RESTful Network API for Network Message Storage
Candidate Version 1.0 – 09 Feb 2016

Open Mobile Alliance
OMA-TS-REST_NetAPI_NMS-V1_0-20160209-C
Contents

1. SCOPE (INFORMATIVE) ........................................................................................................................................ 15
2. REFERENCES .......................................................................................................................................................... 16
  2.1 NORMATIVE REFERENCES ................................................................................................................................. 16
  2.2 INFORMATIVE REFERENCES ............................................................................................................................... 17
3. TERMINOLOGY AND CONVENTIONS .................................................................................................................. 18
  3.1 CONVENTIONS ....................................................................................................................................................... 18
  3.2 DEFINITIONS .......................................................................................................................................................... 18
  3.3 ABBREVIATIONS ..................................................................................................................................................... 18
4. INTRODUCTION (INFORMATIVE) ............................................................................................................................. 20
  4.1 VERSION 1.0 ............................................................................................................................................................ 20
5. NETWORK MESSAGE STORAGE API DEFINITION .................................................................................................. 22
  5.1 CONCEPTS .............................................................................................................................................................. 22
   5.1.1 Object ................................................................................................................................................................. 22
   5.1.2 Folder ................................................................................................................................................................. 23
   5.1.3 Box ..................................................................................................................................................................... 23
   5.1.4 Subscriptions and notifications .......................................................................................................................... 23
      5.1.4.1 Notifications .................................................................................................................................................. 24
      5.1.4.2 Tracked storage changes ............................................................................................................................... 24
         5.1.4.2.1 Box reset .................................................................................................................................................... 25
      5.1.4.3 Requesting past notifications using restartToken ............................................................................................. 25
         5.1.4.3.1 Implementation note (Informative) ........................................................................................................... 26
      5.1.4.4 Handling reordering and duplication with lastModSeq ..................................................................................... 26
      5.1.4.5 Handling loss with index and subscription update ........................................................................................... 26
   5.1.5 Managing local storage mirror (cache) at the client ............................................................................................ 27
      5.1.5.1 Strict Synchronization .................................................................................................................................... 27
      5.1.5.2 Simplified Synchronization .......................................................................................................................... 27
   5.1.6 Root folder(s) discovery ...................................................................................................................................... 28
   5.1.7 Deletion ............................................................................................................................................................. 28
   5.1.8 External content .................................................................................................................................................. 29
   5.1.9 Inline content ...................................................................................................................................................... 29
   5.1.10 Batched retrieval ................................................................................................................................................ 29
   5.1.11 Bulk Creation .................................................................................................................................................... 31
   5.1.12 Correlation (Informative) ................................................................................................................................ 31
      5.1.12.1 Introduction .................................................................................................................................................. 31
      5.1.12.2 Correlation ID .............................................................................................................................................. 31
      5.1.12.3 Correlation tag .............................................................................................................................................. 31
  5.2 RESOURCES SUMMARY ...................................................................................................................................... 32
   5.2.1 Resources allowing a client to manage individual objects .................................................................................... 34
   5.2.2 Resources allowing a client to retrieve stored content of an object payload or payload part .................................... 35
   5.2.3 Resources allowing a client to manage individual folders .................................................................................... 36
   5.2.4 Resources allowing a client to retrieve information and/or perform operations on objects ..................................... 38
   5.2.5 Resources allowing a client to retrieve information and/or perform operations on folders ...................................... 39
   5.2.6 Resources allowing a client to manage subscriptions for storage changes ............................................................ 41
   5.2.7 Resources allowing the server to notify a client about storage changes ................................................................. 41
  5.3 DATA TYPES .......................................................................................................................................................... 42
   5.3.1 XML Namespaces ............................................................................................................................................... 42
   5.3.2 Structures ........................................................................................................................................................... 42
      5.3.2.1 Type: Object .................................................................................................................................................... 42
      5.3.2.2 Type: ObjectList ............................................................................................................................................. 45
      5.3.2.3 Type: Flag ....................................................................................................................................................... 45
      5.3.2.4 Type: FlagList ................................................................................................................................................ 46
      5.3.2.5 Type: Attribute .............................................................................................................................................. 47
      5.3.2.6 Type: AttributeList ....................................................................................................................................... 47
      5.3.2.7 Type: PayloadPartInfo .................................................................................................................................... 48
5.3.2.8 Type: Folder .......................................................................................................................................................... 49
5.3.2.9 Type: FolderList ................................................................................................................................................... 53
5.3.2.10 Type: FolderReferenceList ....................................................................................................................................... 53
5.3.2.11 Type: Reference ................................................................................................................................................. 53
5.3.2.12 Type: ObjectReferenceList .................................................................................................................................. 53
5.3.2.13 Type: TargetSourceRef ......................................................................................................................................... 54
5.3.2.14 Type: ReferenceList ............................................................................................................................................... 54
5.3.2.15 Type: BulkUpdate .................................................................................................................................................. 54
5.3.2.16 Type: BulkDelete ................................................................................................................................................... 54
5.3.2.17 Type: SelectionCriteria ...................................................................................................................................... 55
5.3.2.18 Type: SearchCriteria ............................................................................................................................................ 56
5.3.2.19 Type: SearchCriterion ........................................................................................................................................... 56
5.3.2.20 Type: SortCriteria ................................................................................................................................................ 57
5.3.2.21 Type: SortCriterion ............................................................................................................................................... 57
5.3.2.22 Type: NmsSubscription ........................................................................................................................................ 57
5.3.2.23 Type: NmsSubscriptionList .................................................................................................................................... 60
5.3.2.24 Type: NmsSubscriptionUpdate .................................................................................................................................... 60
5.3.2.25 Type: NmsEvent .................................................................................................................................................... 61
5.3.2.26 Type: NmsEventList ............................................................................................................................................... 62
5.3.2.27 Type: DeletedObject ............................................................................................................................................. 63
5.3.2.28 Type: DeletedFolder .............................................................................................................................................. 64
5.3.2.29 Type: ChangedObject ........................................................................................................................................... 65
5.3.2.30 Type: ChangedFolder ............................................................................................................................................ 65
5.3.2.31 Type: ResetBox ..................................................................................................................................................... 66
5.3.2.32 Type: PathList ........................................................................................................................................................ 67
5.3.2.33 Type: Response ..................................................................................................................................................... 67
5.3.2.34 Type: BulkResponseList ...................................................................................................................................... 68
5.3.2.35 Type: Empty ........................................................................................................................................................ 68

5.3.3 Enumerations ........................................................................................................................................................ 69
5.3.3.1 Enumeration: SearchTypeEnum ................................................................................................................................. 69
5.3.3.2 Enumeration: LogicalOperatorEnum ...................................................................................................................................... 70
5.3.3.3 Enumeration: SortTypeEnum ...................................................................................................................................... 70
5.3.3.4 Enumeration: SortOrderEnum ..................................................................................................................................... 70
5.3.3.5 Enumeration: OperationEnum ...................................................................................................................................... 70
5.3.4 Values of the Link “rel” attribute ........................................................................................................................................... 71

5.4 SEQUENCE DIAGRAMS ............................................................................................................................................... 71
5.4.1 Subscription to NMS notifications .................................................................................................................................. 71
5.4.2 Synchronization with NMS ........................................................................................................................................... 73
5.4.2.1 Strict Synchronization ............................................................................................................................................. 73
5.4.2.2 Simplified Synchronization ........................................................................................................................................ 75
5.4.3 Subscription to filtered NMS Notifications ...................................................................................................................... 76
5.4.4 Operations on folders ..................................................................................................................................................... 77
5.4.5 Operations on objects ................................................................................................................................................... 80
5.4.6 Retrieving a large list of objects ......................................................................................................................................... 83
5.4.7 Discovering the user’s storage hierarchical structure ..................................................................................................... 85
5.4.8 Bulk object creation ....................................................................................................................................................... 87
5.4.9 Bulk object deletion ........................................................................................................................................................ 88
5.4.10 Bulk object update ....................................................................................................................................................... 90

6. DETAILED SPECIFICATION OF THE RESOURCES ............................................................................................................. 92
6.1 RESOURCE: RESOURCE CONTAINING ALL OBJECTS ................................................................................................. 93
6.1.1 Request URL variables ................................................................................................................................................ 93
6.1.2 Response Codes and Error Handling ............................................................................................................................ 93
6.1.3 GET ........................................................................................................................................................................... 93
6.1.4 PUT .......................................................................................................................................................................... 93
6.1.5 POST ....................................................................................................................................................................... 93
6.1.5.1 Example 1: Object creation by parentFolder, response with a location of created resource (Informative) .................. 94
6.1.5.1.1 Request ................................................................................................................................................................. 94
6.1.5.1.2 Response .......................................................................................................................................................... 95
6.1.5.2 Example 2: Object creation by parentFolderPath, response with a location of the created resource while the non-existing parent folder is auto-created (Informative) ............................................................................................ 95
6.1.5.2.1 Request .......................................................................................................................................................... 95
6.1.5.2.2 Response.............................................................................................................................................. 96

6.1.5.3 Example 3: Object creation by parentFolderPath, response creation failure due to prohibited location (i.e. requested parent folder) (Informative)................................................................................................................................. 96

6.1.5.3.1 Request.................................................................................................................................................. 96

6.1.5.3.2 Response.............................................................................................................................................. 97

6.1.5.4 Example 4: Object creation without parent folder, response with a location of created resource and full object (Informative)........................................................................................................................................................................... 97

6.1.5.4.1 Request.................................................................................................................................................. 97

6.1.5.4.2 Response.............................................................................................................................................. 98

6.1.5.5 Example 5: Creation of multipart object with presentation part, response with a location of created resource (Informative)........................................................................................................................................................................................................................................... 99

6.1.5.5.1 Request.................................................................................................................................................. 99

6.1.5.5.2 Response.............................................................................................................................................. 100

6.1.5.6 Example 6: Creation of simple text object, response with a location of created resource (Informative)........................................................................................................................................................................................................................................... 101

6.1.5.6.1 Request.................................................................................................................................................. 101

6.1.5.6.2 Response.............................................................................................................................................. 101

6.1.5.7 Example 7: Creation of simple object with no payload, response with a location of created resource (Informative)........................................................................................................................................................................................................................................... 101

6.1.5.7.1 Request.................................................................................................................................................. 102

6.1.5.7.2 Response.............................................................................................................................................. 102

6.1.6 DELETE.......................................................................................................................................................... 102

6.2 RESOURCE: A STORED OBJECT.................................................................................................................. 103

6.2.1 Request URL variables.................................................................................................................................. 103

6.2.2 Response Codes and Error Handling........................................................................................................ 103

6.2.3 GET................................................................................................................................................................ 103

6.2.3.1 Example 1: Retrieve information about an object (Informative).................................................................. 104

6.2.3.1.1 Request.................................................................................................................................................. 104

6.2.3.1.2 Response.............................................................................................................................................. 104

6.2.3.2 Example 2: Retrieve information about multipart object (Informative)...................................................... 105

6.2.3.2.1 Request.................................................................................................................................................. 105

6.2.3.2.2 Response.............................................................................................................................................. 105

6.2.3.3 Example 3: Retrieve information about object with inline content (Informative)...................................... 106

6.2.3.3.1 Request.................................................................................................................................................. 106

6.2.3.3.2 Response.............................................................................................................................................. 107

6.2.4 PUT................................................................................................................................................................ 107

6.2.5 POST............................................................................................................................................................ 107

6.2.6 DELETE........................................................................................................................................................ 107

6.2.6.1 Example: Delete an object, response with “204 No Content” (Informative)................................................ 107

6.2.6.1.1 Request.................................................................................................................................................. 107

6.2.6.1.2 Response.............................................................................................................................................. 107

6.3 RESOURCE: FLAGS ASSOCIATED WITH THE STORED OBJECT.................................................................. 108

6.3.1 Request URL variables.................................................................................................................................. 108

6.3.2 Response Codes and Error Handling........................................................................................................ 108

6.3.3 GET................................................................................................................................................................ 108

6.3.3.1 Example 1: Retrieve flags associated with an object (Informative)............................................................. 108

6.3.3.1.1 Request.................................................................................................................................................. 108

6.3.3.1.2 Response.............................................................................................................................................. 109

6.3.3.2 Example 2: Retrieve flags associated with an object, failure due to an invalid object (Informative).............. 109

6.3.3.2.1 Request.................................................................................................................................................. 109

6.3.3.2.2 Response.............................................................................................................................................. 109

6.3.4 PUT................................................................................................................................................................ 109

6.3.4.1 Add a flag to flagslist of an object (Informative).......................................................................................... 109

6.3.4.1.1 Request.................................................................................................................................................. 110

6.3.4.1.2 Response.............................................................................................................................................. 110

6.3.5 POST............................................................................................................................................................ 110

6.3.6 DELETE........................................................................................................................................................ 110

6.4 RESOURCE: INDIVIDUAL FLAG.................................................................................................................. 111

6.4.1 Request URL variables.................................................................................................................................. 111

6.4.2 Response Codes and Error Handling........................................................................................................ 111

6.4.3 GET................................................................................................................................................................ 111

6.4.3.1 Example 1: Read an existing individual flag (Informative)........................................................................ 111

6.4.3.1.1 Request.................................................................................................................................................. 112
6.7.5.1 Example 1: Search for objects with certain criteria (Informative) .......................................................... 122
  6.7.5.1.1 Request ............................................................................................................................................. 122
  6.7.5.1.2 Response ......................................................................................................................................... 122

6.7.5.2 Example 2: Retrieve the remaining search response list (Informative) .................................................. 125
  6.7.5.2.1 Request ............................................................................................................................................. 125
  6.7.5.2.2 Response ......................................................................................................................................... 126

6.7.5.3 Example 3: Search for a substring in all searchable text attributes and bodies (Informative) .............. 128
  6.7.5.3.1 Request ............................................................................................................................................. 128
  6.7.5.3.2 Response ......................................................................................................................................... 128

6.7.5.4 Example 4: Search for CreatedObjects not supported (Informative) .................................................... 129
  6.7.5.4.1 Request ............................................................................................................................................. 129
  6.7.5.4.2 Response ......................................................................................................................................... 130

6.7.6 DELETE .................................................................................................................................................... 130

6.8 Resource: Resource URLs of a selected set of objects in the storage ......................................................... 130
  6.8.1 Request URL variables ............................................................................................................................ 131
  6.8.2 Response Codes and Error Handling ....................................................................................................... 131
  6.8.3 GET .......................................................................................................................................................... 131

6.8.3.1 Example 1: Retrieve object’s resource URL based on its path (Informative) ........................................ 131
  6.8.3.1.1 Request ............................................................................................................................................. 131
  6.8.3.1.2 Response ......................................................................................................................................... 132

6.8.3.2 Example 2: Retrieve object’s resource URL based on its path, failure due to an invalid path (Informative) 132
6.10 Request URL variables
6.10.1 GET
6.10.2 PUT
6.10.3 POST
6.10.4 DELETE

6.10.5 Example 1: Bulk update a requested list of objects (Informative)
6.10.5.1 Request
6.10.5.2 Response

6.10.6 Example 2: Bulk update objects meeting certain criteria (Informative)
6.10.6.1 Request
6.10.6.2 Response

6.11 Request URL variables
6.11.1 GET
6.11.2 PUT
6.11.3 POST
6.11.4 DELETE

6.11.5 Example 1: Bulk delete a given list of objects (Informative)
6.11.5.1 Request
6.11.5.2 Response

6.11.6 Example 2: Bulk delete objects meeting certain criteria (Informative)
6.11.6.1 Request
6.11.6.2 Response

6.11.7 Example 3: Retrieve the remaining bulk delete response list (Informative)
6.11.7.1 Request
6.11.7.2 Response

6.12 Request URL variables
6.12.1 GET
6.12.2 PUT
6.12.3 POST
6.12.4 DELETE

6.12.5 Example 1: Folder creation by parentFolder path, response with a location of created resource (Informative)
6.12.5.2 Example: Folder creation by parentFolder path, response with a copy of created resource (Informative) ........................................ 151
6.12.5.2.1 Request....................................................................................................................................................................... 152
6.12.5.2.2 Response..................................................................................................................................................................... 152
6.12.5.3 Example: Folder creation by parentFolder path, response creation failure due to an invalid folder path (Informative). ... 153
6.12.5.3.1 Request....................................................................................................................................................................... 153
6.12.5.3.2 Response..................................................................................................................................................................... 153
6.12.5.4 Example: Folder creation by parentFolder resourceURL, response with a copy of created resource (Informative) ........ 153
6.12.5.4.1 Request....................................................................................................................................................................... 153
6.12.5.4.2 Response..................................................................................................................................................................... 154
6.12.5.5 Example: Folder creation failure due to request missing the parent folder element (Informative) ................................. 154
6.12.5.5.1 Request....................................................................................................................................................................... 154
6.12.5.5.2 Response..................................................................................................................................................................... 155
6.12.5.6 Example: Folder creation by parentFolder path, response creation failure due to prohibited location (i.e. requested 
parent folder) (Informative) ............................................................................................................................................................ 155
6.12.5.6.1 Request....................................................................................................................................................................... 155
6.12.5.6.2 Response..................................................................................................................................................................... 155
6.12.6 DELETE .............................................................................................................................................................................. 156

6.13 RESOURCE: A FOLDER ............................................................................................................................................................ 156
6.13.1 Request URL variables .......................................................................................................................................................... 156
6.13.2 Response Codes and Error Handling .................................................................................................................................. 156
6.13.3 GET .......................................................................................................................................................................................... 156
6.13.3.1 Example: Retrieve information about a folder (Informative) ............................................................................................. 158
6.13.3.1.1 Request....................................................................................................................................................................... 158
6.13.3.1.2 Response..................................................................................................................................................................... 158
6.13.3.2 Example: Retrieve information about a non-existent folder (Informative) ................................................................. 159
6.13.3.2.1 Request....................................................................................................................................................................... 159
6.13.3.2.2 Response..................................................................................................................................................................... 159
6.13.3.3 Example: Retrieve information about a large folder (Informative) ............................................................................... 159
6.13.3.3.1 Request....................................................................................................................................................................... 159
6.13.3.3.2 Response..................................................................................................................................................................... 159
6.13.3.4 Example: Retrieve information about a large folder (Informative) ............................................................................... 160
6.13.3.4.1 Request....................................................................................................................................................................... 160
6.13.3.4.2 Response..................................................................................................................................................................... 160
6.13.4 PUT ....................................................................................................................................................................................... 161
6.13.5 POST ..................................................................................................................................................................................... 161
6.13.6 DELETE .................................................................................................................................................................................. 161
6.13.6.1 Example: Delete a folder, response with “204 No Content” (Informative) ................................................................. 161
6.13.6.1.1 Request....................................................................................................................................................................... 161
6.13.6.1.2 Response..................................................................................................................................................................... 161

6.14 RESOURCE: INDIVIDUAL FOLDER DATA ............................................................................................................................ 162
6.14.1 Request URL variables .......................................................................................................................................................... 162
6.14.2 Response Codes and Error Handling .................................................................................................................................. 162
6.14.3 GET .......................................................................................................................................................................................... 163
6.14.3.1 Example: Retrieve a folder’s name (Informative) ........................................................................................................... 163
6.14.3.1.1 Request....................................................................................................................................................................... 163
6.14.3.1.2 Response..................................................................................................................................................................... 163
6.14.4 PUT ....................................................................................................................................................................................... 163
6.14.4.1 Example: Change folder name (Informative) .................................................................................................................. 163
6.14.4.1.1 Request....................................................................................................................................................................... 163
6.14.4.1.2 Response..................................................................................................................................................................... 163
6.14.4.2 Example: Change folder name, failure due to Policy error (Informative) ................................................................. 163
6.14.4.2.1 Request....................................................................................................................................................................... 164
6.14.4.2.2 Response..................................................................................................................................................................... 164
6.14.5 POST ..................................................................................................................................................................................... 164
6.14.6 DELETE .................................................................................................................................................................................. 164

6.15 RESOURCE: INFORMATION ABOUT A SELECTED SET OF FOLDERS IN THE STORAGE .............................................................................................................. 164
6.15.1 Request URL variables .......................................................................................................................................................... 165
6.15.2 Response Codes and Error Handling .................................................................................................................................. 165
6.16  RESOURCE: RESOURCE URLS OF A SELECTED SET OF FOLDERS IN THE STORAGE

6.16.1  Request URL variables

6.16.2  Response Codes and Error Handling

6.16.3  GET

6.16.3.1  Example 1: Retrieve folder’s resource URL based on its path (Informative)

6.16.3.1.1  Request

6.16.3.1.2  Response

6.16.3.2  Example 2: Retrieve root folder’s resource URL (Informative)

6.16.3.2.1  Request

6.16.3.2.2  Response

6.16.3.3  Example 3: Retrieve root folder’s resource URL, failure due to missing path parameter while multiple root folders exist (Informative)

6.16.3.3.1  Request

6.16.3.3.2  Response

6.16.4  PUT

6.16.5  POST

6.16.5.1  Example 1: Retrieve list of folders’ resource URLs based on their paths (Informative)

6.16.5.1.1  Request

6.16.5.1.2  Response

6.16.5.2  Example 2: Retrieve list of folders’ resource URLs based on their paths, two invalid paths in the list (Informative)

6.16.5.2.1  Request

6.16.5.2.2  Response

6.16.6  DELETE

6.17  RESOURCE: RESOURCE FOR TRIGGERING OBJECT(S)/FOLDER(S) COPYING

6.17.1  Request URL variables

6.17.2  Response Codes and Error Handling

6.17.3  GET

6.17.4  PUT

6.17.5  POST

6.17.5.1  Example 1: Copy objects to a target folder (Informative)

6.17.5.1.1  Request

6.17.5.1.2  Response

6.17.5.2  Example 2: Copy a folder with containing objects to a target folder (Informative)

6.17.5.2.1  Request

6.17.5.2.2  Response

6.17.5.3  Example 3: Copy objects/folders to a target folder, failure due to at lease one invalid source object or folder reference (Informative)

6.17.5.3.1  Request

6.17.5.3.2  Response

6.17.6  DELETE

6.18  RESOURCE: RESOURCE FOR TRIGGERING OBJECT(S)/FOLDER(S) MOVING

6.18.1  Request URL variables

6.18.2  Response Codes and Error Handling

6.18.3  GET

6.18.4  PUT
6.18.5 POST

6.18.5.1 Example 1: Move objects to a target folder (Informative) ................................................................. 182
6.18.5.1.1 Request .............................................................................................................................................. 183
6.18.5.1.2 Response......................................................................................................................................... 183
6.18.5.2 Example 2: Move a folder with containing objects to a target folder (Informative) ................................. 184
6.18.5.2.1 Request .............................................................................................................................................. 184
6.18.5.2.2 Response ......................................................................................................................................... 184
6.18.5.3 Example 3: Move objects/folders to a target folder, failure due to a forbidden target folder (Informative) ..................................................................................................................... 185
6.18.5.3.1 Request .............................................................................................................................................. 185
6.18.5.3.2 Response ......................................................................................................................................... 185
6.18.6 DELETE ................................................................................................................................................ 185

6.19 RESOURCE: ALL SUBSCRIPTIONS IN THE STORAGE ............................................................................. 186

6.19.1 Request URL variables .......................................................................................................................... 187
6.19.2 Response Codes and Error Handling ...................................................................................................... 187
6.19.3 GET ........................................................................................................................................................ 187

6.19.3.1 Example: Reading all active subscriptions (Informative) .................................................................... 188
6.19.3.1.1 Request .............................................................................................................................................. 188
6.19.3.1.2 Response......................................................................................................................................... 188
6.19.4 PUT ....................................................................................................................................................... 188
6.19.5 POST .................................................................................................................................................... 188

6.19.5.1 Example: Creating a new subscription, response with copy of created resource (Informative) ............ 188
6.19.5.1.1 Request .............................................................................................................................................. 189
6.19.5.1.2 Response......................................................................................................................................... 190
6.19.6 DELETE ................................................................................................................................................ 190

6.20 RESOURCE: INDIVIDUAL SUBSCRIPTION ......................................................................................... 190

6.20.1 Request URL variables .......................................................................................................................... 191
6.20.2 Response Codes and Error Handling ...................................................................................................... 191
6.20.3 GET ........................................................................................................................................................ 191

6.20.3.1 Example 1: Reading an individual subscription (Informative) ............................................................ 191
6.20.3.1.1 Request .............................................................................................................................................. 191
6.20.3.1.2 Response......................................................................................................................................... 192
6.20.3.2 Example 2: Retrieve information about a non-existent individual subscription (Informative) ............. 192
6.20.3.2.1 Request .............................................................................................................................................. 192
6.20.3.2.2 Response......................................................................................................................................... 192
6.20.4 PUT ....................................................................................................................................................... 192
6.20.5 POST .................................................................................................................................................... 193

6.20.5.1 Example: Updating the existing subscription (Informative) .............................................................. 193
6.20.5.1.1 Request .............................................................................................................................................. 193
6.20.5.1.2 Response......................................................................................................................................... 193
6.20.6 DELETE ................................................................................................................................................ 193

6.20.6.1 Example: Cancelling a subscription (Informative) ............................................................................. 193
6.20.6.1.1 Request .............................................................................................................................................. 194
6.20.6.1.2 Response......................................................................................................................................... 194

6.21 RESOURCE: CLIENT NOTIFICATION ABOUT STORAGE CHANGES .................................................. 194

6.21.1 Request URL variables .......................................................................................................................... 194
6.21.2 Response Codes and Error Handling ...................................................................................................... 194
6.21.3 GET ........................................................................................................................................................ 194
6.21.4 PUT ....................................................................................................................................................... 195
6.21.5 POST .................................................................................................................................................... 195

6.21.5.1 Example 1: Notify a client about NMS object changes (Informative) ................................................... 195
6.21.5.1.1 Request .............................................................................................................................................. 195
6.21.5.1.2 Response......................................................................................................................................... 196
6.21.5.2 Example 2: Notify a client about NMS folder changes (Informative) .................................................... 196
6.21.5.2.1 Request .............................................................................................................................................. 196
6.21.5.2.2 Response......................................................................................................................................... 196
6.21.1 DELETE ................................................................................................................................................ 196

7. FAULT DEFINITIONS ................................................................................................................................. 197

7.1 SERVICE EXCEPTIONS ............................................................................................................................ 197
7.1.1 SVC1009: Folder’s path needed .............................................................................................................. 197

7.2 POLICY EXCEPTIONS .............................................................................................................................. 197
APPENDIX D. JSON EXAMPLES (INFORMATIVE) ................................................................. 214

D.1 OBJECT CREATION BY PARENT FOLDER, RESPONSE WITH A LOCATION OF CREATED RESOURCE (SECTION 6.1.5.1) ................................................................................................................................................... 214

D.2 OBJECT CREATION BY PARENT FOLDER PATH, RESPONSE WITH A LOCATION OF THE CREATED RESOURCE WHILE THE NON-EXISTENT PARENT FOLDER IS AUTO-CREATED (SECTION 6.1.5.2) ................................................................................................................................................... 215

D.3 OBJECT CREATION BY PARENT FOLDER PATH, RESPONSE CREATION FAILURE DUE TO PROHIBITED LOCATION (I.E. REQUESTED PARENT FOLDER) (SECTION 6.1.5.3) ................................................................................................................................................... 216

D.4 OBJECT CREATION WITHOUT PARENT FOLDER, RESPONSE WITH A LOCATION OF CREATED RESOURCE AND FULL OBJECT (SECTION 6.1.5.4) ................................................................................................................................................... 217

D.5 CREATION OF MULTIPART OBJECT WITH PRESENTATION PART, RESPONSE WITH A LOCATION OF CREATED RESOURCE (SECTION 6.1.5.5) ................................................................................................................................................... 219

D.6 CREATION OF SIMPLE TEXT OBJECT, RESPONSE WITH A LOCATION OF CREATED RESOURCE (SECTION 6.1.5.6) 221

D.7 CREATION OF SIMPLE OBJECT WITH NO PAYLOAD, RESPONSE WITH A LOCATION OF CREATED RESOURCE (SECTION 6.1.5.7) ................................................................................................................................................... 222

D.8 RETRIEVE INFORMATION ABOUT AN OBJECT (SECTION 6.2.3.1) ................................................................................................................................................... 223

D.9 RETRIEVE INFORMATION ABOUT MULTIPART OBJECT (SECTION 6.2.3.2) ................................................................................................................................................... 224

D.10 RETRIEVE INFORMATION ABOUT OBJECT WITH INLINE CONTENT (SECTION 6.2.3.3) ................................................................................................................................................... 225

D.11 DELETE AN OBJECT, RESPONSE WITH "204 NO CONTENT" (SECTION 6.2.6.1) ................................................................................................................................................... 226

D.12 RETRIEVE FLAGS ASSOCIATED WITH AN OBJECT (SECTION 6.3.3.1) ................................................................................................................................................... 226

D.13 RETRIEVE FLAGS ASSOCIATED WITH AN OBJECT, FAILURE DUE TO AN INVALID OBJECT (SECTION 6.3.3.2) ................................................................................................................................................... 226

D.14 ADD A FLAG TO FLAGLIST OF AN OBJECT (SECTION 6.3.4.1) ................................................................................................................................................... 227

D.15 READ AN EXISTING INDIVIDUAL FLAG (SECTION 6.4.3.1) ................................................................................................................................................... 228

D.16 READ A NON-EXISTING INDIVIDUAL FLAG USING ACR: AUTH (SECTION 6.4.3.2) ................................................................................................................................................... 228

D.17 ADD "ANSWERED" FLAG TO FLAGLIST OF AN OBJECT (SECTION 6.4.4.1) ................................................................................................................................................... 228

D.18 DELETE "SEEN" FLAG FROM FLAGLIST OF AN OBJECT (SECTION 6.4.6.1) ................................................................................................................................................... 229

D.19 READ PAYLOAD OF THE STORED OBJECT VIA AN EXTERNAL REFERENCE (SECTION 6.5.3.1) ................................................................................................................................................... 230

D.20 READ AN OBJECT PAYLOAD PART VIA THE NMS RESOURCE TREE (SECTION 6.6.3.1) ................................................................................................................................................... 230
APPENDIX E. OPERATIONS MAPPING TO A PRE-EXISTING BASELINE SPECIFICATION (INFORMATIVE) .............................................................. 281

APPENDIX F. LIGHT-WEIGHT RESOURCES (INFORMATIVE) .................................................................................................................. 282

APPENDIX G. AUTHORIZATION ASPECTS (NORMATIVE) .................................................................................................................. 283

G.1 USE WITH OMA AUTHORIZATION FRAMEWORK FOR NETWORK APIS ........................................................................................................... 283

G.1.1 Scope values .................................................................................................................................................... 283

G.1.1.1 Definitions .................................................................................................................................................... 283

G.1.1.2 Downscoping .................................................................................................................................................... 283

G.1.1.3 Mapping with resources and methods .............................................................................................................. 284

G.1.2 Use of ‘acr:auth’ .................................................................................................................................................... 291

APPENDIX H. FLAG NAMES TABLE (NORMATIVE) .................................................................................................................. 292

APPENDIX I. RCS OBJECT ATTRIBUTES TABLE (INFORMATIVE) ............................................................................................................ 293

APPENDIX J. STANDARD FOLDER ATTRIBUTES (NORMATIVE) .................................................................................................................. 295

APPENDIX K. RCS FOLDER ATTRIBUTES (INFORMATIVE) .................................................................................................................. 296

Figures

Figure 1 Resource structure defined by this specification .............................................................................................................. 33

Figure 2: NMS notifications subscription .............................................................................................................................................. 72

Figure 3: Strict synchronization with NMS .............................................................................................................................................. 74

Figure 4: Simplified synchronization with NMS .............................................................................................................................................. 76

Figure 5: Subscribing to filtered NMS notifications .............................................................................................................................................. 77

Figure 6: Operations on folders .................................................................................................................................................... 79

Figure 7: Operations on objects .................................................................................................................................................... 82

Figure 8: Retrieving a large list of objects .................................................................................................................................................... 84

Figure 9: Discovering the user’s storage hierarchical structure .............................................................................................................................................. 86

Figure 10: Bulk object creation .................................................................................................................................................... 87

Figure 11: Bulk object delete .................................................................................................................................................... 89

Figure 12: Bulk object update .................................................................................................................................................... 90

Tables

Table 1: 1-1 NMS event notification .............................................................................................................................................. 194

Table 2: Scope values for RESTful NMS API .............................................................................................................................................. 283

Table 3 Flag Names .................................................................................................................................................... 292

Table 4 Object Attributes .................................................................................................................................................... 294
Table 5 Standard Folder Attributes

Table 6 RCS Folder Attributes
1. **Scope**

   (Informative)

   This specification defines a RESTful Network API for Network Message Storage using HTTP protocol bindings.
2. References

2.1 Normative References


[REST_NetAPI_Common] “Common definitions for RESTful Network APIs”, Open Mobile Alliance™, OMA-REST_NetAPI_Common-V1_0, URL: http://www.openmobilealliance.org/

[REST_NetAPI_Notification Channel] “RESTful Network API for Notification Channel”, Open Mobile Alliance™, OMA-REST_NetAPI_NotificationChannel-V1_0, URL: http://www.openmobilealliance.org/

[REST_SUP_NMS] “XML schema for the RESTful Network API for Network Message Storage”, Open Mobile Alliance™, OMA-SUP-XSD_rest_netsapi_nms-V1_0, URL: http://www.openmobilealliance.org/


2.2 Informative References

[OMA-CPM-SD] “Converged IP Messaging System Description”, Open Mobile Alliance™, OMA-TC-CPM_System_Description-V2_0, URL: http://www.openmobilealliance.org/


[REST_WP] “Guidelines for RESTful Network APIs”, Open Mobile Alliance™, OMA-TP-Guidelines_for_RESTful_Network_APIs, URL: http://www.openmobilealliance.org/


3. Terminology and Conventions

3.1 Conventions

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].

All sections and appendixes are normative, unless explicitly indicated to be informative.

3.2 Definitions

For the purpose of this TS, all definitions from the OMA Dictionary [OMADICT] apply.

- **Client-side Notification URL**: An HTTP URL exposed by a client, on which it is capable of receiving notifications and that can be used by the client when subscribing to notifications.
- **Heavy-weight Resource**: A resource which is identified by a resource URL which is then used by HTTP methods to operate on the entire data structure representing the resource.
- **Light-weight Resource**: A subordinate resource of a Heavy-weight Resource which is identified by its own resource URL which is then used by HTTP methods to operate on a part of the data structure representing the Heavy-weight Resource. The Light-weight Resource URL can be seen as an extension of the Heavy-weight Resource URL.
- **Long Polling**: A variation of the traditional polling technique, where the server does not reply to a request unless a particular event, status or timeout has occurred. Once the server has sent a response, it closes the connection, and typically the client immediately sends a new request. This allows the emulation of an information push from a server to a client.
- **Notification Channel**: A channel created on the request of the client and used to deliver notifications from a server to a client. The channel is represented as a resource and provides means for the server to post notifications and for the client to receive them via specified delivery mechanisms.
  
  For example in the case of Long Polling the channel resource is defined by a pair of URLs. One of the URLs is used by the client as a call-back URL when subscribing for notifications. The other URL is used by the client to retrieve notifications from the Notification Server.
- **Notification Server**: A server that is capable of creating and maintaining Notification Channels.
- **Server-side Notification URL**: An HTTP URL exposed by a Notification Server, that identifies a Notification Channel and that can be used by a client when subscribing to notifications.

3.3 Abbreviations

- **ACR**: Anonymous Customer Reference
- **API**: Application Programming Interface
- **CPM**: Converged IP Messaging
- **GSM**: Global System for Mobile
- **GSMA**: GSM Association
- **HTTP**: HyperText Transfer Protocol
- **ID**: Identity
- **JSON**: JavaScript Object Notation
- **MIME**: Multipurpose Internet Mail Extensions
- **MMS**: Multimedia Messaging Service
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSISDN</td>
<td>Mobile Subscriber ISDN Number</td>
</tr>
<tr>
<td>NMS</td>
<td>Network Message Storage</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>RCS</td>
<td>Rich Communication Suite</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>SCR</td>
<td>Static Conformance Requirements</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>TS</td>
<td>Technical Specification</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WP</td>
<td>White Paper</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
</tbody>
</table>
4. Introduction

The Technical Specification of the RESTful Network API for Network Message Storage contains HTTP protocol bindings for Network Message Storage, using the REST architectural style. The specification provides resource definitions, the HTTP verbs applicable for each of these resources, and the element data structures, as well as support material including flow diagrams and examples using the various supported message body formats (i.e. XML and JSON).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations:

- Create a folder.
- Delete a folder
- Rename a folder
- Move folder(s)
- Copy folder(s)
- Store an object (e.g., message, file, etc.) in the storage, in a particular folder
- Delete an object
- Move object(s)
- Copy object(s)
- Bulk Creation of objects (e.g. bulk upload of objects from the device storage to the network storage)
- Bulk Deletion of objects
- Bulk Update of objects (note: in this version of the specification, only flags of objects can be updated)
- Retrieve information about a stored object (e.g. message, file, etc.) such as:
  - size
  - hierarchical location (i.e. the equivalent of a full pathname in a file system)
  - flags (string labels) associated with the object
- Update flags (string labels) associated with an object
- Search and retrieve information about a set of selected objects, for example a list of messages, including associated header information such as subject, date and time
- Retrieve the payload (i.e. stream of bytes) of a stored object
- Retrieve individual attachments of an object
- Retrieve information about a folder, such as:
  - hierarchical location (i.e. the equivalent of a full pathname in a file system)
  - identification of the contained objects and/or sub-folders (i.e. children in the hierarchy of the storage)
- Search and retrieve information about a set of folders (e.g. root folder)
- Manage subscriptions to event notifications on changes occurring in the network storage
- Notify client(s) about network message storage events
− Manage synchronization between client local storage and network storage
− Subscribe to filtered notifications

In addition, this specification provides:

− Support for scope values used with the authorization framework defined in [Autho4API_10]
− Support for Anonymous Customer Reference (ACR) as an end user identifier
− Support for “acr:auth” as a reserved keyword in an ACR
5. Network Message Storage API definition

This section is organized to support a comprehensive understanding of the Network Message Storage API design. It specifies the definition of all resources, definition of all data structures, and definitions of all operations permitted on the specified resources.

Common data types, naming conventions, fault definitions and namespaces are defined in [REST_NetAPI_Common].

The remainder of this document is structured as follows:

Section 5 starts with a description of the concepts used by this API (section 5.1). This is followed by a diagram representing the resources hierarchy followed by a table listing all the resources (and their URL) used by this API, along with the data structure and the supported HTTP verbs (section 5.2). What follows are the data structures (section 5.3). A sample of typical use cases is included in section 5.4, described as high level flow diagrams.

Section 6 contains detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP methods, and the supported HTTP verbs. For each supported HTTP verb, a description of the functionality is provided, along with at least one example of a request and the corresponding response. For each unsupported HTTP verb, the returned HTTP error status is specified, as well as what should be returned in the Allow header. Note, throughout this specification the terms: “HTTP method” and “HTTP verb” are used interchangeably.

All examples in section 6 use XML as the format for the message body, while JSON examples are provided in Appendix D.

Section 7 contains fault definition details such as Service Exceptions and Policy Exceptions.

