Use of this document is subject to all of the terms and conditions of the Use Agreement located at http://www.openmobilealliance.org/UseAgreement.html.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the “OMA IPR Declarations” list at http://www.openmobilealliance.org/ipr.html. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR’S REPRESENTED ON THE “OMA IPR DECLARATIONS” LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Used with the permission of the Open Mobile Alliance Ltd. under the terms set forth above.
Contents

1. SCOPE ......................................................................................................................................................... 11
2. REFERENCES .................................................................................................................................................. 12
   2.1 NORMATIVE REFERENCES ................................................................................................................... 12
   2.2 INFORMATIVE REFERENCES ................................................................................................................ 12
3. TERMINOLOGY AND CONVENTIONS ......................................................................................................... 13
   3.1 CONVENTIONS ......................................................................................................................................... 13
   3.2 DEFINITIONS ........................................................................................................................................... 13
   3.3 ABBREVIATIONS .................................................................................................................................. 13
4. INTRODUCTION ............................................................................................................................................ 15
   4.1 VERSION 1.0 ........................................................................................................................................ 15
5. ADDRESS BOOK API DEFINITION ............................................................................................................ 16
   5.1 RESOURCES SUMMARY ......................................................................................................................... 16
   5.2 DATA TYPES ......................................................................................................................................... 26
      5.2.1 XML Namespaces .............................................................................................................................. 26
      5.2.2 Structures ........................................................................................................................................ 26
         5.2.2.1 Type: ContactCollection ........................................................................................................ 26
         5.2.2.2 Type: Contact ......................................................................................................................... 27
         5.2.2.3 Type: SharedIdentity ................................................................................................................. 27
         5.2.2.4 Type: ListCollection ................................................................................................................. 28
         5.2.2.5 Type: List .................................................................................................................................. 28
         5.2.2.6 Type: AttributeList .................................................................................................................. 28
         5.2.2.7 Type: Attribute ....................................................................................................................... 29
         5.2.2.8 Type: MemberCollection ......................................................................................................... 29
         5.2.2.9 Type: Member .......................................................................................................................... 30
         5.2.2.10 Type: ListReferenceCollection ............................................................................................... 30
         5.2.2.11 Type: AbChangesSubscriptionCollection ........................................................................ 31
         5.2.2.12 Type: AbChangesSubscription ............................................................................................ 31
         5.2.2.13 Type: AbChangesNotification ............................................................................................... 32
         5.2.2.14 Type: RuleList ...................................................................................................................... 32
         5.2.2.15 Type: Rule ................................................................................................................................ 33
         5.2.2.16 Type: ApplicableToContacts ................................................................................................. 34
         5.2.2.17 Type: ApplicableToLists ......................................................................................................... 34
         5.2.2.18 Type: ApplicableToListsAndLinkedContacts ...................................................................... 34
         5.2.2.19 Type: AuthorizedTo ............................................................................................................... 34
         5.2.2.20 Type: MemberTransferParameters ....................................................................................... 35
      5.2.3 Enumerations ..................................................................................................................................... 35
         5.2.3.1 Enumeration: ListType ............................................................................................................ 35
         5.2.3.2 Enumeration: ResourceStatus ................................................................................................. 35
      5.2.4 Values of the Link “rel” attribute ..................................................................................................... 36
   5.3 SEQUENCE DIAGRAMS .......................................................................................................................... 36
      5.3.1 Managing contacts in the collection of contacts ........................................................................ 36
      5.3.2 Accessing the lists and members .................................................................................................. 38
      5.3.3 Subscribing to address book changes and receiving notifications ............................................. 39
      5.3.4 Managing shared lists ...................................................................................................................... 41
      5.3.5 Managing authorization rules ....................................................................................................... 42
6. DETAILED SPECIFICATION OF THE RESOURCES ................................................................................. 44
   6.1 RESOURCE: COLLECTION OF CONTACTS .............................................................................................. 44
      6.1.1 Request URL variables .................................................................................................................... 44
      6.1.2 Response Codes and Error Handling ............................................................................................ 44
         For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7 ......................................................................................................................... 45
      6.1.3 GET .................................................................................................................................................. 45
         6.1.3.1 Example 1: Retrieve all contacts with all attributes (default) (Informative) .......................... 45
         6.1.3.2 Example 2: Retrieve all contacts with selected attributes (Informative) ............................. 46
      6.1.4 PUT .................................................................................................................................................. 47
      6.1.5 POST ............................................................................................................................................... 47

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
6.2 RESOURCE: INDIVIDUAL CONTACT ........................................................................................................ 48
  6.2.1 Request URL variables ......................................................................................................................... 48
  6.2.2 Response Codes and Error Handling .................................................................................................... 48
  
  For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7. 48
  6.2.3 GET ........................................................................................................................................................ 48
    6.2.3.1 Example 1: Retrieve a contact with all attributes (default) (Informative) ..................................... 48
    6.2.3.2 Example 2: Retrieve a contact with selected attributes (Informative) ........................................ 49
    6.2.3.3 Example 3: Retrieve a contact with only vCard2.1 as selected attribute (Informative) .............. 50
  6.2.4 PUT ......................................................................................................................................................... 51
    6.2.4.1 Example 1: Create a new contact (Informative) ............................................................................. 51
    6.2.4.2 Example 2: Update an existing contact with links to an existing member (Informative) ............ 51
    6.2.4.3 Example 3: Update an existing contact with links to non existing member (Informative) ........... 54
  6.2.5 POST .................................................................................................................................................... 54
  6.2.6 DELETE ................................................................................................................................................ 54
    6.2.6.1 Example: Delete a contact (Informative) ......................................................................................... 55

6.3 RESOURCE: ATTRIBUTES FOR A CONTACT ..................................................................................... 55
  6.3.1 Request URL variables .......................................................................................................................... 55
  6.3.2 Response Codes and Error Handling .................................................................................................... 55
    6.3.3 GET ......................................................................................................................................................... 55
      6.3.3.1 Example: Retrieve all attributes of a contact (Informative) ......................................................... 55
    6.3.4 PUT ....................................................................................................................................................... 56
    6.3.5 POST .................................................................................................................................................... 56
    6.3.6 DELETE ................................................................................................................................................ 56

6.4 RESOURCE: INDIVIDUAL ATTRIBUTE FOR A CONTACT ............................................................. 56
  6.4.1 Request URL variables .......................................................................................................................... 56
  6.4.2 Response Codes and Error Handling .................................................................................................... 57
    6.4.3 GET ......................................................................................................................................................... 57
      6.4.3.1 Example 1: Retrieve a contact’s attribute (Informative) ................................................................. 57
      6.4.3.2 Example 2: Retrieve a contact’s non existing attribute (Informative). ......................................... 57
    6.4.4 PUT ....................................................................................................................................................... 58
      6.4.4.1 Example: Create or update a contact’s attribute (Informative) .................................................... 58
    6.4.5 POST .................................................................................................................................................... 58
    6.4.6 DELETE ................................................................................................................................................ 58
      6.4.6.1 Example: Delete a contact’s attribute (Informative) ..................................................................... 58

6.5 RESOURCE: LISTS ................................................................................................................................. 59
  6.5.1 Request URL variables .......................................................................................................................... 59
  6.5.2 Response Codes ..................................................................................................................................... 59
    6.5.3 GET ......................................................................................................................................................... 59
      6.5.3.1 Example 1: Retrieve lists with all attributes and all members with all member attributes (default) (Informative).................. 60
      6.5.3.2 Example 2: Retrieve all lists belonging to a user without list attributes and without members (Informative) ................... 62
      6.5.3.3 Example 3: Retrieve specific lists attributes and specific member attributes (Informative) .............. 63
      6.5.3.4 Example 4: Retrieve lists with specific lists attributes and without members data (Informative) ............. 64
      6.5.3.5 Example 5: Retrieve specific lists attributes and list of members without attributes (Informative) ............ 65
      6.5.3.6 Example 6: Retrieve lists without attributes and members with all member attributes (Informative) .......... 67
    6.5.4 PUT ....................................................................................................................................................... 68
    6.5.5 POST .................................................................................................................................................... 68
    6.5.6 DELETE ................................................................................................................................................ 68

6.6 RESOURCE: INDIVIDUAL LIST ........................................................................................................... 68
  6.6.1 Request URL variables .......................................................................................................................... 68
  6.6.2 Response Codes and Error Handling .................................................................................................... 69
    6.6.3 GET ......................................................................................................................................................... 69
      6.6.3.1 Example 1: Retrieve a list with all attributes and all member attributes (default) (Informative) ...... 70
      6.6.3.2 Example 2: Retrieve a list without list and member attributes (Informative) ................................. 71
      6.6.3.3 Example 3: Retrieve a list with all attributes and selected member attributes (Informative) ........... 71
      6.6.3.4 Example 4: Retrieve a non existing list (Informative) ................................................................. 72
    6.6.4 PUT ....................................................................................................................................................... 72
      6.6.4.1 Example: Create a list (Informative) ............................................................................................. 72
    6.6.5 POST .................................................................................................................................................... 73
    6.6.6 DELETE ................................................................................................................................................ 73
6.7 RESOURCE: ATTRIBUTES FOR A LIST ................................................................. 74
  6.7.1 Request URL variables ................................................................. 74
  6.7.2 Response Codes and Error Handling ........................................ 74
  6.7.3 GET ............................................................................................. 74
    6.7.3.1 Example: Retrieve all attributes for a list belonging to Bob (Informative) ............................................................. 74
  6.7.4 PUT ............................................................................................. 75
  6.7.5 POST .......................................................................................... 75
  6.7.6 DELETE ....................................................................................... 75

6.8 RESOURCE: INDIVIDUAL ATTRIBUTE FOR A LIST ........................................ 75
  6.8.1 Request URL variables ................................................................. 75
  6.8.2 Response Codes and Error Handling ........................................ 75
  6.8.3 GET ............................................................................................. 76
    6.8.3.1 Example 1: Retrieve an attribute value (Informative) ............................................................................................ 76
    6.8.3.2 Example 2: Retrieve a non existing attribute (Informative) .......................................................................................... 76
  6.8.4 PUT ............................................................................................. 77
    6.8.4.1 Example: Create an attribute (Informative) ................................................................................................................. 77
  6.8.5 POST .......................................................................................... 77
  6.8.6 DELETE ....................................................................................... 77
    6.8.6.1 Example: Delete an attribute (Informative) ...................................................................................................................... 77

6.9 RESOURCE: MEMBERS IN A LIST ................................................................. 78
  6.9.1 Request URL variables ................................................................. 78
  6.9.2 Response Codes and Error Handling ........................................ 78
  6.9.3 GET ............................................................................................. 78
    6.9.3.1 Example 1: Retrieve all members for a list belonging to a user with all attributes (default) (Informative) ........ 79
    6.9.3.2 Example 2: Retrieve all members for a list belonging to a user, without member attributes (Informative) ........ 80
  6.9.4 PUT ............................................................................................. 80
  6.9.5 POST .......................................................................................... 80
  6.9.6 DELETE ....................................................................................... 80

6.10 RESOURCE: INDIVIDUAL MEMBER IN A LIST ......................................... 81
  6.10.1 Request URL variables ................................................................. 81
  6.10.2 Response Codes and Error Handling ........................................ 81
  6.10.3 GET ............................................................................................. 81
    6.10.3.1 Example 1: Retrieve a member (Informative) ................................................................................................................. 82
    6.10.3.2 Example 2: Retrieve a member and selected attributes (Informative) ........................................................................... 82
    6.10.3.3 Example 3: Retrieve a non existing member (Informative) ............................................................................................ 83
  6.10.4 PUT ............................................................................................. 83
    6.10.4.1 Example 1: Create a member without a link to the contact (Informative) ................................................................. 83
    6.10.4.2 Example 2: Update member with relation to non-existing contact (Informative) ....................................................... 84
  6.10.5 POST .......................................................................................... 85
  6.10.6 DELETE ....................................................................................... 85
    6.10.6.1 Example: Delete a member (Informative) ...................................................................................................................... 85

6.11 RESOURCE: ATTRIBUTES FOR A MEMBER IN A LIST ............................... 85
  6.11.1 Request URL variables ................................................................. 85
  6.11.2 Response Codes and Error Handling ........................................ 86
  6.11.3 GET ............................................................................................. 86
    6.11.3.1 Example: Retrieve all attributes of a member of a list (Informative) ................................................................. 86
  6.11.4 PUT ............................................................................................. 86
  6.11.5 POST .......................................................................................... 87
  6.11.6 DELETE ....................................................................................... 87

6.12 RESOURCE: INDIVIDUAL ATTRIBUTE FOR A MEMBER IN A LIST ............ 87
  6.12.1 Request URL variables ................................................................. 87
  6.12.2 Response Codes and Error Handling ........................................ 87
  6.12.3 GET ............................................................................................. 87
    6.12.3.1 Example 1: Retrieve a member’s attribute (Informative) ............................................................................................ 87
    6.12.3.2 Example 2: Retrieve a member’s non existing attribute (Informative) ........................................................................... 88
  6.12.4 PUT ............................................................................................. 88
    6.12.4.1 Example: Create a member’s attribute (Informative) ............................................................................................... 88
  6.12.5 POST .......................................................................................... 89
6.12.6 DELETE
6.12.6.1 Example: Delete a member’s attribute (Informative) ................................................................. 89

6.13 RESOURCE: LIST REFERENCES ........................................................................................................ 89
6.13.1 Request URL variables .................................................................................................................. 89
6.13.2 Response Codes and Error Handling ............................................................................................... 90
6.13.3 GET .................................................................................................................................................. 90
   6.13.3.1 Example: Retrieve all list references in a list (Informative) ......................................................... 90
6.13.4 PUT .................................................................................................................................................. 90
6.13.5 POST ............................................................................................................................................... 90
6.13.6 DELETE ......................................................................................................................................... 90

6.14 RESOURCE: INDIVIDUAL LIST REFERENCE .............................................................................. 91
6.14.1 Request URL variables .................................................................................................................. 91
6.14.2 Response Codes and Error Handling ............................................................................................... 91
6.14.3 GET .................................................................................................................................................. 91
   6.14.3.1 Example 1: Retrieve a list reference (Informative) .................................................................. 91
   6.14.3.2 Example 2: Retrieve a non existing list reference (Informative) ............................................. 92
6.14.4 PUT .................................................................................................................................................. 92
   6.14.4.1 Example: Create a list reference (Informative) ..................................................................... 92
6.14.5 POST ............................................................................................................................................... 93
6.14.6 DELETE ......................................................................................................................................... 93
   6.14.6.1 Example: Delete a list reference (Informative) ..................................................................... 93

6.15 RESOURCE: ADDRESS BOOK CHANGES SUBSCRIPTIONS ....................................................... 93
6.15.1 Request URL variables .................................................................................................................. 93
6.15.2 Response Codes and Error Handling ............................................................................................... 94
6.15.3 GET .................................................................................................................................................. 94
   6.15.3.1 Example: Retrieve all list changes subscriptions (Informative) ............................................. 94
6.15.4 PUT .................................................................................................................................................. 95
6.15.5 POST ............................................................................................................................................... 95
   6.15.5.1 Example 1: Create new subscription for list changes notification using ‘tel’ URI (Informative) ....... 95
   6.15.5.2 Example 2: Create new subscription for list changes notification using ‘acr’ URI (Informative) ...... 96
6.15.6 DELETE ......................................................................................................................................... 96

6.16 RESOURCE: INDIVIDUAL LIST CHANGES SUBSCRIPTION ....................................................... 97
6.16.1 Request URL variables .................................................................................................................. 97
6.16.2 Response Codes and Error Handling ............................................................................................... 97
6.16.3 GET .................................................................................................................................................. 97
   6.16.3.1 Example: Retrieve individual list changes subscription (Informative) ....................................... 97
6.16.4 PUT .................................................................................................................................................. 98
   6.16.4.1 Example: Update/Extend duration of subscription (Informative) .............................................. 98
6.16.5 POST ............................................................................................................................................... 99
6.16.6 DELETE ......................................................................................................................................... 99
   6.16.6.1 Example: Delete a list changes subscription (Informative) ...................................................... 99

6.17 RESOURCE: CLIENT RESOURCE FOR LIST CHANGES NOTIFICATIONS .................................. 99
6.17.1 Request URL variables .................................................................................................................. 100
6.17.2 Response Codes and Error Handling ............................................................................................... 100
   For HTTP response codes, see [REST NetAPI Common]. ...................................................................... 100
6.17.3 GET ............................................................................................................................................... 100
6.17.4 PUT ............................................................................................................................................... 100
6.17.5 POST ............................................................................................................................................. 100
   6.17.5.1 Example 1: Notification for subscription with sendFullListContent=true (default) (Informative) .. 100
   6.17.5.2 Example 2: Notification for contact change subscription (Informative) ................................... 101
   6.17.5.3 Example 3: Notification for expired contact subscription (Informative) ................................. 101
6.17.6 DELETE ....................................................................................................................................... 102

6.18 RESOURCE: COLLECTION OF CONTACTS SHARED BY OTHER USER ................................. 102
6.18.1 Request URL variables .................................................................................................................. 102
6.18.2 Response Codes and Error Handling ............................................................................................... 102
6.18.3 GET ............................................................................................................................................... 102
   6.18.3.1 Example 1: Retrieve contacts shared by a user identified by otherUserId with all contact attributes (default) (Informative) 103
6.19 RESOURCE: INDIVIDUAL SHARED CONTACT ............................................................................. 105

6.19.1 Request URL variables ........................................................................................................... 105
6.19.2 Response Codes and Error Handling ...................................................................................... 105
6.19.3 GET ........................................................................................................................................... 105

6.19.3.1 Example 1: Retrieve information about individual shared contact using contactId (Informative) .......................................................... 106
6.19.3.2 Example 2: Retrieve information about individual shared contact using sharedIdentity (Informative) .......................................................... 107

6.19.3.3 Example 3: Retrieve information about individual shared contact, with selected contact attributes (default) (Informative) ........................................................................ 108

6.19.4 PUT ........................................................................................................................................... 108
6.19.5 POST ....................................................................................................................................... 108
6.19.6 DELETE ................................................................................................................................... 108

6.20 RESOURCE: COLLECTION OF SHARED LISTS .................................................................... 108

6.20.1 Request URL variables ........................................................................................................... 109
6.20.2 Response Codes and Error Handling ...................................................................................... 109
6.20.3 GET ........................................................................................................................................... 109

6.20.3.1 Example 1: Retrieve lists shared by a user identified by otherUserId with all list attributes and all member attributes (default) (Informative) ........................................................................ 110
6.20.3.2 Example 2: Retrieve lists shared by a user identified by otherUserId without lists attributes, but with selected member attributes (Informative) ........................................................................ 112

6.20.4 PUT ........................................................................................................................................... 113
6.20.5 POST ....................................................................................................................................... 113
6.20.6 DELETE ................................................................................................................................... 113

6.21 RESOURCE: INDIVIDUAL SHARED LIST ............................................................................. 113

6.21.1 Request URL variables ........................................................................................................... 114
6.21.2 Response Codes and Error Handling ...................................................................................... 114
6.21.3 GET ........................................................................................................................................... 114

6.21.3.1 Example 1: Retrieve individual list shared by a user identified by otherUserId, with all list attributes and all member attributes (default) (Informative) ........................................................................ 115
6.21.3.2 Example 2: Retrieve individual list shared by a user identified by otherUserId, with all list attributes and no member attributes (default) (Informative) ........................................................................ 116

6.21.4 PUT ........................................................................................................................................... 117
6.21.5 POST ....................................................................................................................................... 117
6.21.6 DELETE ................................................................................................................................... 117

6.22 RESOURCE: MEMBERS IN A SHARED LIST .......................................................................... 117

6.22.1 Request URL variables ........................................................................................................... 117
6.22.2 Response Codes and Error Handling ...................................................................................... 118
6.22.3 GET ........................................................................................................................................... 118

6.22.3.1 Example: Retrieve all members for a shared list belonging to a user with all attributes (default) (Informative) ........................................................................ 118

6.22.4 PUT ........................................................................................................................................... 119
6.22.5 POST ....................................................................................................................................... 119
6.22.6 DELETE ................................................................................................................................... 119

6.23 RESOURCE: INDIVIDUAL MEMBER INFORMATION FROM SHARED LIST .................. 120

6.23.1 Request URL variables ........................................................................................................... 120
6.23.2 Response Codes and Error Handling ...................................................................................... 120
6.23.3 GET ........................................................................................................................................... 120

6.23.3.1 Example 1: Retrieve all member attributes about user identified as memberld from the list shared by a user identified by otherUserId (default) (Informative) ........................................................................ 121
6.23.3.2 Example 2: Retrieve selected member attributes about user identified as memberld from the list shared by a user identified by otherUserId (default) (Informative) ........................................................................ 121

6.23.4 PUT ........................................................................................................................................... 122
6.23.5 POST ....................................................................................................................................... 122
6.23.6 DELETE ................................................................................................................................... 122

6.24 RESOURCE: AUTHORIZATION RULES .............................................................................. 122

6.24.1 Request URL variables ........................................................................................................... 122
6.24.2 Response Codes and Error Handling .................................................................................................. 123
6.24.3 GET .................................................................................................................................................. 123
6.24.3.1 Example: Retrieve all authorization rules (Informative) ................................................................. 123
6.24.4 PUT .................................................................................................................................................. 124
6.24.5 POST ............................................................................................................................................... 124
6.24.5.1 Example: Create new rule for list and referenced contacts (Informative) ........................................ 124
6.24.6 DELETE ........................................................................................................................................... 125

6.25 RESOURCE: INDIVIDUAL AUTHORIZATION RULE ........................................................................ 125
6.25.1 Request URL variables ...................................................................................................................... 125
6.25.2 Response Codes and Error Handling ................................................................................................. 125
6.25.3 GET .................................................................................................................................................. 125
6.25.3.1 Example: Retrieve an authorization rule (Informative) .................................................................. 125
6.25.4 PUT .................................................................................................................................................. 126
6.25.4.1 Example: Update an existing authorization rule (Informative) ...................................................... 126
6.25.5 POST ............................................................................................................................................... 126
6.25.6 DELETE ........................................................................................................................................... 127

6.26 RESOURCE: INDIVIDUAL AUTHORIZATION RULE DATA ............................................................... 127
6.26.1 Request URL variables ...................................................................................................................... 128
6.26.2 Response Codes and Error Handling ................................................................................................. 128
6.26.3 GET .................................................................................................................................................. 128
6.26.3.1 Example: Retrieve data from an authorization rule (Informative) .................................................. 128
6.26.4 PUT .................................................................................................................................................. 128
6.26.4.1 Example: Authorize user by updating an existing authorization rule (Informative) ....................... 129
6.26.5 POST ............................................................................................................................................... 129
6.26.6 DELETE ........................................................................................................................................... 129
6.26.6.1 Example: Delete an authorized identity (Informative) ................................................................... 129

6.27 RESOURCE: INDIVIDUAL MEMBER TRANSFER ........................................................................ 130
6.27.1 Request URL variables ...................................................................................................................... 130
6.27.2 Response Codes and Error Handling ................................................................................................. 130
6.27.3 GET .................................................................................................................................................. 130
6.27.4 PUT .................................................................................................................................................. 130
6.27.5 POST ............................................................................................................................................... 130
6.27.5.1 Example: Transfer a member from one list to another (Informative) ........................................... 130
6.27.6 DELETE ........................................................................................................................................... 131

7. FAULT DEFINITIONS ............................................................................................................................... 132

7.1 SERVICE EXCEPTIONS ......................................................................................................................... 132
7.1.1 SVC0240: Key property changes not allowed .................................................................................. 132

7.2 POLICY EXCEPTIONS ........................................................................................................................... 132
7.2.1 POL0214: Too many resources ......................................................................................................... 132

APPENDIX A. CHANGE HISTORY (INFORMATIVE) .............................................................................. 133
A.1 APPROVED VERSION HISTORY .......................................................................................................... 133
A.2 DRAFT/CANDIDATE VERSION 1.0 HISTORY ..................................................................................... 133

APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) ............................................. 134
B.1 SCR FOR REST.AB SERVER ................................................................................................................ 135
B.1.1 SCR for REST.AB.ContactCollection Server .................................................................................. 135
B.1.2 SCR for REST.AB.IndividualContact Server .................................................................................. 135
B.1.3 SCR for REST.AB.AttributesForAContact Server ........................................................................... 135
B.1.4 SCR for REST.AB.IndividualAttributeForAContact Server ........................................................... 136
B.1.5 SCR for REST.AB.Lists Server ........................................................................................................... 136
B.1.6 SCR for REST.AB.IndividualList Server .......................................................................................... 136
B.1.7 SCR for REST.AB.AttributesForAList Server .................................................................................. 137
B.1.8 SCR for REST.AB.IndividualAttributeForAList Server ................................................................. 137
B.1.9 SCR for REST.AB.MemberInAList Server .......................................................................................... 137
B.1.10 SCR for REST.AB.IndividualMemberInAList Server ....................................................................... 137
B.1.11 SCR for REST.AB.AttributesForAMemberInAList Server .............................................................. 138
B.1.12 SCR for REST.AB.IndividualAttributeForAMemberInAList Server ............................................... 138
B.1.13 SCR for REST.AB.ListReferences Server ................................................................. 138
B.1.14 SCR for REST.AB.IndivListReference Server ....................................................... 139
B.1.15 SCR for REST.AB.List.Subscr Server ................................................................. 139
B.1.16 SCR for REST.AB.Individual.Subscr Server ....................................................... 139
B.1.17 SCR for REST.AB.Notif Server ............................................................................. 140
B.1.18 SCR for REST.AB.Shared.ContactCollection Server ............................................ 140
B.1.19 SCR for REST.AB.Individual.Shared.Contact Server ......................................... 140
B.1.20 SCR for REST.AB.Shared.Lists Server ................................................................. 141
B.1.21 SCR for REST.AB.Individual.Shared.List Server ................................................. 141
B.1.22 SCR for REST.AB.Members.Ind.Shared.List Server ........................................... 141
B.1.23 SCR for REST.AB.Member.Shared.List Server ..................................................... 141
B.1.24 SCR for REST.AB.Authorization.RuleList Server ............................................... 142
B.1.26 SCR for REST.AB.Indiv.AuthorizationRuleData Server ....................................... 142
B.1.27 SCR for REST.AB.IndividualMemberTransfer Server ........................................ 143

APPENDIX C. APPLICATION/X-WWW-FORM-URLENCODED REQUEST FORMAT FOR POST OPERATIONS (NORMATIVE) .................................................................................................................. 144

C.1 CREATE A SUBSCRIPTION FOR LIST CHANGES NOTIFICATIONS ................................................................. 144
C.1.1 Example using ‘tel’ URI (Informative) .................................................................. 145
C.1.1.1 Request ............................................................................................................... 145
C.1.1.2 Response .......................................................................................................... 146
C.1.2 Example using ‘acr’ URI (Informative) ................................................................. 146
C.1.2.1 Request ............................................................................................................... 146
C.1.2.2 Response .......................................................................................................... 146

C.2 CREATE AN AUTHORIZATION RULE ................................................................................................. 147
C.2.1 Example (Informative) ........................................................................................ 149
C.2.1.1 Request ............................................................................................................... 149
C.2.1.2 Response .......................................................................................................... 149

C.3 TRANSFER A MEMBER FROM ONE LIST TO ANOTHER ......................................................................................... 149
C.3.1 Example: (Informative) ..................................................................................... 150
C.3.1.1 Request ............................................................................................................... 150
C.3.1.2 Response .......................................................................................................... 150

APPENDIX D. JSON EXAMPLES (INFORMATIVE) ................................................................................................. 151

D.1 RETRIEVE ALL CONTACTS WITH ALL ATTRIBUTES (DEFAULT) (SECTION 6.1.3.1) ...................................................... 151
D.2 RETRIEVE A CONTACT WITH ALL ATTRIBUTES (DEFAULT) (SECTION 6.2.3.1) ...................................................... 152
D.3 RETRIEVE A CONTACT WITH ONLY vCard2.1 AS SELECTED ATTRIBUTE (SECTION 6.2.3.3) ................................ 153
D.4 CREATE A NEW CONTACT (SECTION 6.2.4.1) ................................................................................................. 154
D.5 RETRIEVE ALL ATTRIBUTES OF A CONTACT (SECTION 6.3.3.1) ................................................................. 155
D.6 RETRIEVE A CONTACT’S ATTRIBUTE (SECTION 6.4.3.1) ..................................................................................... 155
D.7 CREATE OR UPDATE A CONTACT’S ATTRIBUTE (SECTION 6.4.4.1) ............................................................................ 156
D.8 RETRIEVE A LIST WITH ALL ATTRIBUTES AND ALL MEMBER ATTRIBUTES (DEFAULT) (SECTION 6.6.3.1) ..................... 158
D.10 RETRIEVE A NON EXISTING LIST (SECTION 6.6.3.4) ..................................................................................... 159
D.11 CREATE A LIST (SECTION 6.6.4.1) ........................................................................ 159
D.12 RETRIEVE ALL ATTRIBUTES FOR A LIST BELONGING TO Bob (SECTION 6.7.3.1) .............................................. 160
D.13 RETRIEVE AN ATTRIBUTE VALUE (SECTION 6.8.3.1) ..................................................................................... 161
D.14 CREATE AN ATTRIBUTE (SECTION 6.8.4.1) ................................................................................................. 161
D.15 RETRIEVE ALL MEMBERS FOR A LIST BELONGING TO A USER WITH ALL ATTRIBUTES (DEFAULT) (SECTION 6.9.3.1) ................................................................................................. 162
D.16 RETRIEVE A MEMBER AND SELECTED ATTRIBUTES (SECTION 6.10.3.2) .................................................... 163
D.17 CREATE A MEMBER WITHOUT A LINK TO THE CONTACT (SECTION 6.10.4.1) ..................................................... 164
D.18 RETRIEVE ALL ATTRIBUTES OF A MEMBER OF A LIST (SECTION 6.11.3.1) ................................................. 164
D.19 RETRIEVE A MEMBER’S NON EXISTING ATTRIBUTE (SECTION 6.12.3.2) ......................................................... 165
D.20 CREATE A MEMBER’S ATTRIBUTE (SECTION 6.12.4.1) ..................................................................................... 165
D.21 RETRIEVE ALL LIST REFERENCES IN A LIST (SECTION 6.13.3.1) ................................................................. 166
D.22 RETRIEVE A LIST REFERENCE (SECTION 6.14.3.1) ......................................................................................... 167
D.23 CREATE A LIST REFERENCE (SECTION 6.14.4.1) ............................................................................................. 167

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
Figures

Figure 1 Resource structure defined by this specification .......................................................... 17
Figure 2 Managing contacts ........................................................................................................ 37
Figure 3 Accessing lists and members ....................................................................................... 38
Figure 4 Subscription to notifications about address book changes ........................................ 40
Figure 5 Flow for managing shared lists .................................................................................... 41
Figure 6 Data types for contacts, members and lists ................................................................. 193

Tables

Table 1: Parlay X operations mapping ....................................................................................... 184
Table 2: Light-weight resources for Address Book ................................................................. 185
1. Scope

This specification defines a RESTful Address Book API using an HTTP protocol binding, based on the similar API defined in [3GPP 29.199-13].
2. References

2.1 Normative References


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

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

[REST_SUP AddressBook] “XML schema for the RESTful Network API for Address Book”, Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_addressbook-V1_0, URL:http://www.openmobilealliance.org/


2.2 Informative References


[ParlayREST AddressListMgmt] “RESTful bindings for Parlay X Web Services – Address List Management”, Version 1.0, Open Mobile Alliance™, OMA-TS-ParlayREST_AddressListManagement-V1_0

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, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

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

<table>
<thead>
<tr>
<th>Client-side Notification URL</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long Polling</td>
<td>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.</td>
</tr>
<tr>
<td>Notification Channel</td>
<td>A channel created on the request of the client and used to deliver notifications from a server to an application. 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.</td>
</tr>
<tr>
<td>Notification Server</td>
<td>A server that is capable of creating and maintaining Notification Channels.</td>
</tr>
<tr>
<td>Server-side Notification URL</td>
<td>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.</td>
</tr>
</tbody>
</table>

3.3 Abbreviations

<p>| AB   | Address Book |
| ACR  | Anonymous Customer Reference |
| API  | Application Programming Interface |
| HTTP | HyperText Transfer Protocol |
| JSON | JavaScript Object Notation |
| OMA  | Open Mobile Alliance |
| MIME | Multipurpose Internet Mail Extensions |
| REST | REpresentational State Transfer |
| SCR  | Static Conformance Requirements |
| SIP  | Session Initiation Protocol |
| TS   | Technical Specification |
| URI  | Uniform Resource Identifier |
| URL  | Uniform Resource Locator |</p>
<table>
<thead>
<tr>
<th>VCard</th>
<th>Virtual-Information Card</th>
</tr>
</thead>
<tbody>
<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 for the RESTful Network API for Address Book (AB) contains the HTTP protocol binding based on the Parlay X Address List Management Web Services [3GPP 29.199-13] specification, 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, JSON, and application/x-www-form-urlencoded).

4.1 Version 1.0

The RESTful Network API for Address Book V1.0 is a republication of the ParlayREST Address List Management API V1.0 [ParlayREST_AddressListMgmt] as part of the suite of OMA RESTful Network APIs. Bug fixes and structural changes to fit that suite, but also functional changes have been applied.

Version 1.0 of the RESTful Network API for Address Book keeps supporting the following operations:

- Manage attributes related to a list
- Manage attributes related to a contact or a member in a list
- Manage contacts
- Manage lists
- Manage members in a list
- Manage nested lists related to a list
- Manage shared contacts or lists
- Manage subscriptions to contact or list changes
- Send notifications about contact or list changes

The following new functionality has been introduced:

- Transfer a member from one list to another
- Support “vCard 2.1” and “vCard 3.0” as contact attributes [RFC2426]
- Support for scope values used with authorization framework defined in [Autho4API_10]
- Support for Anonymous Customer Reference (ACR) as an end user identifier
- Support for “acr:Authorization” as a reserved keyword in a resource URL variable that identifies an end user
- Resources, data types and element names have been changed to allow future extensions and reflect the API renaming from Address List Management to Address Book
5. Address Book API definition

This section is organized to support a comprehensive understanding of the RESTful Address Book 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 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 5.1). What follows are the data structures (section Error! Reference source not found.). A sample of typical use cases is included in section 5.3, described as high level flow diagrams.

Section 6 contains the detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP commands, the possible HTTP response codes, and the supported HTTP verbs. For each supported HTTP verb, a description of the functionality is provided, along with an example of a request and an example of a response. For each unsupported HTTP verb, the returned HTTP error status is specified, as well as what should be returned in the Allow header.

All examples in section 6 use XML as the format for the message body. Application/x-www-form-urlencoded examples are provided in Appendix C, while JSON examples are provided in Appendix D. 0 provides the Static Conformance Requirements (SCR).

Appendix E lists the Parlay X equivalent method for each supported REST resource and method combination, 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.

Finally, Appendix A illustrates the most common data structures and their relationships in Address Book.

Note: Throughout this document client and application can be used interchangeably.

5.1 Resources Summary

This section summarizes all the resources used by the RESTful Address Book API.

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 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.
Figure 1 Resource structure defined by this specification

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.
Purpose: To allow client to manage all contacts

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</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>Collection of contacts</td>
<td>/{userId}/contacts</td>
<td>ContactCollection</td>
<td>This operation retrieves information about all user’s contacts.</td>
</tr>
<tr>
<td>Individual contact</td>
<td>/{userId}/contacts/{contactId}</td>
<td>Contact</td>
<td>This operation retrieves information about individual contact from the collection of user’s contacts.</td>
</tr>
<tr>
<td>Attributes for a contact</td>
<td>/{userId}/contacts/{contactId}/attributes</td>
<td>AttributeList</td>
<td>This operation returns all attributes for a contact.</td>
</tr>
<tr>
<td>Individual attribute for a contact</td>
<td>/{userId}/contacts/{contactId}/attributes/[ResourceRelPath]</td>
<td>Attribute</td>
<td>This operation returns the value of the attribute for a contact.</td>
</tr>
</tbody>
</table>
### Purpose
To allow client to manage its lists and members in lists

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists</td>
<td>/{userId}/lists</td>
<td>ListCollection</td>
<td>GET: This operation retrieves all lists the user has created. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual list</td>
<td>/{userId}/lists/{listId}</td>
<td>List</td>
<td>GET: This operation retrieves the list. PUT: This operation creates or updates the entire list. POST: no DELETE: This operation removes the list from the system including attributes and members.</td>
</tr>
<tr>
<td>Attributes for a list</td>
<td>/{userId}/lists/{listId}/attributes</td>
<td>AttributeList</td>
<td>GET: This operation returns all attributes for a list. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual attribute for a list</td>
<td>/{userId}/lists/{listId}/attributes/[ResourceRelPath ]</td>
<td>Attribute</td>
<td>GET: This operation returns the value of the list attribute. PUT: This operation creates or updates an attribute POST: no DELETE: This operation deletes an attribute</td>
</tr>
<tr>
<td>Members in a list</td>
<td>/{userId}/lists/{listId}/members</td>
<td>MemberCollection</td>
<td>GET: This operation retrieves the members in the list. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}</td>
<td>Member</td>
<td>GET: This operation retrieves a user from a list. (Normally only used to verify the existence of the member) PUT: This operation creates and updates a member of the list POST: no DELETE: This operation removes the member from the list</td>
</tr>
<tr>
<td>Attributes for a member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}/attributes</td>
<td>AttributeList</td>
<td>GET: This operation returns all attributes for a member. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual attribute for a member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}/attributes/[ResourceRelPath]</td>
<td>Attribute</td>
<td>GET: This operation returns the value of the attribute for a member. PUT: This operation creates or updates an attribute for a POST: no DELETE: This operation removes an attribute for a</td>
</tr>
<tr>
<td>Resource</td>
<td>Base URL: http://{serverRoot}/addressbook/{apiVersion}</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>MemberTransferParameters</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(used for POST request) commo:ResourceReference (used for POST response)</td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>DELETE</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>member.</td>
<td></td>
</tr>
<tr>
<td>Individual member transfer</td>
<td>/{userId}/lists/{listId}/members/{memberId}/transfer</td>
<td>MemberTransferParameters</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(used for POST request) commo:ResourceReference (used for POST response)</td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This operation transfers a member from one list to another list</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>List references</td>
<td>/{userId}/lists/{listId}/listReferences</td>
<td>ListReferenceCollection</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This operation returns a list of references to other lists.</td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>DELETE</td>
</tr>
<tr>
<td>Individual list reference</td>
<td>/{userId}/lists/{listId}/listReferences/[ResourceRelPath]</td>
<td>common:Link</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This operation returns the value of list reference.</td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This operation creates a new list reference.</td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This operation removes a list reference.</td>
<td></td>
</tr>
</tbody>
</table>
Purpose: To allow client to manage subscriptions to address book changes

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Book changes subscriptions</td>
<td>/{userId}/subscriptions/abChanges</td>
<td>AbChangesSubscriptionCollection</td>
<td>This operation retrieves all active subscriptions for address book changes.</td>
</tr>
<tr>
<td>Individual address book changes subscription</td>
<td>/{userId}/subscriptions/abChanges/{subscriptionId}</td>
<td>AbChangesSubscription</td>
<td>This operation retrieves information about individual subscription.</td>
</tr>
</tbody>
</table>
Purpose: To allow client to receive notifications about address book changes

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client resource for address book changes notifications</td>
<td>&lt;specified by client in the subscription&gt;</td>
<td>AbChangesNotification</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>
### Purpose: To allow client access to the contacts shared by another user

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of shared contacts</td>
<td>/{userId}/shared/{otherUserId}/contacts</td>
<td>ContactCollection</td>
<td><strong>GET</strong> This operation retrieves all contacts shared by another user identified by <code>{otherUserId}</code> with the requesting user identified by <code>{userId}</code>.</td>
</tr>
<tr>
<td>Individual shared contact</td>
<td>/{userId}/shared/{otherUserId}/contacts/{sharedIdentity}</td>
<td>Contact</td>
<td><strong>GET</strong> This operation retrieves information about individual contact shared by user identified by <code>{otherUserId}</code> with requesting user identified by <code>{userId}</code>.</td>
</tr>
</tbody>
</table>
Purpose: To allow client access to the lists shared by another user

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of shared lists</td>
<td>/{userId}/shared/{otherUserId}/lists</td>
<td>ListCollection</td>
<td>GET: This operation retrieves all lists shared by another user identified by {otherUserId} with the requesting user identified by {userId}. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual shared list</td>
<td>/{userId}/shared/{otherUserId}/lists/{sharedListId}</td>
<td>List</td>
<td>GET: This operation retrieves information about individual list shared by user identified by {otherUserId} with requesting user identified by {userId}. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Members in a shared list</td>
<td>/{userId}/shared/{otherUserId}/lists/{sharedListId}/members</td>
<td>MemberCollection</td>
<td>GET: This operation retrieves the members in the list shared by user identified by {otherUserId} with requesting user identified by {userId}. PUT: no POST: no DELETE: no</td>
</tr>
<tr>
<td>Individual member information from shared list</td>
<td>/{userId}/shared/lists/{otherUserId}/{sharedListId}/members/{memberId}</td>
<td>Member</td>
<td>GET: This operation retrieves individual member information from the shared list. PUT: no POST: no DELETE: no</td>
</tr>
</tbody>
</table>
Purpose: To allow client access to authorization rules for shared contacts and lists

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Data Structures</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization Rules</td>
<td>/(userId)/authorizationRules</td>
<td>RuleList</td>
<td>GET: This operation retrieves all authorization rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>PUT: no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>common:ResourceReference (optional alternative for POST response)</td>
<td>POST: This operation creates new authorization rule</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>DELETE: no</td>
</tr>
<tr>
<td>Individual authorization rule</td>
<td>/(userId)/authorizationRules/ {ruleId}</td>
<td>Rule (Used for PUT/GET)</td>
<td>GET: This operation retrieves individual authorization rule</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>PUT: This operation updates individual authorization rule</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>DELETE: This operation removes individual authorization rule</td>
</tr>
<tr>
<td>Individual authorization rule data</td>
<td>/(userId)/authorizationRules/ {ruleId}/[ResourceRelPath]</td>
<td>The data structure corresponds to the element pointed out by the request-URL. (Used for PUT/GET)</td>
<td>GET: This operation retrieves data as specified in the URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>PUT: This operation creates or updates data as specified in the URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule</td>
<td>DELETE: This operation removes the data as specified in the URL</td>
</tr>
</tbody>
</table>
5.2 Data Types

5.2.1 XML Namespaces

The namespace for the Address Book data types is:

```
urn:oma:xml:rest:netapi:addressbook:1
```

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

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

Applications following the RESTful Network API for Address Book V 1.0 specification SHALL use the namespace urn:oma:xml:rest:netapi:addressbook:1.

5.2.2 Structures

The subsections of this section define the data structures used in the RESTful Address Book API.

Some of the structures can be instantiated as so-called root elements, i.e. they define the type of a representation of a so-called heavy-weight resource.

The column [ResourceRelPath] in the tables below, if used, includes relative resource paths for light-weight resource URLs that are used to access individual elements in the data structure (so-called light-weight resources). A string from this column needs to be appended to the corresponding heavy-weight resource URL in order to create light-weight resource URL for that particular element in the data structure. “Not applicable” means that individual access to that element is not supported. The root element and data type of the resource associated with the [ResourceRelPath] are defined by the Element and Type columns in the row that defines the [ResourceRelPath].

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

5.2.2.1 Type: ContactCollection

Collection of all individual contacts for a user, in the form of a flat unique list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contact</td>
<td>Contact [0..unbounded]</td>
<td>Yes</td>
<td>Contains a list of user’s contacts</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 contactCollection of type ContactCollection is allowed in response bodies.
### 5.2.2.2 Type: Contact

Individual contact information in the collection of contacts

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contactId</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Contains a relative URI of the contact (e.g. “Alice”). It is a key property of the element and SHALL NOT be altered when included in the light-weight resource URL. If contactId is also part of the request URL, the two MUST have the same value.</td>
</tr>
<tr>
<td>sharedIdentity</td>
<td>SharedIdentity</td>
<td>Yes</td>
<td>Contains a list of publicly known identities (absolute URIs) of the contact. A shared identity MAY be used by a requesting user to identify the contact in addition to the contactId (since the contactId may not be known by others).</td>
</tr>
<tr>
<td>attributeList</td>
<td>AttributeList</td>
<td>Yes</td>
<td>Contains a list of attributes (e.g. “display-name”) related to a contact.</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>
<tr>
<td>link</td>
<td>common:Link [0..unbounded]</td>
<td>Yes</td>
<td>MAY contain references to members (rel=“Member”) in type List associated with the contact. When a member containing a link to a contact is added to a list (or updated), a reverse link SHALL be created automatically by the server pointing from the contact to the actual member. When a member containing a link to a contact is removed from a list (or member is updated and link removed), the reverse link pointing from the contact to the member SHALL be removed automatically by the server. Links SHALL NOT be included in responses for a shared contact, unless the member referred by the link is a member in a list shared with the requester.</td>
</tr>
</tbody>
</table>

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

### 5.2.2.3 Type: SharedIdentity

List of publicly known identities

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sharedId</td>
<td>xsd:anyURI [0..unbounded]</td>
<td>Yes</td>
<td>Contains a list of absolute URIs (e.g. ‘sip’ URI, ‘tel’ URI, ‘acr’ URI) representing contact identities or list identities known to other users, used when retrieving a contact, respectively alist.</td>
</tr>
</tbody>
</table>
5.2.2.4 Type: ListCollection

List of lists

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>List</td>
<td>[0..unbounded]</td>
<td>Yes</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 listCollection of type ListCollection is allowed in response bodies.

5.2.2.5 Type: List

List of identities

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Contains a relative URI representing the list identity (e.g. “friends”).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It is a key property of the element and SHALL NOT be altered when included in the light-weight resource URL. If listId is also part of the request URL, the two MUST have the same value.</td>
</tr>
<tr>
<td>memberCollection</td>
<td>MemberCollection</td>
<td>Yes</td>
<td>Members in the list</td>
</tr>
<tr>
<td>listReferenceCollection</td>
<td>ListReferenceCollection</td>
<td>Yes</td>
<td>Contains references to other lists.</td>
</tr>
<tr>
<td>category</td>
<td>ListType</td>
<td>[0..unbounded]</td>
<td>Yes</td>
</tr>
<tr>
<td>sharedListIdentity</td>
<td>SharedIdentity</td>
<td>Yes</td>
<td>Contains a list of publicly known identities (absolute URIs) of the list (e.g. <a href="mailto:friendslist@example.com">mailto:friendslist@example.com</a>). A shared identity MAY be used by a requesting user to identify the list in addition to the listId (since the listId may not be known by others).</td>
</tr>
<tr>
<td>attributeList</td>
<td>AttributeList</td>
<td>Yes</td>
<td>Contains a list of attributes related to a list.</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 list of type List is allowed in response bodies.

5.2.2.6 Type: AttributeList

Attributes of a list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>[ResourceRelPath]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>Attribute</td>
<td>[0..unbounded]</td>
<td>Yes</td>
<td>{name}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The sub-element ‘name’ of the type Attribute SHALL NOT be altered when this element is accessed as a light-weight resource.</td>
</tr>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Not applicable</td>
<td>Self referring URL</td>
</tr>
</tbody>
</table>

A root element named attributeList of type AttributeList is allowed in request and/or response bodies.
Column [ResourceRelPath] includes relative resource paths for light-weight resource URLs that are used to access individual elements in the data structure. A string from this column needs to be appended to the corresponding heavy-weight resource URL in order to create light-weight resource URL for that particular element in the data structure. “Not applicable” means that individual access to that element is not supported. The root element and data type of the resource associated with the [ResourceRelPath] are defined by the Element and Type columns in the row that defines the [ResourceRelPath].

