

```
<facet-definition id="id1" group-by="alphanrange">
  <attributes>
    <attribute>r_object_name</attribute>
  </attributes>
  <properties>
    <property name="range">a:m,n:r,s:z</property>
  </properties>
  <sort>FREQUENCY</sort>
</facet-definition>
```

**Example E-19. Alphanrange Grouping Strategy in JSON**

```
{
  "id": "id1",
  "attributes": [
    "r_object_name"
  ],
  "group-by": "alphanrange",
  "sort": "FREQUENCY",
  "properties": {
    "range": "a:m,n:r,s:z"
  }
}
```

The `alphanrange` grouping strategy is commonly used when a string property is set as the facet, for example, `r_object_name`. This grouping strategy has the following option:

`range` (required): specifies a list of buckets for grouping. Each bucket is separated by comma (`,`). A bucket is composed of a lower and upper bounds separated by colon (`:`). You can create open ended buckets by omitting one of the bounds but keeping the comma separator. Overlapping buckets are not supported and may result in inconsistent results.

**Note:**

- This grouping strategy is case-insensitive.
- The boundaries are inclusive.
- The boundaries are parsed as characters. Example: `a:m,n:r,s:z`

**Example E-20. Date Grouping Strategy in XML**

```
<facet-definition id="id1" group-by="week">
  <attributes>
    <attribute>r_modify_date</attribute>
  </attributes>
  <sort>FREQUENCY</sort>
</facet-definition>
```

**Example E-21. Date Grouping Strategy in JSON**

```
{
  "id": "id1",
  "attributes": [
    "r_modify_date"
  ],
  "group-by": "week",
  "sort": "FREQUENCY"
}
```

The date grouping strategy has the following types:

- day: groups results by day
- week: groups results by week
- month: groups results by month
- quarter (since xPlore 1.3): groups results by quarter
- year: groups results by year
- relativeDate: a pre-defined grouping, which contains the following groups: OLDER, LAST\_YEAR, LAST\_MONTH, LAST\_WEEK, THIS\_YEAR, THIS\_WEEK, YESTERDAY, TODAY

For more information on the options for this grouping strategy, see the [Grouping Strategies table](#).

## Facet Ordering

Facet ordering is configured in the `sort` element of a facet definition. The following ordering policies are supported:

- FREQUENCY: sort by the result count in descending order.
- NONE: keeps the order returned by the facet handler.
- VALUE\_ASCENDING: sort by the facet value ID in ascending order.
- VALUE\_DESCENDING: sort by the facet value ID in descending order.