Appendix B provides the Static Conformance Requirements (SCR).

Appendix C provides application/x-www-form-urlencoded examples, where applicable.

Appendix F provides a list of all Light-weight Resources, where applicable.

Appendix G defines authorization aspects to control access to the resources defined in this specification.

Appendix H provides a list of the most common object flags.

Appendix I provides a list of object attributes.

Appendix J provides a list of folder attributes.

Note: Throughout this document client and application are used interchangeably.

5.1 Concepts

5.1.1 Object

The object resource in the context of this specification comprises of:

- a payload (sequence of bytes, divided into one or more payload parts)
- flags, which are string labels that may have applicative meaning, such as:
  - \Flagged
  - \Seen
- attributes that contain metadata, such as:
  - Message-Context (e.g., voice-message, multimedia-message, pager-message)
  - top-level MIME headers (e.g., Content-Type, Content-Location)
- location (i.e. the equivalent of a full pathname in a hierarchical file system)

Each object resource in a given storage is identified by “resourceURL” containing an “objectId”, which is a string that MUST be unique in the context of that storage (under the same box and over the lifetime of that box, even if the object is deleted). The objectId is assigned by the storage server. Note: “operations” is a reserved keyword and MUST NOT be used as an objectId.
The path for any object in the network storage is made up of a sequence of folder names starting from the root folder and ending with the given object’s objectId where the folder names and the objectId are separated by a “/” (U+002F) character.

### 5.1.2 Folder

Similar to the abstract model of a file system, a folder in the context of this specification is a container with a designated location (pathname) that can contain objects and/or sub-folders (i.e. be considered as their parent in the location hierarchy). The folder resource in the context of this specification comprises of:

- attributes
- name
- location (i.e. the equivalent of a full pathname in a hierarchical file system)
- identification of the contained objects and/or sub-folders (i.e. children in the hierarchy of the storage)

Similar to object, as listed above, a folder can also be assigned attributes that contain metadata. A client can perform batch search for objects/folders by their attributes.

Each folder resource in a given storage is identified by resourceURL containing a “folderId”, which is a string that MUST be unique in the context of that storage (under the same box and over the lifetime of that box, even if the folder is deleted). The folderId is assigned by the storage server. Note: “operations” is a reserved keyword and MUST NOT be used as a folderId.

Each folder also has a name, which is used to construct the location (path). The name is a string which MUST be unique in the context of the folder’s parent folder in the hierarchy of the storage.

The path for any folder in the network storage is made up of a sequence of folder names starting from the root folder and ending with the given folder’s name where the folder names are separated by a “/” (U+002F) character.

While a folder in a file system model conceptually contains objects and sub-folders, the resource tree shown in Figure 1 does not mimic that hierarchy. For the purpose of the API, objects and folders are identified by resourceURL containing a unique objectId and folderId respectively, and not by their location (pathname). However, through RESTful operations and queries on the resource tree, it is possible for a client to discover the location (full pathname) of all objects and folder in the storage; hence it is possible to map the full hierarchy. It is also possible to perform a query to resolve a pathname (of a folder or an object) to the equivalent resourceURL (containing a folderId or objectId).

Folder can be created explicitly using POST /folders or implicitly while creating a new object given a parent folder path which doesn’t already exist. This results in implicit creation of the non-existent folder(s).

### 5.1.3 Box

A box represents the logical store that belongs to designated owner(s). For example, a subscriber could be assigned her own private box, a group of subscribers who need a shared store could be assigned a shared box. In other deployment scenarios a box could be assigned to a non-human owner, e.g. a connected camera which is assigned a dedicated box. Each box is identified by “boxId”, which is a string that MUST be unique in the context of that storage. The creation and deletion of boxes, and the assignment of boxIds, are out of the scope of this specification.

### 5.1.4 Subscriptions and notifications

Clients can subscribe for asynchronous notification of changes in the network message storage.

1. The basic mechanism is that a client subscribes for changes to the box (possibly restricted by a filter to a particular area of interest), and the server notifies that client by sending it change notifications that describe each change that occurs.

2. If the client wishes to catch up from some previous point (e.g., if it has been offline for a period), it subscribes for changes from that point, or updates an existing subscription to restart from that point. The point is identified by a server-provided string called a “restartToken”. The server then sends the client change notifications that describe each change that has occurred from that point – first all those that have already occurred, and then any changes which occur subsequently. Hence both catch-up and ongoing changes are handled by the same notifications.

3. To avoid lost updates, every object and folder has a “lastModSeq”. This is a strictly-increasing integer which allows the client to determine unambiguously whether or not to apply a change it has received. This allows
clients to behave correctly in the presence of both notification reordering and interleaving of notifications with
direct retrievals or searches (see section 5.1.4.4). It also permits server implementations more freedom in computing
change notifications.

4. To detect lost notifications, every notification has an index. This increments by 1 for each notification. When a
client detects a missing notification, it updates its existing subscription with the previous restartToken, causing the
server to re-issue the missing change notifications. This is similar to the catch-up case, except that a new
subscription is not required.

The subsequent sections provide more detail regarding each of these aspects. Example flows are given in section 5.4.2.1.

5.1.4.1 Notifications

A client may subscribe for notification of changes. Such clients will receive a change notification whenever the storage is
considered to be changed (subject to any filtering rules specified when the subscription was created).

Each change notification relates to a single object or folder. It describes the resulting state of that object or folder after the
change (not the change itself) – this ensures it is idempotent. It describes the entire mutable state of the object or folder (not
just the part that has changed) – this ensures it is safe to ignore all but the latest change notification for that object or folder.

The server combines one or more change notifications into a list before sending them to the client. Thus each notification
POSTed by the server to the client is a list of change notifications (“NmsEventList”), not an individual change notification
(“NmsEvent”). Furthermore, if Notification Channel [REST_NetAPI_Notification_Channel] is in use it typically groups
multiple notifications into a list (“NotificationList”) before delivery. Hence the client will typically receive a list of lists of
change notifications in each delivery.

When the server notifies a sequence of changes e.g., in response to a subscription with a restartToken, that sequence of
changes notified may be different from the actual sequence, but it MUST have the same net effect as the actual sequence of
changes that occurred. For example, if within the relevant period

- an object’s \Seen flag starts out absent, is added and then removed again, the server may notify only that the object’s
  \Seen flag was removed;
- if an object is marked \Flagged and then the object is deleted, the server may notify only that the object was deleted;
- an object which already has the \Flagged flag is not changed at all, the server may notify that its \Flagged flag was
  added.

5.1.4.2 Tracked storage changes

This section defines what is considered to be a change to the network message storage, i.e., what triggers a change
notification (section 5.1.4.1) and an update of the object’s or folder’s lastModSeq (section 5.1.4.4).

The following operations (and only these operations) are considered to cause a change in an object in the storage:

- creation (whether direct or as part of a recursive operation)
- user-initiated deletion (whether direct or as part of a recursive operation)
- expiry
- “parentFolder” change
- flag change

The following operations (and only these operations) are considered to cause a change in a folder in the storage:

- creation (whether direct or as part of a recursive operation)
- user-initiated deletion (whether direct or as part of a recursive operation)
- expiry
• “parentFolder” change
• change of any Light-weight Resource within a folder, as described in Section 6.14.1.1 “Light-weight relative resource paths”, e.g., changing its “folderName”.

This means that an object or folder is considered to have changed only when the change affects the object or folder itself. That is, changes to the object’s or folder’s derived properties (such as its children or the name or location of one of its parent folders) is not considered a change to the object or folder itself. In particular:

• Changing the “folderName” of a folder is considered to change that folder, but is not considered to change any subfolders or objects within it (even though their “path” and “parentFolderPath” values are now different).
• Adding or deleting an object within a folder is considered to change that object, but is not considered to change the folder itself (even though its “objects” value is now different).
• Adding or deleting a subfolder within a parent folder is considered to change the subfolder, but is not considered to change the parent folder itself (even though its “subFolders” value is now different).
• Moving a folder to a new location is considered to change the moved folder (since its parentFolder changes), but is not considered to change any subfolders or objects within it (even though their path and parentFolderPath values are now different), nor the target (folder) (even though its “subFolders” value is now different).

In each of these cases, it is understood that the client will infer any derived changes it needs in order to capture the overall change to the storage. For example, when a parent “folderName” changes, even though the storage does not report a change against the subfolders’ path values, it is understood that the client derives this change locally based on the notification of the parent folderName change.

5.1.4.2.1 Box reset

A box reset is an event where the entire contents of a box are removed or replaced (e.g., by an administrative operation). This is also considered to be a change to the network message storage. However, in this case the individual changes may be too many to list, or may not even be available any more.

A box reset notification informs the client that the entire contents of the box have changed – it corresponds to a UIDVALIDITY change in IMAP [RFC3501]. The client should dispose of all existing object, folder and subscription references (except for the subscription on which it received the ResetBox notification), and use NMS API requests to determine the current content of the box. The server will not issue individual notifications for objects or folders which are removed, changed, or created as a result of a box reset. Notifications issued after the box reset will notify the client of changes subsequent to the reset. The server SHOULD resend box reset notifications if requested, in the same way as for other notifications (see Section 5.1.4.3). Thus a lost box reset notification will be discovered when the client attempts to refresh its subscription, discovers that it is no longer valid, and re-subscribes.

5.1.4.3 Requesting past notifications using restartToken

The box has a parameter called “restartToken”, which is a varying implementation-specific string value which represents the current point in the stream of changes (see section 5.1.4.2). Informally, it can be thought of as a timestamp or sequence number, though it can be implemented in other ways.

Each subscription contains a restartToken field. This field is set to the box’s current restartToken when the subscription is created, and continually updated as notifications are issued. It is returned to the client in the restartToken element when the subscription is created or updated. Also, each notification (NmsEventList) contains the box’s restartToken at the point immediately after the notification was issued.

When a client subscribes for changes, it may specify a restartToken (which must have been obtained from a previous notification or subscription). Alternatively, at any point the client can update an existing subscription to set a previous value of the restartToken. In either case, the server then issues notifications representing all changes since the point indicated by the restartToken. This will cause some change notifications to be duplicated. The client can use the lastModSeq values to ensure only non-duplicated changes are applied. It then continues to send notifications of changes which occur subsequently.
5.1.4.3.1 Implementation note (Informative)

The server implementation can freely choose the content of the restartToken. However, a good implementation will not require unbounded storage of previous notifications, nor require per-client storage on the server. This section gives one concrete example of how an implementation could achieve this.

One way in which an implementation can handle requests for past notifications is by storing a representation of a timestamp in the restartToken, and storing a last-changed timestamp in every object and folder in the storage (including recently deleted objects). As the contents of the storage are modified, the corresponding last-changed timestamps are updated. Then when the client presents a restartToken, the server simply identifies all objects and folders in the subscriber’s box which have a last-changed timestamp more recent than the one in the restartToken and generates notifications for them with their current state.

The specification intentionally permits a range of alternative implementations. For example, a practical implementation may choose to use sequence numbers (strictly increasing over the whole box) instead of a timestamps. And even though the NMS API only requires that lastModSeq values be increasing for the particular object or folder they apply to, for convenience an implementation may choose to use these sequence numbers directly as lastModSeq values.

If the restartToken also contains information about the validity of the box for which it was generated (e.g., UIDVALIDITY – see section 5.1.4.2.1), the server can also detect if the validity has changed. If the validity of the restartToken does not match that of the box, this indicates that the box has been reset since the client last obtained change notifications from the server, and the server can issue a ResetBox notification. The validity check ensures the ResetBox notification will be resent if it is lost.

5.1.4.4 Handling reordering and duplication with lastModSeq

Notifications are subject to loss, reordering, and duplication. Every object and folder has a lastModSeq which allows the client to determine unambiguously whether or not to apply a change it has received. This allows clients to behave correctly in the presence of both notification reordering and interleaving of notifications with direct retrievals or searches.

Each object has a lastModSeq, which is a mod-sequence (modification sequence number – see below) value used to determine the relative order of changes to the object. Whenever the object is changed the lastModSeq value MUST be updated. A change that makes no difference (e.g., setting a flag which is already set) SHOULD NOT change the lastModSeq.

Each folder has a lastModSeq, which is a mod-sequence value used to determine the relative order of changes to the folder. Whenever the metadata changes the lastModSeq value MUST be updated. A change that makes no difference (e.g., setting the folderName to the value which it already has) SHOULD NOT change the lastModSeq.

A mod-sequence is a positive unsigned 64-bit value. When a relevant operation is performed the storage MUST obtain a mod-sequence value, and MUST set the lastModSeq value of the object or folder being acted upon to that value. The server MUST guarantee that each relevant operation performed on the same object or folder (including simultaneous operations on different metadata items from different connections) will get a different mod-sequence value. Also, for any two successful relevant operations performed on the same object or folder, the mod-sequence of the second completed operation MUST be greater than the mod-sequence of the first completed. Note that the latter rule disallows the use of the system clock as a mod-sequence, because if the system time changes (e.g., an NTP (Network Time Protocol) client adjusting the time), the next generated value might be less than the previous one. See [RFC4551] for an informative discussion of mod-sequences, though unlike IMAP mod-sequences, NMS mod-sequences of different objects or folders cannot be compared.

A client which is intending to keep in sync with the server should record the lastModSeq value of each object and folder. When it receives a change notification for a particular object or folder, it should compare the lastModSeq value of the notification with the current lastModSeq value of the object or folder. If the lastModSeq of the notification is less than or equal to the current value, the notification is out of order and should be ignored. This ensures correct operation even when notifications are reordered or duplicated, or arrive interleaved with objects and folders retrieved by other means (e.g., search).

5.1.4.5 Handling loss with index and subscription update

If the notification mechanism in use is unreliable, clients must detect and recover from loss.

Within a particular subscription, each notification (NmsEventList) contains an index. This starts at 1 when the subscription is created, and increments for each notification. When a subscription is updated the index is not reset, but continues to increment, so that within the sequence of notifications from a particular subscription, each index is unique. The response to
the subscription update contains the current index, so the client knows which index to expect after the update. For further information and an example see sections 5.4.2.1 and 6.20.5.1.

Lost notifications are detected as follows. The client observes the index values of the notifications which are received. If a particular index is not received, even though a greater index has been received, after an appropriate timeout the client can infer that a notification has been lost.

Lost notifications are recovered as follows. The client keeps track of the most recent restartToken received in consecutive notifications. If a lost notification is detected, the client simply updates the subscription to restart from the restartToken received immediately before the lost notification.

A special case of loss is when a client has been shut down or disconnected. In this case, when the client restarts it should supply the latest previously-received restartToken when it makes an NmsSubscription request, so that the server will notify all changes since that point. If there were already lost notifications when it was shut down or disconnected, the restartToken supplied should be the restartToken received immediately before the first lost notification.

### 5.1.5 Managing local storage mirror (cache) at the client

Clients may need to have a local cache, representing the storage at the server. In order to keep it up-to-date, any change made on the server needs to be mirrored in the local cache, which requires tracking of storage changes. Tracking such changes in a multi-device (multi-client) environment is a complicated task that requires extended state management. This tracking incurs overhead both in complexity (cost) of the client and server implementations and in their runtime performance when synchronizing the changes between the client(s) and the server.

Different deployment scenarios have different requirements with regards to the tradeoff made between full and accurate change tracking at the expense of complexity and partial and approximate tracking at the expense of excluding some information from the scope of changes synchronization. Tradeoffs regarding performance and bandwidth should also be considered.

The NMS API offers two alternatives for synchronizing their local cache.

- **Strict synchronization** informs clients of all changes to the storage asynchronously, using subscriptions and notifications (and is dependent on support for asynchronous notifications and the management of lastModSeq).
- **Simplified synchronization** informs clients of significant changes to the storage when required, using search (and is dependent on support for the CreatedObjects and VanishedObjects search types).

In both cases, synchronization is one-way: it provides a way for clients to learn of changes that have occurred on the server, but does not assist clients in informing the server of changes that have occurred locally.

In some use-cases a user may only be interested in selective tracking (e.g. user may only care about the most recent changes and consider older objects/folders irrelevant). The NMS API supports this by the filter parameter on subscriptions and by additional search criteria on searches.

### 5.1.5.1 Strict Synchronization

Strict synchronization uses subscriptions and notifications to keep the client informed of changes to the network message storage.

Strict synchronization can be used both online (where a client receives a stream of change notifications) and offline (where a client asks the server to be told of changes that have occurred since it was last connected). Strict synchronization is reliable as it can recover from notification loss, reordering, or duplication. During synchronization only the changes and a per-box token are exchanged.

To perform strict synchronization, the client should follow the steps for establishing a subscription and processing notifications, as described in section 5.1.4.

### 5.1.5.2 Simplified Synchronization

Each box has a “creationCursor”, which is a varying implementation-specific string value. The creationCursor enables a client to request retrieval of objects that were created in the store but not yet known to the client.
The creationCursor is set by the server and returned to the client in each “CreatedObjects” search response. At any point, the client can use a known creationCursor (that was returned in a previous CreatedObjects search response). The server then returns all existing objects in the store created since the point indicated by the creationCursor.

An illustrative example for creationCursor implementation can use an unsigned long integer (converted to a string representation) that is managed by the server as follows: as each object is added to the box it is assigned a creation sequence number which is higher than all the creation sequence numbers previously assigned (but not necessarily contiguous). The creation sequence number is immutable. When responding to CreatedObjects search the server will return the highest creation sequence number in the box as the value of the creationCursor.

To perform the simplified synchronization the client SHOULD follow the following steps:

1. Synchronize new objects: Fetch all objects created since the last synchronization performed by the client (using the creationCursor).
2. Synchronize purged objects: Fetch objectIds of objects that have been permanently deleted.
3. Synchronize significant flag changes:
   a. Sync Read/Unread flag for all objects: search for objects that do not carry the “Seen” flag. All objectIds returned by the search have the flag unset, and therefore the client infers that all other objects have the “Seen” flag set.
      This step assumes that most objects in the store are read (seen), therefore searching for non “Seen” objects optimizes the retrieval of a relatively short list of objectIds.
   b. Optionally, use similar approach to synchronize other significant flags. The rest of the flags will not be synchronized into the local store.

### 5.1.6 Root folder(s) discovery

A client can perform traversal of the storage hierarchical structure, provided that it can discover the root folder(s) of the hierarchy (i.e. the starting point(s) for traversal). The search by folder attributes operation is used to identify root folder(s) in the message storage. A folder attribute named “Root” with the value “Yes” designates such a starting point. In some deployment scenarios other well-known attribute values may be used and other restrictions may apply (e.g. mandating only single root folder). For further information see section 5.4.7.

When the client retrieves the root folder it will discover its name. By default, the name of the root folder is an empty string unless specifically assigned to be some other name by the server.

Renaming a folder (e.g. a root folder) may be prohibited by server policy.

### 5.1.7 Deletion

There are two ways in which an object or folder can be deleted:

- A client may delete an object or folder by supplying a DELETE request over the API. This is called “user-initiated deletion”.
- The storage server MAY at any time spontaneously delete an object or folder. This is called “expiry”. Expiry typically occurs according to service provider policy, e.g., older objects may be expired automatically in order to maintain a maximum of (say) 1000 messages or an age limit of (say) 90 days on the contents of the box.

Both of these kinds of delete update the object or folder’s lastModSeq and trigger a notification. The notification indicates which has occurred.

Clients may choose to associate different semantics with these different kinds of delete, e.g., user-initiated deletion may result in the object or folder being removed from local storage, whereas expiry may be ignored (for instance to allow the user to hold onto the local copy of a server-deleted object).

If the server receives a subscription request that includes a restartToken from before the point at which the object or folder was deleted, the server SHOULD return a notification indicating the user-initiated deletion or expiry. However the server MAY omit this notification. The server SHOULD NOT omit the notification unless a reasonable period of time has elapsed since the delete occurred (i.e., such that the client could reasonably be expected to have issued a subscription request within this period).

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document
If the server receives a search request for “VanishedObjects”, it SHOULD return a reference to every object that has been user-deleted recently, where the definition of “recently” is subject to service provider’s policy (e.g., last 30 days, last 1000 objects). Reference to objects that have recently expired MAY also be returned in this response, subject to service provider’s policy. Note that repeated search requests for VanishedObject that are submitted within short period are likely to generate (at least partially) the same list of objects in the response.

In order for the storage to correctly return deletion event notifications and/or VanishedObjects search responses, it is necessary for the storage to retain, for each deleted object or folder:

- the objectId or folderId of the deleted object or folder
- the lastModSeq value of the deletion operation (see Section 5.1.4.4)
- the correlationId and correlationTag of each deleted object (see Section 5.1.12.1)
- a subset of the attributes of each deleted object, as determined by server policy (see Section 5.3.2.27).

5.1.8 External content

The content (i.e., payload and payload parts) of an object is made available from one of two places. It can be available via the NMS resource tree, or it can be available via an external reference. This allows NMS implementations to place the payload and/or payload parts in an external server, so that clients can access them without placing load on the NMS itself.

The server indicates the location of the entire payload and any individual payload parts through “payloadURL” and “payloadPart” elements of Object data structure respectively. The location (URL) can refer to a resource within the NMS resource tree, or external to it.

Given the fact that the object’s payload or payload part location is not under client control, clients SHOULD obtain this location from the appropriate object’s payloadURL and payloadPart elements. Clients SHOULD NOT attempt to construct this URL themselves, e.g. using the resource tree shown in Figure 1 and assuming that content is always served directly by NMS.

5.1.9 Inline content

If an object payload can be represented as a textual string of a reasonable length then the NMS server MAY convert a payload resource to a string and add a “TextContent” attribute (see Appendix I) that will carry this string.

When performing the conversion the server MUST add the new attribute and MUST delete the original payload resource. A common use-case for this conversion is an SMS object.

The decision on when to perform the conversion depends on service provider’s policy (e.g. if the resulting text is too long, it will not be converted) and is out of scope of this spec.

5.1.10 Batched retrieval

Folder retrieval and object and folder search allow the client to retrieve a list of entries. This list of entries might be larger than the server or the client is prepared to handle at one time, and so these operations provide a mechanism for batched retrieval.

Batched retrieval uses the following elements of the search request and response data types:

- In search requests, the client supplies a maximum number of entries in the “maxEntries” element.
  1. The maxEntries element indicates the maximum number of entries the client is prepared to accept in a single batch.
  2. The server MUST NOT return more than this many entries in the response. It MAY choose to return fewer entries.
• In search responses, in addition to the batch of entries the server can also supply a cursor value (in the cursor element).

  1. If the cursor element is present, it indicates that there may be further entries in the list beyond the end of this batch. (It does not indicate that there certainly are further entries. For example, if the server is performing a search it may return a cursor to indicate the search is not yet complete. It may in fact be the case that there are no further matches beyond this point, but because the server has not yet determined this it cannot omit the cursor.)

  2. If the cursor element is absent, it indicates that there are no further entries, i.e., that the list is now complete.

  3. The value and format of the string are implementation specific. Clients SHOULD NOT attempt to interpret or alter the cursor value.

• In subsequent search requests, the client can supply a cursor value (in the “fromCursor” element) indicating the previous batch to be continued, in addition to the maximum number of entries (in the maxEntries element).

  1. If the fromCursor element is absent, the batch starts from the first matching entry.

  2. If the fromCursor element is present:

     i. It MUST contain a cursor value obtained from a previous search response.

     ii. This subsequent request MUST be to the same resource URL and have precisely the same “searchCriteria”, “searchScope”, and “sortCriteria” as the previous search request corresponding to the previous search response.

     iii. The batch is a continuation of the previous batch, i.e., it starts from the first matching entry after the last entry of the previous search response. The server SHOULD make best efforts to start the response from at or near this position, or from the start of the matches if this is not possible.

  3. Since the cursor encapsulates server state information which might be volatile, especially in a multi-device environment, the server is not required to ensure that each batch is a precise continuation of the previous batch. However, the server must make best efforts to ensure this is so. The cursor mechanism guarantees that:

     i. If there are no intervening changes to the box (such as object or folder creations or deletions), the batch MUST be a precise continuation of the previous batch.

     ii. If this is an object (or folder) search, with default selection criteria (i.e., the searchCriteria, searchScope, and sortCriteria are all absent), then every object (or folder) which existed in the box at the point of the first request and still exists at the point of the final response MUST appear at least in one of the batches (i.e. if the client retrieves all the batches it will not miss any stored object (or folder)).

  4. If the fromCursor is invalid (e.g., it has been modified by the client, or it came from a request with different selection criteria), the server MAY return either an HTTP error response or an arbitrary subset of matches.

Section 5.4.6 demonstrates the expected flow of requests and responses.

Similarly, folder retrieval allows the client to retrieve a large list of references to contained objects and subfolders. Folder retrieval for a large folder uses the same batch retrieval mechanism as search (with cursor, fromCursor, and maxEntries), with the difference that fromCursor and maxEntries are provided through query parameters of the GET request. For further information see section 6.13.3.3.
5.1.11 Bulk Creation

In bulk creation operation, the client intends to create multiple objects using a single request. The list of objects are passed in a single HTTP POST request (invoked on “/bulkCreation” resource). If the identified parent folder of a given object in the list does not exist, the server SHALL create the parent folder before creating and placing the object in the folder.

The response body includes a list of success or failure status for each object in the request list respectively. The HTTP response code reflects the status of the bulk operation as a whole, which is considered successful if at least one object was created successfully.

5.1.12 Correlation (Informative)

5.1.12.1 Introduction

Some clients receive objects via another transport mechanism (e.g., CPM messaging [OMA-CPM-SD]) as well as via the NMS. Such clients generally wish to correlate these objects with notifications received from the NMS, e.g., to avoid unnecessary downloads, to avoid duplications, and to ensure flag changes and deletions made by other NMS clients are correctly applied to local objects. Clients may perform this correlation by means of the “correlationId” or “correlationTag” elements contained in all relevant notifications.

Where an object has an ID which uniquely identifies the object and which is transported over the relevant mechanism, that unique ID is the best choice for correlation. This is stored in the object’s correlationId element in the NMS.

Not all objects have a unique ID which can be used for correlation; e.g., CPM messages do not necessarily contain a Message-ID header, and the Contribution-ID is not unique for chat messages within a session. Furthermore, some mechanisms (e.g., SMS) can transport object content but not additional headers and hence cannot transport a unique ID. For these reasons, a correlation tag can be provided as a secondary means of correlation.

In some circumstances both correlationId and correlationTag are required. For example, if the transport mechanism of an object is chosen after it is deposited in NMS and the means of correlation to be used depends on the transport mechanism, then the client should supply both the correlationId and the correlationTag when depositing the object so as to ensure that all necessary correlation information is available to the receiving client whichever transport mechanism is chosen.

To enhance interoperability, it is desirable that clients agree on what value is used for the correlationId and correlationTag. The sections below suggest a possible value that could be used for each. Profiles may impose stricter requirements.

In some deployment scenarios the server will be able to generate the agreed value for the correlationId and correlationTag from other information contained within the object. If the client knows that the server will do this, it may omit them when creating the object. This is controlled by service provider policy.

5.1.12.2 Correlation ID

If an object contains a Message-ID attribute (as provided by, e.g., [RFC5322] MIME messages or [RFC5438] IMs), the client may supply the value of this attribute as the correlationId field.

Otherwise, if the object contains an attribute which is defined to be a globally-unique ID, the client may supply the value of this attribute as the correlationId field.

Otherwise, the client may omit the correlationId field.

5.1.12.3 Correlation tag

The correlation tag is weaker than a correlation ID: whereas a correlation ID uniquely identifies an object, a correlation tag identifies a particular object only probabilistically. Despite this, it is still useful in scenarios where no unique correlation ID is available.

The appropriate method for generating the correlation tag is beyond the scope of the NMS specification, since it depends upon the kind of objects being stored, the information available via the non-NMS mechanism, and other considerations (e.g.
the computational ability available to the clients). For example, a suitable method might involve a hash function of certain attributes and/or body parts of the object.

Clients should be aware of the limitations of the correlation tags they use – for example, if two distinct objects are given the same correlation tag value, then any correlation matching must fall back on heuristics such as order of arrival to resolve the ambiguity. In this case the client cannot guarantee correct correlation, and so it must not depend on achieving this.

5.2 Resources Summary

This section summarizes all the resources used by the RESTful Network API for Network Message Storage.

The "apiVersion" URL variable SHALL have the value “v1” to indicate that the API corresponds to this version of the specification. See [REST_NetAPI_Common] which specifies the semantics of this variable.

The "storeName" URL variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value for that variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).

The "boxId" URL variable can be used to identify specific area (or a ‘box’) allocated within the store. The value for this variable depends on the deployment scenario and the service provider’s policy. For example:

- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId)
- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group
- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine

The figure below visualizes the resource structure defined by this specification. Note that those nodes in the resource tree which have associated HTTP methods defined in this specification are depicted by solid boxes.

Note: client resources such as callback URL (used in POSTing notifications) and server arbitrary chosen external URL where an object’s payload can be fetched from are not depicted in the resource tree.
Figure 1 Resource structure defined by this specification

Note: “pathToId” resource is read as path-To-Id.

The following tables give a detailed overview of the resources defined in this specification, the data type of their representation and the allowed HTTP methods.
## 5.2.1 Resources allowing a client to manage individual objects

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td>Resource containing all objects</td>
<td>/objects</td>
<td>Object</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reference (optional alternative for POST response) Empty (used for some responses)</td>
<td></td>
</tr>
<tr>
<td>A stored object</td>
<td>/objects/{objectId}</td>
<td>Object</td>
<td>Retrieve the attributes (metadata) associated with the object</td>
</tr>
<tr>
<td>Flags associated with the stored object</td>
<td>/objects/{objectId}/flags</td>
<td>FlagList</td>
<td>Retrieve the flags (string labels) associated with the object Create or update the flags (string labels) associated with the object</td>
</tr>
<tr>
<td>Individual flag</td>
<td>/objects/{objectId}/flags/{flagName}</td>
<td>Empty (used for some responses and for PUT requests) Retrieve/check existence of an individual flag (string label)</td>
<td>Retrieve/check existence of an individual flag (string label) Add individual flag (string label)</td>
</tr>
</tbody>
</table>
### 5.2.2 Resources allowing a client to retrieve stored content of an object payload or payload part

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored content of an object payload</td>
<td>&lt;payload URL specified by the server in the payloadURL field of Object&gt;</td>
<td>Any MIME content (the one of the object e.g. multipart/mixed or image/jpeg)</td>
<td>Retrieve the payload (stream of bytes) of the object no PUT POST DELETE</td>
</tr>
<tr>
<td></td>
<td>For content available via the NMS resource tree this URL is:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>//({serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payload</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For externally referenced content, this is an arbitrary URL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload part of the stored object</td>
<td>&lt;payload part URL specified by the server in the payloadPart field of Object&gt;</td>
<td>Any MIME content (the one of the payload part)</td>
<td>Retrieve individual object payload part no PUT POST DELETE</td>
</tr>
<tr>
<td></td>
<td>For content available via the NMS resource tree this URL is:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>//({serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payloadParts/{payloadPartId}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For externally referenced content, this is an arbitrary URL.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.2.3 Resources allowing a client to manage individual folders

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE</td>
</tr>
<tr>
<td>Resource containing all folders</td>
<td>/folders</td>
<td>Folder</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reference (optional alternative for POST response)</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Create a folder</td>
</tr>
<tr>
<td>A folder</td>
<td>/folders/{folderId}</td>
<td>Folder</td>
<td>Retrieve the folder properties (such as its location and list of contained objects/subfolders)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: Query string parameter controls the amount of information returned in the response (e.g. response to include subFolders element and not objects element)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Delete a folder from the storage, including contained folders and objects (with their payload)</td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}</td>
<td>Data Structures</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Individual folder data</td>
<td>/folders/{folderId}/[ResourceRelPath]</td>
<td>The data structure corresponds to an element within the Folder structure indicated by the resource URL.</td>
<td>GET: Retrieve individual folder information parameters (e.g. &quot;name&quot; parameter) PUT: Update individual folder information parameters (e.g. rename the folder by changing its &quot;name&quot; parameter) Note: Renaming a folder (e.g. a root folder) may be prohibited by server policy.</td>
</tr>
</tbody>
</table>
### 5.2.4 Resources allowing a client to retrieve information and/or perform operations on objects

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}/objects/operations</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE</td>
</tr>
<tr>
<td>Information about a selected set of objects in the storage</td>
<td>/search</td>
<td>SelectionCriteria (used for POST request)</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ObjectList (used for POST response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ObjectReferenceList (optional alternative for POST response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Response to POST is always ObjectList except where otherwise specified.</td>
<td></td>
</tr>
<tr>
<td>Resource URLs of a selected set of objects in the storage</td>
<td>/pathToId</td>
<td>Reference (used for GET response)</td>
<td>Retrieve resource URL for an object, based on its pathname which is provided via query string</td>
</tr>
<tr>
<td></td>
<td>Note: read as path-To-Id</td>
<td>PathList (used for POST request)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BulkResponseList (used for POST response)</td>
<td></td>
</tr>
<tr>
<td>Bulk creation of objects</td>
<td>/bulkCreation</td>
<td>ObjectList (used for POST request)</td>
<td>Create objects</td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}/objects/operations</td>
<td>Data Structures</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td>Bulk update of objects</td>
<td>/bulkUpdate</td>
<td>BulkUpdate (used for POST request)</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BulkResponseList (used for POST response)</td>
<td></td>
</tr>
<tr>
<td>Bulk deletion of objects</td>
<td>/bulkDelete</td>
<td>BulkDelete (used for POST request)</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BulkResponseList (used for POST response)</td>
<td></td>
</tr>
</tbody>
</table>

### 5.2.5 Resources allowing a client to retrieve information and/or perform operations on folders

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}/folders/operations</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
</tr>
<tr>
<td>Information about a selected set of folders in the storage</td>
<td>/search</td>
<td>SelectionCriteria (used for POST request)</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FolderList (used for POST response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FolderReferenceList (optional alternative for POST response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Response to POST is always FolderList except where otherwise specified.</td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{api Version}/ {storeName}/{boxId}/folders/operations</td>
<td>Data Structures</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------</td>
</tr>
<tr>
<td><strong>Resource URLs of a selected set of folders in the storage</strong></td>
<td>/pathToId Note: read as path-To-Id</td>
<td><strong>Reference</strong> (used for GET response) <strong>PathList</strong> (used for POST request) <strong>BulkResponseList</strong> (used for POST response)</td>
<td><strong>GET</strong> Retrieve resource URL for a folder, based on its pathname which is provided via query string <strong>PUT</strong> no <strong>POST</strong> Retrieve resource URLs for a list of folders, based on their pathnames <strong>DELETE</strong> no</td>
</tr>
<tr>
<td><strong>Resource for triggering object(s)/folder(s) copying</strong></td>
<td>/copyToFolder</td>
<td><strong>TargetSourceRef</strong> (used for POST request) <strong>BulkResponseList</strong> (used for POST response)</td>
<td><strong>GET</strong> no <strong>PUT</strong> no <strong>POST</strong> Copy referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder <strong>DELETE</strong> no</td>
</tr>
<tr>
<td><strong>Resource for triggering object(s)/folder(s) moving</strong></td>
<td>/moveToFolder</td>
<td><strong>TargetSourceRef</strong> (used for POST request) <strong>BulkResponseList</strong> (used for POST response)</td>
<td><strong>GET</strong> no <strong>PUT</strong> no <strong>POST</strong> Move referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder <strong>DELETE</strong> no</td>
</tr>
</tbody>
</table>
### 5.2.6 Resources allowing a client to manage subscriptions for storage changes

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL:  //{serverRoot}/nms/{apiVersion}/ {storeName}/ {boxId}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>All subscriptions in the storage</td>
<td>/subscriptions</td>
<td>NmsSubscriptionList (used for GET)</td>
<td>GET Retrieve all active NMS notification subscriptions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NmsSubscription (used for POST)</td>
<td>PUT no Create new subscription for notification for NMS changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NmsSubscriptionUpdate (used for POST request)</td>
<td>POST no</td>
</tr>
<tr>
<td>Individual subscription</td>
<td>/subscriptions/{subscriptionId}</td>
<td>NmsSubscription (used for GET and POST response)</td>
<td>DELETE no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NmsSubscriptionUpdate (used for POST request)</td>
<td></td>
</tr>
</tbody>
</table>

### 5.2.7 Resources allowing the server to notify a client about storage changes

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL &lt;specified by the client&gt;</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client notification about storage changes</td>
<td>&lt;specified by the client when subscription is created or during provisioning process&gt;</td>
<td>NmsEventList</td>
<td>GET no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST No Notifications of client about storage changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE no</td>
</tr>
</tbody>
</table>
5.3 Data Types

5.3.1 XML Namespaces

The XML namespace for the Network Message Storage data types is:

urn:oma:xml:rest:netapi:nms:1

The 'xsd' namespace prefix is used in the present document to refer to the XML Schema data types defined in XML Schema [XMLSchema1, XMLSchema2]. The 'common' namespace prefix is used in the present document to refer to the data types defined in [REST_NetAPI_Common]. The use of namespace prefixes such as 'xsd' is not semantically significant.

The XML schema for the data structures defined in the section below is given in [REST_SUP_NMS].

5.3.2 Structures

The subsections of this section define the data structures used in the NMS API.

Some of the structures can be instantiated as so-called root elements.

For structures that contain elements which describe a user identifier, the statements in section 6 regarding 'tel', 'sip' and 'acr' URI schemes apply.

For optional elements that contain lists (such as Folder.objects): if the list is absent it indicates only that the list is being omitted in this request or response, not that the list is necessarily empty.

5.3.2.1 Type: Object

Individual object

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentFolder</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>Resource URL of the parent folder that contains the object. In object creation requests this element specifies the folder that will contain the new object. If neither parentFolder nor parentFolderPath are included by the client in an object creation request, the server chooses a parent folder according to service provider policy (e.g., the root folder, or a folder based on this object's other available metadata). The server MUST include this element in responses.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>parentFolderPath</td>
<td>xsd:string</td>
<td>Choice</td>
<td>The location in the hierarchical storage of the folder that contains this object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In object creation requests this element specifies the folder that will contain the new object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server will internally resolve the parentFolderPath to an equivalent of the parentFolder URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of the requested parent folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If neither parentFolder nor parentFolderPath are included by the client in an object creation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>request, the server chooses a parent folder according to service provider policy (e.g., the root</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>folder, or a folder based on this object's other available metadata).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server SHALL implicitly create any folder(s) (referred to by the parentFolderPath element)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>which do not already exist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server MUST NOT include this element in responses.</td>
</tr>
<tr>
<td>attributes</td>
<td>AttributeList</td>
<td>No</td>
<td>List of attributes associated with the object.</td>
</tr>
<tr>
<td>flags</td>
<td>FlagList</td>
<td>No</td>
<td>List of flags associated with the object.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>Yes</td>
<td>Self referring URL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST requests representing notifications by the server to the client, when a complete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>representation of the resource is embedded in the notification. The resourceURL MUST also be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>included in responses to any HTTP method that returns an entity body, and in PUT requests.</td>
</tr>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The location of the object in the hierarchical storage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This element SHALL NOT be included in POST requests by the client but MAY be included in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>responses by the server to the client to any HTTP method that returns an Object entity body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See section 5.1.1 for further information on how an object’s path is constructed.</td>
</tr>
<tr>
<td>payloadURL</td>
<td>xsd:anyURI</td>
<td>Yes</td>
<td>Information about the location of the entire payload.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server MUST include this element if the object is non-empty unless the server determines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>that the payload can be represented using the inline method (see section 5.1.9).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Otherwise, the server MUST omit this element.</td>
</tr>
</tbody>
</table>
| payloadPart | PayloadPartInfo | Yes | Information about individual payload parts. This element SHALL NOT be included in POST requests by the client but MAY be included in responses by the server to the client. 