### 5.2.2.7 Type: Attribute

Individual attribute of a list

<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>Name of the attribute. The following names are predefined by this specification: “display-name”. It is a key property of the element and SHALL NOT be altered when included in the light-weight resource URL. “vCard2.1” and “vCard3.0” are reserved names, and SHALL only be used to access the entire vCard as an opaque object [RFC2426]. When ‘name’ is “vCard2.1” or “vCard3.0”, it MUST be associated with the ‘objectValue’ choice to access an entire opaque vCard2.1 or vCard3.0, respectively.</td>
</tr>
<tr>
<td>value</td>
<td>xsd:string</td>
<td>Choice</td>
<td>Optional element; if present it provides the value of the attribute.</td>
</tr>
<tr>
<td>objectValue</td>
<td>xsd:base64Binary</td>
<td>Choice</td>
<td>Optional element; if present it contains a base64Binary object. In case name is “vCard2.1” this element SHALL contain a base64Binary encoded vCard2.1 object. In case name is “vCard3.0” this element SHALL contain a base64Binary encoded vCard3.0 object.</td>
</tr>
<tr>
<td>&lt;any element&gt;</td>
<td>&lt; type is defined in a schema implementing the element&gt;</td>
<td>Choice</td>
<td>Optional element; if present it provides the value of the attribute. Note that element ‘any element’ can be any element from any other namespace (schema) than the target namespace. Type of such element is defined by the schema implementing the element. In XML implementations, element “any” must be qualified with the namespace prefix.</td>
</tr>
</tbody>
</table>

XSD modelling uses an optional “choice” to select either a value or an objectValue or <any element>, or none of them.

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

### 5.2.2.8 Type: MemberCollection

Collection of members that are part of a list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>member</td>
<td>Member [0..unbounded]</td>
<td>Yes</td>
<td>Contains a list of members related to a members list.</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 memberCollection of type MemberCollection is allowed in response bodies.
5.2.2.9 Type: Member

Individual member information in a list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memberId</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Contains an identifier of a single member (e.g. 'sip' URI, 'tel' URI, 'acr' URI). It is a key property of the element and SHALL NOT be altered when included in the light-weight resource URL. If memberId is also part of the request URL, the two MUST have the same value.</td>
</tr>
<tr>
<td>attributeList</td>
<td>AttributeList</td>
<td>Yes</td>
<td>Contains a list of attributes (e.g. display name) related to a member.</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>
<tr>
<td>link</td>
<td>common:Link [0..unbounded]</td>
<td>Yes</td>
<td>MAY contain references to contacts (rel=&quot;Contact&quot;) in type 'ContactCollection' associated with the member. When a contact containing a link to a member is added to the 'ContactCollection' (or updated), a reverse link SHALL be created automatically by the server pointing from the member to the actual contact. When a contact containing a link to a member is removed from the 'ContactCollection' (or contact is updated and link removed), the reverse link pointing from the member to the contact SHALL be removed automatically by the server. Links SHALL NOT be included in responses for a shared list, unless the contact referred by the link is also shared with the requester.</td>
</tr>
</tbody>
</table>

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

5.2.2.10 Type: ListReferenceCollection

List of references to other lists

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>[ResourceRelPath]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Not applicable</td>
<td>Self referring URL</td>
</tr>
<tr>
<td>link</td>
<td>common:Link [0..unbounded]</td>
<td>Yes</td>
<td>{href}</td>
<td>Contains references to other lists (rel=&quot;List&quot;).</td>
</tr>
</tbody>
</table>

A root element named listReferenceCollection of type ListReferenceCollection is allowed in response bodies.
5.2.2.11 Type: AbChangesSubscriptionCollection

Collection of subscriptions to notifications of changes in contacts and/or lists

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abChangesSubscription</td>
<td>AbChangesSubscription [0..unbounded]</td>
<td>Yes</td>
<td>Contains a list of subscriptions related to contacts changes and/or list changes.</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 abChangesSubscriptionCollection of type AbChangesSubscriptionCollection is allowed in response bodies.

5.2.2.12 Type: AbChangesSubscription

Individual subscription to notifications of changes in contacts and/or a list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anyContacts</td>
<td>(empty)</td>
<td>Choice</td>
<td>Indicates a subscription for changes about any contacts in the 'ContactCollection'.</td>
</tr>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>Indicates a subscription for changes about the specified list.</td>
</tr>
<tr>
<td>callbackReference</td>
<td>common:CallbackReference</td>
<td>No</td>
<td>Client's notification URL and OPTIONAL callbackData</td>
</tr>
<tr>
<td>clientCorrelator</td>
<td>xsd:string</td>
<td>Yes</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 field SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids re-sending the message in such situations. In case the field 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>
</tr>
<tr>
<td>applicationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>A tag that the client MAY use to tag this particular resource on the server. In case the field 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>
</tr>
<tr>
<td>duration</td>
<td>xsd:int</td>
<td>Yes</td>
<td>Subscription duration in seconds. Server would expire subscription after specified number of seconds after subscription creation. If not specified – default value assigned 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>
XSD modelling uses a “choice” to select either anyContacts or listId.
A root element named abChangesSubscription of type AbChangesSubscription is allowed in request and 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 document [REST_NetAPI_Common] provides a recommendation regarding the generation of the value of this field.

### 5.2.2.13 Type: AbChangesNotification

Notifications of changes in contacts and/or a list

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 of changes in contacts and/or a list. See [REST_NetAPI_Common] for details.</td>
</tr>
<tr>
<td>resourceStatus</td>
<td>ResourceStatus</td>
<td>No</td>
<td>Indicates the state for the subscription.</td>
</tr>
<tr>
<td>duration</td>
<td>xsd:int</td>
<td>Yes</td>
<td>Represents the time in seconds that the subscription is active, starting from the time the subscription is created by the server.</td>
</tr>
<tr>
<td>link</td>
<td>common:Link [1..unbounded]</td>
<td>No</td>
<td>SHALL contain reference to the added or updated member or contact (rel=&quot;Member&quot; or rel=&quot;Contact&quot; respectively). A notification MAY include several links for each member or contact. SHALL contain reference to the list in case a member or contact has been removed (rel=&quot;List&quot; or rel=&quot;ContactCollection&quot; respectively). SHALL contain reference to the subscription (rel=&quot;AbChangesSubscription&quot;) if resourceStatus is ‘Active’.</td>
</tr>
</tbody>
</table>

A root element named abChangesNotification of type AbChangesNotification is used by server in the POST request to the client informing it about change in the list that client is monitoring (using one of the subscriptions).

### 5.2.2.14 Type: RuleList

List of authorization rules

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule</td>
<td>Rule [0..unbounded]</td>
<td>Yes</td>
<td>Contains a list of all authorization rules related to contacts and lists.</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 ruleList of type RuleList is allowed in response bodies.
### 5.2.2.15 Type: Rule

Individual authorization rule

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleName</td>
<td>xsd:ID</td>
<td>No</td>
<td>A name associated with the rule. It is a key property of the rule and SHALL NOT be altered when included in the light-weight resource URL.</td>
</tr>
<tr>
<td>clientCorrelator</td>
<td>xsd:string</td>
<td>Yes</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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This field SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids re-sending the message in such situations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In case the field 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>
</tr>
<tr>
<td>applicationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>A tag that the client MAY use to tag this particular resource on the server. In case the field 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>
</tr>
<tr>
<td>applicableToContacts</td>
<td>ApplicableToContacts</td>
<td>Yes</td>
<td>Defines for which contacts the rule applies for.</td>
</tr>
<tr>
<td>applicableToLists</td>
<td>ApplicableToLists</td>
<td>Yes</td>
<td>Defines for which lists the rule applies for.</td>
</tr>
<tr>
<td>applicableToListsAndContacts</td>
<td>ApplicableToListsAndLinkedContacts</td>
<td>Yes</td>
<td>Defines for which lists and referenced contacts the rule applies for.</td>
</tr>
<tr>
<td>authorizedTo</td>
<td>AuthorizedTo</td>
<td>No</td>
<td>Defines who have access to lists and contacts.</td>
</tr>
<tr>
<td>listFilter</td>
<td>xsd:string</td>
<td>[0..unbounded] Yes</td>
<td>Filter for list attributes (section 6.5) the retrieving user is allowed to see. An empty filter means that the user has access to all attributes.</td>
</tr>
<tr>
<td>indivFilter</td>
<td>xsd:string</td>
<td>[0..unbounded] Yes</td>
<td>Filter for member and contact attributes (section 6.1 and 6.9) the retrieving user is allowed to see. An empty filter means that the user has access to all attributes.</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 rule of type Rule is allowed in request and response bodies.

Regarding the clientCorrelator field, the note in section 5.2.2.12 applies.
5.2.2.16 Type: ApplicableToContacts

Contacts the rule applies to

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contactId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>The rule applies to the specified list of 'contactId' that are part of the 'ContactCollection'.</td>
</tr>
<tr>
<td>allContacts</td>
<td>(empty)</td>
<td>Choice</td>
<td>The rule applies to all contacts.</td>
</tr>
</tbody>
</table>

XSD modelling uses an optional “choice” to select either contactId or allContacts.

5.2.2.17 Type: ApplicableToLists

Lists the rule applies to

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>The rule applies to the specified list of 'listId'.</td>
</tr>
<tr>
<td>allLists</td>
<td>(empty)</td>
<td>Choice</td>
<td>The rule applies to all lists.</td>
</tr>
</tbody>
</table>

XSD modelling uses an optional “choice” to select either listId or allLists.

5.2.2.18 Type: ApplicableToListsAndLinkedContacts

Lists and referenced contacts the rule applies to

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>The rule applies to the specified list of 'listId'. The rule applies also to contacts that are referenced from the specified list.</td>
</tr>
<tr>
<td>allLists</td>
<td>(empty)</td>
<td>Choice</td>
<td>The rule applies for all lists and referenced contacts.</td>
</tr>
</tbody>
</table>

XSD modelling uses an optional “choice” to select either listId or allLists.

5.2.2.19 Type: AuthorizedTo

Authorized identities

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>[ResourceRelPath]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>users/{userId}</td>
<td>User is authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>allSharedIds</td>
<td>(empty)</td>
<td>Choice</td>
<td>Not applicable</td>
<td>All contacts with shared identities are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>lists/{listId}</td>
<td>Users within the list are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>allLists</td>
<td>(empty)</td>
<td>Choice</td>
<td>Not applicable</td>
<td>Users within all lists are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>link</td>
<td>common:Link</td>
<td>Choice</td>
<td>externalLists/{href}</td>
<td>Users within the list (owned by another user) are authorized to access a resource for which the rule is applicable to. Link specified using rel=&quot;List&quot;.</td>
</tr>
</tbody>
</table>
domainName xsd:string 
[0..unbounded] Choice domains/{domainName} Users within the domain are authorized to access a resource for which the rule is applicable to.

otherUser (empty) Choice Not applicable Users not authorized by any other rule are authorized to access a resource for which the rule is applicable to.

XSD modelling uses an optional “choice” to select either userId or allSharedIdentities or listId or allLists or link or domainName or otherUser.

5.2.2.20 Type: MemberTransferParameters

Set of parameters for the member transfer request

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Contains a relative URI representing the list where the member is to be transferred.</td>
</tr>
</tbody>
</table>

A root element named memberTransferParameters of type MemberTransferParameters is allowed in request bodies.

5.2.3 Enumerations

The subsections of this section define the enumerations used in the RESTful Address Book API.

5.2.3.1 Enumeration: ListType

Enumeration of types of lists

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URIList</td>
<td>Indicates that list is used as a list of URIs, such as a presence list, email distribution list.</td>
</tr>
<tr>
<td>GroupURIList</td>
<td>Indicates that list is used to store a list of URIs intended for group communication.</td>
</tr>
<tr>
<td>Group</td>
<td>Indicates that list is used for group communication, such as conferencing etc.</td>
</tr>
</tbody>
</table>

5.2.3.2 Enumeration: ResourceStatus

Enumeration of possible status of a subscription

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Indicates that the subscription is active and authorized.</td>
</tr>
<tr>
<td>TerminatedTimeout</td>
<td>Indicates that the subscription has been terminated. The subscription was not refreshed in time before it expired.</td>
</tr>
<tr>
<td>Pending</td>
<td>Indicates that the subscription is awaiting an authorization decision.</td>
</tr>
<tr>
<td>TerminatedBlocked</td>
<td>Indicates that the subscription has been terminated. The subscription was blocked.</td>
</tr>
<tr>
<td>TerminatedNoResource</td>
<td>Indicates that the subscription has been terminated. The intended resource does not exist.</td>
</tr>
<tr>
<td>TerminatedOther</td>
<td>Indicates that the subscription has been terminated of an unknown reason.</td>
</tr>
</tbody>
</table>
5.2.4 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 (list is non-exhaustive, and can be extended):

- Attribute
- AttributeList
- Contact
- AbChangesSubscription
- AbChangesSubscriptionCollection
- Member
- MemberCollection
- List
- ListCollection
- ListReferenceCollection
- MemberTransferParameters

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

5.3 Sequence Diagrams

The following sub-sections describe the resources, methods and steps involved in typical scenarios.

5.3.1 Managing contacts in the collection of contacts

The figure below shows various ways to retrieve and manage contacts in the collection of contacts. There is one application acting on behalf of userId. The application retrieves all contacts of userId, creates/updates a new contact for userId or deletes a contact for userId in the collection of contacts.

The resources:

- To retrieve all the user’s contacts the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts

- To retrieve, add, update or delete one contact in the collection of contacts the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}

- To retrieve all attributes of a specific contact the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes

- To retrieve, create, update or delete a single attribute of a specific contact the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/{ResourceRelPath}
1. The application retrieves all contacts for userId, by performing a GET on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts
   The result contains all the contacts for userId.

2. The application retrieves one contact for userId, by performing a GET on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}
   The result contains the information for one selected contact.

3. The application adds/updates a new contact by performing a PUT on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}
   The result contains a return code indicating whether or not the operation was successful.

4. The application deletes one contact from the collection of all contacts by performing a DELETE on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}
   The result contains a return code indicating whether or not the operation was successful.

5. The application retrieves all attributes of a specific contact by performing a GET on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes

6. The application retrieves a single attribute of a specific contact by performing GET on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/{ResourceRelPath}

7. The application creates or updates a single attribute of a specific contact by performing PUT on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/{ResourceRelPath}

8. The application deletes a single attribute of a specific contact by performing DELETE on the following resource:
   http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/{ResourceRelPath}
5.3.2 Accessing the lists and members

The figure below shows various ways to retrieve and manipulate data in lists. There is one application acting on behalf of Alice. The application is interested in one specific list, the list with id 'myFriends'. In that list there is one specific member, called Bob that is used in this example.

The resources:
- To fetch all the lists the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists
- To add a new list the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}
- To fetch all the members of a list the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members
- To add a specific member to a list the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}
- To fetch all attributes for a list this resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/attributes
- To fetch all attributes of a specific member the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}/attributes
- To fetch a single attribute of a specific member the following resource is used:
  http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}/attributes/[ResourceRelPath]

Figure 3 Accessing lists and members
1. The application fetches all the lists for the userId Alice by doing a GET on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists\]
   
   The result contains all the lists for Alice.

2. The application adds a new list called myFriends by doing a PUT on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends\]
   
   The result contains a return code indicating whether or not the operation was successful.

3. The application fetches all the members of the myFriends list by doing a GET on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/members\]
   
   The response will contain all the members of the myFriends list from Alice, but will not contain the attributes of the list.

4. The application adds a new member called Bob to the myFriends list doing a PUT on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/members/\{bobUserId\}\]
   
   The result contains a return code indicating whether or not the operation was successful.

5. The application fetches all attributes for the myFriends list by doing a GET on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/attributes\]
   
   The response will contain all attributes of the myFriends list.

6. The application fetches all attributes of Bob in the myFriends list by doing a GET on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/members/\{bobUserId\}/attributes\]
   
   The result contains all the attributes of Bob in this list.

7. The application fetches the display name attribute of Bob in the myFriends list by doing a GET on the following resource:
   
   \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/members/\{bobUserId\}/attributes/display-name\]
   
   The result contains Bob’s display name attribute.

### 5.3.3 Subscribing to address book changes and receiving notifications

The figure below shows how a client can subscribe for notifications about contacts and list changes and receive notifications.

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 via Long Polling.

The resources:

- To retrieve all the subscriptions to lists changes, and to create a new subscription the following resource is used:
  
  \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{userId\}/subscriptions/abChanges\]

- To retrieve or delete an individual subscription, the following resource is used:
  
  \[http://\{serverRoot\}/addressbook/\{apiVersion\}/\{userId\}/subscriptions/abChanges/\{subscriptionId\}\]
1. The application retrieves list of all subscriptions for list changes for the userId by performing a GET on the following resource:

   \[ http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/subscriptions/abChanges \]

   The result contains all the subscriptions created by Alice.

2. The application retrieves an individual subscription identified by subscriptionId by performing a GET on the following resource:

   \[ http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/subscriptions/abChanges/\{subscriptionId\} \]

   The result contains a return code indicating whether or not the operation was successful.

3. The application creates a new subscription for address book changes notification by performing a POST on the following resource, including in the request body a client application supplied notifyURL:

   \[ http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/subscriptions/abChanges \]

   If successful (as shown), the response will contain the complete representation of the newly created resource for subscription, including a self reference to the created resource.

4. The application adds a new member called Bob to the myFriends list doing a PUT on the following resource:

   \[ http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/lists/myFriends/members/\{bobUserId\} \]

   The result contains a return code indicating whether or not the operation was successful.

5. The server notifies the application about the changes in his myFriends list, by performing a POST on the client application supplied notifyURL in the newly created subscriptionId:

   \[ http://\{client-supplied notifyURL\} \]

   The result contains a return code indicating whether or not the operation was successful.

6. The application deletes the subscription identified by subscriptionId by performing a DELETE on the resource:

   \[ http://\{serverRoot\}/addressbook/\{apiVersion\}/\{aliceUserId\}/subscription/abChanges/\{subscriptionId\} \]

   The result was successful.
5.3.4 Managing shared lists

The figure below shows how to manage shared lists.

The resources:

- To retrieve all lists shared by another user the following resource is used:
  \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists\]

- To retrieve an individual list shared by another user the following resource is used:
  \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}\]

- To retrieve a member’s information from a list shared by another user the following resource is used:
  \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}/members/{memberId}\]

![Flow for managing shared lists](image)

Figure 5 Flow for managing shared lists

1. The application retrieves a list of all lists shared by the user identified by otherUserId with the user identified by userId, by performing a GET on the following resource:
   \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists\]
   
   The result contains all the lists that otherUserId shares with userId.

2. The application retrieves an individual list shared by the user identified by otherUserId with the user identified by userId, by performing a GET on the following resource:
   \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}\]
   
   The result contains all the members in the shared list.

3. The application retrieves member’s information from a list shared by the user identified by otherUserId with the user identified by userId, by performing a GET on the following resource:
   \[http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}/members/{sharedMemberId}\]
The result contains the selected member’s information.

Note: sequence flows for managing shared contacts are similar.

5.3.5 Managing authorization rules

The figure below shows how to manage authorization rules for sharing lists.

The resources:

- To retrieve all authorization rules and to create a new authorization rule the following resource is used:
  
  \[ \text{http://}\{\text{serverRoot}\}/\text{addressbook/}{\text{apiVersion}}/\{\text{userId}\}/\text{authorizationRules} \]

- To retrieve, update or delete an individual authorization rule the following resource is used:
  
  \[ \text{http://}\{\text{serverRoot}\}/\text{addressbook/}{\text{apiVersion}}/\{\text{userId}\}/\text{authorizationRules/}{\text{ruleId}} \]

- To retrieve, update or delete the authorized entity in an authorization rule the following resource is used:
  
  \[ \text{http://}\{\text{serverRoot}\}/\text{addressbook/}{\text{apiVersion}}/\{\text{userId}\}/\text{authorizationRules/}{\text{ruleId}}/\{\text{ResourceRelPath}\} \]

![Figure 6 Flow for managing authorization rules](image)

1. The application retrieves a list of all authorization rules for a given user by performing a GET on the following resource:

   \[ \text{http://}\{\text{serverRoot}\}/\text{addressbook/}{\text{apiVersion}}/\{\text{userId}\}/\text{authorizationRules} \]

   The result contains all the authorization rules for the given user.

2. The application creates a new authorization rule for a given user identified by userId by performing a POST on the following resource:

   \[ \text{http://}\{\text{serverRoot}\}/\text{addressbook/}{\text{apiVersion}}/\{\text{userId}\}/\text{authorizationRules} \]

   The result contains a reference to new resource including the newly created ruleId.
3. The application updates an authorization rule for the user identified by userId by performing a PUT on the following resource:

   http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules/{ruleId}

   The result contains a return code indicating whether or not the operation was successful.

4. The application adds an authorized entity (e.g. Alice) to the authorization rule for the user identified by userId by performing a PUT on the following resource:

   http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules/{ruleId}/users/{aliceUserId}

   The result contains a return code indicating whether or not the operation was successful.

5. The application deletes an authorization rule for the user identified by userId by performing a DELETE on the following resource:

   http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules/{ruleId}

   The result contains a return code indicating whether or not the operation was successful.
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, application/x-www-form-urlencoded):

- 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, userId, etc) of type anyURI 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 a user identifier (e.g. address, userId, etc) of type anyURI is in the form of a SIP URI, it MUST be defined according to [RFC3261].

- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of an Anonymous Customer Reference (ACR), it MUST be defined according to [IETF_ACR_draft], i.e. it MUST include the protocol prefix ‘acr:’ followed by the ACR.
  - The ACR ‘Authorization’ 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, and MAY support application/x-www-form-urlencoded 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: Collection of contacts

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts

This resource is used to retrieve a collection of all contacts owned by a given user.

6.1.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</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 the RESTful Address Book API, see section 7.

### 6.1.3 GET

This resource is used to retrieve a collection of contacts and their information.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a contact in a collection of contacts, returned in a GET response body.</td>
</tr>
</tbody>
</table>

If indivFilter is absent, GET response body SHALL include all contact sub-elements, plus all contact attributes (default behaviour).

If indivFilterValue = Contact attribute name (e.g. name1), GET response body SHALL include all contact sub-elements, plus only selected contact attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by & are allowed.

If indivFilterValue = ~noAttr, GET response body SHALL include all contact sub-elements and none of the contact attributes.

If indivFilterValue = ~none, GET response body SHALL include none of the contact information (contact structure to be omitted).

No other indivFilterValue is supported.

#### 6.1.3.1 Example 1: Retrieve all contacts with all attributes (default)  (Informative)

**6.1.3.1.1 Request**

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts HTTP/1.1  
Accept: application/xml  
Host: example.com

**6.1.3.1.2 Response**

HTTP/1.1 200 OK  
Date: Fri, 10 Dec 2010 12:51:59 GMT  
Content-Type: application/xml  
Content-Length: nnnn

```xml
<?xml version="1.0" encoding="UTF-8"?>  
<ab:contactCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">  
  <contact>  
    <contactId>alice</contactId>  
    <attributeList>  
      <attribute>
```
<name>display-name</name>
  <value>Alice</value>
</attribute>
<attribute>
  <name>cellphone</name>
  <value>tel:+19585550109</value>
</attribute>
<attribute>
  <name>state</name>
  <value>California</value>
</attribute>
</attributeList>
</contact>
<contact>
  <contactId>sam</contactId>
  <sharedIdentity>
    <sharedId>tel:+19585550121</sharedId>
  </sharedIdentity>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Sam</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550108</value>
    </attribute>
    <attribute>
      <name>state</name>
      <value>New Jersey</value>
    </attribute>
  </attributeList>
</contact>
</ab:contactCollection>

6.1.3.2 Example 2: Retrieve all contacts with selected attributes  (Informative)

6.1.3.2.1 Request

GET/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts?indivFilter=cellphone HTTP/1.1
Accept: application/xml
Host: example.com
6.1.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contactCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contact>
    <contactId>alice</contactId>
    <attributeList>
      <attribute>
        <name>cellphone</name>
        <value>tel:+19585550109</value>
      </attribute>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice</resourceURL>
    <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550122"/>
  </contact>
  <contact>
    <contactId>sam</contactId>
    <sharedIdentity>
      <sharedId>tel:+19585550121</sharedId>
    </sharedIdentity>
    <attributeList>
      <attribute>
        <name>cellphone</name>
        <value>tel:+19585550108</value>
      </attribute>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/sam/attributes</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/sam</resourceURL>
    <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550121"/>
    <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/tel%3A%2B19585550121"/>
  </contact>
</ab:contactCollection>

6.1.4 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 14.7 of [RFC 2616].

6.1.5 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 14.7 of [RFC 2616].

6.1.6 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 14.7 of [RFC 2616].
6.2 Resource: Individual contact

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}

This resource is used to retrieve, create, update or delete individual contact from the collection of contacts of a given user.

6.2.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>contactId</td>
<td>contact identifier. Example: alice</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 the RESTful Address Book API, see section 7.

6.2.3 GET

This resource is used to retrieve individual member data from the collection of contacts of a given user.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a contact in a collection of contacts, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilter is absent, GET response body SHALL include all contact sub-elements, plus all contact attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Contact attribute name (e.g. name1), GET response body SHALL include all contact sub-elements, plus only selected contact attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No other indivFilterValue is supported.</td>
</tr>
</tbody>
</table>

6.2.3.1 Example 1: Retrieve a contact with all attributes (default) (Informative)

6.2.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice HTTP/1.1
6.2.3.2  Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Alice</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550109</value>
    </attribute>
    <attribute>
      <name>state</name>
      <value>California</value>
    </attribute>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550109/contacts/alice/attributes</resourceURL>
  <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550109/lists/1234/members/tel%3A%2B19585550122"/>
</ab:contact>

6.2.3.2  Example 2: Retrieve a contact with selected attributes  (Informative)

6.2.3.2.1  Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550109/contacts/alice?indivFilter=cellphone HTTP/1.1
Accept: application/xml
Host: example.com

6.2.3.2.2  Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
    <attribute>
      <name>cellphone</name>
6.2.3.3 Example 3: Retrieve a contact with only vCard2.1 as selected attribute

(Informative)

6.2.3.3.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice?indivFilter=vCard2.1 HTTP/1.1
Accept: application/xml
Host: example.com

6.2.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
    <attribute>
      <name>vCard2.1</name>
      <objectValue>
        QkVHSU46VkNBUkQNClZFUlNJT046Mi4xvDQpOOkd1bXAX7RM9ycmVzdA0KRk46Rm9ycmVzdCBHdW1wDQpPkc6QnVYmEgR3Vt
CBTahJpbXAgQ28sdDQpUSVRMRTpTaJpbXAgTWFuDQpURU7vV09SSztWTOoMTEk1NTU1MzgKfkgMh0hPTU7
Vk9JQJ0U6KDQwNCkgTU1LTEyMTINCkFEUjtXT1JLojs7MTAwIFdhdGVkIFN0YXRlcyBBbWVyaWNhDQpMQUJFTUtMT01
QmF5dG93biwgTEEgMzAzMTQ9MEQ9MEFVbml0ZWQgU3RhdGVzIG9mIEFtZXJpY2E7
</objectValue>
    </attribute>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes</resourceURL>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice</resourceURL>
  <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550122"/>
</ab:contact>
6.2.4 PUT

This operation is used to create or update a contact in the collection of contacts.

6.2.4.1 Example 1: Create a new contact (Informative)

6.2.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"><contactId>maria</contactId><attributeList><attribute><name>cellphone</name><value>tel:+19585550106</value></attribute><resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes</resourceURL></attributeList><resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria</resourceURL></ab:contact>

6.2.4.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?><ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"><contactId>maria</contactId><attributeList><attribute><name>cellphone</name><value>tel:+19585550106</value></attribute><resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes</resourceURL></attributeList><resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria</resourceURL></ab:contact>

6.2.4.2 Example 2: Update an existing contact with links to an existing member (Informative)

The PUT example is followed by a GET example on a list that was updated as a result of updating a Contact request. The Member is updated with the reverse link to the Contact.

6.2.4.2.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria HTTP/1.1
6.2.4.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/v1/addressbook/tel%3A%2B19585550100/contacts/maria
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>maria</contactId>
  <attributeList>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550106</value>
    </attribute>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes</resourceURL>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria</resourceURL>
  <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/tel%3A%2B19585550106" />
</ab:contact>

6.2.4.2.3 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678 HTTP/1.1
Accept: application/xml
Host: example.com

6.2.4.2.4 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
   <listId>5678</listId>
   <memberCollection>
      <member>
         <memberId>+19585550106</memberId>
         <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com</resourceURL>
         <link rel="Contact" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria"/>
      </member>
      <member>
         <memberId>mailto:liza@example.com</memberId>
         <attributeList>
            <attribute>
               <name>display-name</name>
               <value>Wife</value>
            </attribute>
            <attribute>
               <name>age</name>
               <value>42</value>
            </attribute>
         </attributeList>
         <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com/attributes</resourceURL>
      </member>
      <member>
         <memberId>mailto:serezha@example.com</memberId>
         <attributeList>
            <attribute>
               <name>display-name</name>
               <value>Son</value>
            </attribute>
            <attribute>
               <name>phone</name>
               <value>tel:+19585550105</value>
            </attribute>
         </attributeList>
         <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com/attributes</resourceURL>
      </member>
   </memberCollection>
   <attributeList>
      <attribute>
         <name>display-name</name>
         <value>Family</value>
      </attribute>
   </attributeList>
</ab:list>
6.2.4.3 Example 3: Update an existing contact with links to non existing member
(Informative)

In this example the user tries to update a contact including a link to a member which does not exists.

6.2.4.3.1 Request

```
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>maria</contactId>
  <attributeList>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550106</value>
    </attribute>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes</resourceURL>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria</resourceURL>
  <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/tel%3A%2B19585550108"/>
</ab:contact>
```

6.2.4.3.2 Response

```
HTTP/1.1 403 Forbidden
Date: Fri, 10 Dec 2010 12:51:59 GMT
```

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, PUT, DELETE’ field in the response as per section 14.7 of [RFC 2616].

6.2.6 DELETE

This operation deletes a contact in a list. It SHALL also automatically delete any reverse links in a referenced member.
6.2.6.1 Example: Delete a contact

6.2.6.1.1 Request

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550105/contacts/sam HTTP/1.1
Accept: application/xml
Host: example.com

6.2.6.1.2 Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.3 Resource: Attributes for a contact

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes

This resource is used to retrieve all attributes for a contact.

6.3.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>contactId</td>
<td>contact identifier. Example: alice</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 the RESTful Address Book API, see section 7.

6.3.3 GET

This operation is used for retrieval of all attributes of a contact for a given user identity.

6.3.3.1 Example: Retrieve all attributes of a contact

Retrieve all attributes of a contact belonging to a user, and return result in XML format, using resFormat parameter in URL.

6.3.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes?resFormat=XML HTTP/1.1
Accept: application/xml
Host: example.com
6.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attributeList xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
<attribute>
  <name>cellphone</name>
  <value>tel:+19585550106</value>
</attribute>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes</resourceURL>
</ab:attributeList>

6.3.4 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 14.7 of [RFC 2616].

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’ field in the response as per section 14.7 of [RFC 2616].

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’ field in the response as per section 14.7 of [RFC 2616].

6.4 Resource: Individual attribute for a contact
The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/contacts/{contactId}/attributes/{ResourceRelPath}

This light-weight resource is used to manage attributes for a contact, which include creation, retrieval, update and delete operations.

6.4.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>contactId</td>
<td>contact identifier. Example: alice</td>
</tr>
<tr>
<td>[ResourceRelPath]</td>
<td>Light-weight relative resource path. The allowed string for this light-weight resource is {name}, as defined in the [ResourceRelPath] column in table 5.2.2.6. {name} indicates the name of the attribute. Example: display-name</td>
</tr>
</tbody>
</table>

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

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
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 the RESTful Address Book API, see section 7.

6.4.3 GET
This operation is used for retrieval of attribute values of a contact of a user.

6.4.3.1 Example 1: Retrieve a contact’s attribute

6.4.3.1.1 Request
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/cellphone HTTP/1.1
Accept: application/xml
Host: example.com

6.4.3.1.2 Response
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn
<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>cellphone</name>
  <value>tel:+19585550109</value>
</ab:attribute>

6.4.3.2 Example 2: Retrieve a contact’s non-existing attribute

6.4.3.2.1 Request
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attribute/alien HTTP/1.1
Accept: application/xml
Host: example.com

6.4.3.2.2 Response
HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 12:55:59 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 Invalid input value for message part %1</text>
    <variables>Alien</variables>
  </serviceException>
</common:requestError>
6.4.4 **PUT**

This operation is used for creation or update of an attribute of a contact. If successful, this SHALL also create or update the attribute for the members in lists that are associated with the contact.

6.4.4.1 **Example: Create or update a contact’s attribute**  (Informative)

6.4.4.1.1 **Request**

```
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/married HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>married</name>
  <value>true</value>
</ab:attribute>
```

6.4.4.1.2 **Response**

```
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location:http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/married
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>married</name>
  <value>true</value>
</ab:attribute>
```

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 14.7 of [RFC 2616].

6.4.6 **DELETE**

This operation deletes an attribute of a contact. If successful, this SHALL also delete the attribute from the members’ attributes in lists associated with the contact.

6.4.6.1 **Example: Delete a contact’s attribute**  (Informative)

6.4.6.1.1 **Request**

```
DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/display-name HTTP/1.1
Accept: application/xml
Host: example.com
```
6.4.6.1.2 Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.5 Resource: Lists

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists

This resource is used to retrieve all lists belonging to a user.

6.5.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>User identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
</tbody>
</table>

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

6.5.2 Response Codes

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.5.3 GET

This operation is used for retrieval of all lists for a given user identity.

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>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a list, returned in a GET response body.</td>
</tr>
</tbody>
</table>

  - If listFilter is absent, GET response body SHALL include all the list sub-elements, plus all list attributes (default behaviour).
  - If listFilterValue = List attribute name (e.g. name1), GET response body SHALL include all list sub-elements, plus only selected list attributes (i.e. name1). Any number of listFilter=listFilterValue pairs separated by & are allowed.
  - If listFilterValue = ~noAttr, GET response body SHALL include all the list sub-elements and none of the list attributes.
6.5.3.1 Example 1: Retrieve lists with all attributes and all members with all member attributes (default) (Informative)

6.5.3.1.1 Request
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.1.2 Response
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <list>
    <listId>1234</listId>
    <memberCollection>
      <member>
        <memberId>sip:alice@example.com</memberId>
        <attributeList>
          <attribute>
            <name>display-name</name>
            <value>Alice</value>
          </attribute>
        </attributeList>
      </member>
    </memberCollection>
  </list>
</ab:listCollection>
<attributeList>
  <attribute>
    <name>display-name</name>
    <value>Friends</value>
  </attribute>
</attributeList>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
</member>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com</resourceURL>
</memberCollection>
<category>URIList</category>
<sharedListIdentity>
  <sharedId>sip:bob@example.com;list=1234</sharedId>
</sharedListIdentity>
<attributeList>
  <attribute>
    <name>display-name</name>
    <value>Friends</value>
  </attribute>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes</resourceURL>
</attributeList>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
</list>
</list>
<listId>5678</listId>
<memberCollection>
  <member>
    <memberId>mailto:liza@example.com</memberId>
    <attributeList>
      <attribute>
        <name>display-name</name>
        <value>Wife</value>
      </attribute>
      <attribute>
        <name>age</name>
        <value>42</value>
      </attribute>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com/attributes</resourceURL>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com</resourceURL>
  </member>
  <member>
    <memberId>mailto:serezha@example.com</memberId>
    <attributeList>
      <attribute>
        <name>display-name</name>
        <value>Son</value>
      </attribute>
      <attribute>
        <name>phone</name>
        <value>tel:+19585550105</value>
      </attribute>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com/attributes</resourceURL>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com</resourceURL>
  </member>
</memberCollection>
6.5.3.2 Example 2: Retrieve all lists belonging to a user without list attributes and without members (Informative)

Retrieve all lists belonging to a user, and return result in XML format.

6.5.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=~noAttr&indivFilter=~none HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <list>
    <listId>1234</listId>
    <category>URIList</category>
    <sharedListIdentity>
      <sharedId>sip:bob@example.com;list=1234</sharedId>
    </sharedListIdentity>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/mailto%3Abob@example.com/lists/1234</resourceURL>
  </list>
  <list>
    <listId>5678</listId>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678</resourceURL>
  </list>
</ab:listCollection>
6.5.3.3 Example 3: Retrieve specific lists attributes and specific member attributes (Informative)

6.5.3.3.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=display-name&indivFilter=display-name&indivFilter=phone
HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <list>
    <listId>1234</listId>
    <memberCollection>
      <member>
        <memberId>sip:alice@example.com</memberId>
        <attributeList>
          <attribute>
            <name>display-name</name>
            <value>Alice</value>
          </attribute>
          <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
        </attributeList>
      </member>
    </memberCollection>
    <category>URIList</category>
    <sharedListIdentity>
      <sharedId>sip:bob@example.com;list=1234</sharedId>
      <attributeList>
        <attribute>
          <name>display-name</name>
          <value>Friends</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes</resourceURL>
      </attributeList>
    </sharedListIdentity>
  </list>
  <list>
    <listId>5678</listId>
    <memberCollection>
      <member>
        <memberId>mailto:liza@example.com</memberId>
        <attributeList>
          <attribute>
            <name>display-name</name>
            <value>Liza</value>
          </attribute>
          <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto:liza@example.com/attributes</resourceURL>
        </attributeList>
      </member>
    </memberCollection>
  </list>
</ab:listCollection>
6.5.3.4 Example 4: Retrieve lists with specific lists attributes and without members data (Informative)

6.5.3.4.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=display-name&indivFilter=~none HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.4.2 Response

HTTP/1.1 200 OK
6.5.3.5 Example 5: Retrieve specific lists attributes and list of members without attributes (Informative)

6.5.3.5.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=display-name&indivFilter=~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.5.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <list>
        <listId>1234</listId>
        <category>URIList</category>
        <sharedListIdentity>
            <sharedId>sip:bob@example.com;list=1234</sharedId>
        </sharedListIdentity>
        <attributeList>
            <attribute>
                <name>display-name</name>
                <value>Friends</value>
            </attribute>
            <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes</resourceURL>
        </attributeList>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
    </list>
    <list>
        <listId>5678</listId>
        <attributeList>
            <attribute>
                <name>display-name</name>
                <value>Family</value>
            </attribute>
            <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/attributes</resourceURL>
        </attributeList>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678</resourceURL>
    </list>
</ab:listCollection>
<list>
  <listId>1234</listId>
  <memberCollection>
    <member>
      <memberId>sip:alice@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com</resourceURL>
    </member>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members</resourceURL>
  </memberCollection>
  <category>URIList</category>
  <sharedListIdentity>
    <sharedId>sip:bob@example.com;list=1234</sharedId>
  </sharedListIdentity>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Friends</value>
    </attribute>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes</resourceURL>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
</list>

<list>
  <listId>5678</listId>
  <memberCollection>
    <member>
      <memberId>mailto:liza@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com</resourceURL>
    </member>
    <member>
      <memberId>mailto:serezha@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com</resourceURL>
    </member>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members</resourceURL>
  </memberCollection>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Family</value>
    </attribute>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/attributes</resourceURL>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678</resourceURL>
</list>

<listCollection>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists</resourceURL>
</listCollection>
6.5.3.6 Example 6: Retrieve lists without attributes and members with all member attributes (Informative)

6.5.3.6.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.5.3.6.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <list>
    <listId>1234</listId>
    <memberCollection>
      <member>
        <memberId>sip:alice@example.com</memberId>
        <attributeList>
          <attribute>
            <name>display-name</name>
            <value>Alice</value>
          </attribute>
        </attributeList>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
      </member>
    </memberCollection>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
  </list>
  <list>
    <listId>5678</listId>
    <memberCollection>
      <member>
        <memberId>mailto:liza@example.com</memberId>
        <attributeList>
          <attribute>
            <name>display-name</name>
            <value>Wife</value>
          </attribute>
          <attribute>
            <name>age</name>
            <value>42</value>
          </attribute>
        </attributeList>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com/attributes</resourceURL>
      </member>
    </memberCollection>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678</resourceURL>
  </list>
</ab:listCollection>
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 14.7 of [RFC 2616].

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 14.7 of [RFC 2616].

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 14.7 of [RFC 2616].

6.6 Resource: Individual list
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}

This resource is used to manage an individual list, which include creation, update, retrieval, and delete operations for a particular list.

6.6.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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:</td>
</tr>
</tbody>
</table>
example.com/exampleAPI

<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilter is absent, GET response body SHALL include all the list sub-elements, plus all list attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = List attribute name (e.g. name1), GET response body SHALL include all list sub-elements, plus only selected list attributes (i.e. name1). Any number of listFilter=listFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = ~noAttr, GET response body SHALL include all the list sub-elements and none of the list attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No other listFilterValue is supported.</td>
</tr>
<tr>
<td>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Member attribute name (e.g. name1), GET response body SHALL include all member sub-elements, plus only selected member attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = ~noAttr, GET response body SHALL include all member sub-elements and none of the member attributes.</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 the RESTful Address Book API, see section 7.

### 6.6.3 GET

This operation is used for retrieval of a given list. (This operation is normally only used to verify the existence of the list).

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

apiVersion version of the API client wants to use. The value of this variable is defined in section 5.1.

userId user identifier. Examples: tel:+19585550100, acr:pseudonym123

listId list identifier. Example: 1234
6.6.3.1 Example 1: Retrieve a list with all attributes and all member attributes (default) (Informative)

6.6.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/xml
Host: example.com

6.6.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <listId>1234</listId>
    <memberCollection>
        <member>
            <memberId>sip:alice@example.com</memberId>
            <attributeList>
                <attribute>
                    <name>display-name</name>
                    <value>Alice</value>
                </attribute>
            </attributeList>
            <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com</resourceURL>
        </member>
        <memberCollection>
            <attributeList>
                <attribute>
                    <name>display-name</name>
                    <value>Friends</value>
                </attribute>
            </attributeList>
            <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
        </memberCollection>
    </memberCollection>
</ab:list>

If indivFilterValue = ~none, GET response body SHALL include none of the member information (member structure to be omitted).
No other indivFilterValue is supported.
6.6.3.2  Example 2: Retrieve a list without list and member attributes (Informative)

6.6.3.2.1  Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234?listFilter=~noAttr&indivFilter=~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.6.3.2.2  Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <memberCollection>
    <member>
      <memberId>sip:alice@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/ sip%3Aalice@example.com</resourceURL>
    </member>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members</resourceURL>
  </memberCollection>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
</ab:list>

6.6.3.3  Example 3: Retrieve a list with all attributes and selected member attributes (Informative)

6.6.3.3.1  Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234?indivFilter=display-name HTTP/1.1
Accept: application/xml
Host: example.com

6.6.3.3.2  Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <memberCollection>
    <member>
      <memberId>sip:alice@example.com</memberId>
      <attributeList>
        <attribute>
          <attributeId>sip:alice@example.com</attributeId>
          <value>display-name</value>
        </attribute>
      </attributeList>
    </member>
  </memberCollection>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/ sip%3Aalice@example.com</resourceURL>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
</ab:list>
<name>display-name</name>
   <value>Alice</value>
</attribute>

<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
</attributeList>

</member>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/resourceURL>
</memberCollection>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234</resourceURL>
</ab:list>

6.6.3.4 Example 4: Retrieve a non existing list

6.6.3.4.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/4321 HTTP/1.1
Accept: application/xml
Host: example.com

6.6.3.4.2 Response

HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 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">
   <link rel="List" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/4321"/>
   <serviceException>
      <messageId>SVC0002</messageId>
      <text>Invalid input value for message part %1</text>
      <variables>listId</variables>
   </serviceException>
</common:requestError>

6.6.4 PUT

This operation is used for creation or update of a list.

6.6.4.1 Example: Create a list

6.6.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn
<?xml version="1.0" encoding="UTF-8"?><ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>bobPublic</listId>
  <memberCollection>
    <member>
      <memberId>mailto:alice@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members/mailto%3Aalice@example.com</resourceURL>
    </member>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members</resourceURL>
  </memberCollection>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic</resourceURL>
</ab:list>

6.6.4.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>bobPublic</listId>
  <memberCollection>
    <member>
      <memberId>mailto:alice@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members/mailto%3Aalice@example.com</resourceURL>
    </member>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members</resourceURL>
  </memberCollection>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic</resourceURL>
</ab:list>

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, PUT, DELETE’ field in the response as per section 14.7 of [RFC 2616].

6.6.6 DELETE

This operation deletes a list.

6.6.6.1 Example: Delete a list (Informative)

6.6.6.1.1 Request

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic HTTP/1.1
Accept: application/xml
Host: example.com
6.6.6.1.2 Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.7 Resource: Attributes for a list

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/attributes

This resource is used to retrieve all attributes of a list belonging to a user.

6.7.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</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 the RESTful Address Book API, see section 7.

6.7.3 GET

This operation is used for retrieval of all attributes for a list for a given user identity.

6.7.3.1 Example: Retrieve all attributes for a list belonging to Bob  (Informative)

Retrieve all attributes for a list belonging to Bob, and return result in XML format.

6.7.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes HTTP/1.1
Accept: application/xml
Host: example.com

6.7.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attributeList xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <attribute>
</attribute>
<name>display-name</name>
<value>Friends</value>
</attribute>
<attribute>
<name>date-created</name>
<value>Fri, 10 Dec 2010 11:28:34 GMT</value>
</attribute>
</ab:attributeList>

6.7.4 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 14.7 of [RFC 2616].

6.7.5 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 14.7 of [RFC 2616].

6.7.6 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 14.7 of [RFC 2616].

6.8 Resource: Individual attribute for a list
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/attributes/{ResourceRelPath}

This resource is used to manage attributes on an individual list, which include creation, update, retrieval, and delete operations for the attributes.

6.8.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: 4567</td>
</tr>
<tr>
<td>ResourceRelPath</td>
<td>Light-weight relative resource path. The allowed string for this light-weight resource is {name}, as defined in the [ResourceRelPath] column in table 5.2.2.6. {name} indicates the name of the attribute. Example: display-name</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 the RESTful Address Book API, see section 7.
6.8.3  GET
This operation is used for retrieval of the value of a given attribute of a list.

6.8.3.1  Example 1: Retrieve an attribute value  (Informative)
6.8.3.1.1  Request
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/date-created HTTP/1.1
Accept: application/xml
Host: example.com

6.8.3.1.2  Response
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn
<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>date-created</name>
  <value>Fri, 10 Dec 2010 11:28:34 GMT</value>
</ab:attribute>

6.8.3.2  Example 2: Retrieve a non existing attribute  (Informative)
6.8.3.2.1  Request
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/alien HTTP/1.1
Accept: application/xml
Host: example.com

6.8.3.2.2  Response
HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 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>alien</variables>
  </serviceException>
</common:requestError>
6.8.4 PUT

This operation is used for creation or update of an attribute of a list.

6.8.4.1 Example: Create an attribute (Informative)

6.8.4.1.1 Request

```
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/display-name HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"><name>display-name</name><value>Close friends</value></ab:attribute>
```

6.8.4.1.2 Response

```
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location:http://example.com/exampleAPI/v1/addressbook/tel%3A%2B19585550123/lists/1234/attributes/display-name
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"><name>display-name</name><value>Close friends</value></ab:attribute>
```

6.8.5 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 14.7 of [RFC 2616].

6.8.6 DELETE

This operation deletes an attribute of a list.

6.8.6.1 Example: Delete an attribute (Informative)

6.8.6.1.1 Request

```
DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/display-name HTTP/1.1
Accept: application/xml
Host: example.com

```

6.8.6.1.2 Response

```
HTTP/1.1 204 No Content
```
6.9 Resource: Members in a list

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members

This resource is used to retrieve all members of a list belonging to a user.

6.9.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: myList</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

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.9.3 GET

This operation is used for retrieval of all members of list for a given user identity.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Member attribute name (e.g. name1), GET response body SHALL include all member sub-elements, plus only selected member attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = ~noAttr, GET response body SHALL include all member sub-elements and none of the member attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = ~none, GET response body SHALL include all member sub-elements and none of the member attributes.</td>
</tr>
</tbody>
</table>
6.9.3.1 Example 1: Retrieve all members for a list belonging to a user with all attributes (default) (Informative)

Retrieve all members of a list belonging to a user, and return result in XML format.