Number and content of payload parts:

- If the object is empty, this element MUST be omitted.
- If the server determines that the payload can be represented using the inline method (see section 5.1.9) then this element MUST be omitted.
- If the object’s payload has a multipart MIME type [RFC2046], the first-level parts of the payload MUST be represented as individual payload parts.
- If the object’s payload is of another type which can be divided into a sequence of parts, those parts SHOULD be represented as individual payload parts.
- Otherwise, this element MUST be omitted.

Only the first-level parts of the payload are represented as payload parts; for example, a nested “multipart/mixed” part is represented as a single payload part, not a sequence of subparts.

The number of payload parts MAY be limited by the server. |
| lastModSeq | xsd:unsignedLong | Yes | Last mod-sequence value associated with the object. A server supporting Strict Synchronization MUST provide this element in responses to the client. The client MUST NOT provide this element in requests to the server. |
| correlationId | xsd:string | Yes | Unique correlation ID associated with the object, if any. See section 5.1.12.2. This element MAY be provided by the client in requests to the server. If it was supplied by the client when the object was created, it MUST be returned unchanged by the server in responses and notification requests to the client. If this element was not supplied by the client when the object was created, the server MAY generate it according to service provider policy. |
correlationTag | xsd:string | Yes | Correlation tag associated with the object, if any. See section 5.1.12.3.  
This element MAY be provided by the client in requests to the server. If it was supplied by the client when the object was created, it MUST be returned unchanged by the server in responses and notification requests to the client.  
If this element was not supplied by the client when the object was created, the server MAY generate it according to service provider policy.

A root element named object of type Object is allowed in request and/or response bodies.

XSD modelling uses a “choice” to select either parentFolder or parentFolderPath, but not both of them.

5.3.2.2 Type: ObjectList

List of objects

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Object[0..unbounded]</td>
<td>Yes</td>
<td>List of objects. Number of objects MAY be limited by the server.</td>
</tr>
</tbody>
</table>
| cursor           | xsd:string            | Yes      | If the list of objects is complete, this element is omitted.  
If there may be more available objects not included in the response list, this element is included. The cursor value encapsulates information on these objects. See section 5.1.10 for how to use the cursor value in a subsequent request.  
This element SHALL NOT be included in requests by the client but MAY be included in responses by the server. |
| creationCursor   | xsd:string            | Yes      | An opaque string that enables the client to request retrieval of objects that were created in the store but not yet known to the client.  
See section 5.1.5.2 for how to use the creationCursor value in a subsequent CreatedObjects search request.  
This element MUST be returned in response to a CreatedObjects search. |

A root element named objectList of type ObjectList is allowed in request and/or response bodies.

5.3.2.3 Type: Flag

Individual flag

Simple type derived from xsd: string. Flag name (case insensitive). See Appendix H.

A root element named flag of type Flag is allowed in request and/or response bodies.
5.3.2.4 **Type: FlagList**

List of flags

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>Flag[0..unbounded]</td>
<td>Yes</td>
<td>Set of flags. Flag names are case-insensitive. Duplicates are ignored and order is not preserved. Appendix H defines the strings for flag names. The number of flags MAY be limited by the server.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>Yes</td>
<td>Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.</td>
</tr>
</tbody>
</table>

A root element named flagList of type FlagList is allowed in request and/or response bodies.
5.3.2.5 Type: Attribute

Individual attribute

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>No</td>
<td>Attribute name. Attribute names are case-insensitive: for example, the attribute names “Message-ID”, “Message-Id”, and “message-id” all refer to the same attribute. If a client assigns Folder attribute with similar semantics to one of the attributes specified in Appendix J, then it SHOULD use the name specified there. Attributes with different semantics can be assigned and named at the discretion of the client. Attribute names for use in RCS profiles are included in Informative Appendix I (for objects) and Appendix K (for folders).</td>
</tr>
<tr>
<td>value</td>
<td>xsd:string[0..unbounded]</td>
<td>Yes</td>
<td>Unless otherwise stated, attribute values are case-sensitive. The server SHOULD preserve the order of the values. Attribute values MUST be unencoded Unicode strings; for example, any transfer encoding such as [RFC2047] must be removed. For example, the [RFC5322]-format header &quot;To: =?ISO-8859-1?q?Keld_J=F8rn_Simonsen?= <a href="mailto:keld@dkuug.dk">keld@dkuug.dk</a>&quot; must be presented to NMS as: &quot;&lt;attribute&gt;&lt;name&gt;To&lt;/name&gt;&lt;value&gt;Keld Jørn Simonsen &amp;60;<a href="mailto:keld@dkuug.dk">keld@dkuug.dk</a>&amp;62;&lt;/value&gt;&lt;/attribute&gt;&quot;. The attributes search method (see sections 5.3.2.19 and 5.3.3.1) can only be used for attributes which have values. That is, searching for the presence of an attribute (i.e. which does not have a value) is not possible in the current (v1.0) NMS API release. Hence, attribute value needs to be present where this search method is expected.</td>
</tr>
</tbody>
</table>

5.3.2.6 Type: AttributeList

List of attributes

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>Attribute[0..unbounded]</td>
<td>Yes</td>
<td>List of attributes. Order is not preserved. The number of attributes MAY be limited by the server. The attribute names within the list MUST be unique.</td>
</tr>
</tbody>
</table>
5.3.2.7  Type: PayloadPartInfo

Information about a payload part

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>href</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Link to the stored content:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For content available via the NMS resource tree:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;href&gt;<a href="http://host/nms/v1/tel%3A%2B19585550100/objects/123/">http://host/nms/v1/tel%3A%2B19585550100/objects/123/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>payloadParts/part123&lt;/href&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For externally referenced content:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;href&gt;<a href="http://cdn/d1/123/p456&amp;lt;/href">http://cdn/d1/123/p456&amp;lt;/href</a>&gt;</td>
</tr>
<tr>
<td>contentType</td>
<td>xsd:string</td>
<td>No</td>
<td>Indicates the MIME content type of the stored content.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example: “image/gif”, “video/3gpp”</td>
</tr>
<tr>
<td>size</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Indicates the size of the stored content in bytes; The value MAY be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>approximate (e.g. it could be the size in its transfer encoding and not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the resulting size after any decoding).</td>
</tr>
<tr>
<td>contentId</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The Content-ID of this part as defined in [RFC2045],</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>without the angle brackets. Used to identify this payload part, e.g. in cid:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>URLs [RFC2392].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, “foo4%<a href="mailto:25foo1@bar.net">25foo1@bar.net</a>”.</td>
</tr>
<tr>
<td>contentLocation</td>
<td>xsd:anyURI</td>
<td>Yes</td>
<td>The Content-Location of this part, as defined in [RFC2557], unfolded and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with any transfer encoding such as [RFC2047] removed. Used to specify a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>URI for this payload part.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, “fiction1/fiction2” or “<a href="http://example.com/logo.png%E2%80%9D">http://example.com/logo.png”</a>.</td>
</tr>
<tr>
<td>contentDisposition</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The Content-Disposition of this part as defined in [RFC2183], unfolded and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with any transfer encoding such as [RFC2184] removed. Used to specify the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>disposition parameters (e.g., preferred filename) for this payload part.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, “inline” or ”attachment; filename=genome.jpeg;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>modification-date=&quot;Wed, 12 Feb 1997 16:29:51 -0500&quot;&quot;.</td>
</tr>
</tbody>
</table>
## 5.3.2.8 Type: Folder

Individual folder

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>[ResourceRelPath]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentFolder</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>Not applicable</td>
<td>Resource URL of the parent folder that contains this folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In folder creation requests this element specifies the folder that will contain the new folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In folder creation requests either the parentFolder or the parentFolderPath element MUST be included by the client. Otherwise, Service Exception SHALL be returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The server MUST include this element in responses, except for a folder that has no parent, i.e. represents a top-level folder in the hierarchy (such a folder would typically be assigned the &quot;Root&quot; attribute with value &quot;Yes&quot;).</td>
</tr>
<tr>
<td>parentFolderPath</td>
<td>xsd:string</td>
<td>Choice</td>
<td>Not applicable</td>
<td>The location in the hierarchical storage of the folder that contains this folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In folder creation requests this element specifies the folder that will contain the new folder. The server will internally resolve the parentFolderPath to an equivalent of the parentFolder URL of the requested parent folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In folder creation requests either the parentFolder or the parentFolderPath element MUST be included by the client. Otherwise, Service Exception SHALL be returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The server MUST NOT include this element in responses.</td>
</tr>
<tr>
<td>attributes</td>
<td>AttributeList</td>
<td>No</td>
<td>Not applicable</td>
<td>List of attributes associated with the folder. See Appendix J.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The attribute name “Name” MUST not be included in POST requests.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Required</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>----------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.</td>
</tr>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Not applicable</td>
<td>The location of the folder in the hierarchical storage. This element SHALL NOT be included in POST requests by the client but MAY be included in responses by the server to the client to any HTTP method that returns a Folder entity body. See section 5.1.2 for further information on how a folder’s path is constructed.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>Yes</td>
<td>folderName</td>
<td>The name of the new folder in folder creation operation. The name MUST NOT include the reserved character “/” (U+002F). The character “/” is used as hierarchy delimiter. The assigned folder name SHALL also be reflected in the read-only folder attribute called “Name” (see Appendix J). This enables search based on folder name. This element MAY be present in POST (folder creation operation) requests. If name is not provided by the client, the server SHALL assign a unique folder name in the context of the parent folder. If folder creation operation is successful, name SHALL be used as part of the path (see path element for further details). For the root folder name, see section 5.1.6. See sections 6.14.1.1 and 6.14.4 for further information on how a folder name can be changed.</td>
</tr>
</tbody>
</table>
| lastModSeq  | xsd:unsignedLong | Yes | Not applicable | Last mod-sequence value associated with the folder.  
The server MUST provide this element in responses to the client. The client MUST NOT provide this element in requests to the server. |
|------------|------------------|-----|----------------|----------------------------------------------------------------------------------------------------------------------------------|
| cursor     | xsd:string       | Yes | Not applicable | If the lists of subfolders and objects are both complete, this element is omitted.  
If there are more available subfolders and/or objects not included in these lists, this element is included. The cursor value encapsulates information on these items. See section 6.13.3.3 for how to use the cursor in a subsequent request. |
| subFolders | FolderReferenceList | Yes | Not applicable | List of sub-folders under this folder.  
Order is not significant.  
If this element is omitted it means that the list is unspecified, not that the list is necessarily empty.  
The client SHALL NOT include this element in POST or PUT requests.  
When it is requested by the client (see section 6.13.3) the server includes this element in response to a GET request for an individual folder.  
The server MAY include this element in response to a POST /search request. However, the server MUST include this element in response to a POST request if the search is for Root folder(s). See sections 5.4.7 and 6.15.5.1. |
| objects | ObjectReferenceList | Yes | Not applicable | List of objects under this folder. Order is not significant.  
| | | | | If this element is omitted it means that the list is unspecified, not that the list is necessarily empty.  
| | | | | The client SHALL NOT include this element in POST or PUT requests.  
| | | | | When it is requested by the client (see section 6.13.3) the server includes this element in response to a GET request for an individual folder.  
| | | | | The server MAY include this element in response to a POST /search request. However, the server MUST include this element in response to a POST request if the search is for Root folder(s). See sections 5.4.7 and 6.15.5.1. |

A root element named folder of type Folder is allowed in request and/or response bodies.

XSD modelling uses a “choice” to select either parentFolder or parentFolderPath, but not both of them.
5.3.2.9 Type: FolderList

List of folders

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder</td>
<td>Folder[0..unbounded]</td>
<td>Yes</td>
<td>List of folders. Number of folders MAY be limited by the server.</td>
</tr>
<tr>
<td>cursor</td>
<td>xsd:string</td>
<td>Yes</td>
<td>If the list of folders is complete, this element is omitted. If there may be more available folders not included in the list, this element is included. The cursor value encapsulates information on these folders. See section 5.1.10 for how to use the cursor in a subsequent request.</td>
</tr>
</tbody>
</table>

A root element named folderList of type FolderList is allowed in response bodies.

5.3.2.10 Type: FolderReferenceList

List of folder references

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderReference</td>
<td>Reference[0..unbounded]</td>
<td>Yes</td>
<td>A list of folder references. The number of references MAY be limited by the server.</td>
</tr>
</tbody>
</table>

A root element named folderReferenceList of type FolderReferenceList is allowed in response bodies.

5.3.2.11 Type: Reference

Reference to a stored object or folder

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>The resource URL.</td>
</tr>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The location of the object or folder in the hierarchical storage.</td>
</tr>
</tbody>
</table>

A root element named reference of type Reference is allowed in request and/or response bodies.

5.3.2.12 Type: ObjectReferenceList

List of object references

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objectReference</td>
<td>Reference[0..unbounded]</td>
<td>Yes</td>
<td>A list of object references. The number of references MAY be limited by the server.</td>
</tr>
</tbody>
</table>

A root element named objectReferenceList of type ObjectReferenceList is allowed in response bodies.
5.3.2.13 Type: TargetSourceRef
References to a target folder and source object(s)/folder(s)

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetRef</td>
<td>Reference</td>
<td>No</td>
<td>Reference to the target folder.</td>
</tr>
<tr>
<td>sourceRefs</td>
<td>ReferenceList</td>
<td>No</td>
<td>References to the source object(s)/folder(s).</td>
</tr>
</tbody>
</table>

A root element named targetSourceRef of type TargetSourceRef is allowed in request bodies.

5.3.2.14 Type: ReferenceList
References to stored folders and/or objects

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folders</td>
<td>FolderReferenceList</td>
<td>No</td>
<td>The referenced folders.</td>
</tr>
<tr>
<td>objects</td>
<td>ObjectReferenceList</td>
<td>No</td>
<td>The referenced objects.</td>
</tr>
</tbody>
</table>

5.3.2.15 Type: BulkUpdate
Bulk update to a set of objects

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>ObjectReferenceList</td>
<td>Choice</td>
<td>Reference to the objects which are to be updated.</td>
</tr>
<tr>
<td>selectionCriteria</td>
<td>SelectionCriteria</td>
<td>Choice</td>
<td>Selection criteria for a set of objects which are to be updated.</td>
</tr>
<tr>
<td>operation</td>
<td>OperationEnum</td>
<td>No</td>
<td>The kind of update operation to perform on the identified objects (e.g. AddFlag).</td>
</tr>
<tr>
<td>flags</td>
<td>FlagList</td>
<td>Yes</td>
<td>This element MUST be present if “operation” is either “AddFlag” or “RemoveFlag”. Flags to be added to or removed from the existing flags of objects identified by “objects” or “selectionCriteria”.</td>
</tr>
</tbody>
</table>

A root element named bulkUpdate of type BulkUpdate is allowed in request bodies.

XSD modelling uses a “choice” to select either objects or selectionCriteria, but not both of them.

5.3.2.16 Type: BulkDelete
Bulk delete a set of objects

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>ObjectReferenceList</td>
<td>Choice</td>
<td>Reference to the objects which are to be deleted.</td>
</tr>
<tr>
<td>selectionCriteria</td>
<td>SelectionCriteria</td>
<td>Choice</td>
<td>Selection criteria for a set of objects which are to be deleted.</td>
</tr>
</tbody>
</table>

A root element named bulkDelete of type BulkDelete is allowed in request bodies.

XSD modelling uses a “choice” to select either objects or selectionCriteria, but not both of them.
### 5.3.2.17 Type: SelectionCriteria

Selection criteria for a set of objects or folders

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromCursor</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The beginning position of the retrieve response. Omitting this value denotes the first position. The fromCursor is a cursor value provided by the server in a previous response to a request with exactly the same SelectionCriteria except for the fromCursor element; see section 5.1.10.</td>
</tr>
<tr>
<td>maxEntries</td>
<td>xsd:unsignedInt</td>
<td>No</td>
<td>Specifies maximum number of entries to be returned in the response. The server MAY return fewer entries than this.</td>
</tr>
<tr>
<td>searchCriteria</td>
<td>SearchCriteria</td>
<td>Yes</td>
<td>The search criteria for the retrieval of elements. Default is no search criterion, i.e. retrieval of all available elements.</td>
</tr>
<tr>
<td>searchScope</td>
<td>Reference</td>
<td>Yes</td>
<td>Reference to folder at which point the search would start. If searchScope is provided, the scope of the search is limited to the subtree starting at this folder (if nonRecursiveScope is a false value or omitted) or to this folder alone (if nonRecursiveScope is a true value). This element MUST refer to a folder within the box identified by the {boxId}. If searchScope is not provided, the search is applied to the box identified by the {boxId}</td>
</tr>
<tr>
<td>nonRecursiveScope</td>
<td>xsd:boolean</td>
<td>Yes</td>
<td>If set to a true value limits the scope of the search (as indicated by searchScope element) to a non-recursive search (no search within sub-folders). If the element is not present or set to a false value, then the search is recursive.</td>
</tr>
<tr>
<td>sortCriteria</td>
<td>SortCriteria</td>
<td>Yes</td>
<td>The sort criteria for the retrieval of elements. Default is random or server preferred sort.</td>
</tr>
</tbody>
</table>

A root element named selectionCriteria of type SelectionCriteria is allowed in request bodies.
### 5.3.2.18 Type: SearchCriteria

Search criteria

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>criterion</td>
<td>SearchCriterion[1..unbounded]</td>
<td>No</td>
<td>The search criteria. In the case of multiple search criteria, the result will include elements that comply with all criteria supplied. The number of criteria MAY be limited by the server.</td>
</tr>
<tr>
<td>operator</td>
<td>LogicalOperatorEnum</td>
<td>Yes</td>
<td>In case there is more than one SearchCriterion, the defined logical operation is applied between them. If the operator is absent (not specified), then the AND operation is applied.</td>
</tr>
</tbody>
</table>

### 5.3.2.19 Type: SearchCriterion

Search criterion

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>SearchTypeEnum</td>
<td>No</td>
<td>The search type (e.g. Attribute search or &quot;WholeWord&quot; search, etc). See section 5.3.3.1.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The name MUST be present for search types that require a name as specified in the SearchTypeEnum description (e.g. for the Attribute search type this element contains the attribute name). See sections 6.7 and 6.15.</td>
</tr>
<tr>
<td>value</td>
<td>xsd:string</td>
<td>No</td>
<td>The value to be matched against by the search operation. Format of the value string MUST follow the format as defined by the SearchTypeEnum. The search is for exact (and case sensitive) match, unless otherwise specified in the SearchTypeEnum description. See section 6.7 and 6.15.</td>
</tr>
</tbody>
</table>
5.3.2.20 Type: SortCriteria

Search criteria

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>criterion</td>
<td>SortCriterion[1..unbounded]</td>
<td>No</td>
<td>The sort criteria. In the case of multiple sort criteria, criteria are applied in order. The first criterion is the most significant. The server MAY ignore sort criteria beyond a certain number of entries in this array, as determined by server policy.</td>
</tr>
</tbody>
</table>

5.3.2.21 Type: SortCriterion

Sort criterion

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>SortTypeEnum</td>
<td>No</td>
<td>The sort type (e.g. sorting by Date).</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The name MUST be present for sort types that require a name (e.g. for the Attribute sort type this element contains the attribute name).</td>
</tr>
<tr>
<td>order</td>
<td>SortOrderEnum</td>
<td>Yes</td>
<td>Specifies order in which elements should be retrieved. Default: Descending.</td>
</tr>
</tbody>
</table>

5.3.2.22 Type: NmsSubscription

Individual subscription to notifications about storage changes

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callbackReference</td>
<td>common:CallbackReference</td>
<td>No</td>
<td>Client's notification endpoint and optional callbackData</td>
</tr>
<tr>
<td>duration</td>
<td>xsd:unsignedInt</td>
<td>Yes</td>
<td>Period of time (in seconds) notifications are provided for. If set to &quot;0&quot; (zero), or omitted, a duration time will be chosen according to service provider policy. This element MAY be given by the client during resource creation in order to signal the desired lifetime of the subscription. The server MUST return in this element the period of time for which the subscription will still be valid.</td>
</tr>
<tr>
<td>filter</td>
<td>SearchCriteria</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A filter which may be used by the client to indicate what kind of network storage changes it is interested to receive notifications about (e.g. only SMS messages or SMS messages from a particular contact/userId). See section 5.3.3.1. For each NmsEvent which relates to an object or folder, if the corresponding object or folder matches the filter then the NmsEvent is sent to the client; otherwise it is omitted. Any NmsEvent which does not relate to an object or folder (e.g., ResetBox) is sent to the client in all cases. By default (i.e. when this parameter is absent), the storage server reports all notifications.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>clientCorrelator</th>
<th>xsd:string</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A correlator that the client can use to tag this particular resource representation during a request to create a resource on the server. This element MAY be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate subscription creation in such situations. In case the element is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>resourceURL</th>
<th>xsd:anyURI</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>index</th>
<th>xsd:unsignedLong</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The index of the next notification to be issued. This value is meaningful only within this subscription. The index SHALL NOT be included in requests by the client, but MAY be included in responses by the server. See section 5.1.4.5. A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>----------</td>
</tr>
<tr>
<td>restartToken</td>
<td>xsd:string</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxEvents</td>
<td>xsd:unsignedInt</td>
<td>Yes</td>
</tr>
<tr>
<td>objectAttributeNames</td>
<td>xsd:string</td>
<td>[0..unbounded]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A root element named nmsSubscription of type NmsSubscription is allowed in request and/or response bodies.

Note that the clientCorrelator is used for purposes of error recovery as specified in [REST_NetAPI_Common], and internal client purposes. The server is NOT REQUIRED to use the clientCorrelator value in any form in the creation of the URL of the resource. The specification [REST_NetAPI_Common] provides a recommendation regarding the client’s generation of the value of this field.
### 5.3.2.23 Type: NmsSubscriptionList

List of subscriptions to notifications about storage changes

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscription</td>
<td>NmsSubscription[0..unbounded]</td>
<td>Yes</td>
<td>List of notification subscriptions.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Self referring URL.</td>
</tr>
</tbody>
</table>

A root element named nmsSubscriptionList of type NmsSubscriptionList is allowed in response bodies.

### 5.3.2.24 Type: NmsSubscriptionUpdate

Change to individual subscription to notifications about storage changes

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>xsd:unsignedInt</td>
<td>Yes</td>
<td>Period of time (in seconds) notifications are provided for. If set to &quot;0&quot; (zero), a duration time will be chosen according to service provider policy. This element MAY be given by the client in order to signal the desired lifetime of the subscription.</td>
</tr>
<tr>
<td>restartToken</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Subscription restart token indicating the point at which this subscription is to start. See section 5.1.4.3. This value applies to the box as a whole, and can be used independently of any particular subscription. If this element is present, all matching changes subsequent to the point indicated by the restart token will be notified by the server in addition to any subsequent notifications. If this element is absent, any changes from the time this subscription is created will be notified by the server.</td>
</tr>
</tbody>
</table>

A root element named nmsSubscriptionUpdate of type NmsSubscriptionUpdate is allowed in request bodies.

A client can update its subscription with a new restartToken, in order to restart the notification stream from where it left off. A previously-created {subscriptionId} MAY be updated only if it has not timed out (i.e. subscription’s “duration” hasn’t expired).

Updating a subscription does not affect the index value of subsequent notifications. The client can determine this index value by examining the index element of the updated NmsSubscription object returned by the POST.
5.3.2.25 Type: NmsEvent

Notification about storage changes. These are combined into an NmsEventList before being sent to clients with an appropriate subscription.

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deletedObject</td>
<td>DeletedObject</td>
<td>Choice</td>
<td>Reference to the user-deleted object.</td>
</tr>
<tr>
<td>deletedFolder</td>
<td>DeletedFolder</td>
<td>Choice</td>
<td>Reference to the user-deleted folder.</td>
</tr>
<tr>
<td>expiredObject</td>
<td>DeletedObject</td>
<td>Choice</td>
<td>Reference to the expired object.</td>
</tr>
<tr>
<td>expiredFolder</td>
<td>DeletedFolder</td>
<td>Choice</td>
<td>Reference to the expired folder.</td>
</tr>
<tr>
<td>changedObject</td>
<td>ChangedObject</td>
<td>Choice</td>
<td>Reference to the new or changed object.</td>
</tr>
<tr>
<td>changedFolder</td>
<td>ChangedFolder</td>
<td>Choice</td>
<td>Reference to the new or changed folder.</td>
</tr>
<tr>
<td>resetBox</td>
<td>ResetBox</td>
<td>Choice</td>
<td>The box has been reset.</td>
</tr>
</tbody>
</table>

XSD modelling uses a “choice” to select either deletedObject, deletedFolder, expiredObject, expiredFolder, changedObject, changedFolder or resetBox.

The server reports the creation of an object with a changedObject notification. Similarly the server reports the creation of a folder with a changedFolder notification.
5.3.2.26  **Type: NmsEventList**

List of notifications about storage changes. This is the data type of notifications sent to clients with an appropriate subscription.

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmsEvent</td>
<td>NmsEvent [0..unbounded]</td>
<td>Yes</td>
<td>May contain an array of storage change notifications.</td>
</tr>
<tr>
<td>callbackData</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The ‘callbackData’ element if it was passed by the application in the ‘callbackReference’ element when creating a subscription to notifications about NMS events. See [REST_NetAPI_Common].</td>
</tr>
<tr>
<td>index</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Index of this notification list in the subscription. Starts at 1 for the first notification list and increments by 1 for each subsequent notification list. See section 5.1.4.5. This value is meaningful only within a single subscription. A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
<tr>
<td>restartToken</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Subscription restart token representing the point after the change(s) being notified. See section 5.1.4.3. This value applies to the box as a whole, and can be used independently of any particular subscription. A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
<tr>
<td>link</td>
<td>common:Link[0..unbounded]</td>
<td>Yes</td>
<td>Link to other resources that are in relationship with this notification. The server SHOULD include a link to the related subscription. No other links are required or suggested by this specification.</td>
</tr>
</tbody>
</table>

A root element named nmsEventList of type NmsEventList is allowed in notification request bodies.
5.3.2.27 Type: DeletedObject

An object that has been deleted

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>The resource URL of the deleted object.</td>
</tr>
<tr>
<td>attributes</td>
<td>AttributeList</td>
<td>Yes</td>
<td>List of attributes which were associated with the object before it was deleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server SHOULD include all attributes which satisfy all of the following criteria: (a) the attribute was associated with the object before the object was deleted, and (b) server policy states that the attribute is retained for this deleted object, and (c) the attribute appears in the subscription's objectAttributeNames (see section 5.3.2.22). The server MAY include attributes which have not been specifically asked for by the client.</td>
</tr>
<tr>
<td>lastModSeq</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Last mod-sequence value associated with the deleted object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
<tr>
<td>correlationId</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Unique correlation ID associated with the object, if any. See section 5.1.12.2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server SHOULD provide the correlationId in deleted object notifications if it was known by the server when the object was created. However the server MAY omit the correlationId from expired object notifications.</td>
</tr>
<tr>
<td>correlationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Correlation tag associated with the object. See section 5.1.12.3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server SHOULD provide the correlationId in deleted object notifications if it was known by the server when the object was created. However the server MAY omit the correlationId from expired object notifications.</td>
</tr>
</tbody>
</table>
### 5.3.2.28 Type: DeletedFolder

A folder that has been deleted

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>The resource URL of the deleted folder.</td>
</tr>
<tr>
<td>lastModSeq</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Last mod-sequence value associated with the deleted folder. A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
</tbody>
</table>
### 5.3.2.29 Type: ChangedObject
An object that has been changed

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentFolder</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Resource URL of the parent folder that contains the object.</td>
</tr>
<tr>
<td>flags</td>
<td>FlagList</td>
<td>No</td>
<td>List of flags associated with the object.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>The resource URL of the changed object.</td>
</tr>
<tr>
<td>attributes</td>
<td>AttributeList</td>
<td>Yes</td>
<td>List of attributes associated with the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server SHOULD include all attributes which satisfy both of the following criteria:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(a) the attribute is associated with the object, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(b) the attribute appears in the subscription’s objectAttributeNames (see section 5.3.2.22).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The server MAY include attributes which have not been specifically asked for by the client.</td>
</tr>
<tr>
<td>lastModSeq</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Last mod-sequence value associated with the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
<tr>
<td>correlationId</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Unique correlation ID associated with the object, if any. See section 5.1.12.2.</td>
</tr>
<tr>
<td>correlationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Correlation tag associated with the object. See section 5.1.12.3.</td>
</tr>
</tbody>
</table>

### 5.3.2.30 Type: ChangedFolder
A folder that has been changed

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentFolder</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Resource URL of the parent folder that contains this folder.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>The resource URL of the changed folder.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>No</td>
<td>The name of the folder.</td>
</tr>
<tr>
<td>lastModSeq</td>
<td>xsd:unsignedLong</td>
<td>Yes</td>
<td>Last mod-sequence value associated with the folder.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A server supporting Strict Synchronization MUST provide this element in responses to the client.</td>
</tr>
</tbody>
</table>
5.3.2.31 **Type: ResetBox**

A box that has been reset

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(empty)</td>
<td></td>
<td></td>
<td>In the current version of this specification, this type is empty.</td>
</tr>
</tbody>
</table>
5.3.2.32 **Type: PathList**

Paths (i.e. location) of a list of objects or folders

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>[1..unbounded]</td>
<td>No</td>
</tr>
</tbody>
</table>

A root element named `pathList` of type `PathList` is allowed in request bodies

5.3.2.33 **Type: Response**

Status of a single operation in a given bulk operation (e.g. bulk creation, bulk pathToId)

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>xsd:unsignedShort</td>
<td>No</td>
<td>HTTP status code (e.g. 200, 400, 404, etc.)</td>
</tr>
<tr>
<td>reason</td>
<td>xsd:string</td>
<td>No</td>
<td>HTTP status string (e.g. OK, Bad Request, Not Found, etc.)</td>
</tr>
<tr>
<td>success</td>
<td>Reference</td>
<td>Choice</td>
<td>Object or folder reference which a given bulk operation (e.g. bulk creation, bulk pathToId) could successfully act upon. This element MUST be present when a bulk operation on a given object/folder was successful.</td>
</tr>
<tr>
<td>failure</td>
<td>common:RequestError</td>
<td>Choice</td>
<td>The error that occurred when attempting to operate on a single object/folder in a bulk operation. This element MUST be present when a bulk operation on a given object/folder resulted in a failure.</td>
</tr>
</tbody>
</table>

XSD modelling uses a “choice” to select either success or failure.
5.3.2.34 Type: BulkResponseList
Response to a bulk operation (e.g. bulk creation, bulk pathToId)

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>Response[1..unbounded]</td>
<td>No</td>
<td>List of responses. See section 6.8.5.3.</td>
</tr>
<tr>
<td>cursor</td>
<td>xsd:string</td>
<td>Yes</td>
<td>For search-based bulk operations (e.g. bulkUpdate, bulkDelete), if the list of response is complete, this element is omitted. If there may be more available responses not included in the response list, this element is included. The cursor value encapsulates information on these objects. See section 5.1.10 for how to use the cursor value in a subsequent request. This element SHALL NOT be included in requests by the client but MAY be included in responses by the server.</td>
</tr>
</tbody>
</table>

A root element named bulkResponseList of type BulkResponseList is allowed in response bodies.

5.3.2.35 Type: Empty
A dummy element

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(empty)</td>
<td></td>
<td>No</td>
<td>This type is empty and used as a dummy element, to overcome some limitations, for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. HTTP PUT requires a body, but the semantics of the request/response may not need that body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Some implementations may not be able to support non-existent (empty) HTTP body.</td>
</tr>
</tbody>
</table>

A root element named empty of type Empty is allowed in request and/or response bodies.
5.3.3 Enumerations

The subsections of this section define the enumerations used in the NMS API.

5.3.3.1 Enumeration: SearchTypeEnum

Type of search or event filtering to perform. For details and the corresponding names and values see sections 6.7 and 6.15.

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Searching for object stored by date.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Searching for objects or folders that contain a specified attribute that matches the given attribute value.</td>
</tr>
<tr>
<td>AllTextAttributes</td>
<td>Denotes case-insensitive substring search across all searchable text attributes (e.g. subject, transcript, name, TextContent etc.).</td>
</tr>
<tr>
<td>Flag</td>
<td>Searching for objects that do or do not have the specified flag.</td>
</tr>
<tr>
<td>WholeWord</td>
<td>Denotes whole word search across all text attributes and textual payload parts (also known as full text search, as opposed to substring search).</td>
</tr>
<tr>
<td>VanishedObjects</td>
<td>Searching for objects that were recently permanently deleted.</td>
</tr>
<tr>
<td>CreatedObjects</td>
<td>Searching for existing objects that were created in the store since a previous CreatedObjects search.</td>
</tr>
<tr>
<td>PresetSearch</td>
<td>This search type allows the client to activate a named pre-configured search on the server.</td>
</tr>
</tbody>
</table>
### 5.3.3.2 Enumeration: LogicalOperatorEnum

Logical operator to apply to multiple search criteria

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>Logical AND.</td>
</tr>
<tr>
<td>Or</td>
<td>Logical OR.</td>
</tr>
<tr>
<td>Not</td>
<td>Logical NOT.</td>
</tr>
</tbody>
</table>

If the operator has the value NOT and there is more than one SearchCriterion, then the resulting operation is equivalent to first applying AND to all criteria and then applying NOT:

\[ \text{NOT (criteria1 AND criteria2 AND ...criteriaN)} \]

### 5.3.3.3 Enumeration: SortTypeEnum

Type of sort to perform

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Sorting elements by date. This sort is by the date recorded internally by the NMS, and may not correspond to the value of any Date attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Sorting elements by a specified attribute.</td>
</tr>
</tbody>
</table>

### 5.3.3.4 Enumeration: SortOrderEnum

Order of sort to perform

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascending</td>
<td>Sort in ascending order.</td>
</tr>
<tr>
<td>Descending</td>
<td>Sort in descending order.</td>
</tr>
</tbody>
</table>

The sort locale and/or collation order is determined by the server according to service provider policy.

### 5.3.3.5 Enumeration: OperationEnum

Type of flag operation to perform

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddFlag</td>
<td>Add the specified flags.</td>
</tr>
<tr>
<td>RemoveFlag</td>
<td>Remove the specified flags.</td>
</tr>
</tbody>
</table>
5.3.4 Values of the Link “rel” attribute

The “rel” attribute of the Link element is a free string set by the server implementation, to indicate a relationship between the current resource and an external resource. The following are possible strings (this list is non-exhaustive, and can be extended):

- folder
- nmsSubscription

These values indicate the kind of resource that the link points to.

5.4 Sequence Diagrams

The following subsections describe the resources, methods and steps involved in typical scenarios.

The notification URL passed by the client during the subscription step can be a Client-side Notification URL, or a Server-side Notification URL. Refer to [REST_NetAPI_NotificationChannel] for sequence flows illustrating the creation of a Notification Channel and obtaining a Server-side Notification URL on the server-side, and its use by the client.

In a sequence diagram, a step which involves delivering a notification is labeled with “POST or NOTIFY”, where “POST” refers to delivery via the HTTP POST method, and “NOTIFY” refers to delivery using the Notification Channel [REST_NetAPI_NotificationChannel].

5.4.1 Subscription to NMS notifications

This figure below shows a scenario for an application subscribing to NMS notifications, querying for a list of active subscriptions, querying information pertaining to a subscription, updating a subscription (e.g. extending its duration) and unsubscribing to NMS notifications.

The resources:

- To subscribe to NMS notifications, create a new resource under http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions
- To retrieve list of active subscriptions, read the following resource http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions
- To retrieve information about an individual subscription, read the following resource http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions/{subscriptionId}
- To extend the life (duration) of a subscription and/or ask the subscription to restart the notification stream from a known past point (restartToken), update the following resource http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions/{subscriptionId}
- To cancel subscription to NMS notifications delete the resource under http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions/{subscriptionId}
Outline of the flows:

1. An application subscribes to NMS notifications using the POST method to submit the NmsSubscription data structure to the resource containing all subscriptions.
2. The application receives the result resource URL containing the subscriptionId.
3. An application requests the list of active subscriptions, using the GET method to the resource containing all subscriptions.
4. The server returns subscriptions list belonging to the application in the response.
5. An application requests information pertaining to an individual subscription using the GET method to the resource.
6. The server returns subscription’s data which includes index, restartToken, duration, etc. in the response.
7. An application extends a subscription’s life (duration) using the POST method to submit the NmsSubscriptionUpdate data structure to the resource.
8. The server extends the subscription’s duration accordingly and returns the update subscription’s data which includes index, restartToken, duration, etc.
9. The application stops receiving notifications (on a given subscription) using DELETE with the resource URL containing the subscriptionId.

Figure 2: NMS notifications subscription
10. Deletion confirmation.

5.4.2 Synchronization with NMS

5.4.2.1 Strict Synchronization

This figure below shows a scenario for an application wishing to synchronize its local message storage with the NMS. Typically, this scenario happens if an application with a local storage is off-line for a period of time (e.g. during a flight) and wishing to sync back with the network message storage.

Synchronization with the NMS is yet another form of subscribing to NMS notifications with the inclusion of the “restartToken” parameter the client application is aware of (from the last notification it received prior to going off-line).

The resources:

- To subscribe to NMS notifications while needing to synchronize (e.g. after being off-line for some time), include “restartToken” parameter in the request to create a new resource under:
  
  http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions

- To recover lost events by requesting the subscription to restart the notification stream from a last known restartToken, update the following resource:
  
  http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions/{subscriptionId}
Figure 3: Strict synchronization with NMS
Outline of the flows:

1. An application subscribes to NMS notifications using the POST method to submit the NmsSubscription data structure including the last known “restartToken” value to the resource containing all subscriptions.
2. The application receives the result resource URL containing the subscriptionId as well as index value which starts at 1 for a new notification subscription.
3. The server compares the client’s “restartToken” received in the subscription request with its own state information. Assuming that there are changes, the server formulates a list of events the client has missed (while being off-line) and sends that in an NmsEventList data structure to the client.
4. After some time new changes take place in the NMS which results in the application receiving a new list of notifications. Note that the notifications list reported in step #3 and #4 may be filtered by the server if instructed by the client application (see section 5.4.3 for further information on notification filtering mechanism use case).
5. Some new changes take place in NMS and the events are reported to the application.
6. More changes take place in NMS and they are reported to the application. However, an object ‘A’ is changed twice and lastModSeq reflects these two changes. The events are reported to the application out of order (for whatever reason). That is the latest change in object ‘A’ (marked by object’s A lastModSeq = 712) is reported to the application.
7. More changes take place in NMS however, the event (marked by index = 5) is lost and the application never receives it. The server is not aware of this fact.
8. More changes happen in NMS and the server formulates the next NmsEventList (marked by index = 6) and send it to the application. The application realizes that it has index = 4 already and instead of event with index = 5, it has received the event marked as index = 6. It realizes that it has lost a event in between.
9. The application updates the subscription using POST method to the resource and submits the NmsSubscriptionUpdate data structure with the last known restartToken = nn999.
10. The server returns the updated subscription’s data which includes the next index, restartToken, and the effective subscription duration, etc.
11. The server assembles all the events since the indicated marker by the client (i.e. restartToken = nn999) and sends the events to the application. The application recognizes that object ‘A’ in its local storage contains a higher lastModSeq value (i.e. lastModSeq = 712) than the one received in the event (i.e. lastModSeq = 709). Similarly, the changes against objects ‘M’ and ‘N’ are duplicates (as it had already been received by the client application). Hence the client ignores the events against objects ‘A’, ‘M’, ‘N’ and processes the other events in the list accordingly.