6.9.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members HTTP/1.1
Accept: application/xml
Host: example.com

6.9.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:memberCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <member>
    <memberId>mailto:alice@example.com</memberId>
    <attributeList>
      <attribute>
        <name>display-name</name>
        <value>Alice</value>
      </attribute>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com</resourceURL>
    </member>
    <member>
      <memberId>tel:+19585550124</memberId>
      <attributeList>
        <attribute>
          <name>display-name</name>
          <value>Bob</value>
        </attribute>
      </attributeList>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550124</resourceURL>
    </member>
  </memberCollection>
6.9.3.2  Example 2: Retrieve all members for a list belonging to a user, without member attributes (Informative)

Retrieve all members of a list belonging to a user, and return result in XML format.

6.9.3.2.1  Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members?indivFilter=~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.9.3.2.2  Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:memberCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <member>
    <memberId>mailto:alice@example.com</memberId>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com</resourceURL>
  </member>
  <member>
    <memberId>tel:+19585550124</memberId>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550124</resourceUR

6.9.4  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 14.7 of [RFC 2616].

6.9.5  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 14.7 of [RFC 2616].

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’ field in the response as per section 14.7 of [RFC 2616].
6.10 Resource: Individual member in a list

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}

This resource is used to manage members on an individual list, which include creation, update, retrieval, and delete operations for the members.

6.10.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: 6789</td>
</tr>
<tr>
<td>memberId</td>
<td>member identifier. Example: <a href="mailto:alice@example.com">mailto:alice@example.com</a></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

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.10.3 GET

This operation is used for retrieval of a member from a list.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Member attribute name (e.g. name1), GET response body SHALL include all member sub-elements, plus only selected member attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No other indivFilterValue is supported.</td>
</tr>
</tbody>
</table>
### 6.10.3.1 Example 1: Retrieve a member (Informative)

#### 6.10.3.1.1 Request

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com HTTP/1.1
Accept: application/xml
Host: example.com
```

#### 6.10.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <memberId>sip:alice@example.com</memberId>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Alice</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550155</value>
    </attribute>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes</resourceURL>
</ab:member>
```

### 6.10.3.2 Example 2: Retrieve a member and selected attributes (Informative)

#### 6.10.3.2.1 Request

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com?indivFilter=cellphone HTTP/1.1
Accept: application/xml
Host: example.com
```

#### 6.10.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
```

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
6.10.3.3 Example 3: Retrieve a non existing member  

6.10.3.3.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Ajoe@example.com HTTP/1.1
Accept: application/xml
Host: example.com

6.10.3.3.2 Response

HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 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>memberId</variables>
  </serviceException>
</common:requestError>

6.10.4 PUT

This operation is used for creation or update of a member on a list.

6.10.4.1 Example 1: Create a member without a link to the contact  

6.10.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
6.10.4.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <memberId>mailto:ole@example.com</memberId>
    <attributeList>
        <attribute>
            <name>display-name</name>
            <value>Ole</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com/resources</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com</resourceURL>
</ab:member>

6.10.4.2 Example 2: Update member with relation to non-existing contact

(Informative)

In this example the user tries to update a member including a link to a contact, which does not exists.

6.10.4.2.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Ajoe@example.com HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <memberId>mailto:joe@example.com</memberId>
    <attributeList>
        <attribute>
            <name>display-name</name>
            <value>Joe</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/resources</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com</resourceURL>
</ab:member>
<name>display-name</name>
  <value>Joe</value>
</attribute>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3AJoe@example.com/attributes</resourceURL>
</attributeList>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3AJoe@example.com</resourceURL>
<link rel="Contact" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/joe"/>
</ab:member>

6.10.4.2.2  Response

HTTP/1.1 403 Forbidden
Date: Fri, 10 Dec 2010 12:51:59 GMT

6.10.5  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 14.7 of [RFC 2616].

6.10.6  DELETE
This operation deletes a member from a list.

6.10.6.1  Example: Delete a member  (Informative)

6.10.6.1.1  Request

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/tel%3A%2B19585550125 HTTP/1.1
Accept: application/xml
Host: example.com

6.10.6.1.2  Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.11  Resource: Attributes for a member in a list
The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}/attributes

This resource is used to retrieve all attributes for a member of a list.

6.11.1  Request URL variables
The following request URL variables are common for all HTTP commands:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
serverRoot | server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
---|---
apiVersion | version of the API client wants to use. The value of this variable is defined in section 5.1.
userId | user identifier. Examples: tel:+19585550100, acr:pseudonym123
listId | list identifier. Example: myList.
memberId | member identifier. Example: tel:+19585550126

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

6.11.2 Response Codes and Error Handling
For HTTP response codes, see [REST_NetAPI_Common].
For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.11.3 GET
This operation is used for retrieval of all members of list for a given user identity.

6.11.3.1 Example: Retrieve all attributes of a member of a list (Informative)
Retrieve all attributes of a member of a list belonging to a user, and return result in XML format.

6.11.3.1.1 Request
```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aalice@example.com/attributes HTTP/1.1
Accept: application/xml
Host: example.com
```

6.11.3.1.2 Response
```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attributeList xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
<attribute>
  <name>display-name</name>
  <value>Alice</value>
</attribute>
<attribute>
  <name>cellphone</name>
  <value>tel:+19585550155</value>
</attribute>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes</resourceURL>
</ab:attributeList>
```

6.11.4 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 14.7 of [RFC 2616].
6.11.5 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 14.7 of [RFC 2616].

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’ field in the response as per section 14.7 of [RFC 2616].

6.12 Resource: Individual attribute for a member in a list

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}/attributes/{[ResourceRelPath]}

This light-weight resource is used to manage attributes for a member in a list, which include creation, update, retrieval, and delete operations.

6.12.1 Request URL variables

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverRoot</td>
<td>server base url: hostname+port+base path. 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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: myFriends</td>
</tr>
<tr>
<td>memberId</td>
<td>member identifier. Example: tel:+19585550126</td>
</tr>
<tr>
<td>[ResourceRelPath]</td>
<td>Light-weight relative resource path. The allowed string for this light-weight resource is {name}, as defined in the [ResourceRelPath] column in table 5.2.2.6. {name} indicates the name of the attribute. Example: display-name</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 the RESTful Address Book API, see section 7.

6.12.3 GET

This operation is used for retrieval of attribute values of a member on a list.

6.12.3.1 Example 1: Retrieve a member’s attribute (Informative)

6.12.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/display-name HTTP/1.1
Accept: application/xml
Host: example.com
6.12.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>display-name</name>
  <value>Alice</value>
</ab:attribute>

6.12.3.2 Example 2: Retrieve a member's non existing attribute (Informative)
6.12.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/pet HTTP/1.1
Accept: application/xml
Host: example.com

6.12.3.2.2 Response

HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 12:55:59 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 Invalid input value for message part %1</text>
    <variables>pet</variables>
  </serviceException>
</common:requestError>

6.12.4 PUT

This operation is used for creation or update of an attribute of a member on a list.

6.12.4.1 Example: Create a member's attribute (Informative)
6.12.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/born HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <name>born</name>
  <value>Alice</value>
</ab:attribute>
6.12.4.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/born
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:attribute xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <name>born</name>
    <value>1984</value>
</ab:attribute>

6.12.5 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 14.7 of [RFC 2616].

6.12.6 DELETE

This operation deletes a member from a list.

6.12.6.1 Example: Delete a member’s attribute (Informative)

6.12.6.1.1 Request

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/born
HTTP/1.1
Accept: application/xml
Host: example.com

6.12.6.1.2 Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.13 Resource: List references

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/listReferences

This resource is used to retrieve list of references for all nested lists.

6.13.1 Request URL variables

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
serverRoot | server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
---|---
apiVersion | version of the API client wants to use. The value of this variable is defined in section 5.1.
userId | user identifier. Examples: tel:+19585550100, acr:pseudonym123
listId | list identifier. Example: myList

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 the RESTful Address Book API, see section 7.

### 6.13.3 GET

This operation is used for retrieval of a list of references to other lists.

#### 6.13.3.1 Example: Retrieve all list references in a list (Informative)

Retrieve all list references in a list belonging to a user, and return result in XML format.

**6.13.3.1.1 Request**

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences HTTP/1.1
Accept: application/xml
Host: example.com
```

**6.13.3.1.2 Response**

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listReferenceCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/friends/listReferences</resourceURL>
    <link rel="List" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234"/>
    <link rel="List" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678"/>
</ab:listReferenceCollection>
```

### 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’ field in the response as per section 14.7 of [RFC 2616].

### 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’ field in the response as per section 14.7 of [RFC 2616].

### 6.13.6 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 14.7 of [RFC 2616].
6.14 Resource: Individual list reference

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/listReferences/{ResourceRelPath}

This light-weight resource is used to manage list references in a nested list, which include creation, update, retrieval, and delete operations.

6.14.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: myList</td>
</tr>
<tr>
<td>[ResourceRelPath]</td>
<td>Light-weight relative resource path. The allowed string for this light-weight resource is {href}, as defined in the [ResourceRelPath] column in table 5.2.2.10. {href} indicates the link to the referenced list.</td>
</tr>
</tbody>
</table>

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

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 the RESTful Address Book API, see section 7.

6.14.3 GET

This operation is used for retrieval of a list reference in a nested list. This is typically used to verify if the list reference exists.

6.14.3.1 Example 1: Retrieve a list reference (Informative)

6.14.3.1.1 Request

GET
/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/xml
Host: example.com

6.14.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:link xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1" rel="List"/>
6.14.3.2 Example 2: Retrieve a non existing list reference (Informative)

6.14.3.2.1 Request

GET
/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1235 HTTP/1.1
Accept: application/xml
Host: example.com

6.14.3.2.2 Response

HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 12:55:59 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 Invalid input value for message part %1</text>
    <variables>href</variables>
  </serviceException>
</common:requestError>

6.14.4 PUT

This operation is used for creation of a list reference in a list.

6.14.4.1 Example: Create a list reference (Informative)

6.14.4.1.1 Request

PUT
/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678 HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

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

6.14.4.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
6.14.5 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 14.7 of [RFC 2616].

6.14.6 DELETE

This operation deletes a list reference in a list.

6.14.6.1 Example: Delete a list reference

6.14.6.1.1 Request

```
DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/
http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.14.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT
```

6.15 Resource: Address Book changes subscriptions

The resource used is:
```
http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges
```

This resource is used to retrieve address book changes subscriptions, as well as for the creation of new subscriptions.

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.

6.15.1 Request URL variables

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

<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</td>
</tr>
</tbody>
</table>
5.1. userId user identifier. Examples: tel:+19585550100, acr:pseudonym123

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 the RESTful Address Book API, see section 7.

6.15.3 GET

This operation is used for retrieval of all list change subscriptions for a list.

6.15.3.1 Example: Retrieve all list changes subscriptions (Informative)

6.15.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges HTTP/1.1
Accept: application/xml
Host: example.com

6.15.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscriptionCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
<abChangesSubscription>
<listId>1234</listId>
<callbackReference>
<notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
<callbackData>12345</callbackData>
</callbackReference>
<clientCorrelator>123</clientCorrelator>
<applicationTag>myApp</applicationTag>
<duration>1958</duration>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321</resourceURL>
</abChangesSubscription>
<abChangesSubscription>
<anyContacts />
<callbackReference>
<notifyURL>http://application.example.com/notifications/contacts</notifyURL>
<callbackData>54321</callbackData>
</callbackReference>
<clientCorrelator>456</clientCorrelator>
<applicationTag>myApp</applicationTag>
<duration>2641</duration>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/123456789</resourceURL>
</abChangesSubscription>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges</resourceURL>
6.15.4 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 14.7 of [RFC 2616].

6.15.5 POST
This operation is used for creation of a new subscription.

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]).

6.15.5.1 Example 1: Create new subscription for list changes notification using ‘tel’ URI (Informative)

6.15.5.1.1 Request

```xml
POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
     <listId>1234</listId>
     <callbackReference>
         <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
         <callbackData>12345</callbackData>
     </callbackReference>
     <clientCorrelator>123</clientCorrelator>
     <applicationTag>myApp</applicationTag>
     <duration>7200</duration>
</ab:abChangesSubscription>
```

6.15.5.1.2 Response

```xml
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
     <listId>1234</listId>
     <callbackReference>
         <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
         <callbackData>12345</callbackData>
     </callbackReference>
     <clientCorrelator>123</clientCorrelator>
</ab:abChangesSubscription>
```
6.15.5.2 Example 2: Create new subscription for list changes notification using ‘acr’ URI (Informative)

6.15.5.2.1 Request

POST /exampleAPI/addressbook/v1/acr%3Apseudonym123/subscriptions/abChanges HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
</ab:abChangesSubscription>

6.15.5.2.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/acr%3Apseudonym123/subscriptions/abChanges/987654321
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/acr%3Apseudonym123/subscriptions/abChanges/987654321</resourceURL>
</ab:abChangesSubscription>

6.15.6 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 14.7 of [RFC 2616].
6.16 Resource: Individual list changes subscription

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/subscriptions/abChanges/{subscriptionId}

This resource is used to retrieve or remove an individual list changes subscription.

6.16.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>identifier associated with this specific subscription</td>
</tr>
</tbody>
</table>

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 the RESTful Address Book API, see section 7.

6.16.3 GET

This operation is used for retrieval of a specific list changes subscription.

6.16.3.1 Example: Retrieve individual list changes subscription (Informative)

Retrieve a specific list changes subscription.

6.16.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321 HTTP/1.1
Accept: application/xml
Host: example.com

6.16.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 21:34:52 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
  </callbackReference>
  <callbackData>12345</callbackData>
  <clientCorrelator>123</clientCorrelator>
</ab:abChangesSubscription>
6.16.4 PUT

This operation is used to update and/or to extend the duration of a subscription.

6.16.4.1 Example: Update/Extend duration of subscription (Informative)

6.16.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321 HTTP/1.1
Host: example.com
Accept: application/xml
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321</resourceURL>
</ab:abChangesSubscription>

6.16.4.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321</resourceURL>
</ab:abChangesSubscription>
6.16.5 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 14.7 of [RFC 2616].

6.16.6 DELETE
This operation deletes a subscription to list changes.

6.16.6.1 Example: Delete a list changes subscription (Informative)

6.16.6.1.1 Request
DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321 HTTP/1.1
Accept: application/xml
Host: example.com

6.16.6.1.2 Response
HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.17 Resource: Client resource for list changes notifications
This resource is a callback URL provided by the client for notification about outbound message delivery status. The RESTful Address Book 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 in order to use Long Polling to obtain the notifications, in order to retrieve the notifications, the client needs to use the Long Polling mechanism described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.17.5.

A notification SHALL be generated by the system in the following occasions:

<table>
<thead>
<tr>
<th>Type of notification</th>
<th>Value of ResourceStatus</th>
<th>Generated in the following occasions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update number of contacts or members</td>
<td>Active</td>
<td>A contact is added/removed to/from the contact collection or a member is added/removed to/from a list</td>
</tr>
<tr>
<td>Remove contacts or members</td>
<td>Active</td>
<td>A contact is removed from the contact collection or a member is removed from a list</td>
</tr>
<tr>
<td>Update contact or member attributes</td>
<td>Active</td>
<td>A contact’s or a member’s attributes are updated (attribute added, changed or removed)</td>
</tr>
<tr>
<td>Update contact collection or list attributes</td>
<td>Active</td>
<td>A contact collection’s or a list’s attributes are updated (attribute added, changed or removed)</td>
</tr>
<tr>
<td>Final notification</td>
<td>TerminatedTimeout</td>
<td>The subscription has expired (time indicated by ‘duration’ has elapsed). Note: A request to delete the subscription does not generate a new notification.</td>
</tr>
<tr>
<td></td>
<td>TerminatedNoResource</td>
<td>The subscription has been terminated since the resource</td>
</tr>
<tr>
<td><strong>TerminatedOther</strong></td>
<td>The subscription has been terminated for an unknown reason.</td>
<td></td>
</tr>
</tbody>
</table>

- to indicate that the subscription has been terminated by the server.

(ResourceStatus=TerminatedTimeout/TerminatedNoResource/TerminatedOther)

### 6.17.1 Request URL variables

Client provided.

### 6.17.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

### 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 14.7 of [RFC 2616].

### 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 14.7 of [RFC 2616].

### 6.17.5 POST

This operation is used to send notification about list change to the application.

#### 6.17.5.1 Example 1: Notification for subscription with sendFullListContent=true (default) (Informative)

**6.17.5.1.1 Request**

```plaintext
POST /notifications/lists/1234 HTTP/1.1
Accept: application/xml
Host: application.example.com
Content-Type: application/xml
Content-Length:nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesNotification xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
<callbackData>12345</callbackData>
<resourceStatus>Active</resourceStatus>
<duration>3480</duration>
<link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com"/>
<link rel="AbChangesSubscription" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321"/>
</ab:abChangesNotification>
```

**6.17.5.1.2 Response**

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
```
6.17.5.2 Example 2: Notification for contact change subscription  (Informative)

6.17.5.2.1 Request

POST /notifications/contacts HTTP/1.1
Accept: application/xml
Host: application.example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesNotification xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <callbackData>54321</callbackData>
  <resourceStatus>Active</resourceStatus>
  <duration>4217</duration>
  <link rel="Contact" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/ole"/>
  <link rel="AbChangesSubscription" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/123456789"/>
</ab:abChangesNotification>

6.17.5.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT

6.17.5.3 Example 3: Notification for expired contact subscription  (Informative)

6.17.5.3.1 Request

POST /notifications/contacts HTTP/1.1
Accept: application/xml
Host: application.example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesNotification xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <callbackData>54321</callbackData>
  <resourceStatus>TerminatedTimeout</resourceStatus>
  <link rel="AbChangesSubscription" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/123456789"/>
</ab:abChangesNotification>

6.17.5.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
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 14.7 of [RFC 2616].

6.18 Resource: Collection of contacts shared by other user
The resource used is:

```
http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/contacts
```

This resource is used to retrieve contacts shared by user identified by otherUserId.

6.18.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>sharing user identifier. Examples: tel:+19585550100, acr:pseudonym123</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 the RESTful Address Book API, see section 7.

6.18.3 GET
This operation is used for retrieval of all contacts shared by user identified by otherUserId.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a contact in a contact collection, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue is absent, GET response body SHALL include all contact sub-elements, plus all contact attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Contact attribute name (e.g. name1), GET response body SHALL include all contact sub-elements, plus only selected contact attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = ~noAttr, GET response body SHALL include all contact sub-elements and none of the contact attributes.</td>
</tr>
</tbody>
</table>
6.18.3.1 Example 1: Retrieve contacts shared by a user identified by otherUserId with all contact attributes (default) (Informative)

6.18.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts HTTP/1.1
Accept: application/xml
Host: example.com

6.18.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:contactCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"><contact>
<contactId>alice</contactId>
<sharedIdentity>
  <sharedId>sip:alice@example.com</sharedId>
</sharedIdentity>
<attributeList>
  <attribute>
    <name>display-name</name>
    <value>Alice</value>
  </attribute>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/sip%3Aalice@example.com/attributes</resourceURL>
</attributeList>
<resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/sip%3Aalice@example.com</resourceURL></contact>

<contact>
<contactId>ole</contactId>
<sharedIdentity>
  <sharedId>mailto:ole@example.com</sharedId>
</sharedIdentity>
<attributeList>
  <attribute>
    <name>display-name</name>
    <value>Ole</value>
  </attribute>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/mailto%3Aole@example.com/attributes</resourceURL>
</attributeList>
</contact>

If indivFilterValue = ~none, GET response body SHALL include none of the contact information (contact structure to be omitted).

No other indivFilterValue is supported.
6.18.3.2 Example 2: Retrieve contacts shared by a user identified by otherUserId without contacts attributes

(Informative)

6.18.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts?indivFilter~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.18.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contactCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contact>
    <contactId>alice</contactId>
    <sharedIdentity>
      <sharedId>sip:alice@example.com</sharedId>
    </sharedIdentity>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/sip%3Aalice@example.com</resourceURL>
  </contact>
  <contact>
    <contactId>ole</contactId>
    <sharedIdentity>
      <sharedId>mailto:ole@example.com</sharedId>
    </sharedIdentity>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/mailto%3Aole@example.com</resourceURL>
    <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/lists/6789/members/mailto%3Aole@example.com"/>
  </contact>
</ab:contactCollection>
6.18.4 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 14.7 of [RFC 2616].

6.18.5 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 14.7 of [RFC 2616].

6.18.6 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 14.7 of [RFC 2616].

6.19 Resource: Individual shared contact
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/contacts/{sharedIdentity}
This resource is used to retrieve a contact shared by user identified by otherUserId.

6.19.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>sharing user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>sharedIdentity</td>
<td>identifier of the shared contact. Examples: alice, <a href="mailto:alice@example.com">mailto:alice@example.com</a></td>
</tr>
</tbody>
</table>

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 the RESTful Address Book API, see section 7.

6.19.3 GET
This resource is used to retrieve individual data about the shared contact.
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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a contact in a contact collection, returned in a GET response body. If indivFilterValue is absent, GET response body SHALL include all contact sub-elements, plus all contact attributes (default behaviour).</td>
</tr>
</tbody>
</table>
6.19.3.1 Example 1: Retrieve information about individual shared contact using contactId (Informative)

In this example the contactId is known to the requestor.

6.19.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B 19585550105/shared/tel%3A%2B19585550100/contacts/alice HTTP/1.1
Accept: application/xml
Host: example.com

6.19.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Alice</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550109</value>
    </attribute>
    <attribute>
      <name>state</name>
      <value>California</value>
    </attribute>
  </attributeList>
</ab:contact>
6.19.3.2 Example 2: Retrieve information about individual shared contact using sharedIdentity (Informative)

In this example the requestor uses the publicly known address (sharedIdentity) of the shared contact to be retrieved.

6.19.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/mailto%3Aalice@example.com HTTP/1.1
Accept: application/xml
Host: example.com

6.19.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Alice</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>tel:+19585550109</value>
    </attribute>
    <attribute>
      <name>state</name>
      <value>California</value>
    </attribute>
  </attributeList>
</ab:contact>
6.19.3.3 Example 3: Retrieve information about individual shared contact, with selected contact attributes (default) (Informative)

6.19.3.3.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/alice?indivFilter=cellphone
HTTP/1.1
Accept: application/xml
Host: example.com

6.19.3.3.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:contact xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <contactId>alice</contactId>
  <attributeList>
  <attribute>
    <name>cellphone</name>
    <value>tel:+19585550109</value>
  </attribute>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/alice</resourceURL>
  <link rel="Member" href="http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550122"/>
</ab:contact>

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’ field in the response as per section 14.7 of [RFC 2616].

6.19.5 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 14.7 of [RFC 2616].

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’ field in the response as per section 14.7 of [RFC 2616].

6.20 Resource: Collection of shared lists

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists

This resource is used to retrieve lists shared by user identified by otherUserId.
6.20.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>sharing user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
</tbody>
</table>

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 the RESTful Address Book API, see section 7.

6.20.3 GET

This operation is used for retrieval of all lists shared by user identified by otherUserId.

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>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilter is absent, GET response body SHALL include all the list sub-elements, plus all list attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = List attribute name (e.g. name1), GET response body SHALL include all list sub-elements, plus only selected list attributes (i.e. name1). Any number of listFilter=listFilterValue pairs separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = ~noAttr, GET response body SHALL include all the list sub-elements and none of the list attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No other listFilterValue is supported.</td>
</tr>
<tr>
<td>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilter is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue = Member attribute name (e.g. name1), GET</td>
</tr>
</tbody>
</table>
#### 6.20.3.1 Example 1: Retrieve lists shared by a user identified by otherUserId with all list attributes and all member attributes (default) (Informative)

**6.20.3.1.1 Request**

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists HTTP/1.1
Accept: application/xml
Host: example.com
```