5.4.2.2 Simplified Synchronization

This figure below shows a scenario for an application wishing to synchronize its local message storage with the NMS, by leveraging the simplified selective synchronization. Typically, this scenario happens if an application with a local storage is off-line for a period of time (e.g. during a flight) and wishing to sync back with the network message storage.

Simplified synchronization with the NMS comprises of a set of search operations based on the last creationCursor value that the client application is aware of (from the last simplified synchronization performed prior to going off-line).

The client can initiate the simplified synchronization either asynchronously (i.e. in a ‘pull’ fashion, for example periodically, or triggered by some user operation) or after receiving some change notification.

The resources:

To search NMS needing to synchronize, use the following resource

http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/search
The client SHALL follow the following steps:

1. Search for new objects - SelectionCriteria.searchCriteria.criterion[1]:type = CreatedObjects , name omitted, value = creationCursor value known to the application (from a previous CreatedObjects search response) or an empty string if no such value is known to the application.
2. The server responds with all existing objects created since the point indicated by the creationCursor provided (in the value element of search criterion).
3. Search for purged objects - SelectionCriteria.searchCriteria.criterion[1]:type = VanishedObjects, name omitted, value = "" (empty string).
4. The server responds with all objects that have recently been permanently deleted.
5. Search for objects that do not carry the \"Seen\" flag.
   SelectionCriteria.searchCriteria.criterion[1]: type = Flag, name = \Seen, value = false.
   This step assumes that most objects in the store are read (seen), therefore searching for non \"Seen\" objects optimizes the retrieval of a relatively short list of objectIds.
6. The server responds with all objects that have the flag unset, and therefore all the others have the flag set.

Optionally (not shown in the diagram), use similar approach to synchronize other significant flags. The rest of the flags will not be synchronized into the local store.

### 5.4.3 Subscription to filtered NMS Notifications

This figure below shows a scenario for an application wishing to subscribe to certain NMS notifications. That is the application is not requiring to receive notifications for all the NMS changes.

Filtering NMS notifications is declared at the time of subscription creation where the client application passes in the request, the filter parameter which informs the server of the type of events it is interested to receive.

The resources:
To subscribe to filtered NMS notifications include “filter” parameter in the request to create a new resource under http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions

Outline of the flows:

1. An application subscribes to NMS notifications using the POST method to submit the NmsSubscription data structure including “filter” element (set as any object tagged urgent (i.e. has flag = 'Flagged') and not yet read (not 'Seen')) to the resource containing all subscriptions.

2. The application receives the result resource URL containing the subscriptionId.

Three events take place in NMS in the following order. However, only event #3 passes the filter.

Event #1: a new object is created with no flags set
Event #2: an object is system-deleted (expired) by NMS
Event #3: a new object is created with flags set to 'Flagged' (and 'Seen' flag is not present). This event passes the filter.

3. The application receives a list of notifications (containing one event) meeting the criteria (i.e. flags element containing the “Flagged” and “Seen” flag is not present) set by the application in the filter element of NmsSubscription data structure when subscribing to the notifications (in step #1).

5.4.4 Operations on folders

This figure below shows a scenario for retrieving properties of a folder and listing its containing objects and subfolders, creating a folder, deleting a folder, renaming a folder, moving a folder, copying a folder, searching and retrieving information about a set of selected folders matching a given criteria and inquiring about a folder’s resource URL using its location/path.

The resources:

- To retrieve properties of a folder including list of containing objects and subfolders, read the following resource http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/{folderId}
• Note: query string parameters control the amount of information returned in the response (e.g. exclude both subFolders element and objects element from the response or only include subFolders element in the response). Also control the maximum number of entries (subfolders/objects) to be returned in the response.

− To create a folder, create a new resource under
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders

− To delete a folder including its containing objects and subfolders, delete the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/[folderId]

− To rename a folder, update the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/[folderId]/[ResourceRelPath]

− To move folder(s) including its containing objects and subfolders to another folder, use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/operations/moveToFolder

− To copy folder(s) including its containing objects and subfolders to another folder, use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/operations/copyToFolder

− To search and retrieve information about a set of selected folders matching a given criteria, use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/operations/search

− To search the user’s network storage for the root folder(s), use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/operations/search

− To inquire about a folder’s resource URL using its location/path, use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/[storeName]/[boxId]/folders/operations/pathToId

• Note: query string parameter can be used to retrieve folder reference (i.e. Resource URL) for a single folder using its path.
Figure 6: Operations on folders

1. GET folder data
2. Response with Folder data
3. POST create a new folder
4. Response with Folder data
5. DELETE folder
6. Response folder deleted
7. PUT rename folder
8. Response folder renamed
9. POST move folder
10. Responses ReferenceList
11. POST copy folder
12. Responses ReferenceList
13. POST search for folders (SelectionCriteria = ConversationId)
14. Response FolderList (no cursor provided)
15. POST inquire folder's resource URL (pathList)
16. Response FolderReferenceList
Outline of the flows:

1. An application requests a folder’s data using the GET method on the resource.
2. The server returns folder’s data in the response.
3. An application requests to create a folder using the POST method on the resource.
4. The server creates the folder, assigns it a unique folderId and confirms folder creation in the response.
5. An application requests a folder to be deleted using the DELETE method on the resource.
6. The server confirms folder deletion in the response. All the contained objects and subfolders are also deleted.
7. An application requests a folder to be renamed using the PUT method on the Light-weight Resource representing the folder name parameter of a given folder.
8. The server returns the folder’s new name in the response.
9. An application requests a folder including all the containing objects and subfolders to be moved to a target folder using the POST method on the resource representing moveToFolder.
10. The server returns a ReferenceList containing a reference to all the objects and folders moved as a result. As part of this operation the paths of the moved objects and folders have been changed which is reflected in the returned ReferenceList.
11. An application requests a folder including all the containing objects and subfolders to be copied to a target folder using the POST method on the resource representing copyToFolder.
12. The server returns (synchronously or asynchronously) a ReferenceList containing a reference to all the new objects and folders which have been created as a result. For further information see section 6.17.5.
13. An application searches to retrieve information about a set of folders meeting a given SelectionCriteria (e.g. all the folders having a certain Conversation-Id value) using the POST method.
14. The server returns a FolderList containing all the folders matching the requested criteria. Here, it is assumed that the list is smaller than the Maximum response size. Hence, the server could return the complete FolderList in one single response. Otherwise, the server had to paginate the response list and flag that to the application by the inclusion of a “cursor” in the FolderList. See section 5.4.6 for an example on the usage of the “cursor”.
15. An application inquires to retrieve folder references (i.e. Resource URL) for a set of folders using folders’ path as the key. The request uses a POST method on the resource representing pathToId.
16. The server returns a FolderReferenceList in the response containing a reference to all the requested folders.

5.4.5 Operations on objects

This figure below shows a scenario for retrieving properties of an object, retrieving just the flags associated with an object, retrieving the entire payload of an object at once or retrieving an individual payload part of an object, updating individual flags associated with an object, creating an object, deleting an object, moving an object, copying an object, searching a folder and the subtree beneath it for a set of objects (e.g. messages) matching a given criteria and inquiring about an object’s resource URL using its location/path.

The resources:

- To retrieve properties (location, associated attributes and flags, parent’s folder and link(s) to its payload) of an object, read the following resource
  http://[serverRoot]/nms/[apiVersion]/[storeName]/[boxId]/objects/[objectId]
- To retrieve the flags associated with an object, read the following resource
  http://[serverRoot]/nms/[apiVersion]/[storeName]/[boxId]/objects/[objectId]/flags
- To retrieve the entire payload of an object at once or retrieve an individual payload part of an object, read a dynamically allocated resource which is provided by the server as a property of the given {objectId}. The resource URL of the payload is not known by the client application in advance and may be of any form (e.g. example/CDNstorage/100/blob456) and outside of the scope of this document.
- To update an individual flag associated with an object, update the following resource
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/objects/\{objectId\}/flags/\{flagName\}\]

- To create an object, create a new resource under
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/objects\]

- To delete an object, delete the following resource
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/objects/\{objectId\}\]

- To move object(s) to a folder, use the following resource
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/folders/operations/moveToFolde\]

- To copy object(s) to a folder, use the following resource
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/folders/operations/copyToFo\]

- To search a folder including its containing objects and subfolders for a set of objects matching a given criteria, use
  the following resource \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/objects/operations/search\]

- To inquire about an object’s resource URL using its location/path, use the following resource
  \[http://\{serverRoot\}/nms/\{apiVersion\}/\{storeName\}/\{boxId\}/objects/operations/pathToId\]

  - Note: query string parameter can be used to retrieve object reference (i.e. Resource URL) for a single object using its path.
Figure 7: Operations on objects
Outline of the flows:

1. An application requests an object’s data using the GET method on the resource.
2. The server returns object’s data in the response.
3. An application requests flags associated with an object using the GET method on the resource.
4. The server returns flags list in the response.
5. An application requests to retrieve the entire payload of an object at once or retrieve an individual payload part of an object using the GET method on the payload link provided as part of Object's data in step #1.
6. The external payload storage server (after authorizing access which is out of scope of this document) returns the payload in the response.
7. An application requests adding/removing an individual flag to/from an object using the PUT/DELETE method on the resource representing the flag name.
8. The server confirms with either the flag name for adding or HTTP 204 (No Content) response for deleting the flag name.
9. An application requests to create an object using the POST method on the resource.
10. The server creates the object, assigns it a unique objectld and confirms object creation in the response.
11. An application requests a object to be deleted using the DELETE method on the resource.
12. The server confirms object deletion in the response.
13. An application requests an object(s) to be moved to a target folder using the POST method on the resource representing moveToFolder.
14. The server returns a ReferenceList containing a reference to all the objects moved. As part of this operation the paths of the moved objects have been changed which is reflected in the returned ReferenceList.
15. An application requests object(s) to be copied to a target folder using the POST method on the resource representing copyToFolders.
16. The server returns a ReferenceList containing a reference to all the new objects which were created as a result.
17. An application searches to retrieve information about a set of objects meeting a given SelectionCriteria (e.g. all the objects having Message-Context = “pager-message”) using the POST method.
18. The server returns a ObjectList containing all the objects matching the requested criteria. Here, it is assumed that the list is smaller than the Maximum response size. Hence, the server could return the complete ObjectList in one single response. Otherwise, the server had to paginate the response list and flag that to the application by the inclusion of a “cursor” in the ObjectList. See section 5.4.6 for an example on the usage of the “cursor”.
19. An application inquires to retrieve object references (i.e. Resource URL) for a set of objects using objects’ path as the key. The request uses a POST method on the resource representing pathToId.
20. The server returns a ObjectReferenceList in the response containing a reference to all the requested objects.

5.4.6 Retrieving a large list of objects

This figure below shows a scenario for retrieving a large list of objects which would require multiple queries in order to retrieve the entire response. Responses containing a large list of objects which exceeds maximum allowable response size would normally result from a search over the entire message store. The retrieval of such a large list is managed by the usage of a “cursor” which is provided by the server in the first batch of the list (i.e. the first response). The client application would then need to use the provided “cursor” in the subsequent retrieval request in order to signal to the server that it is interested to receive the remaining portion of the list. This rendezvous mechanism using the “cursor” (i.e. cursor” element of ObjectList) continues until the server signals the end of the list by omitting the “cursor” from the final response.

The resources:
To search and retrieve all the messages meeting a given criteria include an appropriate “SelectionCriteria” parameter in the request using the following resource

http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/search

Note: The same cursor mechanism is employed for the retrieval of a large list of folders which exceeds the maximum allowable response size. Except that, the resource used is the following instead (this is not depicted in the figure below) http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/operations/search

Outline of the flows:

1. An application searches the user’s message store using the POST method with the SelectionCriteria data structure appropriately set (e.g. all messages between 2013-01-01 and 2013-11-10).

2. The server, finds all the objects (e.g. messages) meeting the selection criteria and since the number of selected objects is larger than the maximum entries allowed to be returned in the response, it paginates the list by responding with the 1st partial list and inclusion of a “cursor”.

3. The application finds the cursor in the response and sets fromCursor=cursor in the subsequent repeated request (i.e. POST SelectionCriteria data where the selection criteria is the same as in the 1st request (i.e. step #1)).

4. The server, again since the remaining number of objects is larger than the maximum entries allowed to be returned in the response, responding with the 2nd partial list and inclusion of a new “cursor”.

Figure 8: Retrieving a large list of objects
5. The application finds the new cursor in the response and sets fromCursor=cursor in the subsequent repeated request (i.e. POST SelectionCriteria data where the selection criteria is the same as in the 1st request (i.e. step #1)).

6. The server, since the remaining number of objects is less than the maximum entries allowed to be returned in the response, it responding with the last remaining list and omission of a “cursor”. The absence of the “cursor” in the last response signals the application that the list is now completed.

5.4.7 Discovering the user’s storage hierarchical structure

This figure below shows a scenario for discovering the “root” folder (assuming a single root in the depicted figure) and subsequently the traversal of the storage hierarchical structure. To discover the root folder, a search operation for a folder containing an attribute named “Root” with the attribute value of “Yes” is used (For further information see section 5.1.6). This way, the client application can discover and retrieve the properties of the root folder and a list of references to its containing objects and subfolders. In turn, the listed subfolders resource URLs can be used to traverse the entire tree, one step at a time leading to the discovery of the hierarchical structure of the user’s message store.

If the client intends to retrieve the full content of the storage, another more straightforward way is to perform an /objects/operation/search with no searchCriteria. This will retrieve the complete content of the storage, in batches of a size specified by the client. The client can then derive the hierarchical structure from the path information contained within each object.

The resources:

- To search the user’s network storage for the root folder, use the following resource
  
  http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/operations/search

- To retrieve the properties of the root folder including the list of containing objects and subfolders, read the following resource http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/{folderId}

  Note: To retrieve properties of a folder (child of the root folder) including the list of containing objects and subfolders, read the following resource recursively until the entire tree is exhausted http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/{folderId}
Figure 9: Discovering the user’s storage hierarchical structure

Outline of the flows:

1. An application searches to retrieve information about a set of folders meeting the given SelectionCriteria of attribute named “root” having the value of “Yes” using the POST method.

2. Assuming there is one root folder, the server returns a FolderList containing one Folder data structure (matching the requested criteria of attribute Root =Yes). The Folder data structure returned contains information about the root folder.
as well as a list of references to its containing objects and subfolders. This list of references can be used by the application to step through the entire user’s message store hierarchy and hence discover its tree structure and content.

3. The application requests to retrieve information about the object contained in the root folder using the GET method. Note: This step and step #4 are repeated for as many stand alone objects exist in the root folder.

4. The server returns Object’s data in the response.

5. The application requests to retrieve information about one of the subfolders contained in the root folder using the GET method. Note: This step and step #4 are repeated for as many subfolders exist in the root folder.

6. The server returns Folder data in the response.

7. The application requests to retrieve information about one of the subfolders contained in the subfolder of the root folder using the GET method. Note: This step and step #8 are repeated for as many subfolders exist in that level of the tree.

8. The server returns Folder data in the response.

Note: the above steps recursively repeats until the entire user’s message store is discovered.

5.4.8 Bulk object creation

This figure below shows a scenario for an application wishing to create a list of three objects in NMS using a single bulkCreation request.

Two out of three objects are created by the server and one object creation fails due to a prohibited parent folder being named in the request.

The HTTP response code is a 200 OK reflecting that at least one out of three object creations succeeded while the response body includes a list of success or failure status for each of the three objects in the request list.

The resources:

- To request a bulk object creation, use the following resource:
  
  \[
  \text{http://(serverRoot)/nms/(apiVersion)/(storeName)/(boxId)/objects/operations/bulkCreation} 
  \]

Outline of the flows:

1. An application wants to create three objects in a single request using POST method to submit the ObjectList data structure to the resource.

   The server successfully creates two out of three objects. Object2 creation fails as the parent folder named in the request (not shown in the diagram above) is a prohibited folder by service provider’s policy.

2. The application receives a response containing a 200 OK for the overall bulk creation operation as some of the objects in the list were created successfully while the response body includes of success or failure status for each of the three objects in the request list. Object2 status in the response body indicates a 403 Forbidden error code.
5.4.9 Bulk object deletion

The figure below shows a scenario for an application wishing to delete a list of three specific objects in NMS using a single bulkDeletion request. Additionally, the figure also depicts deletion of multiple objects meeting certain criteria (e.g. having attribute of From = tel:+19585550100)

In the scenario where the request contained a list of objects to be deleted, two out of three objects are deleted by the server and one object deletion fails due to a non-existent object being listed in the request. The HTTP response code is a 200 OK reflecting that at least one out of three object creations succeeded while the response body includes a list of success or failure status for each of the three objects in the request.

In the scenario where the request contained a SelectionCriteria to have all the matching objects deleted, 15 objects are deleted by the server. However, since MaxEntries = 5, the server uses a “cursor” to inform the client that it needs to repeat the request in order to delete the remaining objects matching the given SelectionCriteria and receive the corresponding responses. In this case only successfully deleted objects are reported in the response. The HTTP response code is a 200 OK reflecting that object deletion succeeded while the response body identifies the list of objects (URLs) deleted successfully.

The resources:

- To request a bulk object deletion, use the following resource
  
  `http://{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/bulkDelete`
Figure 11: Bulk object delete

Outline of the flows:

1. An application wants to delete three objects in a single request using POST method to submit the BulkDelete data structure to the resource.
   The server successfully deletes two out of three objects. Object3 deletion fails as the object was not found.

2. The application receives a response containing a 200 OK for the overall bulk update operation as some of the objects in the list were updated successfully while the response body includes success or failure status for each of the three objects in the request list. Object3 status in the response body indicates a “404 Not Found” error code.

3. An application wants to delete all incoming messages received from a person in a single request. It uses the POST method to submit the BulkDelete data structure to the resource. The BulkDelete data structure specifies a SelectionCriteria in order to delete all the objects meeting certain criteria (e.g. objects having attribute of From = tel:+19585550100)
4. The server, finds all the objects (e.g. messages) meeting the selection criteria and since the number of selected objects (to be deleted) is larger than the MaxEntries allowed to be returned in the response, it responds with the 1st partial BulkResponseList and inclusion of a “cursor”.

5. The application finds the cursor in the response and sets fromCursor=cursor in the subsequent repeated request (i.e. POST SelectionCriteria data where the selection criteria is the same as the request in step #3).

6. The server, again since the remaining number of objects (to be deleted) is larger than the MaxEntries allowed to be returned in the response, responds with the 2nd partial BulkResponseList and inclusion of a new “cursor”.

7. The application finds the new cursor in the response and sets fromCursor=cursor in the subsequent repeated request (i.e. POST SelectionCriteria data where the selection criteria is the same as the request in step #3).

8. The server, since the remaining number of objects (to be deleted) is less than the MaxEntries allowed to be returned in the response, responds with the last remaining list of deleted objects and omission of a “cursor”. The absence of the “cursor” in the last response signals the application that the deleted list of objects is now completed.

5.4.10 Bulk object update

The figure below shows a scenario for an application wishing to update a list of specific three objects in NMS using a single bulkUpdate request.

Note: in this version of the specification, only flags of objects can be updated. Similar to bulkDelete explained above, bulk update request may be invoked using a SelectionCriteria (instead of listing the objects in the request). BulkUpdate using SelectionCriteria is not shown in the following figure.

Figure below demonstrates, two out of three objects are updated successfully by the server and one object update fails due to a non-existent object being listed in the request.

The HTTP response code is a 200 OK reflecting that at least one out of three object updates succeeded while the response body includes a list of success or failure status for each of the three objects in the request list.

The resources:

- To request a bulk object update, use the following resource

http://[serverRoot]/nms/[apiVersion]/[storeName]/[boxId]/objects/operations/bulkUpdate

![Diagram of Bulk object update](image)

**Figure 12: Bulk object update**

Outline of the flows:

1. An application wants to update three objects in a single request using POST method to submit the BulkUpdate data structure to the resource

   The server successfully updates two out of three objects. Object3 update fails as the object was not found.
2. The application receives a response containing a 200 OK for the overall bulk update operation as some of the objects in the list were updated successfully while the response body includes success or failure status for each of the three objects in the request list. Object3 status in the response body indicates a “404 Not Found” error code.
6. Detailed specification of the resources

The following applies to all resources defined in this specification regardless of the representation format (i.e. XML, JSON):

- Reserved characters in URL variables (parts of a URL denoted below by a name in curly brackets) MUST be percent-encoded according to [RFC3986]. Note that this always applies, no matter whether the URL is used as a Request URL or inside the representation of a resource (such as in “resourceURL” and “link” elements).

- If a user identifier (e.g. address, participantAddress, etc.) is in the form of an MSISDN, it MUST be defined as a global number according to [RFC3966] (e.g. tel:+19585550100). The use of characters other than digits and the leading “+” sign SHOULD be avoided in order to ensure uniqueness of the resource URL. This applies regardless of whether the user identifier appears in a URL variable or in a parameter in the body of an HTTP message.

- If an equipment identifier is in the form of a SIP URI, it MUST be defined according to [RFC3261].

- If a user identifier (e.g. address, userId, etc) is in the form of an Anonymous Customer Reference (ACR), it MUST be defined according to Appendix H of [REST_NetAPI_ACR].
  - The ACR ‘auth’ is a supported reserved keyword, and MUST NOT be assigned as an ACR to any particular end user. See G.1.2 for details regarding the use of this reserved keyword.

- For requests and responses that have a body, the following applies: in the requests received, the server SHALL support JSON and XML encoding of the parameters in the body. The Server SHALL return either JSON or XML encoded parameters in the response body, according to the result of the content type negotiation as specified in [REST_NetAPI_Common]. In notifications to the Client, the server SHALL use either XML or JSON encoding, depending on which format the client has specified in the related subscription. The generation and handling of the JSON representations SHALL follow the rules for JSON encoding in HTTP Requests/Responses as specified in [REST_NetAPI_Common].
6.1 Resource: Resource containing all objects

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects

This resource is used for creating a new object (message, file, etc.).

6.1.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.1.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.1.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.1.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.1.5 POST

This operation is used for creating a new object.

The request contains both the Object data structure and the payload.

The NMS server MUST support a representation of objects as multipart/form-data entity bodies, where the first entry of the form are the root fields and the second entry of the form are the payload parts. Details about the structure of such objects are
defined in [REST_NetAPI_Common] and [REST_WP]. The type of the form entry carrying the root fields part of such an object MUST be Object in this API.

Other formats MAY be supported at the server’s discretion.

Note: An object returned by the server in response to a client request, or sent by the server to a client in a notification, can alternatively be represented as a list of link elements to the individual payload parts.

The response is usually 201 Created, with either the created Object or a Reference to it.

However if the server has not completed processing of the request it MAY return 202 Accepted [RFC7231] instead. In this case the server SHOULD also include a dummy body in the POST response, containing the Empty element.

### 6.1.5.1 Example 1: Object creation by parentFolder, response with a location of created resource

(Informative)

The following example shows a request for creation of an Object of MIME type multipart/mixed, to be stored under folder id “fld123”, with assigned flags “Seen” and “Flagged”.

#### 6.1.5.1.1 Request

```plaintext
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
</nms:object>

--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep-=--"
Content-Disposition: form-data; name="attachments"

-----=-sep=-
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

See attached photo

-----=-sep=-
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...
```
6.1.5.1.2 Response

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123</resourceURL>
  <path>/main/conversation5/obj123</path>
</nms:reference>

6.1.5.2 Example 2: Object creation by parentFolderPath, response with a location of the created resource while the non-existent parent folder is auto-created (Informative)

The following example shows a request for creation of an object under a non-existent parent folder which is auto-created by the server prior to placing the new object in it.

6.1.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolderPath>/main/myBackups/Football/SavedPictures</parentFolderPath>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
</nms:object>

--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep-=--"
Content-Disposition: form-data; name="attachments"

----=-sep-=--
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"
See attached photo

Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

6.1.5.2.2 Response

HTTP/1.1 201 Created
Date: Fri, 23 May 2014 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj4141
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj4141</resourceURL>
  <path>/main/myBackups/Football/SavedPictures</path>
</nms:reference>

6.1.5.3 Example 3: Object creation by parentFolderPath, response creation failure due to prohibited location (i.e. requested parent folder) (Informative)

The following example shows a request for creation of an object under a prohibited system folder called /Default which is allowed to be used by CPM participating function only. Other client’s attempt to create an object (or folder) under /Default folder is rejected as shown below.

6.1.5.3.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolderPath>/Default</parentFolderPath>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
</nms:object>
6.1.5.3.2 Response

HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 20:51:51 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1"><policyException><messageId>POL1031</messageId><text>Attempt to create objects or folders under %1 is prohibited</text><variables>/Default</variables></policyException></common:requestError>

6.1.5.4 Example 4: Object creation without parent folder, response with a location of created resource and full object (Informative)

The following example shows a request for creation of an Object of MIME type multipart/mixed, to be stored under a folder chosen by the server, with assigned flags “Seen” and “Flagged”. The server chooses to store it under “/main/inbox”. This also shows that object creation may return the created Object rather than just a Reference.

6.1.5.4.1 Request

POST /exampleAPI/nms/v1/myStore/te%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="

See attached photo

---==--
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

See attached photo

---==--
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

---==--

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document
Content-Disposition: form-data; name="root-fields"

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
<attributes/>
<flags>
  <flag>\Seen</flag>
  <flag>\Flagged</flag>
</flags>
</nms:object>

--=-=-=-=-outer123456==
Content-Type: multipart/mixed; boundary="--=-sep=---"
Content-Disposition: form-data; name="attachments"

----=-sep=---
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"
See attached photo

----=-sep=---
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"
GIF89a...binary image data...

----=-sep=---
--=-=-=-=-outer123456==--

6.1.5.4.2 Response

HTTP/1.1 201 Created
Date: Tue, 20 Apr 2014 09:43:02 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj741
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId185</parentFolder>
  <attributes>
    <attribute>
      <name>Date</name>
      <value>2013-11-12T08:30:10Z</value>
    </attribute>
    <attribute>
      <name>Content-Type</name>
      <value>multipart/mixed</value>
    </attribute>
  </attributes>
  <flags>
    <flag>\Seen</flag>
  </flags>
</nms:object>
6.1.5.5 Example 5: Creation of multipart object with presentation part, response with a location of created resource (Informative)

The following example shows a request for creation of an Object of MIME type multipart/related [RFC2387], to be stored under folder id “fld123”, with assigned flags “Seen” and “Flagged”. The start parameter on the multipart/related Content-Type indicates the Content-ID of the presentation part, which is the second of the three payload parts.

6.1.5.5.1 Request

```plaintext
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"
<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1"
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj741</resourceURL>
  <path>/main/inbox/obj741</path>
  <payloadURL>/nms/v1/myStore/tel%3A%2B19585550100/objects/obj741/payload</payloadURL>
  <payloadPart>
    <contentType>text/plain</contentType>
    <size>20</size>
    <href>/nms/v1/myStore/tel%3A%2B19585550100/objects/obj741/payloadParts/blob123</href>
  </payloadPart>
  <payloadPart>
    <contentType>image/gif</contentType>
    <size>16384</size>
    <href>/example/storage/102/blob498</href>
  </payloadPart>
  <lastModSeq>1416230968000</lastModSeq>
</nms:object>

```

--===============outer123456==
Content-Type: multipart/related; start="28186490"; boundary="--sep--"
Content-Disposition: form-data; name="attachments"

-----sep-----
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline;
filename=text.txt
Content-ID: <28184720>

See attached photo.

-----sep-----
Content-Type: application/smil; charset=US-ASCII
Content-Transfer-Encoding: 7bit
Content-ID: <28186490>
Content-Disposition: inline;
filename=slideshow.smil
<smil><head><layout><root-layout background-color="#F7F56D" width="480"
height="640"/></region id="First" height="320" width="480" left="0"
top="0" fit="meet"/></region id="Second" height="320" width="480"
left="0" top="320" fit="meet"/></layout><head><body><par
dur="5000ms"><text src="cid:28184720" region="Second"><param
name="fontSize" value="23"/><param name="textsize" value="medium"/></text><img src="cid:28186480"
region="First"/></par></body></smil>

-----sep-----
Content-Type: image/jpeg
Content-Disposition: inline; filename=IMAG0067.jpg
Content-Transfer-Encoding: binary
Content-ID: <28186480>
JFIF...binary image data...

-----sep----

==outer123456==

6.1.5.5.2 Response

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542</resourceURL>
<path>/main/conversation5/obj542</path>
</nms:reference>
6.1.5.6 Example 6: Creation of simple text object, response with a location of created resource (Informative)

The following example shows a request for creation of a simple text object, which can be represented by the inline method (see section 5.1.9).

6.1.5.6.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"
<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
</nms:object>
--===============outer123456==
Content-Type: text/plain
Content-Disposition: form-data; name="attachments"
The quick brown fox rushed to Montreal.
--===============outer123456==
```

6.1.5.6.2 Response

```
HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543</resourceURL>
  <path>/main/conversation5/obj543</path>
</nms:reference>
```

6.1.5.7 Example 7: Creation of simple object with no payload, response with a location of created resource (Informative)

The following example shows a request for creation of a simple object with no payload.
6.1.5.7.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456==";
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes>
    <attribute>
      <name>Subject</name>
      <value>A historical search engine</value>
    </attribute>
    <attribute>
      <name>URL</name>
      <value>http://altavista.digital.com</value>
    </attribute>
  </attributes>
  <flags>
    <flag>Flagged</flag>
  </flags>
</nms:object>

--===============outer123456==--

6.1.5.7.2 Response

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:52:58 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj547
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj547</resourceURL>
  <path>/main/conversation5/obj547</path>
</nms:reference>

6.1.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.2 Resource: A stored object

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}

This resource is used for managing a stored object such as retrieving information about the object or deleting the object.

6.2.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
<tr>
<td>objectId</td>
<td>Object identifier</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.2.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.2.3 GET

This operation is used for retrieval of an object’s properties such as its location, its list of attributes and flags.
6.2.3.1 Example 1: Retrieve information about an object (Informative)

6.2.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.2.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId567</parentFolder>
  <attributes>
    <attribute>
      <name>Message-Context</name>
      <value>multimedia-message</value>
    </attribute>
    <attribute>
      <name>Direction</name>
      <value>In</value>
    </attribute>
    <attribute>
      <name>From</name>
      <value>tel:+19585550100</value>
    </attribute>
    <attribute>
      <name>To</name>
      <value>tel:+19585550210</value>
      <value>tel:+19585550320</value>
    </attribute>
    <attribute>
      <name>Date</name>
      <value>2013-11-12T08:30:10Z</value>
    </attribute>
    <attribute>
      <name>Subject</name>
      <value>Weekend Trip to Seattle</value>
    </attribute>
    <attribute>
      <name>Content-Type</name>
      <value>multipart/mixed</value>
    </attribute>
  </attributes>
  <flags>
    <flag>Seen</flag>
    <flag>Answered</flag>
  </flags>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999</resourceURL>
  <path>/main/conversation5/oId999</path>
</nms:object>
6.2.3.2 Example 2: Retrieve information about multipart object  (Informative)

The following example shows a client retrieving information about the object deposited in section 6.1.5.5. Notice that the “start” parameter of the Content-Type is presented to the client in the Content-Type attribute, so the client is able to identify that the presentation part is the second payload part [RFC2387].

On receiving the response, a client might proceed as follows:

- Inspect the “start” parameter of the Content-Type attribute in the object GET response to determine the Content-ID of the presentation part: “28186490”.
- Inspect the object GET response again to determine which part has this Content-ID (the second payload part), and what the corresponding URL is: http://example.com/nms/v1/myStore/tel%3A%2B195855550100/objects/obj542/payloadParts/blob124.
- Retrieve the presentation part from the specified URL.
- Interpret the SMIL and determine it needs to display the image with Content-ID “28186480”.
- Inspect the object GET response again to determine which payload part has this Content-ID, and what the corresponding URL is. It is the third part, with URL http://example.com/example/storage/100/blob457.
- Retrieve the image and display it.

6.2.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/obj542 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.2.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fId123</parentFolder>
    <attributes>
        <attribute>
            <name>Content-Type</name>
6.2.3.3 Example 3: Retrieve information about object with inline content

(Informative)

The following example shows a client retrieving information about the object deposited in section 6.1.5.6. Notice that the text content has been extracted into the TextContent attribute, following section 5.1.9. There are no payload parts and no payloadURL.

6.2.3.3.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
6.2.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:object xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId123</parentFolder>
  <attributes>
    <attribute>
      <name>TextContent</name>
      <value>The quick brown fox rushed to Montreal.</value>
    </attribute>
  </attributes>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543</resourceURL>
  <path>/main/conversation5/obj543</path>
  <lastModSeq>48</lastModSeq>
</nms:object>

6.2.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.2.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.2.6 DELETE

This operation is used to delete an object including its payload.

The server responds to a DELETE request with an HTTP 204 No Content response.

6.2.6.1 Example: Delete an object, response with “204 No Content” (Informative)

6.2.6.1.1 Request

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old999 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.2.6.1.2 Response

HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 05:55:59 GMT
6.3 Resource: Flags associated with the stored object

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/flags

This resource is used to manage flags list associated with an object.

6.3.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td></td>
<td>Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use.</td>
</tr>
<tr>
<td></td>
<td>The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e.</td>
</tr>
<tr>
<td></td>
<td>a server hosting multiple independent stores). The value of this variable</td>
</tr>
<tr>
<td></td>
<td>is deployment dependent (e.g. in simple deployment scenarios it could be a</td>
</tr>
<tr>
<td></td>
<td>fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this</td>
</tr>
<tr>
<td></td>
<td>variable depends on the deployment scenario and the service provider’s</td>
</tr>
<tr>
<td></td>
<td>policy. For example:</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where each user is allocated a ‘box’ of its own,</td>
</tr>
<tr>
<td></td>
<td>the value of “boxId” can be equivalent to the unique identifier of the user</td>
</tr>
<tr>
<td></td>
<td>(e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where a ‘box’ is allocated to a group of multiple</td>
</tr>
<tr>
<td></td>
<td>users (or machines), the value of “boxId” can be a unique identifier of the</td>
</tr>
<tr>
<td></td>
<td>group</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenarios where a ‘box’ is allocated to a machine (non-human</td>
</tr>
<tr>
<td></td>
<td>user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
<tr>
<td>objectId</td>
<td>Object identifier</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.3.3 GET

Retrieve the flags (string labels) associated with the object.

6.3.3.1 Example 1: Retrieve flags associated with an object (Informative)

6.3.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old999/flags HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
6.3.3.1.2 Response

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2013 03:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:flagList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <flag>Seen</flag>
  <flag>Flagged</flag>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags</resourceURL>
</nms:flagList>

6.3.3.2 Example 2: Retrieve flags associated with an object, failure due to an invalid object (Informative)

6.3.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objNotThere/flags HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5c903
Accept: application/xml

6.3.3.2.2 Response

HTTP/1.1 404 Not Found
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
    <messageId>SVC0004</messageId>
    <text>No valid addresses provided in message part %1</text>
    <variables>/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objNotThere/flags</variables>
  </serviceException>
</common:requestError>

6.3.4 PUT

Create or update the flags (string labels) associated with the object.

6.3.4.1 Add a flag to flaglist of an object (Informative)

Add “Answered” flag to the flaglist of an object which already contains other flags (as shown in previous example containing: ‘Seen’ and ‘Flagged”).
6.3.4.1.1 Request

PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<nms:flagList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <flag>
    Seen
  </flag>
  <flag>
    Flagged
  </flag>
  <flag>
    Answered
  </flag>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags</resourceURL>
</nms:flagList>

6.3.4.1.2 Response

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2013 03:58:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:flagList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <flag>
    Seen
  </flag>
  <flag>
    Flagged
  </flag>
  <flag>
    Answered
  </flag>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags</resourceURL>
</nms:flagList>

6.3.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, PUT’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.3.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, PUT’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.4 Resource: Individual flag

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/flags/{flagName}

This resource is used to manage an individual flag associated with a given object.

6.4.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td></td>
<td>Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined</td>
</tr>
<tr>
<td></td>
<td>in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server</td>
</tr>
<tr>
<td></td>
<td>hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value</td>
</tr>
<tr>
<td></td>
<td>of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
<tr>
<td>objectId</td>
<td>Object identifier</td>
</tr>
<tr>
<td>flagName</td>
<td>Flag name (case sensitive). See Appendix H.</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.4.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.4.3 GET

Retrieve/check existence of an individual flag (string label).

If the flag is present, the server SHOULD respond with 204 No Content.

If the flag is absent, the server SHOULD respond with 404 Not Found. In that case the server SHOULD also include a dummy body in the GET response, containing the Empty element (this indicates that the 404 response is an expected condition, i.e. not an exception).

6.4.3.1 Example 1: Read an existing individual flag (Informative)

The following example shows a request checking to find out if a given object (e.g. a message) has already been flagged as read (‘Seen’) or not. This is done by checking the existence of the “‘Seen’” flag. In this example, the flag exists, which results in the successful response with no content.
6.4.3.1.1 Request
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CSeen HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

6.4.3.1.2 Response
HTTP/1.1 204 No Content
Date: Sat, 05 Oct 2013 03:55:00 GMT

6.4.3.2 Example 2: Read a non-existing individual flag using acr:auth (Informative)
The following example shows a request checking for the existence of the “Answered” flag associated with an object. In this example, the flag does not exist and the user (i.e. boxId) is identified by the access token present in the Authorization header (hence {boxId} in request URL is “acr:auth”).

6.4.3.2.1 Request
GET /exampleAPI/nms/v1/myStore/acr%3Aauth/objects/oId999/flags/%5CAnswered HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

6.4.3.2.2 Response
HTTP/1.1 404 Not Found
Date: Sat, 05 Oct 2013 03:55:00 GMT
Content-Type: application/xml
Content-Length: nnnn
<?xml version="1.0" encoding="UTF-8"?>

6.4.4 PUT
Add individual flag (string label).
The client SHOULD include a dummy body in the PUT request, containing the Empty element (this helps overcome some web infrastructure limitations, which require a body when using PUT).

If the flag did not exist and is now created, the server SHOULD respond with 201 Created, with a Location header containing the resource URL of the flag. In that case the server SHOULD also include a dummy body in the PUT response, containing the Empty element (this helps overcome some web infrastructure limitations which require a body when using PUT).

If the flag already existed, the server SHOULD respond with 204 No Content.
6.4.4.1 Add "\Answered\" flag to flaglist of an object  (Informative)

6.4.4.1.1 Request

PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CAnswered HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>

6.4.4.1.2 Response

HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CAnswered
Date: Sat, 05 Oct 2013 03:58:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>

6.4.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, PUT, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.4.6 DELETE

Remove individual flag (string label).

If the flag was present, the server SHOULD respond with 204 No Content.

If the flag was already absent, the server SHOULD respond with 404 Not Found. In that case the server SHOULD also include a dummy body in the DELETE response, containing the Empty element (this indicates that the 404 response is an expected condition, i.e. not an exception).

6.4.6.1 Delete "\Seen\" flag from flaglist of an object  (Informative)

6.4.6.1.1 Request

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CSeen HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.4.6.1.2 Response

HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 05:55:59 GMT
6.5 Resource: Stored content of an object payload

The resource used is a URL chosen by the server and reported in the payloadURL element of an Object data structure.

If the content is available via the NMS resource tree this is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payload

For externally referenced content, the resource can be any URL.

This resource is used for retrieving the entire payload of an object at once.

6.5.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
<tr>
<td>objectId</td>
<td>Object identifier</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.5.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.5.3 GET

This operation is used to read the object payload from the storage server.
6.5.3.1 Example: Read payload of the stored object via an external reference (Informative)

6.5.3.1.1 Request

GET /exampleAPI/storage/100/blob456 HTTP/1.1
Accept: image/gif, image/png, image/jpeg, text/html, application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

6.5.3.1.2 Response

HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 03:52:01 GMT
Content-Length: nnnn
Content-Type: multipart/mixed; boundary="--=-sep=---"

-----=-sep=---
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

Are you coming to the football today? See attached photo

-----=-sep=---
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=-sep=---

6.5.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.5.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.5.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.6 Resource: Payload part of the stored object

The resource used is a URL chosen by the server and reported in the href element of a PayloadPartInfo data structure.

If the content is available via the NMS resource tree this is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payloadParts/{payloadPartId}

For externally referenced content, the resource can be any URL.

This resource is used for retrieving an individual payload part of an object.

6.6.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
<tr>
<td>objectId</td>
<td>Object identifier</td>
</tr>
<tr>
<td>payloadPartId</td>
<td>Unique payload part identifier generated by the storage server.</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.6.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.6.3 GET

This operation is used to read one payload part from the storage server using standard HTTP [RFC7231]. It MUST return the binary content and MIME Content-Type header of the indicated payload part.
6.6.3.1 Example: Read an object payload part via the NMS resource tree  
(Informative)

6.6.3.1.1 Request
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123/payloadParts/part123 HTTP/1.1
Accept: image/gif, image/png, image/jpeg, text/html, application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

6.6.3.1.2 Response
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 03:51:59 GMT
Content-Length: nnnn
Content-Type: image/gif

...GIF89a...binary image data

6.6.4 PUT
Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.6.5 POST
Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.6.6 DELETE
Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.7 Resource: Information about a selected set of objects in the storage

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/search

This resource is used for retrieving information about a set of selected objects.

The following table applies to each SearchCriterion:

<table>
<thead>
<tr>
<th>SearchCriterion type</th>
<th>Description</th>
<th>SearchCriterion.name</th>
<th>SearchCriterion.value</th>
</tr>
</thead>
</table>
| Date                 | Searching for object stored by date. | n/a | Contains a query string of the following format:  
• minDate={minDate} - all object stored from a starting (internal date) {minDate} inclusive  
• minDate={minDate}&maxDate={maxDate} - all objects stored between minDate inclusive and maxDate exclusive.  
• maxDate={maxDate} - all objects stored up to maxDate exclusive.  
Where the string format of {minDate} and {maxDate} is xsd:dateTimeStamp as defined in [XMLSchema2]. |
| Attribute            | Searching for objects or folders that contain a specified attribute that matches the given attribute value. | The attribute's name. | The attribute's value for search.  
Examples:  
– to search for messaging objects from sender address bob@example.com use SearchCriterion.name = “From”, SearchCriterion.value = “bob@example.com”  
– to search for messaging objects to recipient address alice@example.com: SearchCriterion.name = “To”, SearchCriterion.value = “alice@example.com” (this search excludes copied recipients, i.e. if alice@example.com is in the “CC” or “BCC”)  
– to search for messaging objects from sender address bob@example.com to recipient address alice@example.com – combine the previous examples in a
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllTextAttributes</td>
<td>Denotes case-insensitive substring search across all searchable text attributes (e.g., subject, transcript, name, TextContent etc.).</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Searching for objects that do or do not have the specified flag.</td>
<td>The flag name.</td>
</tr>
<tr>
<td>WholeWord</td>
<td>Denotes whole word search across all text attributes and textual payload parts (also known as full text search, as opposed to substring search).</td>
<td>n/a</td>
</tr>
<tr>
<td>VanishedObjects</td>
<td>Searching for objects that were recently permanently deleted:</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>Note: VanishedObjects cannot be negated or combined with any other criterion. The search response SHALL be an ObjectReferenceList identifying the deleted objects.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When “VanishedObjects” is used as part of a subscription to filter events, it matches user-deleted or expired objects only (see section 5.3.2.22).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See section 5.1.7 for further information on VanishedObjects.</td>
<td></td>
</tr>
<tr>
<td>CreatedObjects</td>
<td>Searching for existing objects that were created in the store since a previous CreatedObjects search.</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>When CreatedObjects is used as part of a subscription to filter events, its value MUST be the empty string. The filter matches objects which have been newly created after this subscription was created (see section 5.3.2.22). Note that this means such notifications are not reliable i.e., a client cannot recover them if they are lost (see section 5.1.5.2).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.1.4.5) or were disconnected at the time (see section 5.1.4.3) – they are only issued to clients which have an active subscription at the point of object creation, and they will not be (re)sent to clients which update or restart their subscription using a restartToken before the point of creation.

| PresetSearch | This search type allows the client to activate a named pre-configured search on the server. The use of this search type implies pre-agreement between server and clients (beyond those defined in this specification), as defined by profiles or by proprietary server policy. It is permitted that such preset search may override or affect the sort processing applied to the search.

Specifying PresetSearch requires the following:

- **PresetSearch name** – unique name identifying the pre-configured search
- **Search semantics** – the purpose and description of the search, including which parts of the objects are included in the scope of the search (e.g. object payload, particular object attributes, etc.)
- **search arguments** and their format (as represented in a search string)
- **pre-configured sort** – sort performed on the matched elements. If pre-configured sort is specified, it SHOULD define its priority relative to the search arguments in the request (e.g. it MAY ignore some input arguments).

The search request includes the PresetSearch name and any other arguments required for the search, as specified in the pre-|

| The name of the PresetSearch. This specification defines no PresetSearch names. |

| Contains the search arguments string, according to the format specified in the pre-configured search. Empty string denotes no arguments. |
configuration.
For example, in a messaging environment the network message store server can be pre-configured with the following named PresetSearch:

“RemoteParty”:
- **Search semantics**: find all objects (messages) exchanged with a designated remote party, while the scope of the search is limited to "To" and "From" attributes.
- **Search arguments**: the remote party’s user identifier. If that party has multiple user identifiers they are provided in a comma-separated list without whitespaces.
- **pre-configured sort** – the selected objects will be sorted based on their “Timestamp” attribute, while ignoring any sort criteria specified in the request.

### 6.7.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a &quot;box&quot;). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userld).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.
6.7.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.7.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.7.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.7.5 POST

This operation is used for retrieving information about a set of selected objects.

6.7.5.1 Example 1: Search for objects with certain criteria (Informative)

In this example the search results in a list of five objects. However, since the client asked for a maximum entries of 3 in the response, the server paginates the response accordingly and signals this fact with the inclusion of a cursor in the first list (i.e. objectList). Example 2 is a continuation of Example 1. The cursor encapsulates server state information which might be volatile - see section 5.1.10 for further information.

6.7.5.1.1 Request

```xml
POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <maxEntries></maxEntries>
  <searchCriteria>
    <criterion>
      <type>Attribute</type>
      <name>Message-Context</name>
      <value>pager-message</value>
    </criterion>
    <criterion>
      <type>Attribute</type>
      <name>Direction</name>
      <value>In</value>
    </criterion>
    <criterion>
      <type>Attribute</type>
      <name>From</name>
      <value>tel:+19585550100</value>
    </criterion>
    <criterion>
      <type>Date</type>
```

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document
6.7.5.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:objectList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
<object>
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId123</parentFolder>
  <attributes>
    <attribute>
      <name>Message-Context</name>
      <value>pager-message</value>
    </attribute>
    <attribute>
      <name>From</name>
      <value>tel:+19585550100</value>
    </attribute>
    <attribute>
      <name>Date</name>
      <value>2013-11-12T08:30:10Z</value>
    </attribute>
    <attribute>
      <name>Direction</name>
      <value>In</value>
    </attribute>
    <attribute>
      <name>Content-Type</name>
      <value>text/plain</value>
    </attribute>
  </attributes>
  <flags>
    <flag>Seen</flag>
    <flag>Answered</flag>
  </flags>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId998/flags</resourceURL>
</object>
<object>
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes>
    <attribute>
      <name>Message-Context</name>
      <value>pager-message</value>
    </attribute>
    <attribute>
      <name>From</name>
      <value>tel:+19585550100</value>
    </attribute>
    <attribute>
      <name>Date</name>
      <value>2013-11-12T09:12:00Z</value>
    </attribute>
    <attribute>
      <name>Direction</name>
      <value>In</value>
    </attribute>
    <attribute>
      <name>Content-Type</name>
      <value>text/plain</value>
    </attribute>
  </attributes>
  <flags>
    <flag>Seen</flag>
    <flag>Answered</flag>
  </flags>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId990/flags</resourceURL>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId990/payload</resourceURL>
  <lastModSeq>101</lastModSeq>
</object>

<object>
  <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes>
    <attribute>
      <name>Message-Context</name>
      <value>pager-message</value>
    </attribute>
    <attribute>
      <name>From</name>
      <value>tel:+19585550100</value>
    </attribute>
    <attribute>
      <name>Date</name>
      <value>2013-12-12T12:30:50Z</value>
    </attribute>
    <attribute>
      <name>Direction</name>
      <value>In</value>
    </attribute>
    <attribute>
      <name>Content-Type</name>
      <value>text/plain</value>
    </attribute>
  </attributes>
</object>
6.7.5.2 Example 2: Retrieve the remaining search response list (Informative)

This example continues on the search which was started in Example 1. Note the usage of cursor element provided in the previous Example 1 and the usage of it as fromCursor in Example 2. Also note that, since the list is complete, the response in this example does not provide the cursor element in the response.

6.7.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <fromCursor>cursor111</fromCursor>
  <maxEntries>3</maxEntries>
  <searchCriteria>
    <criterion>
      <type>Attribute</type>
      <name>Message-Context</name>
      <value>pager-message</value>
    </criterion>
    <criterion>
      <type>Attribute</type>
      <name>Direction</name>
      <value>In</value>
    </criterion>
    <criterion>
      <type>Attribute</type>
      <name>From</name>
      <value>tel:+195855550100</value>
    </criterion>
    <criterion>
      <type>Date</type>
      <value>minDate=2013-11-11T09:30:10Z</value>
    </criterion>
    <operator>And</operator>
  </searchCriteria>
</nms:selectionCriteria>
6.7.5.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:objectList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
<object>
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId223</parentFolder>
    <attributes>
        <attribute>
            <name>Message-Context</name>
            <value>pager-message</value>
        </attribute>
        <attribute>
            <name>From</name>
            <value>tel:+19585550100</value>
        </attribute>
        <attribute>
            <name>Date</name>
            <value>2013-11-14T06:20:10Z</value>
        </attribute>
        <attribute>
            <name>Direction</name>
            <value>In</value>
        </attribute>
        <attribute>
            <name>Content-Type</name>
            <value>text/plain</value>
        </attribute>
    </attributes>
    <flags>
        <flag>Seen</flag>
        <flag>Answered</flag>
    </flags>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005/flags</resourceURL>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005</resourceURL>
    <path>/main/StaffMeeting/oid1005</path>
    <payloadURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005/payload</payloadURL>
    <lastModSeq>180</lastModSeq>
</object>
<object>
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId223</parentFolder>
    <attributes>
        <attribute>
            <name>Message-Context</name>
            <value>pager-message</value>
        </attribute>
        <attribute>
            <name>From</name>
            <value>tel:+19585550100</value>
        </attribute>
        <attribute>
            <name>Date</name>
            <value>2013-11-14T06:20:10Z</value>
        </attribute>
        <attribute>
            <name>Direction</name>
            <value>In</value>
        </attribute>
        <attribute>
            <name>Content-Type</name>
            <value>text/plain</value>
        </attribute>
    </attributes>
    <flags>
        <flag>Seen</flag>
        <flag>Answered</flag>
    </flags>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005/flags</resourceURL>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005</resourceURL>
    <path>/main/StaffMeeting/oid1005</path>
    <payloadURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oid1005/payload</payloadURL>
    <lastModSeq>180</lastModSeq>
</object>
</nms:objectList>
<name>Message-Context</name>
  <value>pager-message</value>
</attribute>

<attribute>
  <name>From</name>
  <value>tel:+19585550100</value>
</attribute>

<attribute>
  <name>Date</name>
  <value>2013-12-14T08:30:50Z</value>
</attribute>

<attribute>
  <name>Direction</name>
  <value>In</value>
</attribute>

<attribute>
  <name>Content-Type</name>
  <value>text/plain</value>
</attribute>

<flags>
  <flag>Recent</flag>
<resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old1010/flags</resourceURL>
</flags>
<resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old1010</resourceURL>
<path>/main/StaffMeeting/old1010</path>
<payloadURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old1010/payload</payloadURL>
<lastModSeq>195</lastModSeq>
</object>
</nms:objectList>
6.7.5.3 Example 3: Search for a substring in all searchable text attributes and bodies (Informative)

In this example a search is made for a matching substring “Football” in all existing user’s messages in the storage. The search uses the search type “AllTextAttributes” to denote a search across all searchable text attributes (e.g., subject, transcript, etc.). See section 5.3.3.1 for further information.

6.7.5.3.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5c903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <maxEntries>3</maxEntries>
  <searchCriteria>
    <criterion>
      <type>AllTextAttributes</type>
      <value>Football</value>
    </criterion>
  </searchCriteria>
</nms:selectionCriteria>
```

6.7.5.3.2 Response

```
HTTP/1.1 200 OK
Date: Fri, 07 Jun 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:objectList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <object>
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld8686</parentFolder>
    <attributes>
      <attribute>
        <name>Message-Context</name>
        <value>text-message</value>
      </attribute>
      <attribute>
        <name>From</name>
        <value>tel:+19587236564</value>
      </attribute>
      <attribute>
        <name>Subject</name>
        <value>R U coming to football game today</value>
      </attribute>
      <attribute>
        <name>Date</name>
        <value>2013-12-02T08:30:10Z</value>
      </attribute>
    </attributes>
  </object>
</nms:objectList>
```
6.7.5.4 Example 4: Search for CreatedObjects not supported (Informative)

In this example the client attempts to perform a CreatedObjects search, but this is not supported by the server since it does not support simplified synchronization. The response indicates this.

6.7.5.4.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <maxEntries>3</maxEntries>
  <searchCriteria>
    <criterion>
      <type>CreatedObjects</type>
      <value></value>
    </criterion>
  </searchCriteria>
</nms:selectionCriteria>
```
6.7.5.4.2 Response

HTTP/1.1 403 Forbidden
Date: Fri, 14 Mar 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <policyException>
    <messageId>POL2006</messageId>
    <text>Requested feature %1 is not available</text>
    <variables>CreatedObjects</variables>
  </policyException>
</common:requestError>

6.7.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.8 Resource: Resource URLs of a selected set of objects in the storage

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/pathToId

This resource is used for retrieving the resource URL for an object based on its pathname or a list of objects, based on their pathnames.
## 6.8.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

## 6.8.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

## 6.8.3 GET

This operation is used for retrieving the resource URL for an object based on its pathname.

Supported parameters in the query string of the Request URL are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>No</td>
<td>The location of the object in the hierarchical storage. If path is absent, malformed or invalid, an appropriate Service Exception (e.g. SVC0004) SHALL be returned.</td>
</tr>
</tbody>
</table>

### 6.8.3.1 Example 1: Retrieve object’s resource URL based on its path (Informative)

#### 6.8.3.1.1 Request

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId?path=/main/conversation5/obj12345
HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
```
6.8.3.1.2 Response

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj12345</resourceURL>
  <path>/main/conversation5/obj12345</path>
</nms:reference>

6.8.3.2 Example 2: Retrieve object’s resource URL based on its path, failure due to an invalid path (Informative)

In this example the path is invalid: it points to a non-existent object/main//conversation5/obj12345.

6.8.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId?path=/main//conversation5/obj12345
HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.8.3.2.2 Response

HTTP/1.1 400 Bad Request
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
    <messageId>SVC0002</messageId>
    <text>Invalid input value for message part %1</text>
    <variables>/main//conversation5/obj12345</variables>
  </serviceException>
</common:requestError>

6.8.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.8.5 POST

This operation is used for retrieving the resource URLs for a list of objects based on their pathnames.
6.8.5.1 Example 1: Retrieve list of objects’ resource URLs based on their paths
(Informative)

6.8.5.1.1 Request

```xml
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:pathList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111</path>
  <path>/main/conversation5/oId221</path>
  <path>/main/conversation5/oId222</path>
</nms:pathList>
```

6.8.5.1.2 Response

```
HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId111</resourceURL>
      <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111</path>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId221</resourceURL>
      <path>/main/conversation5/oId221</path>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId222</resourceURL>
      <path>/main/conversation5/oId222</path>
    </success>
  </response>
</nms:bulkResponseList>
```
6.8.5.2  Example 2: Retrieve list of objects’ resource URLs based on their paths, at least one path to a non-existing object  (Informative)

In this example there is an invalid path which does not match an existing object. Response to the request is a 200 OK while the actual result of the success or failure for each object in the request are reported in the response body.

6.8.5.2.1  Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc1956fd903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:pathList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111</path>
  <path>/cleanups/conversation5/oId221</path>
</nms:pathList>
```

6.8.5.2.2  Response

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 12:09:09 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId111</resourceURL>
      <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111</path>
    </success>
  </response>
  <response>
    <code>400</code>
    <reason>Bad Request</reason>
    <failure>
      <serviceException>
        <messageId>SVC0002</messageId>
        <text>Invalid input value for message part %1</text>
        <variables>/cleanups/conversation5/old221</variables>
      </serviceException>
    </failure>
  </response>
</nms:bulkResponseList>
```

6.8.5.3  (void)

6.8.6  DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.9 Resource: Bulk creation of objects

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/bulkCreation

This resource is used for creating multiple objects using a single request.

6.9.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a 'box' of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a 'box' is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.9.2 Response Codes and Error Handling

If the creation of all the objects failed the HTTP response code SHALL be 4xx or 5xx.
Otherwise the HTTP response code SHALL be 2xx, even if the creation of some (but not all) objects failed.
For HTTP response codes, see [REST_NetAPI_Common].

The response body includes a list of success or failure status for each object in the request list respectively.

The maximum size of bulk creation request MAY be limited subject to server’s pre-defined policy, e.g. by number of objects, object size, total request size. For this reason the client SHOULD NOT make unreasonably large bulk creation requests.

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.9.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.9.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.9.5 **POST**

This operation is used for creating multiple objects in a single request.

The request contains both the ObjectList data structure and the payloads of each uploaded object.

The request body is a multipart/form-data entity, where the first entry of the form is the ObjectList and the subsequent entries of the form are the payloads of the objects. This format is an extension of that defined in [REST_NetAPI_Common] and [REST_WP], and follows [RFC2388].

- The root fields are as described in [REST_NetAPI_Common]. The type of this form entry MUST be ObjectList.

- For each object, in the same order as they appear in the ObjectList, the multimedia contents are represented as described in [REST_NetAPI_Common]. The resulting form-data parts each have the same name (“attachments”).

  The concerns of [RFC2388] section 5.5 do not apply: NMS servers, clients, and intermediaries MUST NOT reorder these fields.

  Where an object contains no content item (i.e., no payload), this is represented by including a MIME body with:

  ```
  Content-Disposition: form-data; name="attachments"
  Content-Length: 0
  and no Content-Type.
  ```

The response body contains a BulkResponseList, and the order of the elements in the list corresponds to the order of the object elements in the request (within ObjectList).

### 6.9.5.1 Example 1: Bulk creation (Informative)

In this example three objects are created using a single POST request. Response to the request is a 200 OK while the actual result of the success or failure for each object in the request are reported in the response body. In this example it is assumed that folder “Pictures” did not exist prior to the request, but is implicitly created upon creation of the first child object.

### 6.9.5.1.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkCreation HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456==";
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/xml
Content-Disposition: form-data; name="root-fields"
<?xml version="1.0" encoding="UTF-8"?>
<nms:objectList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
<object>
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123</parentFolder>
  <attributes/>
  <flags>
    <flag>Seen</flag>
    <flag>Flagged</flag>
  </flags>
</object>
<object>
</object>
```
<parentFolderPath>/Pictures</parentFolderPath>
<attributes/>
<flags>
  <flag>\Seen</flag>
</flags>
</object>
<parentFolderPath>/Pictures</parentFolderPath>
<attributes/>
<flags>
  <flag>\Seen</flag>
  <flag>\Flagged</flag>
</flags>
</object>
</nms:objectList>
--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep=---"
Content-Disposition: form-data; name="attachments"

-----=-sep=---
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

See attached photo

-----=-sep=---
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=-sep=---
--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep=---"
Content-Disposition: form-data; name="attachments"

-----=-sep=---
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

Photo from trip to Vancouver

-----=-sep=---
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=-sep=---
--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep=---"
Content-Disposition: form-data; name="attachments"

-----=-sep=---
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"
Photo from Sorrento Meeting

---==--
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

---==--
================outer123456==

6.9.5.2 Response

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 12:09:09 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>201</code>
    <reason>Created</reason>
    <success>
      <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj412</resourceURL>
      <path>/SpringBreak2014/obj412</path>
    </success>
  </response>
  <response>
    <code>403</code>
    <reason>Forbidden</reason>
    <failure>
      <policyException>
        <messageId>POL0001</messageId>
        <text>A policy error occurred. Error code is %1</text>
        <variables>E42</variables>
      </policyException>
    </failure>
  </response>
  <response>
    <code>201</code>
    <reason>Created</reason>
    <success>
      <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj413</resourceURL>
      <path>/Pictures/obj413</path>
    </success>
  </response>
</nms:bulkResponseList>

6.9.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.10 Resource: Bulk update of objects

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/bulkUpdate

This resource is used for updating multiple objects using a single request.

6.10.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL.</td>
</tr>
<tr>
<td></td>
<td>Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined</td>
</tr>
<tr>
<td></td>
<td>in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e.</td>
</tr>
<tr>
<td></td>
<td>a server hosting multiple independent stores). The value of this variable is</td>
</tr>
<tr>
<td></td>
<td>deployment dependent (e.g. in simple deployment scenarios it could be a fixed</td>
</tr>
<tr>
<td></td>
<td>literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this</td>
</tr>
<tr>
<td></td>
<td>variable depends on the deployment scenario and the service provider's policy.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own,</td>
</tr>
<tr>
<td></td>
<td>the value of “boxId” can be equivalent to the unique identifier of the user</td>
</tr>
<tr>
<td></td>
<td>(e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple</td>
</tr>
<tr>
<td></td>
<td>users (or machines), the value of “boxId” can be a unique identifier of the</td>
</tr>
<tr>
<td></td>
<td>group in deployment scenarios where a ‘box’ is allocated to a machine (non-</td>
</tr>
<tr>
<td></td>
<td>human user), the value of the “boxId” can be a unique identifier of the</td>
</tr>
<tr>
<td></td>
<td>machine.</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.10.2 Response Codes and Error Handling

If the update of all the objects failed the HTTP response code SHALL be 4xx or 5xx.
Otherwise the HTTP response code SHALL be 2xx, even if the update of some (but not all) objects failed.

For HTTP response codes, see [REST_NetAPI_Common].

The response body includes a list of success or failure status for each object in the request list (if objects are specified in the request) or for each matching object (if selectionCriteria are specified in the request) respectively.

The maximum size of bulk update request MAY be limited subject to server’s pre-defined policy, e.g. by number of objects, object size, total request size. For this reason the client SHOULD NOT make unreasonably large bulk update requests.

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.10.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.10.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.10.5 POST

This operation is used for updating flags of multiple objects in a single request.

The response includes only the references of the objects which are directly updated, i.e., objects named in the request. For search-based bulkUpdate requests however, the response includes all the matching objects which were successfully updated.

For search-based bulkUpdate requests, selectionCriteria.fromCursor and selectionCriteria.maxEntries SHALL be applied in such a way that only up to the maxEntries or the server’s limit (whichever is less) objects are updated at a time, and subsequent requests are required to update the rest of the matching objects.

Note that the client could also receive notifications about the objects updated during the bulkUpdate operation, e.g., if it is subscribed for notifications and the scope and filter include these objects.

6.10.5.1 Example 1: Bulk update a requested list of objects (Informative)

6.10.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkUpdate HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkUpdate xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <objects>
    <objectReference>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
    </objectReference>
    <objectReference>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
    </objectReference>
  </objects>
  <operation>AddFlag</operation>
  <flags>
    <flag>Seen</flag>
  </flags>
</nms:bulkUpdate>
6.10.5.1.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>

<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
      <path>/main/SummerHolidayPlan/objId001</path>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
      <path>/main/SummerHolidayPlan/objId002</path>
    </success>
  </response>
</nms:bulkResponseList>
6.10.5.2 Example 2: Bulk update objects meeting certain criteria  (Informative)

This example demonstrates that only two objects matched the selectionCriteria. Response to the request is a 200 OK while the actual result of the successful update for each matching object are reported in the response body. In this case, the number of matches is less than both maxEntries and the server’s limit, so all matching objects are updated and reported in the bulkResponseList. For an example showing an incomplete bulkResponseList and the correct use of the cursor to complete it, see Section 6.11.5.3.

6.10.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkUpdate HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkUpdate xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <selectionCriteria>
    <maxEntries>3</maxEntries>
    <searchCriteria>
      <criterion>
        <type>Attribute</type>
        <name>Direction</name>
        <value>In</value>
      </criterion>
      <criterion>
        <type>Attribute</type>
        <name>From</name>
        <value>tel:+19585550100</value>
      </criterion>
      <criterion>
        <type>Date</type>
        <value>minDate=2014-11-11T09:30:10Z</value>
      </criterion>
    </searchCriteria>
  </selectionCriteria>
  <operation>AddFlag</operation>
  <flags>
    <flag>Seen</flag>
  </flags>
</nms:bulkUpdate>
6.10.5.2.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8" ?>

<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId008</resourceURL>
      <path>/main/SummerHolidayPlan/objId008</path>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId009</resourceURL>
      <path>/main/SummerHolidayPlan/objId009</path>
    </success>
  </response>
</nms:bulkResponseList>
6.10.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.11 Resource: Bulk deletion of objects

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/operations/bulkdelete

This resource is used for deleting multiple objects using a single request.

6.11.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.11.2 Response Codes and Error Handling

If the delete of all the objects failed the HTTP response code SHALL be 4xx or 5xx.

Otherwise the HTTP response code SHALL be 2xx, even if the delete of some (but not all) objects failed.

For HTTP response codes, see [REST_NetAPI_Common].

The response body includes a list of success or failure status for each object in the request list (if objects are specified in the request) or for each matching object (if selectionCriteria are specified in the request) respectively.

The maximum size of bulk delete request MAY be limited subject to server’s pre-defined policy, e.g. by number of objects, object size, total request size. For this reason the client SHOULD NOT make unreasonably large bulk delete requests.

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.11.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.11.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.11.5 POST

This operation is used for deleting multiple objects in a single request.

The response body contains a BulkResponseList of the objects which were successfully deleted.

The response includes only the references of the objects which are directly deleted, i.e., objects named in the request. For search-based bulkDelete requests however, the response includes all the matching objects which were successfully deleted.

For search-based bulkDelete requests, selectionCriteria.fromCursor and selectionCriteria.maxEntries SHALL be applied in such a way that only up to the maxEntries or the server’s limit (whichever is less) objects are deleted at a time, and subsequent requests are required to delete the rest of the matching objects.

Note that the client could also receive notifications about the objects deleted during the bulkDelete operation, e.g., if it is subscribed for notifications and the scope and filter include objects.

6.11.5.1 Example 1: Bulk delete a given list of objects (Informative)

6.11.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkDelete xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
<object>
<objectReference>
<resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
</objectReference>
<objectReference>
<resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
</objectReference>
</objects>
</nms:bulkDelete>
6.11.5.1.2  Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>

<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
    </success>
  </response>
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
    </success>
  </response>
</nms:bulkResponseList>

6.11.5.2  Example 2: Bulk delete objects meeting certain criteria  (Informative)

In this example the search (using the given selectionCriteria) results in a list of 10 objects. However, since the client asked for a maximum entries (i.e. maxEntries) of 5 in the response, the server paginates the response accordingly and signals this fact with the inclusion of a cursor in the response (i.e.bulkResponseList). The client is required to repeat the same request in order to retrieve the rest. Example 3 is a continuation of Example 2. The cursor encapsulates server state information which might be volatile - see section 5.1.10 for further information.

6.11.5.2.1  Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>

<nms:bulkDelete xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <selectionCriteria>
    <maxEntries>5</maxEntries>
  </selectionCriteria>
</nms:bulkDelete>
6.11.5.2.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId021</resourceURL>
    </success>
  </response>
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId032</resourceURL>
    </success>
  </response>
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId039</resourceURL>
    </success>
  </response>
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId012</resourceURL>
    </success>
  </response>
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId014</resourceURL>
    </success>
  </response>
  <cursor>cursor944</cursor>
</nms:bulkResponseList>
6.11.5.3  Example 3: Retrieve the remaining bulk delete response list
(Informative)

This example continues on the bulk delete search which was started in Example 2. Note the usage of cursor element provided
in the previous Example 2 and the usage of it as fromCursor in Example 3. Also note that, since the list is complete, the
response in this example does not provide the cursor element in the response

6.11.5.3.1  Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkDelete xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <selectionCriteria>
    <fromCursor>cursor944</fromCursor>
    <maxEntries>5</maxEntries>
    <searchCriteria>
      <criterion>
        <type>Attribute</type>
        <name>From</name>
        <value>tel:+19585550100</value>
      </criterion>
      <criterion>
        <type>Date</type>
        <value>maxDate=2014-12-30T09:30:10Z</value>
      </criterion>
    </searchCriteria>
  </selectionCriteria>
</nms:bulkDelete>

6.11.5.3.2  Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:52:01 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>204</code>
    <reason>No Content</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId081</resourceURL>
    </success>
  </response>
</nms:bulkResponseList>
6.11.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.12 Resource: Resource containing all folders

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders

This resource is used for creating a new folder.

6.12.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>– in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>– in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>– in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.12.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.12.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.12.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.12.5 POST

This operation is used for creating a new folder.
6.12.5.1 Example 1: Folder creation by parentFolder path, response with a location of created resource

The following example shows a request for creating a new folder called BoardMeeting to be created under the folder with path “/main”. This example assumes that a folder with path “/main” already exists.

6.12.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1"
  <parentFolderPath>/main</parentFolderPath>
  <attributes></attributes>
  <name>BoardMeeting</name>
</nms:folder>

6.12.5.1.2 Response

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
</nms:reference>

6.12.5.2 Example 2: Folder creation by parentFolder path, response with a copy of created resource

The following example shows a request for creating a new folder called NMSdiscussion to be created under the folder with path “/main”. This example assumes that a folder with path “/main” already exists.
6.12.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?><nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1"><parentFolderPath>/main</parentFolderPath><attributes></attributes><name>NMSdiscussion</name></nms:folder>

6.12.5.2.2 Response

HTTP/1.1 201 Created
Date: Tue, 15 Apr 2014 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld559
Content-Type: application/xml
Content-Length: nnnn

6.12.5.3 Example 3: Folder creation by parentFolder path, response creation failure due to an invalid folder path  
(Informative)

The following example shows a request for creating a new folder called “WorldCup2014” under the folder with path “/main/myBackups/Football”. This example assumes that the parent folder “/main/myBackups/Football” does not exist.

6.12.5.3.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?><nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1"><parentFolderPath>/main/myBackups/Football</parentFolderPath><attributes></attributes><name>WorldCup2014</name></nms:folder>
```

6.12.5.3.2 Response

```
HTTP/1.1 400 Bad request
Date: Wed, 20 Nov 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1"><serviceException><messageId>SVC0002</messageId><text>Invalid input value for message part %1</text><variables>/main/myBackups/Football</variables></serviceException></common:requestError>
```

6.12.5.4 Example 4: Folder creation by parentFolder resourceURL, response with a copy of created resource  
(Informative)

The following example shows a request for creating a new folder called SorrentoMeeting to be created under the folder with Id of fld559. See previous example where NMSdiscussion was created under “/main”. In this example we create SorrentoMeeting folder under NMSdiscussion folder using its folderId = fld559. This example assumes that folderId=fld559 already exists.

6.12.5.4.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

```
<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559</parentFolder>
  <attributes></attributes>
  <name>SorrentoMeeting</name>
</nms:folder>

6.12.5.4.2 Response

HTTP/1.1 201 Created
Date: Mon, 20 Jan 2014 11:20:13 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld560
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559</parentFolder>
  <attributes>
    <attribute>
      <name>Date</name>
      <value>2014-01-20T11:20:10Z</value>
    </attribute>
    <attribute>
      <name>Name</name>
      <value>SorrentoMeeting</value>
    </attribute>
  </attributes>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld560</resourceURL>
  <path>/main/NMSdiscussion/SorrentoMeeting</path>
  <name>SorrentoMeeting</name>
  <lastModSeq>977</lastModSeq>
</nms:folder>

6.12.5.5 Example 5: Folder creation failure due to request missing the parent folder element (Informative)

6.12.5.5.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <attributes></attributes>
  <name>VegasMeeting</name>
</nms:folder>
6.12.5.5.2 Response

HTTP/1.1 400 Bad Request
Date: Thu, 20 Nov 2014 20:51:51 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
    <messageId>SVC0002</messageId>
    <text>Invalid input value for message part %1</text>
    <variables>Parent folder (missing</variables>
  </serviceException>
</common:requestError>

6.12.5.6 Example 6: Folder creation by parentFolder path, response creation failure due to prohibited location (i.e. requested parent folder) (Informative)

The following example shows a request for creation of user-defined folder under a prohibited system folder called /Default which is allowed to be used by CPM participating function only.

6.12.5.6.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/xml
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/xml
Content-Length: nnnn
MIME-Version: 1.0

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolderPath>/Default</parentFolderPath>
  <attributes></attributes>
  <name>MySavedPictures</name>
</nms:folder>

6.12.5.6.2 Response

HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 21:01:11 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <policyException>
    <messageId>POL1031</messageId>
    <text>Attempt to create objects or folders under %1 is prohibited</text>
    <variables>/Default</variables>
  </policyException>
</common:requestError>
6.12.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.13 Resource: A folder

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/{folderId}

This resource is used for managing a folder such as retrieving information about the contents of a folder or deleting a folder, including contained folders and objects (with their payload).

6.13.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

folderId | Folder identifier.                                                                

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.13.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.13.3 GET

This operation is used for retrieval of a folder’s properties such as its location and the list of contained subfolders and objects. Supported parameters in the query string of the Request URL are:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromCursor</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The beginning position of the retrieve response. Omitting this value denotes the first position. The fromCursor is a cursor value provided by the server in a previous response to a request for the same folder; see section 5.1.10.</td>
</tr>
<tr>
<td>maxEntries</td>
<td>xsd:unsignedInt</td>
<td>Yes</td>
<td>Specifies maximum number of subfolders/objects entries to be returned in the response. The server MAY return fewer entries than this. Default is provided by server policy.</td>
</tr>
<tr>
<td>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Controls whether subfolders and/or objects (contained in the folder) is required to be returned in the body of the GET response. If listFilter is absent, GET response body SHALL NOT include subfolders nor objects. If listFilter is absent, maxEntries SHOULD be omitted (if present it SHALL be ignored). If listFilter = Subfolders, GET response body SHALL include only subfolders list. If listFilter = Objects, GET response body SHALL include only objects list. If listFilter = All, GET response body SHALL include both Subfolders and objects lists.</td>
</tr>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Controls whether folder’s path is required to be returned in the body of the GET response. If path is absent, GET response body SHALL NOT include folder’s path. If path = Yes, GET response body SHALL include the folder’s path.</td>
</tr>
<tr>
<td>attrFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines selected attribute(s) to be returned in the body of the GET response. If attrFilter is absent, GET response body SHALL NOT include the following attributes which are considered to be processing-intensive from server’s perspective:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• MsgCount</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• UnreadMsgCount</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SubtreeMsgCount</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SubtreeUnreadMsgCount</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SubtreeSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The syntax of this query parameter is: attrFilter=attrFilterValue, where attrFilterValue is an attribute name from the supported folder attribute names as listed in Appendix J. Multiple attrFilter=attrFilterValue pairs separated by “&amp;” are allowed.</td>
</tr>
</tbody>
</table>
6.13.3.1 Example 1: Retrieve information about a folder (Informative)

6.13.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld608?path=Yes&amp;attrFilter=SubtreeMsgCount&amp;attrFilter=SubtreeSize HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.13.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld554</parentFolder>
  <attributes>
    <attribute>
      <name>Conversation-ID</name>
      <value>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</value>
    </attribute>
    <attribute>
      <name>Contribution-ID</name>
      <value>abcdef-1234-5678-90ab-cdef01234567</value>
    </attribute>
    <attribute>
      <name>Date</name>
      <value>2013-11-19T08:30:50Z</value>
    </attribute>
    <attribute>
      <name>Name</name>
      <value>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</value>
    </attribute>
    <attribute>
      <name>SubtreeMsgCount</name>
      <value>140</value>
    </attribute>
    <attribute>
      <name>SubtreeSize</name>
      <value>18766988</value>
    </attribute>
  </attributes>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld608</resourceURL>
  <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
  <name>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</name>
  <lastModSeq>600</lastModSeq>
</nms:folder>
6.13.3.2 Example 2: Retrieve information about a non-existent folder (Informative)

6.13.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld444777 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.13.3.2.2 Response

HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: nnnn
Date: Fri, 17 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="folder" href="http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld444777"/>
  <serviceException>
    <messageId>SVC0004</messageId>
    <text>No valid addresses provided in message part %1</text>
    <variables>Request-URI</variables>
  </serviceException>
</common:requestError>

6.13.3.3 Example 3: Retrieve information about a large folder (Informative)

6.13.3.3.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608?listFilter=All&maxEntries=3 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.13.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554</parentFolder>
  <attributes>
    <attribute>
      <name>Conversation-ID</name>
      <value>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</value>
    </attribute>
    <attribute>
      <name>Contribution-ID</name>
      <value>abcdef-1234-5678-90ab-cdef01234567</value>
    </attribute>
  </attributes>
</nms:folder>
6.13.3.4 Example 4: Retrieve information about a large folder (Informative)

This example continues the previous one, by passing back the cursor provided by the server.

6.13.3.4.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608?fromCursor=abcdef&cur%26194&listFilter=All&maxEntries=3 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.13.3.4.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folder xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <parentFolder>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554</parentFolder>
  <attributes>
    <attribute>
      <name>Conversation-ID</name>
      <value>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</value>
    </attribute>
    <attribute>
      <name>Contribution-ID</name>
      <value>abcdef-1234-5678-90ab-cdef01234567</value>
    </attribute>
  </attributes>
</nms:folder>
6.13.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.13.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.13.6 DELETE

This operation is used to delete a folder. All the contained folders and objects (including their payload) in the targeted folder SHALL be deleted as well.

The server responds to a DELETE request with an HTTP 204 No Content response.

6.13.6.1 Example 1: Delete a folder, response with “204 No Content” (Informative)

6.13.6.1.1 Request

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId567 HTTP/1.1
Host: example.com
Accept: application/xml

6.13.6.1.2 Response

HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 06:05:09 GMT
6.14 Resource: Individual folder data

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/{folderId}/[ResourceRelPath]

This resource is used for changing a folder’s name. It can also be used to retrieve the folder’s name.

6.14.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td>folderId</td>
<td>Folder identifier.</td>
</tr>
<tr>
<td>[ResourceRelPath]</td>
<td>Relative resource path for a Light-weight Resource, consisting of a relative path down to an element in the data structure. For more information about the applicable values (strings) for this variable, see 6.14.1.1.</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.14.1.1 Light-weight relative resource paths

The following table describes the type of Light-weight Resources that can be accessed by using this resource, applicable methods, and the link to a data structure that contains values (strings) for those relative resource paths.

<table>
<thead>
<tr>
<th>Light-weight Resource type</th>
<th>Method supported</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual folder data</td>
<td>GET, PUT</td>
<td>Enables access to folderName data element of a folder. See column [ResourceRelPath] for element “name” in section 5.3.2.8 for possible values for the Light-weight relative resource path.</td>
</tr>
</tbody>
</table>

6.14.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.
6.14.3 GET

This operation is used for retrieval of a folder’s name.

6.14.3.1 Example: Retrieve a folder’s name (Informative)

6.14.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName
HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.14.3.1.2 Response

HTTP/1.1 200 OK
Date: Mon, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:name xmlns:nms="urn:oma:xml:rest:netapi:nms:1">BoardMeeting</nms:name>

6.14.4 PUT

This operation is used for changing a folder’s name.

6.14.4.1 Example 1: Change folder name (Informative)

6.14.4.1.1 Request

PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:name xmlns:nms="urn:oma:xml:rest:netapi:nms:1">BoardSession1</nms:name>

6.14.4.1.2 Response

HTTP/1.1 200 OK
Date: Mon, 04 Jun 2012 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:name xmlns:nms="urn:oma:xml:rest:netapi:nms:1">BoardSession1</nms:name>

6.14.4.2 Example 2: Change folder name, failure due to Policy error (Informative)

Certain folders may not be allowed to be renamed by the server. Attempting to rename such folders (e.g. /Default) would result in a Policy error.
6.14.4.2.1 Request