**6.20.3.1.2 Response**

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
    <list>
        <listId>1234</listId>
        <memberCollection>
            <member>
                <memberId>sip:alice@example.com</memberId>
                <attributeList>
                    <attribute>
                        <name>display-name</name>
                        <value>Alice</value>
                    </attribute>
                    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
                </attributeList>
            </member>
        </memberCollection>
    </list>
</ab:listCollection>
```
<memberCollection>
<category>URLList</category>
<sharedListIdentity>
  <sharedId>sip:bob@example.com;list=1234</sharedId>
</sharedListIdentity>
<attributeList>
  <attribute>
    <name>display-name</name>
    <value>Friends</value>
  </attribute>
</attributeList>
</list>
<list>
  <listId>5678</listId>
  <memberCollection>
    <member>
      <memberId>mailto:liza@example.com</memberId>
      <attributeList>
        <attribute>
          <name>display-name</name>
          <value>Wife</value>
        </attribute>
        <attribute>
          <name>age</name>
          <value>42</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com/attributes</resourceURL>
      </attributeList>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com</resourceURL>
    </member>
    <member>
      <memberId>mailto:serezha@example.com</memberId>
      <attributeList>
        <attribute>
          <name>display-name</name>
          <value>Son</value>
        </attribute>
        <attribute>
          <name>phone</name>
          <value>tel:+19585550105</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com/attributes</resourceURL>
      </attributeList>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com</resourceURL>
    </member>
  </memberCollection>
</list>
</memberCollection>

6.20.3.2 Example 2: Retrieve lists shared by a user identified by otherUserId without lists attributes, but with selected member attributes (Informative)

6.20.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists?listFilter=~noAttr&indivFilter=phone
HTTP/1.1
Accept: application/xml
Host: example.com

6.20.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:listCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <list>
    <listId>1234</listId>
    <memberCollection>
      <member>
        <memberId>sip:alice@example.com</memberId>
      </member>
    </memberCollection>
    <category>URIList</category>
    <sharedListIdentity>
      <sharedId>sip:bob@example.com;list=1234</sharedId>
    </sharedListIdentity>
  </list>
</ab:listCollection>
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’ field in the response as per section 14.7 of [RFC 2616].

6.20.5 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 14.7 of [RFC 2616].

6.20.6 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 14.7 of [RFC 2616].

6.21 Resource: Individual shared list
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}

This resource is used to retrieve individual list identified by sharedListIdentity shared by user identified by otherUserId with user identified by userId.

### 6.21.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>sharing user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>sharedListIdentity</td>
<td>shared list identifier. Example: myList</td>
</tr>
</tbody>
</table>

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

### 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 the RESTful Address Book API, see section 7.

### 6.21.3 GET

This operation is used for retrieval of all individual shared list information for a given user identity.

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>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a list, returned in a GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilter is absent, GET response body SHALL include all the list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sub-elements, plus all list attributes (default behaviour).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = List attribute name (e.g. name1), GET response</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>body SHALL include all list sub-elements, plus only selected list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>attributes (i.e. name1). Any number of listFilter=listFilterValue pairs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>separated by &amp; are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If listFilterValue = ~noAttr, GET response body SHALL include all the list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sub-elements and none of the list attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No other listFilterValue is supported.</td>
</tr>
<tr>
<td>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in a GET response body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If indivFilterValue is absent, GET response body SHALL</td>
</tr>
</tbody>
</table>

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
6.21.3.1 Example 1: Retrieve individual list shared by a user identified by otherUserId, with all list attributes and all member attributes (default) (Informative)

6.21.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/xml
Host: example.com

6.21.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <memberCollection>
    <member>
      <memberId>sip:alice@example.com</memberId>
      <attributeList>
        <attribute>
          <name>display-name</name>
          <value>Alice</value>
        </attribute>
        <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
      </attributeList>
    </member>
  </memberCollection>
</ab:list>
6.21.3.2 Example 2: Retrieve individual list shared by a user identified by otherUserId, with all list attributes and no member attributes (default) (Informative)

6.21.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234?indivFilter=~noAttr HTTP/1.1
Accept: application/xml
Host: example.com

6.21.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:list xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <memberCollection>
    <member>
      <memberId>sip:alice@example.com</memberId>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com</resourceURL>
    </member>
  </memberCollection>
</ab:list>
6.21.4 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 14.7 of [RFC 2616].

6.21.5 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 14.7 of [RFC 2616].

6.21.6 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 14.7 of [RFC 2616].

6.22 Resource: Members in a shared list
The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}/members

This resource is used to retrieve all members in the list identified by sharedListIdentity shared by user identified by otherUserId with user identified by userId.

6.22.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>sharing user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>sharedListIdentity</td>
<td>shared list identifier. Example: myList</td>
</tr>
</tbody>
</table>

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

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.22.3 GET

This operation is used for retrieval of all members of a shared list for a given user identity.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body.</td>
</tr>
</tbody>
</table>

- If indivFilterValue is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour).
- If indivFilterValue = Member attribute name (e.g. name1), GET response body SHALL include all member sub-elements, plus only selected member attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by & are allowed.
- If indivFilterValue = ~noAttr, GET response body SHALL include all member sub-elements and none of the member attributes.
- If indivFilterValue = ~none, GET response body SHALL include none of the member information (member structure to be omitted).

No other indivFilterValue is supported.

6.22.3.1 Example: Retrieve all members for a shared list belonging to a user with all attributes (default) (Informative)

6.22.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members HTTP/1.1
Accept: application/xml
Host: example.com

6.22.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:memberCollection xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <member>
    <memberId>mailto:alice@example.com</memberId>
    <attributeList>
      <attribute>
        <name>display-name</name>
        <value>Alice</value>
      </attribute>
      <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes</resourceURL>
    </attributeList>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com</resourceURL>
  </member>

  <member>
    <memberId>tel:+19585550124</memberId>
    <attributeList>
      <attribute>
        <name>display-name</name>
        <value>Bob</value>
      </attribute>
    </attributeList>
  </member>

</ab:memberCollection>

### 6.22.4 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 14.7 of [RFC 2616].

### 6.22.5 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 14.7 of [RFC 2616].

### 6.22.6 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 14.7 of [RFC 2616].
6.23 Resource: Individual member information from shared list

The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/sharedBy/{otherUserId}/{sharedListId}/members/{sharedMemberId}

This resource is used to retrieve individual member information from the list shared by user identified by otherUserId.

6.23.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>requesting user identifier. Examples: tel:+19585550105, acr:pseudonym456</td>
</tr>
<tr>
<td>otherUserId</td>
<td>other user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>sharedListIdentity</td>
<td>shared list identifier. Examples: myList</td>
</tr>
<tr>
<td>memberId</td>
<td>list member identifier. Example: <a href="mailto:alice@example.com">mailto:alice@example.com</a></td>
</tr>
</tbody>
</table>

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

6.23.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.23.3 GET

This operation is used for retrieval of all information about a member in a shared list.

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>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Defines the level of information related to a member in a list, returned in a GET response body. If indivFilterValue is absent, GET response body SHALL include all member sub-elements, plus all member attributes (default behaviour). If indivFilterValue = Member attribute name (e.g. name1), GET response body SHALL include all member sub-elements, plus only selected member attributes (i.e. name1). Any number of indivFilter=indivFilterValue pairs separated by &amp; are allowed. No other indivFilterValue is supported.</td>
</tr>
</tbody>
</table>
6.23.3.1 Example 1: Retrieve all member attributes about user identified as memberId from the list shared by a user identified by otherUserId (default) (Informative)

6.23.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com HTTP/1.1
Accept: application/xml
Host: example.com

6.23.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:member xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <memberId>sip:alice@example.com</memberId>
  <attributeList>
    <attribute>
      <name>display-name</name>
      <value>Alice</value>
    </attribute>
    <attribute>
      <name>cellphone</name>
      <value>19585550105</value>
    </attribute>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes</resourceURL>
  </attributeList>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com</resourceURL>
</ab:member>

6.23.3.2 Example 2: Retrieve selected member attributes about user identified as memberId from the list shared by a user identified by otherUserId (default) (Informative)

6.23.3.2.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com?indivFilter=cellphone HTTP/1.1
Accept: application/xml
Host: example.com

6.23.3.2.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
6.23.4 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 14.7 of [RFC 2616].

6.23.5 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 14.7 of [RFC 2616].

6.23.6 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 14.7 of [RFC 2616].

6.24 Resource: Authorization rules
The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules

This resource is used to retrieve rules for access to shared lists, as well as for the creation of new authorization rules.

6.24.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
</tbody>
</table>

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

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.24.3 GET

This operation is used for retrieval of all authorization rules.

6.24.3.1 Example: Retrieve all authorization rules (Informative)

6.24.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules HTTP/1.1
Accept: application/xml
Host: example.com

6.24.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:ruleList xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <rule>
    <ruleName>allowAllContacts</ruleName>
    <clientCorrelator>123</clientCorrelator>
    <applicationTag>myApp</applicationTag>
    <applToContacts>
      <allContacts/>
    </applToContacts>
    <authorizedTo>
      <listId>1234</listId>
    </authorizedTo>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1</resourceURL>
  </rule>
  <rule>
    <ruleName>allowOwnContact</ruleName>
    <clientCorrelator>456</clientCorrelator>
    <applicationTag>myApp</applicationTag>
    <applToContacts>
      <contactId>bob</contactId>
    </applToContacts>
    <authorizedTo>
      <allSharedIdentities/>
    </authorizedTo>
    <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2</resourceURL>
  </rule>
  <rule>
    <ruleName>allowFriendsList</ruleName>
    <clientCorrelator>789</clientCorrelator>
    <applicationTag>myApp</applicationTag>
    <applToLists>
      <listId>1234</listId>
    </applToLists>
    <authorizedTo>
      <allSharedIdentities/>
    </authorizedTo>
  </rule>
</ab:ruleList>
6.24.4 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 14.7 of [RFC 2616].

6.24.5 POST

This operation is used for creation of a new authorization rule.

6.24.5.1 Example: Create new rule for list and referenced contacts  (Informative)

This rule allows users in the specified list (authorizedTo element) to retrieve the members in the same list and also the contacts that have references to the members in the list.

6.24.5.1.1 Request

```
POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <ruleName>combinedRule</ruleName>
  <clientCorrelator>135</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <applToListsAndContacts>
    <listId>5678</listId>
  </applToListsAndContacts>
  <authorizedTo>
    <listId>5678</listId>
  </authorizedTo>
</ab:rule>
```

6.24.5.1.2 Response

```
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule3
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
```

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
6.24.6 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 14.7 of [RFC 2616].

6.25 Resource: Individual authorization rule
The resource used is:

http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules/{ruleId}

This resource is used to retrieve an individual authorization rule for access to shared lists.

6.25.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>ruleId</td>
<td>rule identifier. Example: 987654321</td>
</tr>
</tbody>
</table>

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

6.25.2 Response Codes and Error Handling
For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.25.3 GET
This operation is used for retrieval of an individual authorization rule.

6.25.3.1 Example: Retrieve an authorization rule (Informative)

6.25.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1 HTTP/1.1
Accept: application/xml
Host: example.com
6.25.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"
  <ruleName>allowAllContacts</ruleName>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <applToContacts>
    <allContacts/>
  </applToContacts>
  <authorizedTo>
    <listId>1234</listId>
    <listId>5678</listId>
  </authorizedTo>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1</resourceURL>
</ab:rule>

6.25.4 PUT

This operation is used to update an individual authorization rule.

6.25.4.1 Example: Update an existing authorization rule (Informative)

6.25.4.1.1 Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1 HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1"
  <ruleName>allowAllContacts</ruleName>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <applToContacts>
    <allContacts/>
  </applToContacts>
  <authorizedTo>
    <listId>1234</listId>
    <listId>5678</listId>
  </authorizedTo>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1</resourceURL>
</ab:rule>

6.25.4.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <ruleName>allowAllContacts</ruleName>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <applToContacts>
    <allContacts/>
  </applToContacts>
  <authorizedTo>
    <listId>1234</listId>
    <listId>5678</listId>
  </authorizedTo>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1</resourceURL>
</ab:rule>

6.25.5 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 14.7 of [RFC 2616].

6.25.6 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 14.7 of [RFC 2616].

6.25.6.1.1 Request
DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2 HTTP/1.1
Accept: application/xml
Host: example.com

6.25.6.1.2 Response
HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT

6.26 Resource: Individual authorization rule data
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/authorizationRules/{ruleId}/{ResourceRelPath}
This resource is used to retrieve an individual authorization rule for access to shared lists.
This lightweight resource is used to manage authorized identities in an existing authorization rule, which include creation, update, retrieval, and delete operations.
### 6.26.1 Request URL variables

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

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>ruleId</td>
<td>rule identifier: Example: 987654321</td>
</tr>
<tr>
<td>[ResourceRelPath]</td>
<td>Light-weight relative resource path. The allowed string for this light-</td>
</tr>
<tr>
<td></td>
<td>weight resource is defined in the [ResourceRelPath] column in table 5.2.2.19</td>
</tr>
</tbody>
</table>

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

### 6.26.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

### 6.26.3 GET

This operation is used for retrieval of authorized identity in an individual authorization rule. It is typically used to verify if it already exists.

#### 6.26.3.1 Example: Retrieve data from an authorization rule (Informative)

##### 6.26.3.1.1 Request

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1/lists/1234 HTTP/1.1
Accept: application/xml
Host: example.com

##### 6.26.3.1.2 Response

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ab:listId xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">1234</ab:listId>
```

### 6.26.4 PUT

This operation is used to add an authorized identity to individual authorization rule.
6.26.4.1  Example: Authorize user by updating an existing authorization rule  
(Informative)

6.26.4.1.1  Request

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:userId xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">mailto:alice@example.com</ab:userId>

6.26.4.1.2  Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com
Content-Type: application/xml
Content-Length : nnnn

<?xml version="1.0" encoding="UTF-8"?><ab:userId xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">mailto:alice@example.com</ab:userId>

6.26.5  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 14.7 of [RFC 2616].

6.26.6  DELETE

This operation removes a previously authorized identity from an authorization rule.

6.26.6.1  Example: Delete an authorized identity  
(Informative)

6.26.6.1.1  Request

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com HTTP/1.1
Accept: application/xml
Host: example.com

6.26.6.1.2  Response

HTTP/1.1 204 No Content
Date: Fri, 10 Dec 2010 12:53:23 GMT
6.27 Resource: Individual member transfer
The resource used is:
http://{serverRoot}/addressbook/{apiVersion}/{userId}/lists/{listId}/members/{memberId}/transfer
This resource is used to transfer a member from one list to another list.

6.27.1 Request URL variables
The following request URL variables are common for all HTTP commands:

<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 5.1.</td>
</tr>
<tr>
<td>userId</td>
<td>user identifier. Examples: tel:+19585550100, acr:pseudonym123</td>
</tr>
<tr>
<td>listId</td>
<td>list identifier. Example: allFriends</td>
</tr>
</tbody>
</table>

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

6.27.2 Response Codes and Error Handling
For HTTP response codes, see [REST_NetAPI_Common].
For Policy Exception and Service Exception fault codes applicable to the RESTful Address Book API, see section 7.

6.27.3 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 14.7 of [RFC 2616].

6.27.4 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 14.7 of [RFC 2616].

6.27.5 POST
This operation is used to transfer a member from one list to another list. In case of successful transfer, “303 See Other” SHALL be returned, providing a Location header and a resourceReference root element with the location representing the member in the destination list.

6.27.5.1 Example: Transfer a member from one list to another (Informative)

6.27.5.1.1 Request

```xml
POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/members/tel%3A%2B19585550199/transfer
HTTP/1.1Accept: application/xml
Host: example.com
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:memberTransferParameters xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <destination>http://exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts</destination>
</ab:memberTransferParameters>
```
6.27.5.1.2 **Response**

HTTP/1.1 303 See Other  
Date: Fri, 10 Dec 2010 12:53:23 GMT  
Location:  
http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts/members/tel%3A%2B19585550199  
Content-Type: application/xml  
Content-Length: nnnn

```xml
<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
</common:resourceReference>
```

6.27.6 **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 14.7 of [RFC 2616].
7. Fault definitions

7.1 Service Exceptions

The following Service Exception codes are defined for the RESTful Address Book API in addition to those in [3GPP 29.199-13].

7.1.1 SVC0240: Key property changes not allowed

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageId</td>
<td>SVC0240</td>
</tr>
<tr>
<td>Text</td>
<td>Key property changes not allowed: key property %1</td>
</tr>
<tr>
<td>Variables</td>
<td>%1 – key property</td>
</tr>
</tbody>
</table>

7.2 Policy Exceptions

The following Policy Exception codes are defined for the RESTful Address Book API in addition to those in [3GPP 29.199-13].

7.2.1 POL0214: Too many resources

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageId</td>
<td>POL0214</td>
</tr>
<tr>
<td>Text</td>
<td>Maximum number of permitted resources exceeded.</td>
</tr>
<tr>
<td>Variables</td>
<td>None</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—or No previous version within OMA</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>Draft version</td>
<td>03 May 2011</td>
<td>Many</td>
<td>Structural changes to fit the OMA RESTful Network API release. This version inherits the technical content of OMA-REST-ParlayREST_AddressListManagement-V1.0-20110111-C and applies changes according to ARC INP 30R01, 98R02, 155R01, 156R01, 167R02, 172R01, 175R01, 186, 187R02, and 167R04 (which also includes the fix equivalent of OMA-REST-M-2011-009R01-CR_ALM_ParlayX_fix).</td>
</tr>
<tr>
<td></td>
<td>06 Jun 2011</td>
<td>Many</td>
<td>Implemented AI: REST-NetAPI-2011-A025</td>
</tr>
<tr>
<td></td>
<td>22 Jun 2011</td>
<td>Section 5.2.2.7</td>
<td>Implemented CR: OMA-ARC-REST-NetAPI-2011-0054-CR_ALM_TS_fix_value_type_in_Attribute</td>
</tr>
<tr>
<td></td>
<td>11 Aug 2011</td>
<td>Many</td>
<td>Implemented OMA-ARC-REST-NetAPI-2011-0190-CR_ALM_address_5_and_7_from_ParlayREST_list</td>
</tr>
<tr>
<td></td>
<td>01 Sep 2011</td>
<td>Many</td>
<td>Implemented: OMA-ARC-REST-NetAPI-2011-0200R04-CR_ALM_ACR</td>
</tr>
<tr>
<td></td>
<td>13 Dec 2011</td>
<td>Many</td>
<td>Implemented: OMA-ARC-REST-NetAPI-2011-0441R01-CR_AddressBook_acr_Auto</td>
</tr>
<tr>
<td></td>
<td>10 Jan 2012</td>
<td>Many</td>
<td>Implemented: OMA-ARC-REST-NetAPI-2012-0001R01-CR_AddressBook_CONR_comments_resolution_1</td>
</tr>
<tr>
<td></td>
<td>13 Jan 2012</td>
<td>Many</td>
<td>No content changes—it is a re-issue of the last draft that had many tables formatting issues. For visibility, all previous changes were preserved. Also reversed to Word 2003 format (.doc)</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>25 Jan 2012</td>
<td>Many</td>
<td>Implemented: OMA-ARC-REST-NetAPI-2012-0020R03-CR_AddressBook_resolve_CONR_A019</td>
</tr>
<tr>
<td></td>
<td>26 Jan 2012</td>
<td>Many</td>
<td>Implemented: OMA-ARC-REST-NetAPI-2012-0030R01-CR_AddressBook_Fix_CONR_A035, editorials</td>
</tr>
<tr>
<td></td>
<td>30 Jan 2012</td>
<td>6.12, 6.27</td>
<td>Changed resource URLs to be all bold. Editorial changes</td>
</tr>
<tr>
<td>Candidate version</td>
<td>07 Feb 2012</td>
<td>n/a</td>
<td>Status changed to Candidate by TP&lt;br&gt;TP Ref # OMA-TP-2012-0029-INP_REST_NetAPI_AddressBook_1.0_ERP_and_ETR_for_Candidate_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.AB Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-SUPPORT-S-001-M</td>
<td>Support for the RESTful AB API</td>
<td>5, 6</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUPPORT-S-002-M</td>
<td>Support for the XML request &amp; response format</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUPPORT-S-003-M</td>
<td>Support for the JSON request &amp; response format</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUPPORT-S-004-O</td>
<td>Support for the application/x-form-urlencoded format</td>
<td>Appendix C</td>
<td></td>
</tr>
</tbody>
</table>

#### B.1.1 SCR for REST.AB.ContactCollection Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-CONTACT-COL-S-001-M</td>
<td>Support contact collection with all of user’s contacts</td>
<td>6.1</td>
<td></td>
</tr>
<tr>
<td>REST-AB-CONTACT-COL-S-002-M</td>
<td>This operation retrieves all contacts in the contact collection - GET</td>
<td>6.1.3</td>
<td></td>
</tr>
</tbody>
</table>

#### B.1.2 SCR for REST.AB.IndividualContact Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-CONTACT-IND--S-001-M</td>
<td>Support individual contact in contact collection</td>
<td>6.2</td>
<td></td>
</tr>
<tr>
<td>REST-AB-CONTACT-IND-S-002-M</td>
<td>This operation retrieves information of a contact in the contact collection - GET</td>
<td>6.2.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-CONTACT-IND-S-003-M</td>
<td>This operation adds or updates a contact in the contact collection - PUT</td>
<td>6.2.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-CONTACT-IND-S-004-M</td>
<td>This operation deletes a contact in the contact collection - DELETE</td>
<td>6.2.6</td>
<td></td>
</tr>
</tbody>
</table>

#### B.1.3 SCR for REST.AB.AttributesForAContact Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-ATTRIB-CONTACT-S-001-M</td>
<td>Supports retrieval of all attributes of a contact</td>
<td>6.3</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-AB-CONTACT-S-002-M</td>
<td>This operation retrieves all attributes for a contact – GET</td>
<td>6.3.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.4 SCR for REST.AB.IndividualAttributeForAContact Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-ATTRIB-CONTACT-S-001-M</td>
<td>Support for management (create, update, retrieve and delete) of a contact’s single attribute.</td>
<td>6.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-CONTACT-S-002-M</td>
<td>This operation returns the value of the attribute for a member – GET</td>
<td>6.4.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-CONTACT-S-003-M</td>
<td>This operation creates or updates an attribute for a member – PUT</td>
<td>6.4.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-CONTACT-S-004-M</td>
<td>This operation removes an attribute for a member - DELETE</td>
<td>6.4.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.5 SCR for REST.AB.Lists Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEM-LS-S-001-M</td>
<td>Support for retrieving all lists belonging to a user.</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>REST-AB-MEM-LS-S-002-M</td>
<td>This operation retrieves all lists the user has created - GET</td>
<td>6.5.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.6 SCR for REST.AB.IndividualList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-MEM-LS-001-M</td>
<td>Support for management (create, update, retrieve and delete) of an individual list.</td>
<td>6.6</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-002-M</td>
<td>This operation retrieves the list - GET</td>
<td>6.6.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-003-M</td>
<td>This operation creates or updates the entire list - PUT</td>
<td>6.6.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-004-M</td>
<td>This operation removes the list including attributes</td>
<td>6.6.6</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></td>
<td>and members - DELETE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B.1.7 SCR for REST.AB.AttributesForAList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-ATTRIB-MEM-LS-S-001-O</td>
<td>Support for retrieving all attributes for a given list belonging to a user.</td>
<td>6.7</td>
<td>REST-AB-ATTRIB-MEM-LS-S-002-O</td>
</tr>
<tr>
<td>REST-AB-ATTRIB-MEM-LS-S-002-O</td>
<td>This operation returns all attributes for a list - GET</td>
<td>6.7.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.8 SCR for REST.AB.IndividualAttributeForAList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-MGT-002-O</td>
<td>This operation returns the value of the list attribute - GET</td>
<td>6.8.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-MGT-003-O</td>
<td>This operation creates or updates an attribute - PUT</td>
<td>6.8.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-MGT-004-O</td>
<td>This operation deletes an attribute - DELETE</td>
<td>6.8.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.9 SCR for REST.AB.MemberInAList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MBR-MEM-LS-S-001-O</td>
<td>Supports retrieval of all members of a list belonging to a user</td>
<td>6.9</td>
<td>REST-AB-MBR-MEM-LS-S-002-O</td>
</tr>
<tr>
<td>REST-AB-MBR-MEM-LS-S-002-O</td>
<td>This operation retrieves the members in the list - GET</td>
<td>6.9.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.10 SCR for REST.AB.IndividualMemberInAList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<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-AB-IND-MBR-MEM-LS-S-002-O</td>
<td>This operation retrieves a user from a list - GET</td>
<td>6.10.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MBR-MEM-LS-S-003-O</td>
<td>This operation creates and updates an entry in the list - PUT</td>
<td>6.10.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MBR-MEM-LS-S-004-O</td>
<td>This operation removes the member from the list - DELETE</td>
<td>6.10.6</td>
<td></td>
</tr>
</tbody>
</table>