```
PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:name xmlns:nms="urn:oma:xml:rest:netapi:nms:1">myDefault</nms:name>
```
6.15.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.15.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.15.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.15.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.15.5 POST

This operation is used for retrieving information about a set of selected folders, where the set is defined by selection criteria.

6.15.5.1 Example 1: Search for root folders (Informative)

In this example the search results in a list of one root folder. The root folder’s parentFolder element is omitted meaning that it does not have a parent folder (see section 5.3.2.8 for further information). Also in this example root folder’s name (i.e. folderName) is empty which determines its path to be an empty string as well.
6.15.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <maxEntries>3</maxEntries>
  <searchCriteria>
    <criterion>
      <type>Attribute</type>
      <name>root</name>
      <value>Yes</value>
    </criterion>
  </searchCriteria>
</nms:selectionCriteria>

6.15.5.1.2 Response

HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folderList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <folder>
    <attributes>
      <attribute>
        <name>Root</name>
        <value>Yes</value>
      </attribute>
    </attributes>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fIdroot1</resourceURL>
    <path></path>
    <lastModSeq>1</lastModSeq>
    <subFolders>
      <folderReference>
        <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554</resourceURL>
        <path>/main</path>
      </folderReference>
      <folderReference>
        <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld664</resourceURL>
        <path>/tmp</path>
      </folderReference>
    </subFolders>
  </folder>
</nms:folderList>
6.15.5.2 Example 2: Search for folders created within a given timeframe (Informative)

In this example the search is performed to look for folder created within a given timeframe while the search is instructed to start at a particular node/folder in the storage hierarchy.

6.15.5.2.1 Request

```xml
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><nms:selectionCriteria xmlns:nms="urn:oma:xml:rest:netapi:nms:1"><maxEntries>10</maxEntries><searchCriteria><criterion><type>Date</type><value>minDate=2013-12-01T08:00:00Z&amp;maxDate=2014-01-01T12:00Z</value></criterion></searchCriteria><searchScope><resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld24</resourceURL><path>/main/projects/APIs</path></searchScope><sortCriteria><criterion><type>Date</type><order>Ascending</order></criterion></sortCriteria></nms:selectionCriteria>
```

6.15.5.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 07:51:50 GMT
Content-Type: application/xml
Content-Length: nnnn

```xml
```
6.15.5.3 Example 3: Search for folders with a given name (Informative)

In this example the search is performed to look for folders with a given name. The search is conducted over the entire user's storage hierarchy.

6.15.5.3.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn
<?xml version="1.0" encoding="UTF-8"?>
  <maxEntries>10</maxEntries>
  <searchCriteria>
    <criterion>
      <type>Attribute</type>
      <name>Name</name>
      <value>projects</value>
    </criterion>
  </searchCriteria>
</nms:selectionCriteria>

6.15.5.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:50:50 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:folderList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <folder>
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fldMain01</parentFolder>
    <attributes>
      <attribute>
        <name>root</name>
        <value>No</value>
      </attribute>
      <attribute>
        <name>Date</name>
        <value>2013-12-10T09:00:10Z</value>
      </attribute>
      <attribute>
        <name>Name</name>
        <value>projects</value>
      </attribute>
    </attributes>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld0044</resourceURL>
    <path>/main/projects</path>
    <name>projects</name>
    <lastModSeq>18</lastModSeq>
  </folder>
  <folder>
    <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld0044</parentFolder>
    <attributes>
      <attribute>
        <name>root</name>
        <value>No</value>
      </attribute>
      <attribute>
        <name>Date</name>
        <value>2014-01-18T19:30:10Z</value>
      </attribute>
      <attribute>
        <name>Name</name>
        <value>projects</value>
      </attribute>
    </attributes>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld0044</resourceURL>
    <path>/main/projects</path>
    <name>projects</name>
    <lastModSeq>18</lastModSeq>
  </folder>
</nms:folderList>
6.15.5.4 Example 4: Search for folders containing a given substring in its name (Informative)

In this example the search is performed to look for folders with a given substring in its name. The search is conducted over the entire user’s storage hierarchy.

6.15.5.4.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn


6.15.5.4.2 Response

HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

6.15.6  DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.16  Resource: Resource URLs of a selected set of folders in the storage

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/operations/pathToId

This resource is used for retrieving the resource URL for a folder based on its pathname or a list of folders, based on their pathnames.
### 6.16.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
</tbody>
</table>
| boxId      | Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider's policy. For example:  
- in deployment scenario where each user is allocated a 'box' of its own, the value of "boxId" can be equivalent to the unique identifier of the user (e.g. userId).  
- in deployment scenario where a 'box' is allocated to a group of multiple users (or machines), the value of "boxId" can be a unique identifier of the group  
- in deployment scenarios where a 'box' is allocated to a machine (non-human user), the value of the "boxId" can be a unique identifier of the machine |

See section 6 for a statement on the escaping of reserved characters in URL variables.

### 6.16.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

### 6.16.3 GET

This operation is used for retrieving the resource URL for a folder based on its pathname.

Supported parameters in the query string of the Request URL are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>xsd:string</td>
<td>Yes</td>
<td>The location of the folder in the hierarchical storage. If &quot;path&quot; element is absent, GET response body SHALL include the root folder’s resource URL provided there is a single root folder otherwise, Service Exception SVC1009 SHALL be returned. For further information see section 7.</td>
</tr>
</tbody>
</table>

### 6.16.3.1 Example 1: Retrieve folder’s resource URL based on its path (Informative)

#### 6.16.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId?path=/RCSMessageStore/footballGame  
HTTP/1.1  
Host: example.com  
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903  
Accept: application/xml
6.16.3.1.2 Response

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId11111</resourceURL>
  <path>/RCSMessageStore/footballGame</path>
</nms:reference>

6.16.3.2 Example 2: Retrieve root folder’s resource URL (Informative)

In this example the absence of a query parameter triggers the server to return the root folder’s resource URL. In this example it is assumed that there is a single root folder in the network storage and the server has configured the root folder name to be an empty string which would result in a response containing an empty string for the root folder’s path.

6.16.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.16.3.2.2 Response

HTTP/1.1 200 OK
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
  <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId001</resourceURL>
  <path></path>
</nms:reference>

6.16.3.3 Example 3: Retrieve root folder’s resource URL, failure due to missing path parameter while multiple root folders exist (Informative)

In this example a failure would result as there are multiple root folders and a root folder’s path parameter is required in the request.

6.16.3.3.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
6.16.3.3.2 Response

HTTP/1.1 400 Bad Request
Date: Thu, 22 May 2014 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
    <messageId>SVC1009</messageId>
    <text>Folder’s path is missing. When more than one root folder exists, folder’s path must be provided</text>
  </serviceException>
</common:requestError>

6.16.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.16.5 POST

This operation is used for retrieving the resource URLs for a list of folders based on their pathnames.

6.16.5.1 Example 1: Retrieve list of folders’ resource URLs based on their paths (Informative)

6.16.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:pathList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
  <path>/main/conversation5</path>
  <path>/main/SorrentoMeeting</path>
</nms:pathList>

6.16.5.1.2 Response

HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
</nms:bulkResponseList>
6.16.5.2 Example 2: Retrieve list of folders’ resource URLs based on their paths, two invalid paths in the list (Informative)

In this example there are two invalid paths which do not match an existing folder. Response to the request is a 200 OK while the actual result of the success or failure for each folder in the request are reported in the response body.

6.16.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:pathList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
  <path>/main/conversation5</path>
  <path>/main/SorrentoMeeting/year2014</path>
  <path>/main/conversation99</path>
</nms:pathList>
6.16.5.2.2 Response

HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ns1:bulkResponseList xmlns:ns1="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId8</resourceURL>
      <path>/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId9</resourceURL>
      <path>/main/conversation5</path>
    </success>
  </response>
  <response>
    <code>400</code>
    <reason>Bad Request</reason>
    <failure>
      <serviceException>
        <messageId>SVC0002</messageId>
        <text>Invalid input value for message part %1</text>
        <variables>/main//SorrentoMeeting/year2014</variables>
      </serviceException>
    </failure>
  </response>
  <response>
    <code>400</code>
    <reason>Bad Request</reason>
    <failure>
      <serviceException>
        <messageId>SVC0002</messageId>
        <text>Invalid input value for message part %1</text>
        <variables>/main/conversation99</variables>
      </serviceException>
    </failure>
  </response>
</ns1:bulkResponseList>

6.16.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.17 Resource: Resource for triggering object(s)/folder(s) copying

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/operations/copyToFolder

This resource is used for copying referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder

6.17.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userld).</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>− in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.17.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.17.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.17.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.17.5 POST

This operation is used for copying referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder.

The response includes only the references of the folders and objects which are directly created, i.e., which are copies of folders and objects named in the request. It does not include references to the folders and objects which are created by recursion. The client can obtain these by recursively retrieving the contents of the folders listed in the response.

Note that the client could also receive notifications about the folders and objects created during the copy operation, e.g., if it is subscribed for notifications and the scope and filter include these folders and/or objects.

6.17.5.1 Example 1: Copy objects to a target folder (Informative)

In this example it is assumed that the target folder already exists. After the copy operation two new objects (with newly assigned objectIds’) are created in the target folder.

6.17.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/copyToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:targetSourceRef xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <targetRef>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
  </targetRef>
  <sourceRefs>
    <folders/>
    <objects>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
      </objectReference>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
      </objectReference>
    </objects>
  </sourceRefs>
</nms:targetSourceRef>

6.17.5.1.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
  </response>
</nms:bulkResponseList>
6.17.5.2 Example 2: Copy a folder with containing objects to a target folder

It is assumed that the target folder (i.e. SummerHolidayPlan with folderId = fld456) already exists. The source folder (with name “f81d4fae-7dec-11d0-a765-00a0c91e6bf6” and folderId = fld111) containing the two objects is copied to the target folder, creating one new folder directly. As a result the content of the source folder are recursively copied over to the target folder, creating two new objects recursively. This example demonstrates copying a system-created folder (in NMS).

6.17.5.2.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/copyToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:targetSourceRef xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <targetRef>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
  </targetRef>
  <sourceRefs>
    <folders>
      <folderReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
      </folderReference>
      <folders>
        <folderReference>
          <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld111</resourceURL>
        </folderReference>
      </folders>
    </sourceRefs>
  </nms:targetSourceRef>
6.17.5.2.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 03:10:19 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/newfId111222</resourceURL>
      <path>/main/SummerHolidayPlan/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
    </success>
  </response>
</nms:bulkResponseList>

6.17.5.3 Example 3: Copy objects/folders to a target folder, failure due to at least one invalid source object or folder reference (Informative)

In this example it is assumed that at least one source object/folder reference is not valid. Response to the request is a 200 OK while the actual result of the success or failure for each object or folder being copied are reported in the response body.

6.17.5.3.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/operations/copyToFolders HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>
<nms:targetSourceRef xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <targetRef>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld456</resourceURL>
  </targetRef>
  <sourceRefs>
    <folders/>
    <objects>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/objId001</resourceURL>
      </objectReference>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/objId009</resourceURL>
      </objectReference>
    </objects>
  </sourceRefs>
</nms:targetSourceRef>
6.17.5.3.2 Response

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 12:09:09 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/newobjId333</resourceURL>
      <path>/main/SummerHolidayPlan/newobjId333</path>
    </success>
  </response>
  <response>
    <code>400</code>
    <reason>Bad Request</reason>
    <failure>
      <serviceException>
        <messageId>SVC0002</messageId>
        <text>Invalid input value for message part %1</text>
        <variables>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId009</variables>
      </serviceException>
    </failure>
  </response>
</nms:bulkResponseList>

6.17.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.18 Resource: Resource for triggering object(s)/folder(s) moving

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/folders/operations/moveToFolder

This resource is used for moving referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder.
6.18.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
<tr>
<td>boxId</td>
<td>Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider's policy. For example:</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group</td>
</tr>
<tr>
<td></td>
<td>- in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine</td>
</tr>
</tbody>
</table>

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.18.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.18.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.18.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.18.5 POST

This operation is used for moving referenced source object(s) and/or folder(s) (including recursive folders’ content) to a designated target folder.
6.18.5.1 Example 1: Move objects to a target folder

In this example it is assumed that the target folder already exists. After the move operation the two existing objects (with the existing objectId’s) are placed in the target folder identified by folderId of fld456.

6.18.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:targetSourceRef xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <targetRef>
    <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
  </targetRef>
  <sourceRefs>
    <folders/>
    <objects>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
      </objectReference>
      <objectReference>
        <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
      </objectReference>
    </objects>
  </sourceRefs>
</nms:targetSourceRef>

6.18.5.1.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 01:11:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8" ?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001</resourceURL>
    </success>
  </response>
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002</resourceURL>
    </success>
  </response>
</nms:bulkResponseList>
6.18.5.2 Example 2: Move a folder with containing objects to a target folder (Informative)

It is assumed that the target folder (i.e. SummerHolidayPlan with folderId = fld456) already exists. As a result the contents of the source folder are recursively moved to the target folder. This example demonstrates moving a system-created folder (in NMS) with folder name of “81d4fae-7dec-11d0-a765-00a0c91e6bf6” and folderId of fld111 as the source reference. As a result of the move operation no new folder/object is created (i.e. just the location of the existing folder/object’s changes).

Note: it is assumed that the client accordingly (locally) moves the containing objects of a moved parent folder even though the response to the folder move operation does not report the containing moved objects (in the NMS).

6.18.5.2.1 Request

POST /exampleAPInms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:targetSourceRef xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <targetRef>
    <resourceURL>http://exampleAPInms/v1/myStore/tel%3A%2B19585550100/folders/fld456</resourceURL>
  </targetRef>
  <sourceRefs>
    <folders>
      <folderReference>
        <resourceURL>http://exampleAPInms/v1/myStore/tel%3A%2B19585550100/folders/fld111</resourceURL>
      </folderReference>
    </folders>
    <objects/>
  </sourceRefs>
</nms:targetSourceRef>

6.18.5.2.2 Response

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 04:10:19 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<nms:bulkResponseList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <response>
    <code>200</code>
    <reason>OK</reason>
    <success>
      <resourceURL>http://exampleAPInms/v1/myStore/tel%3A%2B19585550100/folders/fld111</resourceURL>
      <path>/main/SummerHolidayPlan/f81d4fae-7dec-11d0-a765-00a0c91e6bf6</path>
    </success>
  </response>
</nms:bulkResponseList>
6.18.5.3 Example 3: Move objects/folders to a target folder, failure due to a forbidden target folder

In this example it is assumed that the target folder is forbidden to move objects/folders to. Such an attempt would result in a policy error.

6.18.5.3.1 Request

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

```

6.18.5.3.2 Response

```
HTTP/1.1 403 Forbidden
Date: Thu, 22 May 2014 12:09:09 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1"><policyException><messageId>POL1031</messageId><text>Attempt to create objects or folders under %1 is prohibited</text><variables>/System</variables></policyException></common:requestError>
```

6.18.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.19 Resource: All subscriptions in the storage

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions

This resource is used to manage subscriptions to NMS event notifications.

This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

The table in Section 6.7 applies to each SearchCriterion.
6.19.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
</tbody>
</table>
| boxId      | Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:  
  - in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).  
  - in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group  
  - in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine |

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.19.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.19.3 GET

This operation is used for reading the list of active NMS notification subscriptions. Only subscriptions this client is authorised to access will be included in the list.
6.19.3.1 Example: Reading all active subscriptions  

Application client reads all active subscriptions.

6.19.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions HTTP/1.1
Accept: application/xml
Host: example.com

6.19.3.1.2 Response

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscriptionList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
    <subscription>
        <callbackReference>
            <notifyURL>http://applicationClient.example.com/nms/notifications/77777</notifyURL>
            <callbackData>abcd</callbackData>
        </callbackReference>
        <duration>6300</duration>
        <clientCorrelator>12345</clientCorrelator>
        <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
        <index>1</index>
        <restartToken>abc123</restartToken>
    </subscription>
    <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions</resourceURL>
</nms:nmsSubscriptionList>

6.19.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.19.5 POST

This operation is used to create a new subscription for NMS notifications.

The notifyURL in the callbackReference either contains the Client-side Notification URL (as defined by the client) or the Server-side Notification URL (as obtained during the creation of the Notification Channel [REST_NetAPI_NotificationChannel]).

The response to the subscription creation MUST contain the index and restartToken of the subscription after it has been created but before any resulting notifications have been issued. In other words, the index MUST be the index of the next notification to be issued after the creation (i.e., it MUST be 1), and if a restartToken is provided in the request, the restartToken in the response MUST be the same as that provided in the request. See section 5.1.4.3 for further information.

6.19.5.1 Example: Creating a new subscription, response with copy of created resource  

(Application Informative)

Application client creates a subscription.
6.19.5.1.1 Request

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscription xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <callbackReference>
    <notifyURL>http://applicationClient.example.com/nms/notifications/77777</notifyURL>
    <callbackData>abcd</callbackData>
  </callbackReference>
  <duration>7200</duration>
  <clientCorrelator>12345</clientCorrelator>
</nms:nmsSubscription>
6.19.5.1.2 Response

HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscription xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
    <callbackReference>
        <notifyURL>http://applicationClient.example.com/nms/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
    <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
    <index>1</index>
    <restartToken>abc123</restartToken>
</nms:nmsSubscription>

6.19.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.20 Resource: Individual subscription

The resource used is:

//{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/subscriptions/{subscriptionId}

This resource is used to manage an individual event subscription. This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.
6.20.1 Request URL variables

The following request URL variables are common for all HTTP methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Version of the API client wants to use. The value of this variable is defined in section 5.2</td>
</tr>
<tr>
<td>storeName</td>
<td>Name of the storage. This variable can be used to allow multi-tenancy (i.e. a server hosting multiple independent stores). The value of this variable is deployment dependent (e.g. in simple deployment scenarios it could be a fixed literal).</td>
</tr>
</tbody>
</table>
| boxId        | Identifier of designated area within the store (a “box”). The value of this variable depends on the deployment scenario and the service provider’s policy. For example:  
|              | - in deployment scenario where each user is allocated a ‘box’ of its own, the value of “boxId” can be equivalent to the unique identifier of the user (e.g. userId).  
|              | - in deployment scenario where a ‘box’ is allocated to a group of multiple users (or machines), the value of “boxId” can be a unique identifier of the group  
|              | - in deployment scenarios where a ‘box’ is allocated to a machine (non-human user), the value of the “boxId” can be a unique identifier of the machine |
| subscriptionId| Identifier of the subscription                                                                                                              |

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.20.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.20.3 GET

This operation is used for reading an individual subscription.

6.20.3.1 Example 1: Reading an individual subscription (Informative)

Application client reads a subscription.

6.20.3.1.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
6.20.3.1.2 Response

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscription xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <callbackReference>
    <notifyURL>http://applicationClient.example.com/nms/notifications/77777</notifyURL>
    <callbackData>abcd</callbackData>
  </callbackReference>
  <duration>7200</duration>
  <clientCorrelator>12345</clientCorrelator>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
  <index>1</index>
  <restartToken>abc123</restartToken>
</nms:nmsSubscription>

6.20.3.2 Example 2: Retrieve information about a non-existent individual subscription (Informative)

6.20.3.2.1 Request

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub6699 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml

6.20.3.2.2 Response

HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="NmsSubscription" href="http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub6699"/>
  <serviceException>
    <messageId>SVC0004</messageId>
    <text>No valid addresses provided in message part %1</text>
    <variables>Request-URI</variables>
  </serviceException>
</common:requestError>

6.20.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST, DELETE’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.20.5  **POST**

This operation is used for updating an individual subscription.

The response to the update MUST contain the index and restartToken of the subscription after it has been updated but before any resulting notifications have been issued. In other words, the index MUST be the index of the next notification to be issued after the update, and if a restartToken is provided in the request, the restartToken in the response MUST be the same as that provided in the request. See section 5.1.4.5 for further information.

6.20.5.1  **Example: Updating the existing subscription**  *(Informative)*

Application client increases the duration (expiry) of an existing subscription while also indicates that it needs to sync up starting at a restart token (e.g. nnn789 in this example which is an opaque string previously reported by the server in a notification list).

6.20.5.1.1  **Request**

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscriptionUpdate xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <duration>10800</duration>
  <restartToken>nnn789</restartToken>
</nms:nmsSubscriptionUpdate>
```

6.20.5.1.2  **Response**

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsSubscription xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <callbackReference>
    <notifyURL>http://applicationClient.example.com/nms/notifications/77777</notifyURL>
    <callbackData>abcd</callbackData>
  </callbackReference>
  <duration>10800</duration>
  <clientCorrelator>12345</clientCorrelator>
  <resourceURL>http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001</resourceURL>
  <index>46</index>
  <restartToken>nnn789</restartToken>
</nms:nmsSubscription>
```

6.20.6  **DELETE**

This operation is used to cancel a subscription and to stop corresponding notifications.

6.20.6.1  **Example: Cancelling a subscription**  *(Informative)*

Application client cancels a subscription.
6.20.6.1.1 Request

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com

6.20.6.1.2 Response

HTTP/1.1 204 No Content
Date: Wed, 15 Jan 2014 18:51:59 GMT

6.21 Resource: Client notification about storage changes

This resource is a callback URL provided by the client for notifications about changes in the network storage. The RESTful NMS API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel to obtain the notifications, the client needs to use the mechanisms described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.21.5.

The following table gives detailed information about NMS storage notification.

<table>
<thead>
<tr>
<th>EventType</th>
<th>Notification Root Element Type</th>
<th>Notification sent to</th>
<th>Response to Notification</th>
<th>Link rel</th>
<th>Link href</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>NmsEventList</td>
<td>client</td>
<td>Success. Content ignored.</td>
<td>NmsSubscription</td>
<td>/subscriptions/{subscriptionId}</td>
</tr>
</tbody>
</table>

Table 1: 1-1 NMS event notification

If the server receives an HTTP 4xx or 5xx error response, from the client or Notification Channel, based on the error and the policy, the server SHOULD determine whether to continue buffering the notifications and retrying in the future or to expire the subscription (e.g. in the event of receiving an HTTP 404, the server can consider this as a permanent error and no reason for retrying and hence expiring the notification subscription or in the event of receiving an HTTP 503, the server can consider this as a temporary error and continue buffering the notifications and attempting to deliver them in retries).

6.21.1 Request URL variables

Client provided.

6.21.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to Network Message Storage, see section 7.

6.21.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
6.21.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.21.5 POST

This operation is used to notify the client about NMS storage events.

6.21.5.1 Example 1: Notify a client about NMS object changes (Informative)

In this example, the application client is notified asynchronously of 3 change events (2 object deletion and 1 object creation). It is assumed that the client has already subscribed for notifications.

6.21.5.1.1 Request

```xml
POST /nms/notifications/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: applicationClient.example.com

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsEventList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <nmsEvent>
    <deletedObject>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999</resourceURL>
      <lastModSeq>133</lastModSeq>
      <correlationId>cId122</correlationId>
    </deletedObject>
    </nmsEvent>
  <nmsEvent>
    <changedObject>
      <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId80</parentFolder>
      <flags>
        <flag>Flagged</flag>
      </flags>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId1000</resourceURL>
      <lastModSeq>134</lastModSeq>
      <correlationId>cId67</correlationId>
    </changedObject>
    </nmsEvent>
  <nmsEvent>
    <expiredObject>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId111</resourceURL>
      <lastModSeq>135</lastModSeq>
      <correlationId>cId9</correlationId>
    </expiredObject>
    </nmsEvent>
  <callbackData>12345</callbackData>
  <index>1</index>
  <restartToken>abc67</restartToken>
  <link rel="NmsSubscription"
    href="http://example.com/exampleAPI/nms/v1/myStore/myStore/tel%3A%2B19585550100/subscriptions/sub001"/>
</nms:nmsEventList>
```
6.21.5.2   Example 2: Notify a client about NMS folder changes   (Informative)

In this example, the application client is notified asynchronously of a folder deletion and a folder creation events. It is assumed that the client is using the same event subscription as in example 1 (hence the index is bumped up to 2 to indicate that this is the 2\(^{nd}\) notification list from the given subscription pointed to by the link).

6.21.5.2.1   Request

POST /nms/notifications/7777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: applicationClient.example.com

<?xml version="1.0" encoding="UTF-8"?>
<nms:nmsEventList xmlns:nms="urn:oma:xml:rest:netapi:nms:1">
  <nmsEvent>
    <deletedFolder>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId20</resourceURL>
      <lastModSeq>136</lastModSeq>
    </deletedFolder>
  </nmsEvent>
  <nmsEvent>
    <changedFolder>
      <parentFolder>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId10</parentFolder>
      <resourceURL>http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId33</resourceURL>
      <name>SorrentoMeeting</name>
      <lastModSeq>137</lastModSeq>
    </changedFolder>
  </nmsEvent>
  <callbackData>12345</callbackData>
  <index>2</index>
  <restartToken>hgf853</restartToken>
  <link rel="NmsSubscription"
        href="http://example.com/exampleAPI/nms/v1/myStore/myStore/tel%3A%2B19585550100/subscriptions/sub001"/>
</nms:nmsEventList>

6.21.5.2.2   Response

HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT

6.21.1   DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].
7. Fault definitions

7.1 Service Exceptions

For common Service Exceptions refer to [REST_NetAPI_Common]. The following additional Service Exception codes are defined for the RESTful Network Message Storage API.

7.1.1 SVC1009: Folder’s path needed

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>SVC1009</td>
</tr>
<tr>
<td>Text</td>
<td>Folder’s path is missing. When more than one root folder exists, folder’s path must be provided.</td>
</tr>
<tr>
<td>Variables</td>
<td>None</td>
</tr>
<tr>
<td>HTTP status code(s)</td>
<td>400 Bad Request</td>
</tr>
</tbody>
</table>

7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common]. The following additional Policy Exception codes are defined for the RESTful Network Message Storage API.

7.2.1 POL1030: Folder cannot be modified, moved or deleted

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>POL1030</td>
</tr>
<tr>
<td>Text</td>
<td>Modifying, moving or deleting this folder is not allowed.</td>
</tr>
<tr>
<td>Variables</td>
<td>None</td>
</tr>
<tr>
<td>HTTP status code(s)</td>
<td>403 Forbidden</td>
</tr>
</tbody>
</table>

7.2.2 POL1031: Object or folder creation under the requested folder not allowed

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>POL1031</td>
</tr>
<tr>
<td>Text</td>
<td>Attempt to create objects or folders under %1 is prohibited</td>
</tr>
<tr>
<td>Variables</td>
<td>%1 –prohibited folders path</td>
</tr>
<tr>
<td>HTTP status code(s)</td>
<td>403 Forbidden</td>
</tr>
</tbody>
</table>
# Appendix A. Change History

## A.1 Approved Version History

<table>
<thead>
<tr>
<th>Reference</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>n/a</td>
<td>No prior version</td>
</tr>
</tbody>
</table>

## A.2 Draft/Candidate Version 1.0 History

<table>
<thead>
<tr>
<th>Document Identifier</th>
<th>Date</th>
<th>Sections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST_NetAPI_NMS-V1_0</td>
<td>19 Mar 2013</td>
<td>All</td>
<td>Initial baseline</td>
</tr>
<tr>
<td></td>
<td>27 Jun 2013</td>
<td>5.1, 5.2.x, 6.x</td>
<td>OMA-ARC-REST-NMS-2013-0004R03 OMA-ARC-REST-NMS-2013-0007R02 OMA-ARC-REST-NMS-2013-0008R02</td>
</tr>
<tr>
<td></td>
<td>08 Aug 2013</td>
<td>5.1</td>
<td>Incorporates: OMA-ARC-REST-NMS-2013-0013R01-CR_set-clear-individual-flags</td>
</tr>
<tr>
<td></td>
<td>20 Aug 2013</td>
<td>5.1, 5.2.2.x</td>
<td>Editorials in sections 6.x</td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>11 Nov 2013</td>
<td>Many sections</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>throughout the</td>
<td>OMA-ARC-REST-NMS-2013-0020R03-C_root</td>
</tr>
<tr>
<td></td>
<td></td>
<td>document</td>
<td>OMA-ARC-REST-NMS-2013-0045R01-C_Concepts_section</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0046-CR_Modification_definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0048R01-C_Update_section_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0049R01-C_search_date_format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR47R02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR51R01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR54R01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Also many editorials (table 5.1 format, cross-references and font changes)</td>
</tr>
<tr>
<td></td>
<td>24 Nov 2013</td>
<td>Many sections</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>throughout the</td>
<td>OMA-ARC-REST-NMS-2013-0070R01-C_search_for_root_folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>document</td>
<td>OMA-ARC-REST-NMS-2013-0062R01-C_FolderList_cursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0064-CR_flags_element_name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20131111-D_CR73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20131111-D_CR72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0065R01-C_Root_path_character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0056R03-CR_Negative_flag_search</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20131111-D_CR71R01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0053R01-CR_System_deletion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0068R02-CR_Notification_simplification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0069R01-CR_Use_xsd_dateTimeStamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR61</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0067R01-CR_Notification_reliability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-TS-REST_NetAPI_NMS-V1_0-20130930-D_CR55R03</td>
</tr>
<tr>
<td></td>
<td>09 Dec 2013</td>
<td>Many sections</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>throughout the</td>
<td>OMA-ARC-REST-NMS-2013-0057R06-CR_External_payload</td>
</tr>
<tr>
<td></td>
<td></td>
<td>document</td>
<td>OMA-ARC-REST-NMS-2013-0066R03-CR_Correlation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0075R01-CR_path_FFS_resolution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0076R01-CR_Attribute_names</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0077R02-CR_clean_up_and_editorial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0078-CR_FlagList_ResourceURL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0079-CR_Further_editorials</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0080R01-CR_Search</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0081-CR_AllSearchableText_clarification</td>
</tr>
<tr>
<td></td>
<td>03 Jan 2014</td>
<td>Many sections</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>throughout the</td>
<td>OMA-ARC-REST-NMS-2013-0074R03-CR_Correlation_with_hash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>document</td>
<td>OMA-ARC-REST-NMS-2013-0084R01-CR_Direction_clarification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0085-CR_Editorials_arising_from_merge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2013-0086R01-CR_Exceptions</td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>29 Jan 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0001R01-CR_Mod_seq_generalisation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0003-CR_Editors_and_appendix_C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0004-CR_Appendix_G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0005-CR_Appendix_F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0006R03-CR_Appendix_B_SCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0007R01-CR_Adding_Sec_6_for_copy_move_to_folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0008-CR_sec_6_xml_examples_for_subscription</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0009-CR_Notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0010-CR_sec_6_further_XML_examples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0011R01-CR_Search_based_on_name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0012R02-CR_Attribute_value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0014-CR_FFS_resolution_prohibited_folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0015-CR_Consistent_use_of_Root_attr_name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0016R01-CR_inline_content_support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0017R01-CR_FFS_subscription_cancel_notif</td>
</tr>
<tr>
<td></td>
<td>10 Feb 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0021-CR_Optional_payload</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0022R02-CR_Cursor_clarification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0023R05-CR_Sync</td>
</tr>
<tr>
<td></td>
<td>11 Mar 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0024R02-CR_Cursor_reliability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0023R05-CR_Sync</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0031R02-CR_FFS_Object_and_Folder_Reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0032R01-CR_FFS_parentFolder_in_notification</td>
</tr>
<tr>
<td></td>
<td>18 Mar 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates the following CONRR resolutions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0036-CR_CONR_editorials_and_small_bugs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0036-CR_CONR_editorials_and_small_bugs</td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>12 Apr 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates the following CONRR resolutions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0037R01-CR_CONR_further_editorials</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0038R02-CR_CONR_order_preservation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0039R01-CR_CONR_sort_criteria</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0040R01-CR_CONR_naming_cleanup</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0042R01-CR_CONR_simplifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0043R01-CR_CONR_redundant_ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0044R01-CR_CONR_search_consistency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0045R01-CR_CONR_absent_lists</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0046R01-CR_CONR_minor_technical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0047R01-CR_CONR_large_folder_support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0048R01-CR_CONR_plural_item_names</td>
</tr>
<tr>
<td></td>
<td>29 Apr 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates the following CONRR resolutions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0053R04-CR_CONR_add_more_pathToId_examples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0054R01-CR_CONR_additional_bugfixes_and_editorials</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0055R01-CR_CONR_data_type_sections_in_alphabetical_order</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0057R01-CR_VanishedObjects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0058R01-CR_CONR_objectId_folderId_definition_update_plus_others</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0060R01-CR_CONR_items_resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0061R01-CR_BulkResponse_merge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0063R01-CR_resolve_A50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0064R01-CR_VanishedObjects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0065R01-CR_tracking_offline_change_in_CR53R04</td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>27 May 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates the following CONRR resolutions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0056R02-CR_Strict_sync_explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0067R03-CR_CONR_Payload</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0068R01-CR_ParentFolder_xml_example_bug_fix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0069-CR_CONR_Unbounded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0071R01-CR_Optional_ParentFolder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0073R01-CR_Box_reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0074R02-CR_CONR_Sync_wording</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0075-CR_CONR_Optional_sync</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0078-CR_CONR_SCR_fix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0080R01-CR_CONR_section_5_1_path_resource</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0081R01-CR_CONR_consistent_resourceURL_def</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0082-CR_CONR_resolutions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0090R01-CR_CONR_uniqueId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0091-CR_CONR_resourceURL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0092-CR_CONR_PayloadPartInfo_link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0094-CR_CreationList</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0096-CR_CONR_NmsEventList_Link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0097-CR_SearchCriterion_Clarification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0098R02-CR_CONR_event_Subscription_Filter_FFS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OMA-ARC-REST-NMS-2014-0099R02-CR_CONR_Conversation_Description</td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>24 Jun 2014</td>
<td>Many sections throughout the document</td>
<td>Incorporates the following CONRR resolutions:</td>
</tr>
<tr>
<td></td>
<td>26 Jun 2014</td>
<td>5.3.3.1, G, 6.1.x, 5.2.x &amp; 5.3.x</td>
<td>Incorporates CR135 and also fixes some typos/editorials.</td>
</tr>
<tr>
<td></td>
<td>02 Jul 2014</td>
<td>6.13.5.1.2, 6.13.5.2.2, 6.13.5.3.2 and 6.13.5.4.2</td>
<td>Incorporates CR136.</td>
</tr>
</tbody>
</table>