**B.1.11 SCR for REST.AB.AttributesForAMemberInAList Server**

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-ATTRIB-MBR-MEM-LS-S-001-O</td>
<td>Supports retrieval of all attributes of a member of a list belonging to a user</td>
<td>6.11</td>
<td></td>
</tr>
<tr>
<td>REST-AB-ATTRIB-MBR-MEM-LS-S-002-O</td>
<td>This operation returns all attributes for a member – GET</td>
<td>6.11.3</td>
<td></td>
</tr>
</tbody>
</table>

**B.1.12 SCR for REST.AB.IndividualAttributeForAMemberInAList Server**

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-S-001-O</td>
<td>Support for management (create, update, retrieve and delete) of member attributes on individual list.</td>
<td>6.12</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-S-002-O</td>
<td>This operation returns the value of the attribute for a member – GET</td>
<td>6.12.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-S-003-O</td>
<td>This operation creates or updates an attribute for a member – PUT</td>
<td>6.12.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-ATTRIB-MEM-LS-S-004-O</td>
<td>This operation removes an attribute for a member - DELETE</td>
<td>6.12.6</td>
<td></td>
</tr>
</tbody>
</table>

**B.1.13 SCR for REST.AB.ListReferences Server**

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEM-LS-REF-S-001-O</td>
<td>Supports retrieval of a list of references for all nested lists.</td>
<td>6.13</td>
<td></td>
</tr>
</tbody>
</table>

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
### B.1.14 SCR for REST.AB.IndivListReference Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEM-LS-REF-S-002-O</td>
<td>This operation returns a list of references to other lists. –GET</td>
<td>6.13.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-REF-S-002-O</td>
<td>This operation returns an individual list reference to another list –GET</td>
<td>6.14.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-REF-S-003-O</td>
<td>This operation creates or updates an individual reference to another list. –PUT</td>
<td>6.14.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM-LS-REF-S-004-O</td>
<td>This operation deletes an individual reference to another list. –DELETE</td>
<td>6.14.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.15 SCR for REST.AB.List.Subscr Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-SUBSCR-S-001-M</td>
<td>Support subscriptions for address book changes notifications</td>
<td>6.15</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUBSCR-S-002-M</td>
<td>This operation retrieves a list of subscriptions - GET</td>
<td>6.15.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUBSCR-S-003-M</td>
<td>This operation creates a subscription for list or contact collection change notifications – POST (XML or JSON)</td>
<td>6.15.5</td>
<td></td>
</tr>
<tr>
<td>REST-AB-SUBSCR-S-004-O</td>
<td>This operation creates a subscription for list or contact collection change notifications – POST (application/x-www-form-urlencoded)</td>
<td>C.1</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.16 SCR for REST.AB.Individual.Subscr Server
### B.1.17 SCR for REST.AB.Notif Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-NOTIF-S-001-M</td>
<td>Support for notifying application about list or contact collection changes</td>
<td>6.17</td>
</tr>
<tr>
<td>REST-AB-NOTIF-S-002-M</td>
<td>This operation notifies the application about list or contact collection changes - POST</td>
<td>6.17.5</td>
</tr>
</tbody>
</table>

### B.1.18 SCR for REST.AB.Shared.ContactCollection Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-SHAREDCONCOLL-S-001-O</td>
<td>Support shared contact collection</td>
<td>6.18</td>
</tr>
<tr>
<td>REST-AB-SHAREDCONCOLL-S-002-O</td>
<td>This operation retrieves the collection of shared contacts by one user with another user - GET</td>
<td>6.18.3</td>
</tr>
</tbody>
</table>

### B.1.19 SCR for REST.AB.Individual.Shared.Contact Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-SHAREDCON-S-001-</td>
<td>Support individual shared contact</td>
<td>6.19</td>
</tr>
<tr>
<td>Item</td>
<td>Function</td>
<td>Reference</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-SHAREDCON-S-002-O</td>
<td>This operation retrieves an individual shared contact by one user with another user - GET</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.20 SCR for REST.AB.Shared.Lists Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-SHAREDLISTS-S-001-O</td>
<td>Support shared lists</td>
<td>6.20</td>
<td>REST-AB-SHAREDLISTS-S-002-O</td>
</tr>
<tr>
<td>REST-AB-SHAREDLISTS-S-002-O</td>
<td>This operation retrieves a collection of all shared lists by one user with another user - GET</td>
<td>6.20.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.21 SCR for REST.AB.Individual.Shared.List Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-SHAREDLIST-S-001-O</td>
<td>Support individual shared list</td>
<td>6.21</td>
<td>REST-AB-IND-SHAREDLIST-S-002-O</td>
</tr>
<tr>
<td>REST-AB-IND-SHAREDLIST-S-002-O</td>
<td>This operation retrieves an individual shared list by one user with another user - GET</td>
<td>6.21.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.22 SCR for REST.AB.Members.Ind.Shared.List Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEM-IND-SHAREDLIST-S-001-O</td>
<td>Support list of members in individual shared list</td>
<td>6.22</td>
<td>REST-AB-MEM-IND-SHAREDLIST-S-002-O</td>
</tr>
<tr>
<td>REST-AB-MEM-IND-SHAREDLIST-S-002-O</td>
<td>This operation retrieves a list of members in an individual shared list by one user with another user - GET</td>
<td>6.22.3</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.23 SCR for REST.AB.Member.Shared.List Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEMBER-SHAREDLIST-S-001-O</td>
<td>Support individual member in shared list</td>
<td>6.23</td>
<td>REST-AB-MEMBER-SHAREDLIST-S-002-O</td>
</tr>
</tbody>
</table>
### B.1.24 SCR for REST.AB.Authorization.RuleList Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-MEMBER-SHAREDLIST-S-002-O</td>
<td>This operation retrieves information of an individual member in a shared list - GET</td>
<td></td>
<td>6.23.3</td>
</tr>
<tr>
<td>REST-AB-AUTH-RULE-LS-S-002-O</td>
<td>This operation retrieves all authorization rules owned by a given user - GET</td>
<td></td>
<td>6.24.3</td>
</tr>
<tr>
<td>REST-AB-AUTH-RULE-LS-S-003-O</td>
<td>This operation creates a new authorization rule – POST (XML or JSON)</td>
<td>6.24.5</td>
<td></td>
</tr>
<tr>
<td>REST-AB-AUTH-RULE-LS-S-004-O</td>
<td>This operation creates a new authorization rule – POST</td>
<td>C.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(application/x-www-form-urlencoded)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B.1.25 SCR for REST.AB.Indiv.AuthorizationRule Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-AUTH-RULE-S-002-O</td>
<td>This operation retrieves an individual authorization rule - GET</td>
<td>6.25.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-AUTH-RULE-S-003-O</td>
<td>This operation updates an individual authorization rule - PUT</td>
<td>6.25.4</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-AUTH-RULE-S-004-O</td>
<td>This operation deletes an individual authorization rule - DELETE</td>
<td>6.25.6</td>
<td></td>
</tr>
</tbody>
</table>

### B.1.26 SCR for REST.AB.Indiv.AuthorizationRuleData Server

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></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-AB-IND-AUTH-RULE-DATA-S-002-O</td>
<td>This operation retrieves an individual authorization rule specific data - GET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-AUTH-RULE-DATA-S-003-O</td>
<td>This operation updates an individual authorization rule specific data - PUT</td>
<td>6.26.3</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-AUTH-RULE-DATA-S-004-O</td>
<td>This operation deletes an individual authorization rule specific data - DELETE</td>
<td>6.26.4</td>
<td></td>
</tr>
</tbody>
</table>

**B.1.27 SCR for REST.AB.IndividualMemberTransfer Server**

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST-AB-IND-MEM TRANS-S-001-M</td>
<td>Supports for transferring a member to another list</td>
<td>6.27</td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM TRANS-S-002-M</td>
<td>Transferring a member to another list – POST (XML or JSON)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REST-AB-IND-MEM TRANS-S-003-O</td>
<td>Transferring a member to another list – POST (application/x-www-form-urlencoded)</td>
<td>C.3</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C. Application/x-www-form-urlencoded Request Format for POST Operations (Normative)

This section defines a format for RESTful Address Book API requests where the body of the request is encoded using the application/x-www-form-urlencoded MIME type.

Note: only the request body is encoded as application/x-www-form-urlencoded, the response is still encoded as XML or JSON depending on the preference of the client and the capabilities of the server. Names and values MUST follow the application/x-www-form-urlencoded character escaping rules from [W3C_URLENC].

The encoding is defined below for the following Address Book REST operations which are based on POST requests:

- Create a subscription for list changes notification
- Create an authorization rule
- Transfer a member from one list to another list

C.1 Create a subscription for list changes notifications

This operation is used to create a new subscription for list changes notifications, see section 6.15.5.

This REST operation is used by the application to start the list changes notifications. It MUST use the HTTP POST operation. If the operation was successful, it returns an HTTP Status of “201 Created”.

The notifyURL 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 following parameters are defined:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anyContacts</td>
<td>xsd:boolean</td>
<td>Choice</td>
<td>Indicates a subscription for changes about any contacts in the ‘ContactCollection’.</td>
</tr>
<tr>
<td>listId</td>
<td>xsd:anyURI</td>
<td>Choice</td>
<td>Indicates a subscription for changes about the specified list.</td>
</tr>
<tr>
<td>notifyURL</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Notification endpoint definition. For the use of Client-side Notification URLs and Server-side Notification URLs in this parameter, see sections 6.15 and 6.15.5.</td>
</tr>
<tr>
<td>callbackData</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Data the application can register with the server when subscribing to notifications, and that are passed back unchanged in each of the related notifications.</td>
</tr>
<tr>
<td>clientCorrelator</td>
<td>xsd:string</td>
<td>Yes</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 field SHOULD be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids re-sending the message in such situations. In case the field 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>
</tr>
<tr>
<td>applicationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>A tag that the client MAY use to tag this particular resource on the server. In case the field 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>
</tr>
<tr>
<td>duration</td>
<td>xsd:int</td>
<td>Yes</td>
<td>Subscription duration in seconds. Server would expire subscription after specified number of seconds after subscription creation. If not specified – default value assigned by the server.</td>
</tr>
<tr>
<td>notificationFormat</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Default: XML Application can specify format of the resource representation in notifications that are related to this subscription. The choice is between {XML, JSON}.</td>
</tr>
</tbody>
</table>

Note: “choice” indicates an optional selection of either anyContacts or listId.

### C.1.1 Example using ‘tel’ URI (Informative)

#### C.1.1.1 Request

```
POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/subscriptions/abChanges HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn

listId=1234&notifyURL=http://application.example.com/notifications/lists/1234&callbackData=12345&
```
C.1.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321</resourceURL>
</ab:abChangesSubscription>

C.1.2 Example using ‘acr’ URI
(Informative)

C.1.2.1 Request

POST /exampleAPI/addressbook/v1/acr%3Apseudonym123/lists/subscriptions HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn

listId=1234&
notifyURL=http://application.example.com/notifications/lists/1234&
callbackData=12345&
clientCorrelator=123&
applicationTag=myApp&
duration=7200&

C.1.2.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/acr%3Apseudonym123/subscriptions/abChanges/987654321
Content-Type: application/xml
Content-Length: nnnn
C.2 Create an authorization rule

This operation is used to create an authorization rule, see section 6.24.5. It MUST use the HTTP POST operation. If the operation was successful, it returns an HTTP Status of “201 Created”.

The following parameters are defined:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ab:abChangesSubscription xmlns:ab="urn:oma:xml:restapi:addressbook:1">
  <listId>1234</listId>
  <callbackReference>
    <notifyURL>http://application.example.com/notifications/lists/1234</notifyURL>
    <callbackData>12345</callbackData>
  </callbackReference>
  <clientCorrelator>123</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <duration>7200</duration>
  <resourceURL>http://example.com/exampleAPI/addressbook/v1/acr%3Apseudonym123/subscriptions/abChanges/987654321</resourceURL>
</ab:abChangesSubscription>
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleName</td>
<td>xsd:ID</td>
<td>No</td>
<td>A name associated with the rule. It is a key property of the rule and when included in the light-weight resource URL then it SHALL NOT be altered.</td>
</tr>
<tr>
<td>clientCorrelator</td>
<td>xsd:string</td>
<td>Yes</td>
<td>A correlator that the client MAY use to tag this particular resource representation during a request to create a resource on the server. In case the field 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>
</tr>
<tr>
<td>applicationTag</td>
<td>xsd:string</td>
<td>Yes</td>
<td>A tag that the client MAY use to tag this particular resource on the server. In case the field 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>
</tr>
<tr>
<td>appToContacts</td>
<td>xsd:anyURI</td>
<td>Choice1</td>
<td>The rule applies to the list of specified ‘contactId’ that are in the ‘ContactCollection’.</td>
</tr>
<tr>
<td>appToAllContacts</td>
<td>xsd:boolean</td>
<td>Choice1</td>
<td>The rule applies to all contacts.</td>
</tr>
<tr>
<td>appToLists</td>
<td>xsd:anyURI</td>
<td>Choice2</td>
<td>The rule applies to the specified list of ‘listId’.</td>
</tr>
<tr>
<td>appToAllLists</td>
<td>xsd:boolean</td>
<td>Choice2</td>
<td>The rule applies to all lists.</td>
</tr>
<tr>
<td>appToListsAndContacts</td>
<td>xsd:anyURI</td>
<td>Choice3</td>
<td>The rule applies to the specified list. The rule applies also to contacts that are referenced from the specified list.</td>
</tr>
<tr>
<td>appToAllListsAndContacts</td>
<td>xsd:boolean</td>
<td>Choice3</td>
<td>The rule applies for all lists and referenced contacts.</td>
</tr>
<tr>
<td>authorizedToUserId</td>
<td>xsd:string</td>
<td>Choice4</td>
<td>User is authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToAllSharedIdentities</td>
<td>xsd:boolean</td>
<td>Choice4</td>
<td>All contacts with shared identities are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToListId</td>
<td>xsd:anyURI</td>
<td>Choice4</td>
<td>Users within the list are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToAllLists</td>
<td>xsd:boolean</td>
<td>Choice4</td>
<td>Users within all lists are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToExternalList</td>
<td>xsd:anyURI</td>
<td>Choice4</td>
<td>Users within the list (owned by another user) are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToDomainName</td>
<td>xsd:string</td>
<td>Choice4</td>
<td>Users within the domain are authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>authorizedToOtherUser</td>
<td>xsd:boolean</td>
<td>Choice4</td>
<td>Any user is authorized to access a resource for which the rule is applicable to.</td>
</tr>
<tr>
<td>listFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Filter for list attributes (section 6.5) the retrieving user is allowed to see. An empty filter means that the user has access to all attributes.</td>
</tr>
<tr>
<td>indivFilter</td>
<td>xsd:string</td>
<td>Yes</td>
<td>Filter for member and contact attributes (section 6.1 and 6.9) the retrieving user is allowed to see. An empty filter means that the user has access to all attributes.</td>
</tr>
</tbody>
</table>

Note:
- “choice1” indicates an optional selection of either applicableToContacts or applicableToAllContacts
- “choice2” indicates an optional selection of either applicableToLists or applicableToAllLists
- “choice3” indicates an optional selection of either applicableToListsAndContacts or applicableToAllListsAndContacts
- “choice4” indicates an optional selection of either authorizedToUserId or authorizedToAllSharedIdentities or authorizedToListId or authorizedToAllLists or authorizedToplicableToContacts or applicableToAllContacts

C.2.1 Example

C.2.1.1 Request

POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn

ruleName=combinedRule&
clientCorrelator=135&
applicationTag=myApp&
applToListsAndContacts=5678&
authorizedToListId=5678

C.2.1.2 Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule3
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<ab:rule xmlns:ab="urn:oma:xml:rest:netapi:addressbook:1">
  <ruleName>combinedRule</ruleName>
  <clientCorrelator>135</clientCorrelator>
  <applicationTag>myApp</applicationTag>
  <applToListsAndContacts>
    <listId>5678</listId>
  </applToListsAndContacts>
  <authorizedTo>
    <listId>5678</listId>
  </authorizedTo>
</ab:rule>

C.3 Transfer a member from one list to another

This operation is used to transfer a member from one list to another, see section 6.27.5.

The following parameters are defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type/Values</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>xsd:anyURI</td>
<td>No</td>
<td>Contains a relative URI representing the list where the member</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is to be transferred.</td>
</tr>
</tbody>
</table>
C.3.1 Example: (Informative)

C.3.1.1 Request

POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/members/tel%3A%2B19585550199/transfer HTTP/1.1
Accept: application/xml
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: nnnn

destination=
http%3A%2F%2Fexample.com%2FexampleAPI%2Faddressbook%2Ftel%3A%2B19585550100%2Fmembers%2FblockedContacts

C.3.1.2 Response

HTTP/1.1 303 See Other
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location:
http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts/members/tel%3A%2B19585550199
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
</common:resourceReference>
Appendix D. JSON examples

(Informative)

JSON (JavaScript Object Notation) is a lightweight, 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 invocations from browsers or other processors with JavaScript engines. Further information on JSON can be found at [RFC 4627].

The following examples show the request and response for various operations using a JSON binding. 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 Retrieve all contacts with all attributes (default) (section 6.1.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactCollection": {
  "contact": [
    {
      "attributeList": {"attribute": [
        {
          "name": "display-name",
          "value": "Alice"
        },
        {
          "name": "cellphone",
          "value": "tel:+19585550109"
        },
        {
          "name": "state",
          "value": "California"
        }
      ],
      "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes"
    },
    {"contactId": "alice",
     "link": {
      "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/tel%3A%2B19585550122",
      "next": 
    }}}}
D.2 Retrieve a contact with all attributes (default) (section 6.2.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice HTTP/1.1
Accept: application/json
Host: example.com

Response:
D.3 Retrieve a contact with only vCard2.1 as selected attribute (section 6.2.3.3)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contact": {
   "attributeList": "{attribute": [
   {
      "name": "vCard2.1",
      "objectValue": "QkVHSU46VkNBUkQNClZFUlNJT046Mi4xDQpOOkd1bXA7Rm9ycmVzdA0KRk46Rm9ycmVzdCBHdW1wDQpPUkc6QnViYmEgR3Vtc"},

   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice"}
}}
D.4 Create a new contact (section 6.2.4.1)

Request:

```
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria HTTP/1.1
Content-Type: application/json
Accept: application/json
Content-Length : nnnn
Host: example.com
```

```
{"contact": {
  "attributeList": {
    "attribute": {
      "name": "cellphone",
      "value": "tel:+19585550106"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes"
  },
  "contactId": "maria",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria"
}}
```

Response:

```
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria
Content-Type: application/json
Content-Length: nnnn
```
D.5 Retrieve all attributes of a contact (section 6.3.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"attributeList": {
    "attribute": {
        "name": "cellphone",
        "value": "tel:+19585550106"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes"
}}

D.6 Retrieve a contact’s attribute (section 6.4.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/cellphone HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json

{"attributeList": {
    "attribute": {
        "name": "cellphone",
        "value": "tel:+19585550106"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/maria/attributes"
}}
D.7  Create or update a contact’s attribute (section 6.4.4.1)

Request:

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/alice/attributes/married  HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length : nnnn

{"attribute": {
 "name": "married",
 "value": "true"
 }}

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"attribute": {
 "name": "married",
 "value": "true"
 }}

D.8  Retrieve specific lists attributes and specific member attributes (section 6.5.3.3)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists?listFilter=display-name&indivFilter=display-name&indivFilter=phone
HTTP/1.1
Accept: application/json
Host: example.com

Response:
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"listCollection": {"list": [
{
  "attributeList": {"attribute": {
    "name": "display-name",
    "value": "Friends"
  },
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes",
},
  "category": "URIList",
  "memberCollection": {"member": {
    "attributeList": {"attribute": {
      "name": "display-name",
      "value": "Alice"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes",
  },
  "memberId": "sip:alice@example.com",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com",
},
  "listId": "1234",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234",
  "sharedListIdentity": {"sharedId": "sip:bob@example.com;list=1234"}
},
{"attributeList": {"attribute": {
  "name": "display-name",
  "value": "Family"
}},
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/attributes",
},
  "memberCollection": {"member": [
    {"attributeList": {"attribute": {
      "name": "display-name",
      "value": "Wife"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/mailto%3Aliza@example.com/attributes",
}}
},
  "memberId": "mailto:liza@example.com",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aliza@example.com",
},
{"attributeList": {"attribute": [
}
D.9 Retrieve a list with all attributes and all member attributes (default) (section 6.6.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"list": {
  "attributeList": {
    "attribute": {
      "name": "display-name",
      "value": "Son"
    },
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes"
  },
  "memberId": "mailto:serezha@example.com",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members",
  "listId": "5678",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678"
}}

resourceURL": http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678/members
"memberCollection": {"member": {  
  "attributeList": {"attribute": {  
   "name": "display-name",
   "value": "Alice"
  },
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes"
  },
  "memberId": "sip:alice@example.com",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com"
},
  "listId": "1234",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234"
}

D.10 Retrieve a non existing list (section 6.6.3.4)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/4321 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {
  "link": {
    "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/4321",
    "rel": "List"
  },
  "serviceException": {
    "messageId": "SVC0002",
    "text": "Invalid input value for message part %1",
    "variables": "listId"
  }
}}

D.11 Create a list (section 6.6.4.1)

Request:
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"list": {
    "memberCollection": {
        "member": {
            "memberId": mailto:alice@example.com,
            "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members/mailto%3Aalice@example.com"
        },
        "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members"
    },
    "listId": "bobPublic",
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic"
}}

Response:

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic
Content-Type: application/json
Content-Length: nnnn

{"list": {
    "memberCollection": {
        "member": {
            "memberId": mailto:alice@example.com,
            "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members/mailto%3Aalice@example.com"
        },
        "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic/members"
    },
    "listId": "bobPublic",
    "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/bobPublic"
}}

D.12 Retrieve all attributes for a list belonging to Bob (section 6.7.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes HTTP/1.1
Accept: application/json
Host: example.com

Response:
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"attributeList": {
  "attribute": [
    {
      "name": "display-name",
      "value": "Friends"
    },
    {
      "name": "date-created",
      "value": "Fri, 10 Dec 2010 11:28:34 GMT"
    }
  ],
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes"
}}

D.13 Retrieve an attribute value  (section 6.8.3.1)
Request:
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/date-created HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"attribute": {
  "name": "date-created",
  "value": "Fri, 10 Dec 2010 11:28:34 GMT"
}}

D.14 Create an attribute (section 6.8.4.1)
Request:
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/attributes/display-name HTTP/1.1
Accept: application/json
Host: example.com
D.15 Retrieve all members for a list belonging to a user with all attributes (default) (section 6.9.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"memberCollection": {
"member": [
{
"attributeList": {
"attribute": {
"name": "display-name",
"value": "Alice"
},
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes"
},
"memberId": "mailto:alice@example.com",
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes"
}
]
D.16 Retrieve a member and selected attributes  (section 6.10.3.2)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com?indivFilter=cellphone
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"member": {
  "attributeList": {"attribute": {
    "name": "cellphone",
    "value": "tel:+19585550155"
  },
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com"
},
"memberId": "sip:alice@example.com",
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com"
}}
D.17 Create a member without a link to the contact (section 6.10.4.1)

Request:

```plaintext
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com   HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length : nnnn

{"member": {
   "attributeList": {
      "attribute": {
         "name": "display-name",
         "value": "Ole"
      },
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com/attributes"
   },
   "memberId": "mailto:ole@example.com",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com"
}}
```

Response:

```plaintext
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com
Content-Type: application/json
Content-Length: nnnn

{"member": {
   "attributeList": {
      "attribute": {
         "name": "display-name",
         "value": "Ole"
      },
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com/attributes"
   },
   "memberId": "mailto:ole@example.com",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aole@example.com"
}}
```

D.18 Retrieve all attributes of a member of a list (section 6.11.3.1)

Request:
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/mailto%3Aalice@example.com/attributes HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"attributeList": {
  "attribute": [
    {
      "name": "display-name",
      "value": "Alice"
    },
    {
      "name": "cellphone",
      "value": "tel:+19585550155"
    }
  ],
  "resourceURL": 
  "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes"
}}

D.19 Retrieve a member's non existing attribute (section 6.12.3.2)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/pet
Accept: application/json
Host: example.com

Response:

HTTP/1.1 404 Not Found
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"serviceException": {
"messageId": "SVC0002",
"text": "Invalid Invalid input value for message part %1",
"variables": "pet"
}}}
D.20 Create a member’s attribute (section 6.12.4.1)

Request:

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/born HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"attribute": {
  "name": "born",
  "value": "1984"
}}

Response:

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/6789/members/sip%3Aalice@example.com/attributes/born
Content-Type: application/json
Content-Length: nnnn

{"attribute": {
  "name": "born",
  "value": "1984"
}}

D.21 Retrieve all list references in a list (section 6.13.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"listReferenceCollection": {
  "link": [
    {
    }}
D.22 Retrieve a list reference (section 6.14.3.1)

Request:

GET
/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/http%3A/http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnn

{"link": {
  "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/1234",
  "rel": "List"
}}

D.23 Create a list reference (section 6.14.4.1)

Request:

PUT
/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/http%3A/example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: example.com
Content-Length: nnnn
Response:

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:51:59 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/listReferences/
http%3A//example.com/exampleAPI/addressbook/v1/tel%3A%2B 19585550100/lists/5678
Content-Type: application/json
Content-Length: nnnn

{"link": {
  "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/5678",
  "rel": "List"
}}

D.24 Retrieve all list changes subscriptions (section 6.15.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscriptionCollection": {
  "abChangesSubscription": [
    {
      "applicationTag": "myApp",
      "callbackReference": {
        "callbackData": "12345",
        "notifyURL": "http://application.example.com/notifications/lists/1234"
      },
      "clientCorrelator": "123",
      "duration": "1958",
      "listId": "1234",
      "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321"
    },
    {
      "anyContacts": null,
    }
  ]
}}
D.25 Create new subscription for list changes notification using ‘tel’ URI (section 6.15.5.1)

Request

POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
  "applicationTag": "myApp",
  "callbackReference": {
    "callbackData": "12345",
    "notifyURL": "http://application.example.com/notifications/lists/1234"
  },
  "clientCorrelator": "123",
  "duration": "7200",
  "listId": "1234"
}}

Response

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
  "applicationTag": "myApp",
  "callbackReference": {
    "callbackData": "12345",
    "notifyURL": "http://application.example.com/notifications/lists/1234"
  },
  "clientCorrelator": "123",
  "duration": "7200",
  "listId": "1234",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321"
}}
D.26  Create new subscription for list changes notification using ‘acr’ URI (section 6.15.5.2)

Request

```
POST /exampleAPI/addressbook/v1/acr%3AsomePseudonym123/subscriptions/abChanges HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
   "applicationTag": "myApp",
   "callbackReference": {
      "callbackData": "12345",
      "notifyURL": "http://application.example.com/notifications/lists/1234"
   },
   "clientCorrelator": "123",
   "duration": "7200",
   "listId": "1234"
}}
```

Response

```
HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/acr%3AsomePseudonym123/subscriptions/abChanges/987654321
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
   "applicationTag": "myApp",
   "callbackReference": {
      "callbackData": "12345",
      "notifyURL": "http://application.example.com/notifications/lists/1234"
   },
   "clientCorrelator": "123",
   "duration": "7200",
   "listId": "1234",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/acr%3AsomePseudonym123/subscriptions/abChanges/987654321"
}}
```

D.27  Retrieve individual list changes subscription (section 6.16.3.1)

Request

```
GET /exampleAPI/addressbook/v1/tele%3A%2B19585550100/subscriptions/abChanges/987654321 HTTP/1.1
Accept: application/json
Host: example.com
```

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
  "applicationTag": "myApp",
  "callbackReference": {
    "callbackData": "12345",
    "notifyURL": "http://application.example.com/notifications/lists/1234"
  },
  "clientCorrelator": "123",
  "duration": "6817",
  "listId": "1234",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321"
}}

D.28 Update/Extend duration of subscription (section 6.16.4.1)

Request:

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321 HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
  "applicationTag": "myApp",
  "callbackReference": {
    "callbackData": "12345",
    "notifyURL": "http://application.example.com/notifications/lists/1234"
  },
  "clientCorrelator": "123",
  "duration": "7200",
  "listId": "1234",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/987654321"
}}

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"abChangesSubscription": {
D.29 Notification for contact change subscription (section 6.17.5.2)

Request:

```plaintext
POST /notifications/contacts HTTP/1.1
Accept: application/json
Host: application.example.com
Content-Type: application/json
Content-Length: nnnn

{"abChangesNotification": {
  "callbackData": "54321",
  "duration": "4217",
  "link": [
    {
      "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/contacts/ole",
      "rel": "Contact"
    },
    {
      "href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/subscriptions/abChanges/123456789",
      "rel": "AbChangesSubscription"
    }
  ],
  "resourceStatus": "Active"
}}
```

Response:

```plaintext
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
```

D.30 Retrieve contacts shared by a user identified by otherUserId with all contact attributes (section 6.18.3.1)

Request:

```plaintext
```

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.
Error! Reference source not found.
GET /exampleAPI/addressbook/v1/tel%3A%2B 19585550105/shared/tel%3A%2B19585550100/contacts HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contactCollection": {
"contact": [

{"attributeList":{"attribute":{
"name": "display-name",
"value": "Alice"
},
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/sip%3Aalice@example.com/attributes"},
"contactId": "alice",
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/sip%3Aalice@example.com",
"sharedIdentity": {"sharedId": "sip:alice@example.com"}
},

{"attributeList":{"attribute":{
"name": "display-name",
"value": "Ole"
},
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/mailto%3Aole@example.com/attributes"},
"contactId": "ole",
"link": {"href": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/shared/lists/6789/members/mailto%3Aole@example.com", "rel": "Member"},
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/mailto%3Aole@example.com",
"sharedIdentity": {"sharedId": "mailto:ole@example.com"}
}
],
"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts"}
D.31 Retrieve information about individual shared contact using contactId (section 6.19.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/contacts/alice HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"contact": {
 "attributeList": {"attribute": [
 {"name": "display-name",
 "value": "Alice"
 },
 {"name": "cellphone",
 "value": "tel:+19585550109"
 },
 {"name": "state",
 "value": "California"
 }
 ],
 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/alice/attributes"
 },
 "contactId": "alice",
 "link": {
 "rel": "Member"
 },
 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/alice"
 }

D.32 Retrieve lists shared by a user identified by otherUserId without lists attributes, but with selected member attributes (section 6.20.3.2)

Request:
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists?listFilter=~noAttr& indivFilter=phone
HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"listCollection": {"list": [
    {
        "category": "URIList",
        "memberCollection": {"member": [
            {"memberId": "sip:alice@example.com",
             "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com"
            },
            {"resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members"
            },
            {"listId": "1234",
             "sharedListIdentity": {"sharedId": "sip:bob@example.com;list=1234"}
            },
            {"memberCollection": {"member": [
                {"memberId": "mailto:liza@example.com",
                 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aliza@example.com"
                },
                {"attributeList": {"attribute": [
                     {"name": "phone",
                      "value": "tel:+19585550105"
                     },
                "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com/attributes"
                }},
                {"memberId": "mailto:serezha@example.com",
                 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members/mailto%3Aserezha@example.com"
                }
            ]}
        }
    }
]}
}
D.33 Retrieve an individual list shared by a user identified by otherUserId, with all list attributes and all member attributes (section 6.21.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"list": {
"attributeList": {"attribute": {
"name": "display-name",
"value": "Friends"
},"resourceURL":
"category": "URIList",
"memberCollection": {"member": {
"attributeList": {"attribute": {
"name": "display-name",
"value": "Alice"
},
"resourceURL":
"http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes",
"memberId": "sip:alice@example.com",
"resourceURL":
"http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes",
"memberId": "sip:alice@example.com",
"resourceURL":
"http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678/members"
"listId": "5678",
"resourceURL":
"http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/5678"
}}
}}
"resourceURL":
"http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists"
}
D.34 Retrieve all members for a shared list with all attributes (default) (section 6.22.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"memberCollection": {
  "member": [
    {
      "attributeList": {
        "attribute": {
          "name": "display-name",
          "value": "Alice"
        },
        "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com/attributes"
      },
      "memberId": "mailto:alice@example.com",
      "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/mailto%3Aalice@example.com"
    },
    {
      "attributeList": {
        "attribute": {
          "name": "display-name",
          "value": "Bob"
        },
      },
      "memberId": "tel:+19585550124",
    }
  ]
}
D.35 Retrieve selected member attributes about user identified as memberId from the list shared by a user identified by otherUserId (section 6.23.3.2)

Request:

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com?indivFilter=cellphone HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:53:23 GMT
Content-Type: application/json
Content-Length: nnnn

{
  "member": {
    "attributeList": {
      "attribute": {
        "name": "cellphone",
        "value": "tel:+19585550105"
      },
      "resourceURL":
      "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com/attributes"
    },
    "memberId": "sip:alice@example.com",
    "resourceURL":
    "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550105/shared/tel%3A%2B19585550100/lists/1234/members/sip%3Aalice@example.com"
  }
}
```

D.36 Retrieve all authorization rules (section 6.24.3.1)

Request:

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules HTTP/1.1
Accept: application/json
Host: example.com
```

© 2012 Open Mobile Alliance Ltd. All Rights Reserved.  Error! Reference source not found.
Response:

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"ruleList": {
 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules",
 "rule": [
  {
   "applToContacts": {"allContacts": null},
   "applicationTag": "myApp",
   "authorizedTo": {"listId": "1234"},
   "clientCorrelator": "123",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1",
   "ruleName": "allowAllContacts"
  },
  {
   "applToContacts": {"contactId": "bob"},
   "applicationTag": "myApp",
   "authorizedTo": {"allSharedIdentities": null},
   "clientCorrelator": "456",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2",
   "ruleName": "allowOwnContact"
  },
  {
   "applToLists": {"listId": "1234"},
   "applicationTag": "myApp",
   "authorizedTo": {"listId": "1234"},
   "clientCorrelator": "789",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule3",
   "ruleName": "allowFriendsList"
  }
 ]
}}
```

D.37 Create new rule for list and referenced contacts (section 6.24.5.1)

Request:

```
POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"rule": {
 "applToListsAndContacts": {"listId": "5678"},
 "applicationTag": "myApp",
 "authorizedTo": {"allContacts": null},
 "clientCorrelator": "123",
 "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule4",
 "ruleName": "allowAllContacts"
}}
```
"authorizedTo": {"listId": "5678"},
"clientCorrelator": "135",
"ruleName": "combinedRule"
}

Response:

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 21:33:52 GMT
Location: http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule3
Content-Type: application/json
Content-Length: nnnn

{"rule": {
   "appToListsAndContacts": {"listId": "5678"},
   "applicationTag": "myApp",
   "authorizedTo": {"listId": "5678"},
   "clientCorrelator": "135",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule3",
   "ruleName": "combinedRule"
}}

D.38 Retrieve an authorization rule (section 6.25.3.1)

Request:

GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1 HTTP/1.1
Accept: application/json
Host: example.com

Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"rule": {
   "appToContacts": {"allContacts": null},
   "applicationTag": "myApp",
   "authorizedTo": {"listId": "1234"},
   "clientCorrelator": "123",
   "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1",
   "ruleName": "allowAllContacts"
}}
D.39  Update an existing authorization rule (section 6.25.4.1)

Request:

```
PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1 HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"rule": {
  "applToContacts": {"allContacts": null},
  "applicationTag": "myApp",
  "authorizedTo": {"listId": [
    "1234",
    "5678"
  ]},
  "clientCorrelator": "123",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1",
  "ruleName": "allowAllContacts"
}}
```

Response:

```
HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"rule": {
  "applToContacts": {"allContacts": null},
  "applicationTag": "myApp",
  "authorizedTo": {"listId": [
    "1234",
    "5678"
  ]},
  "clientCorrelator": "123",
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1",
  "ruleName": "allowAllContacts"
}}
```

D.40  Retrieve data from an authorization rule (section 6.26.3.1)

Request:

```
GET /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule1/lists/1234 HTTP/1.1
Accept: application/json
Host: example.com
```
Response:

HTTP/1.1 200 OK
Date: Fri, 10 Dec 2010 12:55:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"listId": "1234"}

D.41 Authorize user by updating an existing authorization rule (section 6.26.4.1)

Request:

PUT /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"userId": "mailto:alice@example.com"}

Response:

HTTP/1.1 201 Created
Date: Fri, 10 Dec 2010 12:55:59 GMT
Location: /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com
Content-Type: application/json
Content-Length: nnnn

{"userId": "mailto:alice@example.com"}

D.42 Delete an authorized identity (section 6.26.6.1)

Request:

DELETE /exampleAPI/addressbook/v1/tel%3A%2B19585550100/authorizationRules/rule2/users/mailto%3Aalice@example.com
Accept: application/json
Host: example.com

Response:

HTTP/1.1 204 No Content
D.43 Transfer a member from one list to another (section 6.27.5.1)

Request:

POST /exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/allFriends/members/tel%3A%2B19585550200/transfer HTTP/1.1
Accept: application/json
Host: example.com
Content-Type: application/json
Content-Length: nnnn

{"memberTransferParameters": {
  "destination": "http://exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts"
}}

Response:

HTTP/1.1 303 See Other
Date: Fri, 10 Dec 2010 12:55:59 GMT
Location:
http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts/members/tel%3A%2B19585550199
Content-Type: application/json
Content-Length: nnnn

{"resourceReference": {
  "resourceURL": "http://example.com/exampleAPI/addressbook/v1/tel%3A%2B19585550100/lists/blockedContacts/members/tel%3A%2B19585550199"
}}
Appendix E.  Parlay X operations mapping  (Informative)

The table below illustrates the mapping between REST resources/methods and Parlay X [3GPP 29.199-13] equivalent operations.

<table>
<thead>
<tr>
<th>REST Resource</th>
<th>REST Method</th>
<th>REST Section reference</th>
<th>Parlay X equivalent operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists</td>
<td>GET</td>
<td>6.5.3</td>
<td>getOwnersGroups</td>
</tr>
<tr>
<td>Individual list</td>
<td>DELETE</td>
<td>6.6.6</td>
<td>deleteGroup</td>
</tr>
<tr>
<td>Attributes for a list</td>
<td>GET</td>
<td>6.7.3</td>
<td>queryGroupAttributes</td>
</tr>
<tr>
<td>Individual attribute for a list</td>
<td>DELETE</td>
<td>6.8.6</td>
<td>deleteGroupAttribute</td>
</tr>
<tr>
<td>Members in a list</td>
<td>GET</td>
<td>6.9.3</td>
<td>queryMembers</td>
</tr>
<tr>
<td>Individual member in a list</td>
<td>DELETE</td>
<td>6.10.6</td>
<td>deleteMember</td>
</tr>
<tr>
<td>Attributes for a member in a list</td>
<td>GET</td>
<td>6.11.3</td>
<td>queryGroupMemberAttributes</td>
</tr>
<tr>
<td>Individual attribute for a member in a list</td>
<td>DELETE</td>
<td>6.12.6</td>
<td>deleteGroupMemberAttribute</td>
</tr>
</tbody>
</table>

Table 1: Parlay X operations mapping
Appendix F. Light-weight resources (Informative)

The following table lists all Address Book data structure elements that can be accessed individually as light-weight resources. For each light-weight resource there are listed: 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>Attributes (5.2.2.6)</td>
<td>attribute</td>
<td>attribute</td>
<td>Attribute</td>
<td>{name}</td>
</tr>
<tr>
<td>List references (5.2.2.10)</td>
<td>link</td>
<td>link</td>
<td>common:Link</td>
<td>{href}</td>
</tr>
<tr>
<td>Authorization (5.2.2.19)</td>
<td>userId</td>
<td>userId</td>
<td>xsd:anyURI</td>
<td>users/{userId}</td>
</tr>
<tr>
<td></td>
<td>listId</td>
<td>listId</td>
<td>xsd:anyURI</td>
<td>lists/{listId}</td>
</tr>
<tr>
<td></td>
<td>link</td>
<td>link</td>
<td>common:Link</td>
<td>externalLists/{href}</td>
</tr>
<tr>
<td></td>
<td>domainName</td>
<td>domainName</td>
<td>xsd:string</td>
<td>domains/{domainName}</td>
</tr>
</tbody>
</table>

Table 2: Light-weight resources for Address Book

Note: When appending [ResourceRelPath] string to its heavy-weight resource URL, all variables within curly brackets "{}" such as: serviceId, version, deviceId, watcherUserId, contactListId, and domainName have to be replaced by their real values.
Appendix G. Authorization aspects (Normative)

This appendix specifies how to use the RESTful Address Book API in combination with some authorization frameworks.

G.1 Use with OMA Authorization Framework for Network APIs

The RESTful Address Book API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Address Book 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 Address Book 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_addressbook.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 “apiVersion” URL variable which is defined in section 5.1. 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_addressbook.contacts</td>
<td>Provide access to all defined operations on contacts.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_addressbook.lists</td>
<td>Provide access to all defined operations on lists.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_addressbook.subscr</td>
<td>Provide access to all defined operations on subscriptions.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_addressbook.shared</td>
<td>Provide access to all defined operations on shared contacts/lists.</td>
<td>No</td>
</tr>
<tr>
<td>oma_rest_addressbook.rules</td>
<td>Provide access to all defined operations on authorization rules for shared contacts/lists.</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3: Scope values for RESTful Address Book API

G.1.1.2 Downscoping

In the case where the client requests authorization for “oma_rest_addressbook.all_{apiVersion}” scope, the authorization server and/or resource owner MAY restrict the granted scope to some of the following scope values:
- “oma_rest_addressbook.contact”
- “oma_rest_addressbook.list”
- “oma_rest_addressbook.subscr”
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 Address Book API map to the REST resources and methods of this API. In these tables, the root “oma_rest_addressbook.” of scope values is omitted for readability reasons.
<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Section reference</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of contacts</td>
<td>/{userId}/contacts</td>
<td>6.1</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE n/a</td>
</tr>
<tr>
<td>Individual contact</td>
<td>/{userId}/contacts/{contactId}</td>
<td>6.2</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE all_{apiVersion} or contact</td>
</tr>
<tr>
<td>Attributes for a contact</td>
<td>/{userId}/contacts/{contactId}/attributes</td>
<td>6.3</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE n/a</td>
</tr>
<tr>
<td>Individual attribute for a contact</td>
<td>/{userId}/contacts/{contactId}/attributes/ResourceRelPath</td>
<td>6.4</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE all_{apiVersion} or contact</td>
</tr>
<tr>
<td>Collection of shared contacts</td>
<td>/{userId}/shared/{otherUserId}/contacts</td>
<td>6.18</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE n/a</td>
</tr>
<tr>
<td>Individual shared contact</td>
<td>/{userId}/shared/{otherUserId}/contacts/{sharedIdentity}</td>
<td>6.19</td>
<td>GET all_{apiVersion} or contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE n/a</td>
</tr>
</tbody>
</table>

Table 4: Required scope values for: managing contacts

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Section referenc e</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists</td>
<td>/{userId}/lists</td>
<td>6.5</td>
<td>GET all_{apiVersion} or list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE n/a</td>
</tr>
<tr>
<td>Individual list</td>
<td>/{userId}/lists/{listId}</td>
<td>6.6</td>
<td>GET all_{apiVersion} or list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PUT all_{apiVersion} or list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POST n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DELETE all_{apiVersion} or list</td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</td>
<td>Section reference</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------</td>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Attributes for a list</td>
<td>/{userId}/lists/{listId}/attributes</td>
<td>6.7</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Individual attribute for a list</td>
<td>/{userId}/lists/{listId}/attributes/[ResourceRelPath]</td>
<td>6.8</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Members in a list</td>
<td>/{userId}/lists/{listId}/members</td>
<td>6.9</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Individual member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}</td>
<td>6.10</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Attributes for a member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}/attributes</td>
<td>6.11</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Individual attribute for a member in a list</td>
<td>/{userId}/lists/{listId}/members/{memberId}/attributes/[ResourceRelPath]</td>
<td>6.12</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Individual member transfer</td>
<td>/{userId}/lists/{listId}/members/{memberId}/transfer</td>
<td>6.27</td>
<td>n/a</td>
</tr>
<tr>
<td>List references</td>
<td>/{userId}/lists/{listId}/ listReferences</td>
<td>6.13</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Individual list reference</td>
<td>/{userId}/lists/{listId}/ listReferences/[ResourceRelPath]</td>
<td>6.14</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Collection of shared lists</td>
<td>/{userId}/shared/{otherUserId}/lists</td>
<td>6.20</td>
<td>all_{apiVersion} or list PUT POST DELETE</td>
</tr>
<tr>
<td>Resource</td>
<td>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</td>
<td>Section reference</td>
<td>HTTP verbs</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Individual shared list</td>
<td>/{userId}/shared/{otherUserId}/lists/{sharedListIdentity}</td>
<td>6.21</td>
<td>all_{apiVersion} or list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET: n/a, PUT: n/a, POST: n/a, DELETE: n/a</td>
</tr>
<tr>
<td>Individual member information from shared list</td>
<td>/{userId}/shared/lists/{otherUserId}/{sharedListIdentity}/{memberId}</td>
<td>6.23</td>
<td>all_{apiVersion} or list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET: n/a, PUT: n/a, POST: n/a, DELETE: n/a</td>
</tr>
</tbody>
</table>

Table 5: Required scope values for: managing lists and members in a list

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Section reference</th>
<th>HTTP verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Book changes subscriptions</td>
<td>/{userId}/subscriptions/abChanges</td>
<td>6.15</td>
<td>all_{apiVersion} or subscr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET: n/a, PUT: n/a, POST: all_{apiVersion} or subscr, DELETE: n/a</td>
</tr>
<tr>
<td>Individual address book changes subscription</td>
<td>/{userId}/subscriptions/abChanges/{subscriptionId}</td>
<td>6.16</td>
<td>all_{apiVersion} or subscr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET: n/a, PUT: all_{apiVersion} or subscr, POST: n/a, DELETE: all_{apiVersion} or subscr</td>
</tr>
</tbody>
</table>

Table 5: Required scope values for: managing subscriptions to notifications about contacts and/or lists changes
<table>
<thead>
<tr>
<th>Resource</th>
<th>URL Base URL: http://{serverRoot}/addressbook/{apiVersion}</th>
<th>Section reference</th>
<th>HTTP verbs</th>
<th>GET</th>
<th>PUT</th>
<th>POST</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorisation rules</td>
<td>/{userId}/authorizationRules</td>
<td>6.24</td>
<td>all_{apiVersion} or contact or list</td>
<td>n/a</td>
<td>all_{apiVersion} or contact or list</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>Individual authorisation rule</td>
<td>/{userId}/authorizationRules/{ruleId}</td>
<td>6.25</td>
<td>all_{apiVersion} or contact or list</td>
<td>all_{apiVersion} or contact or list</td>
<td>all_{apiVersion} or contact or list</td>
<td>all_{apiVersion} or contact or list</td>
<td></td>
</tr>
<tr>
<td>Individual authorisation rule data</td>
<td>/{userId}/authorizationRules/{ruleId}/[ResourceRelPath]</td>
<td>6.26</td>
<td>all_{apiVersion} or contact or list</td>
<td>all_{apiVersion} or contact or list</td>
<td>all_{apiVersion} or contact or list</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Required scope values for: managing authorization rules for shared contacts/lists/members
G.1.2 Use of ‘acr:Authorization’
This section specifies the use of ‘acr:Authorization’ in place of an end user identifier in a resource URL path.
An ‘acr’ URI of the form ‘acr:Authorization’, where ‘Authorization’ 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:Authorization’ in a resource URL in place of the {endUserId} resource URL variable in the resource URL path, when the RESTful AddressBook API is used in combination with [Autho4API_10].
In the case the RESTful Payment supports [Autho4API_10], the server:
- SHALL accept ‘acr:Authorization’ as a valid value for the resource URL variable {endUserId}.
- SHALL conform to [REST_Common_TS] section 5.8.1.1 regarding the processing of ‘acr:Authorization’.
Appendix H. Overview of data types

The section provides an overview of the relation between the data types used for contacts, members and lists.

Figure 6 Data types for contacts, members and lists