Incorporates the following CONRR resolutions:
- OMA-ARC-REST-NMS-2014-0077R01-CR_CONR_SCR_FFS_Resolution
- OMA-TS-REST_NetAPI_NMS-V1.0-20140429-D_CR79R01
- OMA-ARC-REST-NMS-2014-0084R01-CR_CONR_section_5_1_Purpose_Headers
- OMA-ARC-REST-NMS-2014-0086R02-CR_CONR_Multiple_attribute_values
- OMA-ARC-REST-NMS-2014-0088-CR_CONR_Combination
- OMA-ARC-REST-NMS-2014-0095R01-CR_Optimal_parent_folder_for_object
- OMA-ARC-REST-NMS-2014-0100R01-CR_CONR_Resource_name_alignment
- OMA-ARC-REST-NMS-2014-0102R01-CR_CONR_XMLexamples_NMS_specific_errors
- OMA-ARC-REST-NMS-2014-0103R03-CR_CONR_XML_error_examples_re_copy_move
- OMA-ARC-REST-NMS-2014-0105R01-CR_Object_creation_behavior
- OMA-ARC-REST-NMS-2014-0107R01-CR_CONR_object_attributes
- OMA-ARC-REST-NMS-2014-0108-CR_CONR_NmsEventList_link
- OMA-ARC-REST-NMS-2014-0109R02-CR_CONR_Search_field_table
- OMA-ARC-REST-NMS-2014-0110-CR_CONR_Locale
- OMA-ARC-REST-NMS-2014-0111-CR_CONR_anyURI
- OMA-ARC-REST-NMS-2014-0112R02-CR_CONR_Flag_simplification
- OMA-ARC-REST-NMS-2014-0113R01-CR_CONR_Normative_folder_attributes
- OMA-ARC-REST-NMS-2014-0114R03-CR_CONR_Deposit_and_retrieval_examples
- OMA-ARC-REST-NMS-2014-0115R02-CR_CONR_Control_GET_folder_response
- OMA-ARC-REST-NMS-2014-0116R02-CR_EXplain_restartToken
- OMA-ARC-REST-NMS-2014-0117R02-CR_New_auth_scopes
- OMA-ARC-REST-NMS-2014-0118-CR_payload_correction
- OMA-ARC-REST-NMS-2014-0119R02-CR_CONR_Subscription_flows
- OMA-ARC-REST-NMS-2014-0120-CR_CONR_Subscription_Filter_flows
- OMA-ARC-REST-NMS-2014-0121-CR_CONR_BulkCreation_flow
- OMA-ARC-REST-NMS-2014-0123-CR_JSON_Examples
- OMA-ARC-REST-NMS-2014-0124R01-CR_Typos
- OMA-ARC-REST-NMS-2014-0125-CR_FFS_Resolution_Resource_Tree
- OMA-ARC-REST-NMS-2014-0126R01-CR_Appendix_I_formatting
- OMA-ARC-REST-NMS-2014-0127R01-CR_CONR_Deposit_and_retrieval_examples
- OMA-ARC-REST-NMS-2014-0129R01-CR_FFS_Resolution_POSTing_Notification_error
- OMA-ARC-REST-NMS-2014-0131R01-CR_Multipart_clarification
- OMA-ARC-REST-NMS-2014-0132-CR_Figure3_correction
- OMA-ARC-REST-NMS-2014-0133R01-CR_Further_explain_restartToken
- OMA-ARC-REST-NMS-2014-0134-CR_minor_clarification
- OMA-ARC-REST-NMS-2014-0135R03-CR_ConversationSearch
- OMA-ARC-REST-NMS-2014-0136R02-CR_CONR_Filter_folder_response
- OMA-ARC-REST-NMS-2014-0137R02-CR_CONR_FMS_Resolution_POSTing_Notification_error
<table>
<thead>
<tr>
<th>Document Identifier</th>
<th>Date</th>
<th>Sections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate Version:</td>
<td>15 Jul 2014</td>
<td>n/a</td>
<td>Status changed to Candidate by TP&lt;br&gt;TP Ref # OMA-TP-2014-0155-&lt;br&gt;INP_REST_NetAPI_NMS_V1_0_ERP_and_ETR_for_Candidate_A&lt;br&gt;pproval</td>
</tr>
<tr>
<td>REST_NetAPI _NMS-V1_0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST_NetAPI _NMS-V1_0</td>
<td></td>
<td>5.1.4.2.1, 5.1.10, 5.3.2.6, 5.4.6, 6, 6.1.5.1, 6.1.5.4, 6.1.5.5.1, 6.1.5.6.1, 6.4.3.1, 6.8.3.2, 6.8.3.2.2, 6.8.5.3, 6.14.5.2, 6.14.5.2.2, D.1, D.3, D.4, D.5, D.6, D.24, D.27, D.51</td>
<td></td>
</tr>
<tr>
<td>Draft Versions:</td>
<td>28 Feb 2015</td>
<td>2.1, 6.1.3, 6.1.4, 6.1.6, 6.2.4, 6.2.5, 6.3.5, 6.3.6, 6.4.5, 6.5.4, 6.5.5, 6.5.6, 6.6.4, 6.6.5, 6.6.6, 6.7.3, 6.7.4, 6.7.6, 6.8.4, 6.8.6, 6.9.3, 6.9.4, 6.10.3, 6.10.4, 6.10.6, 6.11.4, 6.11.5, 6.12.5, 6.12.6, 6.13.3, 6.13.4, 6.13.6, 6.14.4, 6.14.6, 6.15.3, 6.15.4, 6.15.6, 6.16.3, 6.16.4, 6.16.6, 6.17.4, 6.17.6, 6.18.4, 6.19.3, 6.19.4, 6.19.1</td>
<td>Incorporated CR:&lt;br&gt;- OMA-ARC-REST-NetAPI-2014-0094R01-CR_NMS_Update_HTTP_RFC&lt;br&gt;- Editorial changes</td>
</tr>
<tr>
<td>REST_NetAPI _NMS-V1_0</td>
<td></td>
<td>5.1.10, 6.7.5.1.1, 6.7.5.2.1, 6.13.5.2.1, D.20, D.21, D.44</td>
<td></td>
</tr>
<tr>
<td>Candidate Version:</td>
<td>16 Dec 2014</td>
<td>n/a</td>
<td>Status changed to Candidate by TP&lt;br&gt;TP Ref # OMA-TP-2014-0278-&lt;br&gt;INP_REST_NetAPI_NMS_V1_0_ERP_for_Candidate_reapproval</td>
</tr>
<tr>
<td>REST_NetAPI _NMS-V1_0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Identifier</td>
<td>Date</td>
<td>Sections</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
|                     | 27 Apr 2015| 2.1, 4.1, 5.2, 5.2.1, 5.2.4, 5.3.2.7, 5.3.2.15, 5.3.2.16, 5.3.3.22, 5.3.2.34, 5.3.3.5, 5.4.9, 5.4.10, 6.1.5, 6.1.5.1.1, 6.1.5.2.1, 6.1.5.3.1, 6.1.5.3.2, 6.1.5.4.1, 6.1.5.5.1, 6.1.5.6.1, 6.1.5.7, 6.2.3.1.2, 6.2.3.2.2, 6.2.3.2.3, 6.3.1.1.1, 6.3.1.1, 6.3.1.2, 6.8.3.1.1, 6.8.3.2.1, 6.8.3.2.2, 6.8.5.1.2, 6.8.9.1.1, 6.9.6, 6.10, 6.11, 6.12.5.3.2, 6.12.5.4.2, 6.12.5.5.2, 6.12.5.6.2, 6.13.3.1.1, 6.14.3.1.2, 6.14.4.1.2, 6.14.4.2.2, 6.14.4.3.2, 6.16.3.1, 6.16.3.2.1, 6.16.5.1.2, 6.16.5.2.2, 6.18.5.3.2, 7.2.1, D.1, D.10, D.12, D.17, D.23, D.27, D.29, D.34, D.37, D.38, D.39, D.40, D.44, D.46, D.47, D.48, D.53, D.54, D.56, D.57, D.63, D.67 | Incorporated CRs:  
OMA-ARC-REST-NetAPI-2015-0027-CR_NMS_TS_bulkUpdate_bulkDelete  
OMA-ARC-REST-NetAPI-2015-0029-CR_NMS_bulkUpdate_bulkDelete_JSON  
OMA-ARC-REST-NetAPI-2015-0030R01-CR_NMS_bulkUpdate_bulkDelete_Seq_diagram  
OMA-ARC-REST-NetAPI-2015-0031R01-CR_NMS_POL1030_generalization  
OMA-ARC-REST-NetAPI-2015-0032R01-CR_NMS_object_exampleClarification  
OMA-ARC-REST-NetAPI-2015-0033-CR_NMS_typo  
OMA-ARC-REST-NetAPI-2015-0036-CR_NMS_unsupported_feature_example  
OMA-ARC-REST-NetAPI-2015-0037-CR_NMS_Stray_Content_Lengeth_headers  
OMA-ARC-REST-NetAPI-2015-0038-CR_NMS_Preserve_Payloads_Content_Disposition  
OMA-ARC-REST-NetAPI-2015-0039-CR_NMS_Invalid_dates  
OMA-ARC-REST-NetAPI-2015-0040-CR_NMS_Inconsistent_URL_escaping  
OMA-ARC-REST-NetAPI-2015-0041R01-CR_NMS_MaxEvents_para  
OMA-ARC-REST-NetAPI-2015-0043-CR_error_code_consistant_usage  |
|                     | 30 Apr 2015| 5.3.2.22, 5.3.2.24, 5.4.8, 5.4.9, 5.4.10, D.29              | Editorial changes                                                                             |
|                     | 01 Sep 2015| 6.10.5.2, D.70, D.71                                       | Incorporated CR:  
OMA-ARC-REST-NetAPI-2015-0075-CR_NMS_editorials                                          |
<table>
<thead>
<tr>
<th>Document Identifier</th>
<th>Date</th>
<th>Sections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 Nov 2015</td>
<td>5.3.2.22,</td>
<td>Incorporated CR:</td>
<td>OMA-ARC-REST-NetAPI-2015-0088R01-</td>
</tr>
<tr>
<td></td>
<td>5.3.2.27,</td>
<td>CR_NMS_TS_on_demand_attr_in_Notification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.3.2.29</td>
<td>Editorial changes</td>
<td></td>
</tr>
<tr>
<td>03 Dec 2015</td>
<td>5.1.4.4,</td>
<td>Incorporated CR:</td>
<td>OMA-ARC-REST-NetAPI-2015-0096R01-</td>
</tr>
<tr>
<td></td>
<td>5.1.12.1,</td>
<td>CR_NMS_TS_further_clarification_on_CR88</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.3.2.22,</td>
<td>Editorial changes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.4.9,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.1.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.2.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.3.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.4.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.4.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.5.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.6.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.1.5.7.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.2.3.2.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.2.3.3.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.3.4.1.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.3.4.1.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.7.5.1.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.7.5.2.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.7.5.3.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.9.5.1.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.10.5.1.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.10.5.2.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.21.5.1.1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A.2,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.21,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.22,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.23,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 Jan 2016</td>
<td>6.2.3.2.2,</td>
<td>Incorporated CR:</td>
<td>OMA-ARC-REST-NetAPI-2015-0098-</td>
</tr>
<tr>
<td></td>
<td>6.7.5.4.1,</td>
<td>CR_NMS_TS_fix_example_validation_issue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Candidate Version:</td>
<td>REST_NetAPI_NMS-V1_0</td>
<td>n/a</td>
<td>Status changed to Candidate by TP</td>
</tr>
<tr>
<td>09 Feb 2016</td>
<td></td>
<td></td>
<td>TP Ref # OMA-TP-2016-0031-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INP_REST_NetAPI_NMS_V1_0_ERP_for_Candidate_re_approval</td>
</tr>
</tbody>
</table>
Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

## B.1 SCR for REST.NMS Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-SUPPORT-S-001-M</td>
<td>Support for the RESTful NMS API</td>
<td>5, 6</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-SUPPORT-S-002-M</td>
<td>Support for the XML request &amp; response format</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-SUPPORT-S-003-M</td>
<td>Support for the JSON request &amp; response format</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.1 SCR for REST.NMS.Objects Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-OBJECTS-S-001-M</td>
<td>Support for object creation (object upload into NMS)</td>
<td>6.1</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-OBJECTS-S-002-M</td>
<td>Upload (create) a new object into an identified folder of NMS – POST</td>
<td>6.1.5</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.2 SCR for REST.NMS.AObject Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AOBJECT-S-001-M</td>
<td>Support for managing individual stored object</td>
<td>6.2</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-S-002-M</td>
<td>Retrieve metadata of an object – GET</td>
<td>6.2.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-S-003-M</td>
<td>Delete an object – DELETE</td>
<td>6.2.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.3 SCR for REST.NMS.AObject.Flags Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AOBJECT-FLAGS-S-001-M</td>
<td>Support for managing flags associated with an object</td>
<td>6.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-FLAGS-S-002-M</td>
<td>Retrieve flag list of an object – GET</td>
<td>6.3.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-FLAGS-S-003-O</td>
<td>Add a new flag to the flag list of an object – PUT</td>
<td>6.3.3.2</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-FLAGS-S-004-M</td>
<td>Update the entire flag list of an object – PUT</td>
<td>6.3.3.2</td>
<td></td>
</tr>
</tbody>
</table>
### B.1.4 SCR for REST.NMS.AObject.IndFlag Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AOBJECT-INDFLAG-S-001-M</td>
<td>Support for managing an individual flag associated with an object</td>
<td></td>
<td>6.4</td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-INDFLAG-S-002-M</td>
<td>Test for existence of a flag in a given object – GET</td>
<td></td>
<td>6.4.3</td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-INDFLAG-S-003-M</td>
<td>Add a new flag to the flag list of an object – PUT</td>
<td></td>
<td>6.4.4</td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-INDFLAG-S-004-M</td>
<td>Remove a flag from flag list of an object – DELETE</td>
<td></td>
<td>6.4.6</td>
</tr>
</tbody>
</table>

### B.1.5 SCR for REST.NMS.AObject.Payload Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AOBJECT-PAYLOAD-S-001-M</td>
<td>Support for retrieving object’s entire payload</td>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-PAYLOAD-S-002-M</td>
<td>Retrieve the entire payload of a given object – GET</td>
<td></td>
<td>6.5.3</td>
</tr>
</tbody>
</table>

### B.1.6 SCR for REST.NMS.AObject.PayloadPart Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AOBJECT-PAYLOADPART-S-001-M</td>
<td>Support for retrieving an individual payload part</td>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td>REST-NMS-AOBJECT-PAYLOADPART-S-002-M</td>
<td>Retrieve an individual payload part of a given object – GET</td>
<td></td>
<td>6.6.3</td>
</tr>
</tbody>
</table>

### B.1.7 SCR for REST.NMS.Objects.Search Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-OBJECTS-SEARCH-S-001-M</td>
<td>Support for searching for objects meeting certain criteria</td>
<td></td>
<td>6.7</td>
</tr>
<tr>
<td>REST-NMS-OBJECTS-SEARCH-S-002-M</td>
<td>Retrieve information about objects meeting certain criteria – POST</td>
<td></td>
<td>6.7.5</td>
</tr>
</tbody>
</table>
B.1.8 SCR for REST.NMS.Objects.PathToId Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-OBJECTS-PATHTOID-S-001-M</td>
<td>Support for looking up object(s) resource URL(s) using its (their) location/path</td>
<td>6.8</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-OBJECTS-PATHTOID-S-002-M</td>
<td>Retrieve an object’s Id (resource URL) using its location/path – GET</td>
<td>6.8.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-OBJECTS-PATHTOID-S-003-M</td>
<td>Retrieve a list of object Ids (resource URLs) using their location/path – POST</td>
<td>6.8.5</td>
<td></td>
</tr>
</tbody>
</table>

B.1.9 SCR for REST.NMS.Objects.bulkCreation Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-OBJECTS-bulkCreation-S-001-O</td>
<td>Support for bulk upload of objects</td>
<td>6.9</td>
<td></td>
</tr>
</tbody>
</table>

B.1.10 SCR for REST.NMS.Folders Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERS-S-001-M</td>
<td>Support for folder creation</td>
<td>6.12</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-S-002-M</td>
<td>Create a new folder under an identified parent folder in NMS – POST</td>
<td>6.12.5</td>
<td></td>
</tr>
</tbody>
</table>

B.1.11 SCR for REST.NMS.AFolders Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-AFOLDERS-S-001-M</td>
<td>Support for folder creation</td>
<td>6.13</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AFOLDERS-S-002-M</td>
<td>Retrieve information about a folder – GET</td>
<td>6.13.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-AFOLDERS-S-003-M</td>
<td>Delete a folder – DELETE</td>
<td>6.13.6</td>
<td></td>
</tr>
</tbody>
</table>

B.1.12 SCR for REST.NMS. FolderName Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERNAMESPACE-S-001-M</td>
<td>Support for managing individual folder data</td>
<td>6.14</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-FOLDERNAMESPACE-S-003-M</td>
<td>Change folder name – PUT</td>
<td>6.14.4</td>
<td></td>
</tr>
</tbody>
</table>

B.1.13 SCR for REST.NMS.Folders.Search Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERS-SEARCH-S-001-M</td>
<td>Support search operation for folders</td>
<td>6.15</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-SEARCH-S-002-M</td>
<td>Search for the root folder(s) – POST</td>
<td>6.15.5</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Function</td>
<td>Reference</td>
<td>Requirement</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-SEARCH-S-003-O</td>
<td>Support additional search operations other than searching for root – POST</td>
<td>6.15.5</td>
<td></td>
</tr>
</tbody>
</table>
### B.1.14 SCR for REST.NMS.Folders.PathToId Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERS-PATHTOID-S-001-M</td>
<td>Support for looking up folder(s) resource URL(s) using its (their) location/path</td>
<td>6.16</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-PATHTOID-S-002-M</td>
<td>Retrieve an folder’s Id (resource URL) using its location/path – GET</td>
<td>6.16.3</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-PATHTOID-S-003-M</td>
<td>Retrieve a list of folders Ids (resource URLs) using their location/path – POST</td>
<td>6.16.5</td>
</tr>
</tbody>
</table>

### B.1.15 SCR for REST.NMS.Folders.Copy Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERS-COPY-S-001-M</td>
<td>Support for copying objects and/or folders to a target folder</td>
<td>6.17</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-COPY-S-002-M</td>
<td>Copy objects into a target folder – POST</td>
<td>6.17.5</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-COPY-S-003-M</td>
<td>Copy a folder containing other folders and objects into a target folder (recursive copy) – POST</td>
<td>6.17.5</td>
</tr>
</tbody>
</table>

### B.1.16 SCR for REST.NMS.Folders.Move Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-FOLDERS-MOVE-S-001-M</td>
<td>Support for moving objects and/or folders to a target folder</td>
<td>6.18</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-MOVE-S-002-M</td>
<td>Move objects into a target folder – POST</td>
<td>6.18.5</td>
</tr>
<tr>
<td>REST-NMS-FOLDERS-MOVE-S-003-M</td>
<td>Move a folder containing other folders and objects into a target folder (recursive move) – POST</td>
<td>6.18.5</td>
</tr>
</tbody>
</table>

### B.1.17 SCR for REST.NMS.Subscriptions Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-SUBSCR-S-001-O</td>
<td>Support for subscriptions to NMS event notifications as well as synchronization with NMS</td>
<td>6.19</td>
</tr>
<tr>
<td>REST-NMS-SUBSCR-S-002-O</td>
<td>Read the list of active subscriptions to NMS event notifications – GET</td>
<td>6.19.3</td>
</tr>
<tr>
<td>REST-NMS-SUBSCR-S-003-O</td>
<td>Create new subscription to NMS event notifications – POST</td>
<td>6.19.5</td>
</tr>
<tr>
<td>Item</td>
<td>Function</td>
<td>Reference</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>REST-NMS-SUBSCR-S-004-O</td>
<td>Create new subscription to NMS event notifications while it syncs the local storage with NMS – POST</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-SUBSCR-S-005-O</td>
<td>Create new subscription to NMS event notifications with filter setup to receive only certain event (e.g. SMS’s) – POST</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.18 SCR for REST.NMS.IndSubscription Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-INDSUBSCR-S-001-O</td>
<td>Support for access to an individual subscription to NMS event notifications</td>
<td>6.20</td>
<td>REST-NMS-INDSUBSCR-S-002-O AND REST-NMS-INDSUBSCR-S-003-O AND REST-NMS-INDSUBSCR-S-004-O</td>
</tr>
<tr>
<td>REST-NMS-INDSUBSCR-S-002-O</td>
<td>Read an individual subscription to NMS event notifications – GET</td>
<td>6.20.3</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-INDSUBSCR-S-003-O</td>
<td>Update an individual subscription to NMS event notifications – POST</td>
<td>6.20.5</td>
<td></td>
</tr>
<tr>
<td>REST-NMS-INDSUBSCR-S-004-O</td>
<td>Cancel subscription and stop corresponding notifications – DELETE</td>
<td>6.20.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.19 SCR for REST.NMS.Notifications Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-NMS-NOTIF-S-001-O</td>
<td>Support for notifications about NMS events</td>
<td>6.21</td>
<td>REST-NMS-NOTIF-S-002-O</td>
</tr>
<tr>
<td>REST-NMS-NOTIF-S-002-O</td>
<td>Notifications about NMS changes – GET</td>
<td>6.21.5</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C.    Application/x-www-form-urlencoded Request Format for POST Operations  

(Normative)

This specification does not define any API request based on application/x-www-form-urlencoded MIME type.
Appendix D. JSON examples (Informative)

JSON (JavaScript Object Notation) is a Light-weight, text-based, language-independent data interchange format. It provides a simple means to represent basic name-value pairs, arrays and objects. JSON is relatively trivial to parse and evaluate using standard JavaScript libraries, and hence is suited for REST invocations from browsers or other processors with JavaScript engines. Further information on JSON can be found at [RFC7159].

The following examples show the request and response for various operations using the JSON data format. The examples follow the XML to JSON serialization rules in [REST_NetAPI_Common]. A JSON response can be obtained by using the content type negotiation mechanism specified in [REST_NetAPI_Common].

For full details on the operations themselves please refer to the section number indicated.

D.1 Object creation by parentFolder, response with a location of created resource (section 6.1.5.1)

Request:

```
POST /exampleAPI/nms/v1/myStore.tel%3A%2B195855550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{ "object": {
  "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld123",
  "attributes": { "attribute": [] },
  "flags": {
    "flag": [
      "\Seen",
      "\Flagged"
    ]
  }
}

--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep-=--"
Content-Disposition: form-data; name="attachments"

--=-sep-=--
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"
See attached photo

--=-sep-=--
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"
```


D.2 Object creation by parentFolderPath, response with a location of the created resource while the non-existent parent folder is auto-created (section 6.1.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="==outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--==outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{ "object": { 
  "parentFolderPath": "/main/myBackups/Football/SavedPictures",
  "attributes": { "attribute": [] },
  "flags": { 
    "flag": [
      "\Seen",
      "\Flagged"
    ]
  }
}
--==outer123456==
Content-Type: multipart/mixed; boundary="--=-sep=---"
Content-Disposition: form-data; name="attachments"

Response:

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123
Content-Type: application/json
Content-Length: nnnn

{ "reference": {
  "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123",
  "path": "/main/conversation5/obj123"
}
}
D.3 Object creation by parentFolderPath, response creation failure due to prohibited location (i.e. requested parent folder) (section 6.1.5.3)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn

MIME-Version: 1.0

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{ "object": { "parentFolderPath": "/Default", "attributes": [ { "attribute": []}, "flags": [ { "flag": [ \"\Seen\", \"\Flagged\" ] } ] } 

Response:

HTTP/1.1 201 Created
Date: Fri, 23 May 2014 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/obj4141
Content-Type: application/json
Content-Length: nnnn

{ "reference": { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/obj4141", "path": "main/myBackups/Football/SavedPictures" } }
D.4 Object creation without parent folder, response with a location of created resource and full object (section 6.1.5.4)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="=outer123456==
Content-Length: nnnn
MIME-Version: 1.0

--=outer123456==
Content-Type: application/json

Response:

HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 20:51:51 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
    "policyException": {
        "messageId": "POL1031",
        "text": "Attempt to create objects or folders under %1 is prohibited",
        "variables": ["/Default"]
    }
}}
Content-Disposition: form-data; name="root-fields"

{"object": {
"attributes": {},
"flags": {
"flag": [
"\Seen",
"\Flagged"
]
}
}}

Response:

HTTP/1.1 201 Created
Date: Tue, 20 Apr 2014 09:43:02 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj741
Content-Type: application/json
Content-Length: nnnn

{"object": {
"parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId185",
"attributes": {
"attribute": []
},
"flags": {
"flag": [
"\Seen",
"\Flagged"
]
}
}}
D.5 Creation of multipart object with presentation part, response with a location of created resource (section 6.1.5.5)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Type: multipart/related; start="28186490"; boundary="--=-sep-=--"

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{"object":
  "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123",
  "attributes": {}},

--=-sep=-
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable
Content-Disposition: inline; filename="text.txt"

See attached photo.

Response:

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542
Content-Type: application/json
Content-Length: nnnn

{ "reference": {
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542",
    "path": "main/conversation5/obj542"
}
}
D.6 Creation of simple text object, response with a location of created resource (section 6.1.5.6)

Request:

```plaintext
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456=="
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{ "object": {
    "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123",
    "attributes": { "attribute": [] },
    "flags": {
        "flag": ["\Seen",
                  "\Flagged"
        ]
    }
}

--===============outer123456==
Content-Type: text/plain
Content-Disposition: form-data; name="attachments"
The quick brown fox rushed to Montreal.

--===============outer123456==--
```

Response:

```plaintext
HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543
Content-Type: application/json
Content-Length: nnnn

{ "reference": {
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543",
    "path": "/main/conversation5/obj543"
}
```

D.7 Creation of simple object with no payload, response with a location of created resource (section 6.1.5.7)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456==";
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"
Content-Length: nnnn

{ "object": {
 "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123",
 "attributes": {
 "attribute": [
 { "name": "Subject",
 "value": [ "A historical search engine" ]
 },
 { "name": "URL",
 "value": [ "http://altavista.digital.com" ]
 }
 ],
 "flags": { "flag": [ "\Flagged" ] }
 }
},
--===============outer123456==

Response:

HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:52:58 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj547
Content-Type: application/json
Content-Length: nnnn

{ "reference": {
 "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj547",
 "path": "/main/conversation5/obj547"
 }
}
D.8 Retrieve information about an object (section 6.2.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{
  "object": {
    "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId567",
    "attributes": {
      "attribute": [
        { "name": "Message-Context", "value": [ "multimedia-message" ] },
        { "name": "Direction", "value": [ "In" ] },
        { "name": "From", "value": [ "tel:+19585550100" ] },
        { "name": "To", "value": [
          "tel:+19585550210",
          "tel:+19585550320"
        ] },
        { "name": "Date", "value": [ "2013-11-12T08:30:10Z" ] },
        { "name": "Subject", "value": [ "Weekend Trip to Seattle" ] },
        { "name": "Content-Type", "value": [ "multipart/mixed" ] }
      ]
    },
    "flags": {
      "flag": [
        "\Seen",
        "\Answered"
      ]
    },
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999",
    "path": "/main/conversation5/old999",
    "payloadURL": "/nms/v1/myStore/tel%3A%2B19585550100/objects/old999/payload"
  }
}
```
D.9 Retrieve information about multipart object (section 6.2.3.2)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "object": {
    "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId123",
    "attributes": {
        "attribute": [
            { "name": "Content-Type", "value": [ "multipart/related; start="28186490" ]
        ]
    },
    "flags": {
        "flag": [
            "\Seen",
            "\Flagged"
        ]
    },
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542",
    "path": "/main/conversation5/obj542",
    "payloadURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj542/payload",
    "payloadPart": [
        { "contentType": "text/plain",
          "contentDisposition": " inline; filename="text.txt"",
          "contentId": "28184720",
          "size": 48,
          "href": "/nms/v1/myStore/tel%3A%2B19585550100/objects/old999/payloadParts/blob123"
        }
    ],
    "lastModSeq": 48
}
D.10 Retrieve information about object with inline content (section 6.2.3.3)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 04 Oct 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{
"object": {
"parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId123",
"attributes": {
"attribute": [
{"name": "TextContent",
"value": ["The quick brown fox rushed to Montreal." ]
}
],
"flags": {
"flag": [
"\Seen",
"\Flagged"
]
},
"resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj543",
"path": "/main/conversation5/obj543",
"lastModSeq": 48
}
D.11 Delete an object, response with “204 No Content” (section 6.2.6.1)

Request:

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId999 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 05:55:59 GMT

D.12 Retrieve flags associated with an object (section 6.3.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId999/flags HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2013 03:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{
  "flagList": {
    "flag": [
      "\Seen",
      "\Flagged"
    ],
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId999/flags"
  }
}
```

D.13 Retrieve flags associated with an object, failure due to an invalid object (section 6.3.3.2)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/objectNotThere/flags HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document
Response:

HTTP/1.1 404 Not Found
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{
"requestError": {
  "serviceException": {
    "messageId": "SVC0004",
    "text": "No valid addresses provided in message part %1",
    "variables": ["/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objNotThere/flags"]
  }
}
}
```

D.14 Add a flag to flaglist of an object (section 6.3.4.1)

Request:

PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

```json
{
  "flagList": {
    "flag": [
      "\Seen",
      "\Flagged",
      "\Answered"
    ],
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags"
  }
}
```

Response:

HTTP/1.1 200 OK
Date: Sat, 05 Oct 2013 03:58:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{
  "flagList": {
    "flag": [
      "\Seen",
      "\Flagged",
      "\Answered"
    ],
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags"
  }
}
```
D.15 Read an existing individual flag (section 6.4.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old999/flags/%5CSeen HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

Response:

HTTP/1.1 204 No Content
Date: Sat, 05 Oct 2013 03:55:00 GMT

D.16 Read a non-existing individual flag using acr:auth (section 6.4.3.2)

Request:

GET /exampleAPI/nms/v1/myStore/acr%3Aauth/objects/old999/flags/%5CAnswered HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

Response:

HTTP/1.1 404 Not Found
Date: Sat, 05 Oct 2013 03:55:00 GMT
Content-Type: application/json
Content-Length: nnnn

{ "empty": null }

D.17 Add “\Answered” flag to flaglist of an object (section 6.4.4.1)

Request:

PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old999/flags/%5CAnswered HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{ "empty": null }
Response:

HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CAnswered
Date: Sat, 05 Oct 2013 03:58:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "empty": null }

D.18 Delete “\Seen” flag from flaglist of an object (section 6.4.6.1)

Request:

DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId999/flags/%5CSeen HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 05:55:59 GMT
D.19 Read payload of the stored object via an external reference (section 6.5.3.1)

Request:

```
GET /exampleAPI/storage/100/blob456 HTTP/1.1
Accept: image/gif, image/png, image/jpeg, text/html, application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 03:52:01 GMT
Content-Length: nnnn
Content-Type: multipart/mixed; boundary="--=-sep-=--"

----=-sep-=--
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"
Are you coming to the football today? See attached photo

----=-sep-=--
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"
GIF89a...binary image data...

----=-sep-=--
```

D.20 Read an object payload part via the NMS resource tree (section 6.6.3.1)

Request:

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj123/payloadParts/part123 HTTP/1.1
Accept: image/gif, image/png, image/jpeg, text/html, application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 20 Aug 2013 03:51:59 GMT
Content-Length: nnnn
Content-Type: image/gif

...GIF89a...binary image data
```
D.21 Search for objects with certain criteria (section 6.7.5.1)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn
{
  "selectionCriteria": {
    "maxEntries": 3,
    "searchCriteria": {
      "criterion": [
        { "type": "Attribute",
          "name": "Message-Context",
          "value": "pager-message"
        },
        { "type": "Attribute",
          "name": "Direction",
          "value": "In"
        },
        { "type": "Attribute",
          "name": "From",
          "value": "tel:+19585550100"
        },
        { "type": "Date",
          "value": "minDate=2013-11-11T09:30:10Z"
        }
      ],
      "operator": "And"
    },
    "sortCriteria": {
      "criterion": [
        { "type": "Date",
          "order": "Ascending"
        }
      ]
    }
  }
}
```
Response:

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```
{
  "objectList": {
    "object": {
      "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123",
      "attributes": {
        "attribute": {
          "name": "Message-Context",
          "value": [ "pager-message" ]
        },
        "name": "From",
        "value": [ "tel:+195855550100" ]
      },
      "name": "Date",
      "value": [ "2013-11-12T08:30:10Z" ]
    },
    "name": "Direction",
    "value": [ "In" ]
  },
  "flags": {
    "flag": [ "\Seen",
      "Answered"
    ],
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId998/flags"
  },
  "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId998",
  "path": "/main/f81d4fae-7dec-11d0-a765-00a0c91e6b6f/old998",
  "payloadURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/oId998/payload",
  "lastModSeq": 100
  },
  "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fld123",
  "attributes": {
    "attribute": {
      "name": "Message-Context",
      "value": [ "pager-message" ]
    },
    "name": "From",
    "value": [ "tel:+195855550100" ]
  },
  "name": "Date",
  "value": [ "2013-11-12T09:12:00Z" ]
},
```


```json
{
    "cursor": "cursor111"
}
```
D.22 Retrieve the remaining search response list (section 6.7.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": {
    "fromCursor": "cursor111",
    "maxEntries": 3,
    "searchCriteria": {
        "criterion": [
            { "type": "Attribute",
              "name": "Message-Context",
              "value": "pager-message"
            },
            { "type": "Attribute",
              "name": "Direction",
              "value": "In"
            },
            { "type": "Attribute",
              "name": "From",
              "value": "tel:+19585550100"
            },
            { "type": "Date",
              "value": "minDate=2013-11-11T09:30:10Z"
            }
        ],
        "operator": "And"
    },
    "sortCriteria": {
        "criterion": [
            { "type": "Date",
              "order": "Ascending"
            }
        ]
    }
}

Response:

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "objectList": [
   "object": [
      { "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId223",
         "attributes": {
            "attribute": [
               { "name": "Message-Context",
                 "value": ["pager-message"]
               },
               { "name": "From",
                 "value": ["tel:+19585550100"]
               },
               { "name": "Date",
                 "value": ["2013-11-14T06:20:10Z"]
               },
               { "name": "Direction",
                 "value": ["In"]
               },
               { "name": "Content-Type",
                 "value": ["text/plain"]
               }
            ]
         }]
   }]
   "flags": {
            "flag": [
               "\Seen",
               "\Answered"
            ],
            "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId1005/flags"
   },
   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId1005",
   "path": "/main/StaffMeeting/oId1005",
   "payloadURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId1005/payload",
   "lastModSeq": 180
   }]
]
D.23 Search for a substring in all searchable text attributes and bodies (section 6.7.5.3)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": {
  "maxEntries": 3,
  "searchCriteria": {
    "criterion": [
      { "type": "AllTextAttributes",
        "value": "Football"
      }
    ]
  }
}
```
Response:

HTTP/1.1 200 OK
Date: Fri, 07 Jun 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "objectList": {
    "object": [
      {
        "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId8686",
        "attributes": {
          "attribute": [
            {
              "name": "Message-Context",
              "value": ["text-message"]
            },
            {
              "name": "From",
              "value": ["tel:+19587236564"]
            },
            {
              "name": "Subject",
              "value": ["R U coming to football game today"]
            },
            {
              "name": "Date",
              "value": ["2013-12-02T08:30:10Z"]
            },
            {
              "name": "Direction",
              "value": ["In"]
            },
            {
              "name": "Content-Type",
              "value": ["multipart/mixed"]
            }
          ]
        },
        "flags": {
          "flag": [
            "\Seen",
            "\Answered"
          ]
        },
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId123",
        "path": "main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/old123",
        "payloadURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId123/payload",
        "lastModSeq": 7688
      }
    ]
  }
}
D.24 Search for CreatedObjects not supported (section 6.7.5.4)

Request:

```plaintext
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{  "selectionCriteria": {  "maxEntries": 3,
      "searchCriteria": {  "criterion": [
          {  "type": "CreatedObjects",
              "value": null
          }
      ],
      "sortCriteria": {  "criterion": [
          {  "type": "Date",
              "order": "Ascending"
          }
      ]
  }
}

Response:

HTTP/1.1 403 Forbidden
Date: Fri, 14 Mar 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{  "requestError": {  "policyException": {  "messageId": "POL2006",
          "text": "Requested feature %1 is not available",
          "variables": [  "CreatedObjects" ]
      }
  }
}
D.25 Retrieve object’s resource URL based on its path (section 6.8.3.1)

Request:

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId?path=/main/conversation5/obj12345 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "reference": {
    "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj12345",
    "path": "/main/conversation5/obj12345"
  }
}
```

D.26 Retrieve object’s resource URL based on its path, failure due to a malformed path (section 6.8.3.2)

Request:

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId?path=/main/conversation5/obj12345 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

Response:

```
HTTP/1.1 400 Bad Request
Date: Wed, 24 Jul 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "requestError": {
    "serviceException": {
      "messageId": "SVC0002",
      "text": "Invalid input value for message part %1",
      "variables": [ "/main/conversation5/obj12345" ]
    }
  }
}
```
D.27 Retrieve list of objects’ resource URLs based on their paths (section 6.8.5.1)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "pathList": {
  "path": [
    "/main/f81d4fiae-7dec-11d0-a765-00a0c91e6bf6/old111",
    "/main/conversation5/old221",
    "/main/conversation5/old222",
  ]
}
```

Response:

```
HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old111",
        "path": "/main/f81d4fiae-7dec-11d0-a765-00a0c91e6bf6/old111"
      }
    },
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old221",
        "path": "/main/conversation5/old221"
      }
    },
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old222",
        "path": "/main/conversation5/old222"
      }
    }
  ]
}
```
D.28 Retrieve list of objects’ resource URLs based on their paths, at least one path to a non-existing object (section 6.8.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "pathList": {
   "path": [
      "/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111",
      "/cleanups/conversation5/oId221"
   ]
}

Response:

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 12:09:09 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
   "response": [
      { "code": 200,
        "reason": "OK",
        "success": {
           "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId111",
           "path": "/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6/oId111"
        }
      },
      { "code": 400,
        "reason": "Bad Request",
        "failure": {
           "serviceException": {
              "messageId": "SVC0002",
              "text": "Invalid input value for message part %1",
              "variables": [ "/cleanups/conversation5/oId221" ]
           }
        }
      }
   ]
}

D.29 Bulk creation (section 6.9.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkCreation HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: multipart/form-data; boundary="===============outer123456==";
Content-Length: nnnn
MIME-Version: 1.0

--===============outer123456==
Content-Type: application/json
Content-Disposition: form-data; name="root-fields"

{ "objectList": {
  "object": {
    "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld123",
    "attributes": { "attribute": [] },
    "flags": {
      "flag": [
        "\Seen",
        "\Flagged"
      ]
    }
  },
  { "parentFolderPath": "/Pictures",
    "attributes": { "attribute": [] },
    "flags": { "flag": [ \Seen ] }
  },
  { "parentFolderPath": "/Pictures",
    "attributes": { "attribute": [] },
    "flags": { "flag": [ \Seen, \Flagged ] }
  }
}

--===============outer123456==
Content-Type: multipart/mixed; boundary="--=-sep-=--"
Content-Disposition: form-data; name="attachments"

-----=-sep=-
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

See attached photo

-----=-sep=-
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=--=
--=outer123456=
Content-Type: multipart/mixed; boundary="--=
Content-Disposition: form-data; name="attachments"

-----=
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

Photo from trip to Vancouver

-----=
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=--=
--=outer123456=
Content-Type: multipart/mixed; boundary="--=
Content-Disposition: form-data; name="attachments"

-----=
Content-Type: text/plain
Content-Disposition: attachment; filename="body.txt"

Photo from Sorrento Meeting

-----=
Content-Type: image/gif
Content-Disposition: attachment; filename="picture.gif"

GIF89a...binary image data...

-----=--=
--=outer123456=--
Response:

```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 12:09:09 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {  
"response": [  
  { "code": 201,  
    "reason": "Created",  
    "success": {  
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj412",  
      "path": "/SpringBreak2014/obj412"  
    }  
  },  
  { "code": 403,  
    "reason": "Forbidden",  
    "failure": {  
      "policyException": {  
        "messageId": "POL0001",  
        "text": "A policy error occurred. Error code is %1",  
        "variables": [ "E42" ]  
      }  
    }  
  },  
  { "code": 201,  
    "reason": "Created",  
    "success": {  
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj413",  
      "path": "/Pictures/obj413"  
    }  
  }  
]  
}
```
D.30 Bulk update a requested list of objects (section 6.10.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkUpdate HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

```json
{  "bulkUpdate": {  
    "objects": [  
      { "objectReference": [  
        { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001" },
        { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002" }
      ]  
    },
    "operation": "AddFlag",
    "flags": { "flag": [ "\Seen" ] }
  }
}
```

Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{  "bulkResponseList": {
  "response": [  
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001",
        "path": "/main/SummerHolidayPlan/objId001"
      }
    },
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002",
        "path": "/main/SummerHolidayPlan/objId002"
      }
    }
  ]
}
```
D.31 Bulk update objects meeting certain criteria (section 6.10.5.2)

Request:

```json
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkUpdate HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "bulkUpdate": {
  "selectionCriteria": {
    "maxEntries": 3,
    "searchCriteria": {
      "criterion": [
        { "type": "Attribute",
          "name": "Direction",
          "value": "In"
        },
        { "type": "Attribute",
          "name": "From",
          "value": "tel:+19585550100"
        },
        { "type": "Date",
          "value": "minDate=2014-11-11T09:30:10Z"
        }
      ],
      "operator": "And"
    }
  },
  "operation": "AddFlag",
  "flags": { "flag": [ "\Seen" ] }
}
```

Response:

```json
HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId008",
        "path": "/main/SummerHolidayPlan/objId008"
      }
    }
  ]
}
```
D.32 Bulk delete a given list of objects (section 6.11.5.1)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": { "response": [ { "code": 204, "reason": "No Content", "success": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001" } }, { "code": 204, "reason": "No Content", "success": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002" } ] }
```
D.33 Bulk delete objects meeting certain criteria (section 6.11.5.2)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{  "bulkDelete": {  
   "selectionCriteria": {  
      "maxEntries": 5,
      "searchCriteria": {  
         "criterion": [  
            {   "type": "Attribute",
                "name": "From",
                "value": "tel:+19585550100"
            },
            {   "type": "Date",
                "value": "maxDate=2014-12-30T09:30:10Z"
            }
         ]
      }
   }
}
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{  "bulkResponseList": {  
   "response": [  
      {   "code": 204,
          "reason": "No Content",
          "success": {   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId021"}
      },
      {   "code": 204,
          "reason": "No Content",
          "success": {   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId032"}
      },
      {   "code": 204,
          "reason": "No Content",
          "success": {   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId039" }
      },
      {   "code": 204,
          "reason": "No Content",
          "success": {   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId012" }
      },
      {   "code": 204,
```
D.34 Retrieve the remaining bulk delete response list (section 6.11.5.3)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/operations/bulkDelete HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "bulkDelete": {
  "selectionCriteria": {
    "fromCursor": "cursor944",
    "maxEntries": 5,
    "searchCriteria": {
      "criterion": [
        { "type": "Attribute",
          "name": "From",
          "value": "tel:+19585550100"
        },
        { "type": "Date",
          "value": "maxDate=2014-12-30T09:30:10Z"
        }
      ]
    }
  }
}

Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:52:01 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [
    { "code": 204,
      "reason": "No Content",
      "success": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId081" }
    }
  ]
}
D.35 Folder creation by parentFolder path, response with a location of created resource (section 6.12.5.1)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
    "parentFolderPath": "/main",
    "attributes": [ "attribute": []
    },
    "name": "BoardMeeting"
}
```

Response:

```
HTTP/1.1 201 Created
Date: Tue, 20 Aug 2013 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456
Content-Type: application/json
Content-Length: nnnn

{ "reference": { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456" } }
```
D.36  Folder creation by parentFolder path, response with a copy of created resource (section 6.12.5.2)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
    "parentFolderPath": "/main",
    "attributes": { "attribute": [] },
    "name": "NMSdiscussion"
}
```

Response:

```
HTTP/1.1 201 Created
Date: Tue, 15 Apr 2014 02:51:59 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559
Content-Type: application/json
Content-Length: nnnn

{ "folder": {
    "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554",
    "attributes": {
        "attribute": [
            { "name": "Date",
              "value": [ "2014-04-15T02:51:55Z" ]
            },
            { "name": "Name",
              "value": [ "NMSdiscussion" ]
            }
        ]
    },
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559",
    "path": "/main/NMSdiscussion",
    "name": "NMSdiscussion",
    "lastModSeq": 555
}
```

D.37 Folder creation by parentFolder path, response creation failure due to an invalid folder path (section 6.12.5.3)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
   "parentFolderPath": "/main/myBackups/Football",
   "attributes": { "attribute": [] },
   "name": "WorldCup2014"
}
```

Response:

```
HTTP/1.1 400 Bad Request
Date: Wed, 20 Nov 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
   "serviceException": {
      "messageId": "SVC0002",
      "text": "Invalid input value for message part %1",
      "variables": [ "/main/myBackups/Football" ]
   }
}
```

D.38 Folder creation by parentFolder resourceURL, response with a copy of created resource (section 6.12.5.4)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
   "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559",
   "attributes": { "attribute": [] },
   "name": "SorrentoMeeting"
}
```
Response:

HTTP/1.1 201 Created
Date: Mon, 20 Jan 2014 11:20:13 GMT
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld560
Content-Type: application/json
Content-Length: nnnn

{ "folder": {
    "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld559",
    "attributes": {
        "attribute": [
            { "name": "Date",
              "value": [ "2014-01-20T11:20:10Z" ]
            },
            { "name": "Name",
              "value": [ "SorrentoMeeting" ]
            }
        ],

        "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld560",
        "path": "/main/NMSdiscussion/SorrentoMeeting",
        "name": "SorrentoMeeting",
        "lastModSeq": 977
    }
}

D.39 Folder creation failure due to request missing the parent folder element (section 6.12.5.5)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
    "attributes": { "attribute": [] },
    "name": "VegasMeeting"
}
}
Response:

HTTP/1.1 400 Bad Request
Date: Thu, 20 Nov 2014 20:51:51 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
   "serviceException": {
      "messageId": "SVC0002",
      "text": "Invalid input value for message part %1",
      "variables": [ "Parent folder (missing)"
   }
}

D.40 Folder creation by parentFolder path, response creation failure due to prohibited location (i.e. requested parent folder) (section 6.12.5.6)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders HTTP/1.1
Accept: application/json
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Content-Type: application/json
Content-Length: nnnn
MIME-Version: 1.0

{ "folder": {
   "parentFolderPath": "/Default",
   "attributes": [ "attribute" ],
   "name": "MySavedPictures"
}

Response:

HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 21:01:11 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
   "policyException": {
      "messageId": "POL1031",
      "text": "Attempt to create objects or folders under %1 is prohibited",
      "variables": [ "/Default"
   }
}

D.41 Retrieve information about a folder (section 6.13.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608?path=Yes&attrFilter=SubtreeMsgCount&attrFilter=SubtreeSize HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{  "folder": {  "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554",
    "attributes": {
      "attribute": [
        { "name": "Conversation-ID",
          "value": [ "f81d4fae-7dec-11d0-a765-00a0c91e6bf6" ]
        },
        { "name": "Contribution-ID",
          "value": [ "abcdef-1234-5678-90ab-cdef01234567" ]
        },
        { "name": "Date",
          "value": [ "2013-11-19T08:30:50Z" ]
        },
        { "name": "Name",
          "value": [ "f81d4fae-7dec-11d0-a765-00a0c91e6bf6" ]
        },
        { "name": "SubtreeMsgCount",
          "value": [ "140" ]
        },
        { "name": "SubtreeSize",
          "value": [ "18766988" ]
        }
      ]
    },
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608",
    "path": "/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
    "name": "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
    "lastModSeq": 600
  }
D.42 Retrieve information about a non-existent folder (section 6.13.3.2)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld444777 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: nnnn
Date: Fri, 17 Jan 2014 17:51:59 GMT

{"requestError": {
  "link": [
    { "rel": "folder",
      "href": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld444777"
    }
  ],
  "serviceException": {
    "messageId": "SVC0004",
    "text": "No valid addresses provided in message part %1",
    "variables": [ "Request-URI" ]
  }
}

D.43 Retrieve information about a large folder (section 6.13.3.3)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608?listFilter=All&maxEntries=3 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"folder": {
  "parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554",
  "attributes": {
    "attribute": {
      "name": "Conversation-ID",
      "value": ["f81d4fae-7dec-11d0-a765-00a0c91e6bf6"]
    }
  }
}
D.44 Retrieve information about a large folder (section 6.13.3.4)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608?fromCursor=abcdef?cur%26194&listFilter=All&maxEntries=3

HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 200 OK
Date: Fri, 14 Mar 2014 09:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "folder": {
"parentFolder": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554",
"attributes": {
"attribute": [ {
"name": "Conversation-ID",
"value": ["f81d4fae-7dec-11d0-a765-00a0c91e6bf6"]
}, {
"name": "Contribution-ID",
"value": ["abcdef-1234-5678-90ab-cdef01234567"]
}, {
"name": "Date",
"value": ["2013-11-19T08:30:50Z"]
} ]
} }
```
{
  "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld608",
  "name": "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
  "lastModSeq": 600,
  "subFolders": {
    "folderReference": [{
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld902"
    }]
  },
  "objects": {
    "objectReference": [{
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/obj881"
    }]
  }
}
```

### D.45 Delete a folder, response with “204 No Content” (section 6.13.6.1)

**Request:**

```
DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld567 HTTP/1.1
Host: example.com
Accept: application/json
```

**Response:**

```
HTTP/1.1 204 No Content
Date: Thu, 05 Sep 2013 06:05:09 GMT
```

### D.46 Retrieve a folder’s name (section 6.14.3.1)

**Request:**

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

**Response:**

```
HTTP/1.1 200 OK
Date: Mon, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "name": "BoardMeeting" }
```
D.47 Change folder name (section 6.14.4.1)

Request:

```
PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn
Host: example.com

{ "name": "BoardSession1" }
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 04 Jun 2012 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "name": "BoardSession1" }
```

D.48 Change folder name, failure due to Policy error (section 6.14.4.2)

Request:

```
PUT /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456/folderName HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "name": "myDefault" }
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Thu, 22 May 2014 21:01:11 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
 "policyException": {
 "messageId": "POL1030",
 "text": "Modifying, moving or deleting this folder is not allowed"
 }
 }
```
D.49 Search for root folders (section 6.15.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": {
  "maxEntries": 3,
  "searchCriteria": {
    "criterion": [
      { "type": "Attribute",
        "name": "root",
        "value": "Yes"
      }
    ]
  }
}

Response:

HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "folderList": {
  "folder": [
    { "attributes": {
      "attribute": [
        { "name": "Root",
          "value": [ "Yes" ]
        }
      ],
      "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fldRoot1",
      "path": null,
      "lastModSeq": 1,
      "subFolders": {
        "folderReference": [
          { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld554",
            "path": "/main"
          },
          { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld664",
            "path": "/tmp"
          }
        ]
      }
    }
  ]
}

D.50 Search for folders created within a given timeframe (section 6.15.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": { "maxEntries": 10, "searchCriteria": { "criterion": [ { "type": "Date", "value": "minDate=2013-12-01T08:00:00Z&maxDate=2014-01-01T12:00Z" } ] }, "searchScope": { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld24", "path": "/main/projects/APIs" }, "sortCriteria": { "criterion": [ { "type": "Date", "order": "Ascending" } ] } } }

Response:

HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "folderList": { "folder": [ { "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld24", "attributes": { "attribute": [ { "name": "root", "value": [ "No" ] } ] } } ] }


```json
{
  "name": "Date",
  "value": ["2013-12-10T09:30:10Z"]
}
```

```
{
  "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld27",
  "path": "/main/projects/APIs/QoS",
  "lastModSeq": 91,
  "subFolders": [
    "folderReference": {
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld44",
      "path": "/main/projects/APIs/QoS/TS-Related"
    },
    "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld45",
    "path": "/main/projects/APIs/QoS/meetingInfo"
  ]
}
```

```
{
  "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld24",
  "attributes": {
    "attribute": [
      { "name": "root", "value": ["No"]
    },
    { "name": "Date", "value": ["2013-12-18T19:30:10Z"]
    }
  ]
}
```

```
{
  "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld30",
  "path": "/main/projects/APIs/WebRTCSignaling",
  "lastModSeq": 129,
  "subFolders": {
    "folderReference": [  
      { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld49",
        "path": "/main/projects/APIs/WebRTCSignaling/TS-Related"
      },
      { "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld50",
        "path": "/main/projects/APIs/WebRTCSignaling/meetingInfo"
      }
    ]
  }
}
```

```
D.51 Search for folders with a given name (section 6.15.5.3)

Request:

```plaintext
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": {
  "maxEntries": 10,
  "searchCriteria": {
    "criterion": [
      { "type": "Attribute",
        "name": "Name",
        "value": "projects"
      }
    ]
  }
}
```

Response:

```plaintext
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "folderList": {
  "folder": [
    { "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fldMain01",
      "attributes": {
        "attribute": [
          { "name": "root",
            "value": [ "No" ]
          },
          { "name": "Date",
            "value": [ "2013-12-10T09:00:10Z" ]
          },
          { "name": "Name",
            "value": [ "projects" ]
          }
        ]
      },
      "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld7",
      "path": "/main/projects",
      "name": "projects",
      "lastModSeq": 18
    },
    { "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld0044",
      "attributes": {
        "attribute": [
          { "name": "name",
            "value": [ "main/projects" ]
          }
        ]
      },
      "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld0044",
      "path": "/main/projects",
      "name": "projects",
      "lastModSeq": 18
    }
  ]
}
```
D.52 Search for folders containing a given substring in its name (section 6.15.5.4)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/search HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "selectionCriteria": {
  "maxEntries": 10,
  "searchCriteria": {
    "criterion": [
      { "type": "AllTextAttributes",
        "value": "QoS"
      }
    ]
  }
}
}
Response:

HTTP/1.1 200 OK
Date: Thu, 14 Nov 2013 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

```json
{
  "folderList": {
    "folder": [
      {
        "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld24",
        "attributes": {
          "attribute": [
            {
              "name": "root",
              "value": [ "No" ]
            },
            {
              "name": "Date",
              "value": [ "2013-12-10T09:30:10Z" ]
            },
            {
              "name": "Name",
              "value": [ "QoS" ]
            }
          ]
        },
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld27",
        "path": "/main/projects/APIs/QoS",
        "name": "QoS",
        "lastModSeq": 93
      },
      {
        "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fldTmp012",
        "attributes": {
          "attribute": [
            {
              "name": "root",
              "value": [ "No" ]
            },
            {
              "name": "Date",
              "value": [ "2013-10-18T19:30:10Z" ]
            },
            {
              "name": "Name",
              "value": [ "QoS-related" ]
            }
          ]
        },
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fldTmp012",
        "path": "/main/tmp/QoS-related",
        "name": "QoS-related",
        "lastModSeq": 96
      }
    ]
  }
}
```
D.53 Retrieve folder’s resource URL based on its path (section 6.16.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId?path=/RCSMessageStore/footballGame HTTP/1.1  
Host: example.com  
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903  
Accept: application/json  

Response:

HTTP/1.1 200 OK  
Date: Wed, 24 Jul 2013 12:51:59 GMT  
Content-Type: application/json  
Content-Length: nnnn  

{ "reference": {  
   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId11111",  
   "path": "/RCSMessageStore/footballGame"  
} }

D.54 Retrieve root folder’s resource URL (section 6.16.3.2)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId HTTP/1.1  
Host: example.com  
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903  
Accept: application/json  

Response:

HTTP/1.1 200 OK  
Date: Wed, 24 Jul 2013 12:51:59 GMT  
Content-Type: application/json  
Content-Length: nnnn  

{ "reference": {  
   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId001",  
   "path": null  
} }

D.55 Retrieve root folder’s resource URL, failure due to missing path parameter while multiple root folders exist (section 6.16.3.3)

Request:
GET /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json

Response:

HTTP/1.1 400 Bad Request
Date: Thu, 22 May 2014 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "requestError": {
  "serviceException": {
    "messageId": "SVC1009",
    "text": "Folder's path is missing. When more than one root folder exists, folder's path must be provided"
  }
}

D.56 Retrieve list of folders’ resource URLs based on their paths (section 6.16.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "pathList": {
  "path": [
    "main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
    "main/conversation5",
    "main/SorrentoMeeting"
  ]
}

Response:

HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B195855550100/folders/fId8",
      }
    ]
  }
}
D.57 Retrieve list of folders’ resource URLs based on their paths, two invalid paths in the list (section 6.16.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/pathToId HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "pathList": {
  "path": [
    "main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
    "main/conversation5",
    "main/SorrentoMeeting/year2014",
    "main/conversation99"
  ]
}

Response:

HTTP/1.1 200 OK
Date: Wed, 20 Nov 2013 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [{
      "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId9",
        "path":="/main/conversation5"
      }
    },
    {"code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId12",
        "path":="/main/SorrentoMeeting"
      }
    }]
}
"success": {
  "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld8",
  "path": "/main/f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
},
},
{"code": 200,
"reason": "OK",
"success": {
  "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld9",
  "path": "/main/conversation5"
}
},
{"code": 400,
"reason": "Bad Request",
"failure": {
  "serviceException": {
    "messageId": "SVC0002",
    "text": "Invalid input value for message part %1",
    "variables": ["/main/SorrentoMeeting/year2014"]
  }
}
},
{"code": 400,
"reason": "Bad Request",
"failure": {
  "serviceException": {
    "messageId": "SVC0002",
    "text": "Invalid input value for message part %1",
    "variables": ["/main/conversation99"]
  }
}
}
}
]
}

D.58 Copy objects to a target folder (section 6.17.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/copyToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{"targetSourceRef": {
  "targetRef": {"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456"},
  "sourceRefs": {
    "folders": {"folderReference": []},
    "objects": {
      "objectReference": [{"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001"},
      {"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002"}]}"}
D.59 Copy a folder with containing objects to a target folder  
(section 6.17.5.2)

Request:

```
POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/copyToFolders HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "targetSourceRef": {
  "targetRef": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456" },
  "sourceRefs": {
    "folders": { "folderReference": [ { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld111" } ] },
    "objects": { "objectReference": [ ] }
  }
}
```
Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": {
  "response": [
    { "code": 200,
      "reason": "OK",
      "success": {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/newfId111222",
        "path": "/main/SummerHolidayPlan/f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
      }
    }
  ]
}

D.60 Copy objects/folders to a target folder, failure due to at least one invalid source object or folder reference (section 6.17.5.3)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/copyToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "targetSourceRef": {
  "targetRef": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456" },
  "sourceRefs": {
    "folders": ["folderReference": []],
    "objects": {
      "objectReference": ["resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001"],
      "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId009" }
  }
}


Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "bulkResponseList": {
    "response": [
      { "code": 200,
        "reason": "OK",
        "success": {
          "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/newobjId333",
          "path": "/main/SummerHolidayPlan/newobjId333"
        }
      },
      { "code": 400,
        "reason": "Bad Request",
        "failure": {
          "serviceException": {
            "messageId": "SVC0002",
            "text": "Invalid input value for message part %1",
            "variables": [ "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId009" ]
          }
        }
      }
    ]
  }
}

D.61 Move objects to a target folder (section 6.18.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "targetSourceRef": {
  "targetRef": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456" },
  "sourceRefs": {
    "folders": [ "folderReference": [] ],
    "objects": {
      "objectReference": [ {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001"
      }, {
        "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002"
      } ]
    }
  }
}
Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": { 
 "response": [ 
 { "code": 200,
   "reason": "OK",
   "success": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId001" } },

 { "code": 200,
   "reason": "OK",
   "success": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/objId002" } }
 ] }
}

D.62 Move objects to a target folder (section 6.18.5.2)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFol
0x0der HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn

{ "targetSourceRef": { 
 "targetRef": { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld456" },

 "sourceRefs": { 
   "folders": { "folderReference": [ { "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld111" } ] },

   "objects": { "objectReference": [] } }

}

Response:

HTTP/1.1 200 OK
Date: Mon, 13 Jan 2014 04:10:19 GMT
Content-Type: application/json
Content-Length: nnnn

{ "bulkResponseList": { 
 "response": [ 
 { "code": 200,
   "reason": "OK",
   "success": { 
   "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld111",
   "path": "/main/SummerHolidayPlan/f81d4fae-7dec-11d0-a765-00a0c91e6bf6" }
 ] }
}
D.63 Move objects/folders to a target folder, failure due to a forbidden target folder (section 6.18.5.3)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/operations/moveToFolder HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
Content-Type: application/json
Content-Length: nnnn


Response:

HTTP/1.1 403 Forbidden
Date: Thu, 22 May 2014 12:09:09 GMT
Content-Type: application/json
Content-Length: nnnn

{n "requestError": {  "policyException": {  "messageId": "POL1031",  "text": "Attempt to create objects or folders under %1 is prohibited",  "variables": ["/System"]  }  }  }
D.64 Reading all active subscriptions (section 6.19.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

{ "nmsSubscriptionList": {
  "subscription": [
    {
      "callbackReference": {
        "notifyURL": "http://applicationClient.example.com/nms/notifications/77777",
        "callbackData": "abcd"
      },
      "duration": 6300,
      "clientCorrelator": "12345",
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001",
      "index": 1,
      "restartToken": "abc123"
    }
  ],
  "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions"
}
}

D.65 Creating a new subscription, response with copy of created resource (section 6.19.5.1)

Request:

POST /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{ "nmsSubscription": {
  "callbackReference": {
    "notifyURL": "http://applicationClient.example.com/nms/notifications/77777",
    "callbackData": "abcd"
  },
  "duration": 7200,
  "clientCorrelator": "12345"
}
}
Response:

HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

{"nmsSubscription": {
"callbackReference": {
"notifyURL": "http://applicationClient.example.com/nms/notifications/77777",
"callbackData": "abcd"
},
"duration": 7200,
"clientCorrelator": "12345",
"resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001",
"index": 1,
"restartToken": "abc123"
}
}

D.66 Reading an individual subscription (section 6.20.3.1)

Request:

GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

{"nmsSubscription": {
"callbackReference": {
"notifyURL": "http://applicationClient.example.com/nms/notifications/77777",
"callbackData": "abcd"
},
"duration": 7200,
"clientCorrelator": "12345",
"resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001",
"index": 1,
"restartToken": "abc123"
}
}
D.67 Retrieve information about a non-existent individual subscription (section 6.20.3.2)

Request:

```
GET /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub6699 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

Response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

{"requestError": {
  "link": [
    {
      "rel": "NmsSubscription",
      "href": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub6699"
    }
  ],
  "serviceException": {
    "messageId": "SVC0004",
    "text": "No valid addresses provided in message part %1",
    "variables": [ "Request-URI" ]
  }
}}
```
Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

{
   "nmsSubscription": {
      "callbackReference": {
         "notifyURL": "http://applicationClient.example.com/nms/notifications/77777",
         "callbackData": "abcd"
      },
      "duration": 10800,
      "clientCorrelator": "12345",
      "resourceURL": "http://example.com/exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001",
      "index": 46,
      "restartToken": "nnn789"
   }
}
```

### D.69 Cancelling a subscription (section 6.20.6.1)

**Request:**

```
DELETE /exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com
```

**Response:**

```
HTTP/1.1 204 No Content
Date: Wed, 15 Jan 2014 18:51:59 GMT
```

### D.70 Notify a client about NMS object changes (section 6.21.5.1)

**Request:**

```
POST /nms/notifications/77777 HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: applicationClient.example.com

{
   "nmsEventList": {
      "nmsEvent": [
         {"deletedObject": {
            "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old999",
            "lastModSeq": 133,
            "correlationId": "cId122"
         }
      ],
      {"changedObject": {
         "parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId80",
         "flags": ["\Flagged"],
         "resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/old1000",
```

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document
"lastModSeq": 134,
"correlationId": "cId67"
}
},
{"expiredObject": {
"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/objects/oId111",
"lastModSeq": 135,
"correlationId": "cId9"
}
}
],
"callbackData": "12345",
"index": 1,
"restartToken": "abc67",
"link": [
{"rel": "NmsSubscription",
"href": "http://example.com/exampleAPI/nms/v1/myStore/myStore/tel%3A%2B19585550100/subscriptions/sub001"
}
]
}

Response:

HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT

D.71 Notify a client about NMS folder changes (section 6.21.5.2)

Request:

POST /nms/notifications/77777 HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: applicationClient.example.com

{ "nmsEventList": { 
"nmsEvent": [ 
{"deletedFolder": { 
"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fId20",
"lastModSeq": 136
}
},
{"changedFolder": { 
"parentFolder": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld10",
"resourceURL": "http://exampleAPI/nms/v1/myStore/tel%3A%2B19585550100/folders/fld33",
"name": "SorrentoMeeting",
"lastModSeq": 137
}
]
"callbackData": "12345",
"index": 2,
"restartToken": "hgf853",
"link": [ 

© 2016 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms as stated in this document [OMA-TEMPLATE-TS_RESTful_Network_API-20160101-I]
{ "rel": "NmsSubscription",
  "href": "http://example.com/exampleAPI/nms/v1/myStore/myStore/tel%3A%2B19585550100/subscriptions/sub001"
}
]
}

Response:

HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT
Appendix E. Operations mapping to a pre-existing baseline specification

(Informative)

As this specification does not have a baseline specification, this appendix is empty.
Appendix F.  Light-weight Resources  (Informative)

The following table lists all NMS data structure elements that can be accessed individually as Light-weight Resources.

For each Light-weight Resource, the following information is provided: corresponding root element name, root element type and [ResourceRelPath] string.

<table>
<thead>
<tr>
<th>Type of Light-weight Resources (and references to data structures)</th>
<th>Element/attribute that can be accessed as Light-weight Resource</th>
<th>Root element name for the Light-weight Resource</th>
<th>Root element type for the Light-weight Resource</th>
<th>[ResourceRelPath] string that needs to be appended to the corresponding Heavy-weight Resource URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder (5.3.2.8)</td>
<td>name</td>
<td>name</td>
<td>xsd:string</td>
<td>folderName</td>
</tr>
</tbody>
</table>
Appendix G. Authorization aspects

This appendix specifies how to use the RESTful Network Message Storage API in combination with some authorization frameworks.

G.1 Use with OMA Authorization Framework for Network APIs

The RESTful NMS API MAY support the authorization framework defined in [Autho4API_10].

A RESTful NMS API supporting [Autho4API_10]:

- SHALL conform to section D.1 of [REST_NetAPI_Common];
- SHALL conform to this section G.1.

G.1.1 Scope values

G.1.1.1 Definitions

In compliance with [Autho4API_10], an authorization server serving clients requests for getting authorized access to the resources exposed by the RESTful NMS API:

- SHALL support the scope values defined in the table below;
- MAY support scope values not defined in this specification.

<table>
<thead>
<tr>
<th>Scope value</th>
<th>Description</th>
<th>For one-time access token</th>
</tr>
</thead>
<tbody>
<tr>
<td>oma_rest_nms.all_{apiVersion}</td>
<td>Provide access to all defined operations on the resources in this version of the API. The {apiVersion} part of this identifier SHALL have the same value as the &quot;apiVersion&quot; URL variable which is defined in section 5.2. This scope value is the union of the other scope values listed in next rows of this table.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_nms.create</td>
<td>Provide access to all defined operations relating to object creation function only.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_nms.read</td>
<td>Provide access to all defined operations relating to reading the contents of the box only.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_nms.modify</td>
<td>Provide access to all defined operations relating to modifying the contents of the box only.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_nms.subscription</td>
<td>Provide access to all defined operations relating to managing notification subscriptions.</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2: Scope values for RESTful NMS API

G.1.1.2 Downscoping

In the case where the client requests authorization for “oma_rest_nms.all_{apiVersion}” scope, the authorization server and/or resource owner MAY restrict the granted scope to some of the following scope values:

- “oma_rest_nms.create”
- “oma_rest_nms.read”
- “oma_rest_nms.modify”
• “oma_rest_nms.subscription”

G.1.1.3 Mapping with resources and methods

Tables in this section specify how the scope values defined in section G.1.1.1 for the RESTful NMS API map to the REST resources and methods of this API. In these tables, the root “oma_rest_nms.” of scope values is omitted for readability reasons.
| Resource | URL Base URL:  
//{serverRoot}/nms/{apiVersion}/
{storeName}/{boxId} | Section reference | HTTP verbs | GET | PUT | POST | DELETE |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource containing all objects</td>
<td>/objects</td>
<td>6.1</td>
<td>n/a</td>
<td>n/a</td>
<td>all_{apiVersion} or create</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>A stored object</td>
<td>/objects/{objectId}</td>
<td>6.2</td>
<td>all_{apiVersion} or read</td>
<td>n/a</td>
<td>n/a</td>
<td>all_{apiVersion} or modify</td>
<td></td>
</tr>
<tr>
<td>Flags associated with the stored object</td>
<td>/objects/{objectId}/flags</td>
<td>6.3</td>
<td>all_{apiVersion} or read</td>
<td>all_{apiVersion} or modify</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>Individual flag</td>
<td>/objects/{objectId}/flags/{flagName}</td>
<td>6.4</td>
<td>all_{apiVersion} or read</td>
<td>all_{apiVersion} or modify</td>
<td>n/a</td>
<td>all_{apiVersion} or modify</td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>URL &lt;specified by server&gt;</td>
<td>Section reference</td>
<td>HTTP verbs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-----------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload of the stored object</td>
<td>&lt;payload URL specified by the server in the payloadURL field of Object&gt;</td>
<td>6.5</td>
<td>GET: all_{apiVersion} or read PUT: n/a POST: n/a DELETE: n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For content available via the NMS resource tree this URL is: //{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For externally referenced content, this is an arbitrary URL.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload part of the stored object</td>
<td>&lt;payload part URL specified by the server in the payloadPart field of Object&gt;</td>
<td>6.6</td>
<td>GET: all_{apiVersion} or read PUT: n/a POST: n/a DELETE: n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For content available via the NMS resource tree this URL is: //{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}/objects/{objectId}/payloadParts/{payloadPartId}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For externally referenced content, this is an arbitrary URL.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{apiVersion}/{storeName}/{boxId}</td>
<td>Section reference</td>
<td>HTTP verbs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>-------------------</td>
<td>------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource containing all folders</td>
<td>/folders</td>
<td>6.12</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>all_{apiVersion}</td>
<td>or modify</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A folder</td>
<td>/folders/{folderId}</td>
<td>6.13</td>
<td>all_{apiVersion} or read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a</td>
<td>all_{apiVersion} or modify</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Individual folder data</td>
<td>/folders/{folderId}/{ResourceRelPath}</td>
<td>6.14</td>
<td>all_{apiVersion} or read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>all_{apiVersion} or modify</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{apiVersion}/ {storeName}/{boxId}/objects/operations</td>
<td>Section reference</td>
<td>HTTP verbs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information about a selected set of objects in the storage</td>
<td>/search</td>
<td>6.7</td>
<td>n/a</td>
<td>n/a</td>
<td>all_{apiVersion} or read</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>
| Resource URLs of a selected set of objects in the storage | /pathToId  
Note: read as path-To-Id | 6.8 | all_{apiVersion} or read | n/a | all_{apiVersion} or read | n/a |
<p>| Bulk creation of objects | /bulkCreation | 6.9 | n/a | n/a | all_{apiVersion} or create | n/a |</p>
<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: //{serverRoot}/nms/{apiVersion}/ {storeName}/ {boxId}/folders/operations</th>
<th>Section reference</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.15</td>
<td>GET</td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>all_{apiVersion} or read</td>
</tr>
<tr>
<td>Information about a selected set of folders in the storage</td>
<td>/search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource URLs of a selected set of folders in the storage</td>
<td>/pathToId</td>
<td>6.16</td>
<td>all_{apiVersion} or read</td>
</tr>
<tr>
<td>Note: read as path-To-Id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource for triggering object(s)/folder(s) copying</td>
<td>/copyToFolder</td>
<td>6.17</td>
<td>n/a</td>
</tr>
<tr>
<td>Resource for triggering object(s)/folder(s) moving</td>
<td>/moveToFolder</td>
<td>6.18</td>
<td>n/a</td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: //{serverRoot}/nms/{apiVersion}/ {storeName}/{boxId}</td>
<td>Section reference</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------</td>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>All subscriptions in the storage</td>
<td>/subscriptions</td>
<td>6.19</td>
<td>all_{apiVersion} or subscription</td>
</tr>
<tr>
<td>Individual subscription</td>
<td>/subscriptions/{subscriptionId}</td>
<td>6.20</td>
<td>all_{apiVersion} or subscription</td>
</tr>
</tbody>
</table>
G.1.2 Use of ‘acr:auth’

This section specifies the use of ‘acr:auth’ in place of an end user identifier in a resource URL path.

An ‘acr’ URI of the form ‘acr:auth’, where ‘auth’ is a reserved keyword MAY be used to avoid exposing a real end user identifier in the resource URL path.

A client MAY use ‘acr:auth’ in a resource URL in place of a {boxId} when the RESTful NMS API is used in combination with [Autho4API_10].

In the case the RESTful NMS API supports [Autho4API_10], the server:

- SHALL accept ‘acr:auth’ as a valid value for the resource URL variable {boxId}
- SHALL conform to [REST_NetAPI_Common] section 5.8.1.1 regarding the processing of ‘acr:auth’.
Appendix H. Flag Names Table (Normative)

The following table lists the most common flag names as defined by [RFC3501], [RFC5788] and [OMA-CPM_TS_MessageStorage].

<table>
<thead>
<tr>
<th>Flag Name</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen</td>
<td>Message has been read</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>Answered</td>
<td>Message has been answered</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>Flagged</td>
<td>Message is &quot;flagged&quot; for urgent and/or special attention</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>Deleted</td>
<td>Message is &quot;deleted&quot; for removal by later internal message store process</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>Draft</td>
<td>Message has not completed composition (marked as a draft)</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>Recent</td>
<td>Message is &quot;recently&quot; arrived in this mailbox</td>
<td>[RFC3501], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>$MDNSent</td>
<td>A disposition notification has been sent for this message</td>
<td>[RFC5788], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>$Forwarded</td>
<td>Message has been forwarded</td>
<td>[RFC5788], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td>\read-report-sent</td>
<td>A read receipt has been sent for this message</td>
<td>[OMA-CPM_TS_MessageStorage]</td>
</tr>
</tbody>
</table>

Table 3 Flag Names

Note that in addition to the strings listed in the above table, deployments MAY also support other strings.
## Appendix I. RCS Object Attributes Table (Informative)

The following common object attributes are suggested by this specification in order to enhance interoperability. See section 5.3.2.5 “Attribute” for details of how these may be used.

The use of these attributes is REQUIRED when NMS is used as part of the GSMA Rich Communications Suite (RCS).

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Format</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong></td>
<td>Date and time of the object. Typically this is the date and time at which the object was complete and ready for transmission or storage.</td>
<td>xsd:dateTimeStamp</td>
<td>[XMLSchema2], [RFC5322], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td><strong>Message-Context</strong></td>
<td>The object’s classification (i.e., what kind of object it is).</td>
<td>Enumeration. Valid attribute values are at least: “voice-message” “video-message” “fax-message” “pager-message” (i.e., SMS) “multimedia-message” (i.e., MMS) “text-message” (i.e., email) “none” Other values may be defined by profiles or other standards. This attribute is case-insensitive.</td>
<td>[RFC3458], [RFC3938]</td>
</tr>
<tr>
<td><strong>Direction</strong></td>
<td>Direction of message.</td>
<td>Enumeration. Valid values are: “In” (i.e., message terminating at this box) “Out” (i.e., message originating from this box). This attribute is case-insensitive.</td>
<td></td>
</tr>
<tr>
<td><strong>From</strong></td>
<td>Address of originator (i.e., sender).</td>
<td>xsd:string</td>
<td>[RFC5322], [OMA-CPM_TS_MessageStorage]</td>
</tr>
<tr>
<td><strong>To</strong></td>
<td>Address of primary recipient(s).</td>
<td>xsd:string</td>
<td>[RFC5322]</td>
</tr>
<tr>
<td><strong>Cc</strong></td>
<td>Address of other recipient(s).</td>
<td>xsd:string</td>
<td>[RFC5322]</td>
</tr>
<tr>
<td><strong>Bcc</strong></td>
<td>Address of recipient(s) whose addresses are not to be revealed to</td>
<td>xsd:string</td>
<td>[RFC5322]</td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td>A short string identifying the topic of the object.</td>
<td>xsd:string</td>
<td>[RFC5322]</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td><strong>TextContent</strong></td>
<td>The stored content, if it is representable as an xsd:string (see section 5.1.9).</td>
<td>xsd:string</td>
<td>[XMLSchema2]</td>
</tr>
<tr>
<td><strong>Content-Type</strong></td>
<td>Indicates the top level MIME content type of the object as a whole, if any. For example: multipart/related; start=&quot;<a href="mailto:950120.aaaCC@example.com">950120.aaaCC@example.com</a>&quot;; type=&quot;application/smii&quot;</td>
<td>xsd:string</td>
<td>[RFC2045]</td>
</tr>
<tr>
<td><strong>Content-Location</strong></td>
<td>The top level Content-Location, if any, as defined in [RFC2557], unfolded and with any transfer encoding such as [RFC2047] removed. Used to specify a base URI for this object. For example, “<a href="http://example.com%E2%80%9D">http://example.com”</a>.</td>
<td>xsd:string</td>
<td>[RFC2557]</td>
</tr>
</tbody>
</table>

**Table 4 Object Attributes**
Appendix J.  Standard Folder Attributes  (Normative)

The following table specifies standard folder attribute names and their associated semantics. See section 5.3.2.5 “Attribute” for details of how these may be used.

Each attribute is optional unless otherwise specified.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Format</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>The value “Yes” denotes the folder is designated as a root folder. For multi-root deployment environment, there may be several folders containing the attribute root=“Yes”. In some deployment scenarios other well-known attribute values may be used and other restrictions may apply (e.g. mandating only single root folder).</td>
<td>xsd:string</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Folder name. Read-only attribute. Where present, the value of this attribute SHALL be the same as that of the “name” element in the Folder data structure (see section 5.3.2.8). This enables search based on folder name.</td>
<td>xsd:string</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 Standard Folder Attributes
Appendix K. RCS Folder Attributes

The following common folder attributes are suggested by this specification in order to enhance interoperability in RCS profiles. Each attribute is optional unless otherwise specified.

See also Appendix J.

Additional RCS attributes names and their associated semantics could be derived from or be equivalent to a message header, including those defined in [RFC5322], [OMA-CPM_TS_MessageStorage], and [IANA_Message_Headers].

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
<th>Format</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date and Time at which the folder was created.</td>
<td>As defined by xsd:dateTimeStamp in [XMLSchema2]</td>
<td></td>
</tr>
<tr>
<td>MsgCount</td>
<td>Total number of message objects in the folder.</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
<tr>
<td>UnreadMsgCount</td>
<td>Total number of message objects in the folder, that are unread (do not carry the /Seen flag)</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Folder size in bytes. The value MAY be approximate or may not include the size of the containing objects.</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
<tr>
<td>SubtreeMsgCount</td>
<td>Total number of message objects in a subtree rooted at the folder.</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
<tr>
<td>SubtreeUnreadMsgCount</td>
<td>Total number of message objects in a subtree rooted at the folder, that are unread (do not carry the /Seen flag)</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
<tr>
<td>SubtreeSize</td>
<td>Total size in bytes of a subtree rooted at the folder. The value MAY be approximate.</td>
<td>xsd:unsignedLong</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 RCS Folder Attributes