ETSI GS NFV-SOL 003 V2.5.1 (2018-09)

Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point

Reference

RGS/NFV-SOL003ed251

Keywords

API, NFV, protocol

Contents

Intellectual Property Rights 18
Foreword 18
Modal verbs terminology 18
1 Scope 19
2 References 19
2.1 Normative references 19
2.2 Informative references 20
3 Definitions and abbreviations 21
3.1 Definitions 21
3.2 Abbreviations 21
4 General aspects 22
4.1 Overview 22
4.2 URI structure and supported content formats 23
4.3 Common procedures 24
4.3.1 Introduction 24
4.3.2 Attribute-based filtering 24
4.3.2.1 Overview and example (informative) 24
4.3.2.2 Specification 25
4.3.3 Attribute selectors 26
4.3.3.1 Overview and example (informative) 26
4.3.3.2 Specification 27
4.3.3.2.1 GET request 27
4.3.3.2.2 GET response 27
4.3.4 Usage of HTTP header fields 28
4.3.4.1 Introduction 28
4.3.4.2 Request header fields 28
4.3.4.3 Response header fields 29
4.3.5 Error reporting 30
4.3.5.1 Introduction 30
4.3.5.2 General mechanism 30
4.3.5.3 Type: ProblemDetails 31
4.3.5.4 Common error situations 31
4.3.5.5 Overview of HTTP error status codes 33
4.4 Common data types 35
4.4.1 Structured data types 35
4.4.1.1 Introduction 35
4.4.1.2 Type: Object 35
4.4.1.3 Type: Link 35
4.4.1.3a Type: NotificationLink 35
4.4.1.4 Type: KeyValuePairs 35
4.4.1.5 Type: VnfInstanceSubscriptionFilter 35
4.4.1.6 Type: VimConnectionInfo 36
4.4.1.7 Type: ResourceHandle 37
4.4.1.8 void 38
4.4.1.9 void 38
4.4.1.10 Type: VnfExtCpData 38
4.4.1.10a Type: VnfExtCpConfig 38
4.4.1.10b Type: CpProtocolData 39
4.4.1.10c Type: IpOverEthernetAddressData 39
4.4.1.11 Type: ExtVirtualLinkData 40
4.4.1.12 Type: ExtManagedVirtualLinkData 41
4.4.1.13 Type: ApiVersionInformation 41
4.4.2 Simple data types and enumerations 42
4.4.2.1 Introduction 42

4.4.2.2 Simple data types ... 42
4.4.2.3 Void ... 43
4.5 Authorization of API requests and notifications ... 43
4.5.1 Introduction ... 43
4.5.2 Flows (informative) ... 43
4.5.2.0 General ... 43
4.5.2.1 Authorization of API requests using OAuth 2.0 access tokens ... 43
4.5.2.1a Authorization of API requests using TLS certificates ... 45
4.5.2.2 Authorization of notifications using the HTTP Basic authentication scheme ... 46
4.5.2.3 Authorization of notifications using OAuth 2.0 access tokens ... 47
4.5.2.4 Authorization of notifications using TLS certificates ... 49
4.5.3 Specification ... 51
4.5.3.1 Introduction ... 51
4.5.3.2 General mechanism ... 51
4.5.3.3 Authorizing API requests ... 51
4.5.3.4 Authorizing the sending of notifications ... 52
4.5.3.5 Client roles ... 53
4.5.3.6 Negotiation of the authorization method ... 54
4.5.3.6.1 Authorization of API requests ... 54
4.5.3.6.2 Authorization of notification requests ... 56
4.6 Version management ... 57
4.6.1 Version identifiers and parameters ... 57
4.6.1.1 Version identifiers ... 57
4.6.1.2 Version parameters ... 57
4.6.2 Rules for incrementing version identifier fields ... 57
4.6.2.1 General ... 57
4.6.2.2 Examples of backward and non-backward compatible changes ... 58
4.6.3 Version information retrieval ... 59
4.6.3.1 General ... 59
4.6.3.2 Resource structure and methods ... 59
4.6.3.3 Resource: API versions ... 60
4.6.3.3.1 Description ... 60
4.6.3.3.2 Resource definition ... 60
4.6.3.3.3 Resource methods ... 60
4.6.4 Version signaling ... 61
4.7 Handling of large query results ... 62
4.7.1 Description ... 62
4.7.2 Specification ... 62
4.7.2.1 Alternatives ... 62
4.7.2.2 Error response ... 62
4.7.2.3 Paged response ... 63

5 VNF Lifecycle Management interface ... 63
5.1 Description ... 63
5.1a API version ... 64
5.2 Resource structure and methods ... 64
5.3 Sequence diagrams (informative) ... 67
5.3.1 Flow of the creation of a VNF instance resource ... 67
5.3.2 Flow of the deletion of a VNF instance resource ... 68
5.3.3 Flow of VNF lifecycle management operations triggered by task resources ... 68
5.3.4 Flow of automatic invocation of VNF scaling and VNF healing ... 71
5.3.5 Flow of the Query VNF operation ... 73
5.3.6 Flow of the Modify VNF Information operation ... 74
5.3.7 Flow of the Get Operation Status operation ... 76
5.3.8 Flow of managing subscriptions ... 77
5.3.9 Flow of sending notifications ... 78
5.3.10 Flow of retrying a VNF lifecycle management operation ... 79
5.3.11 Flow of rolling back a VNF lifecycle management operation ... 80
5.3.12 Flow of failing a VNF lifecycle management operation ... 81
5.3.13 Flow of cancelling a VNF lifecycle management operation ... 82
5.4 Resources ... 84
5.4.1 Introduction ... 84

5.4.1a Resource: API versions 84
5.4.2 Resource: VNF instances 85
5.4.2.1 Description 85
5.4.2.2 Resource definition 85
5.4.2.3 Resource methods 85
5.4.2.3.1 POST 85
5.4.2.3.2 GET 86
5.4.2.3.3 PUT 87
5.4.2.3.4 PATCH 87
5.4.2.3.5 DELETE 87
5.4.3 Resource: Individual VNF instance 87
5.4.3.1 Description 87
5.4.3.2 Resource definition 88
5.4.3.3 Resource methods 88
5.4.3.3.1 POST 88
5.4.3.3.2 GET 88
5.4.3.3.3 PUT 88
5.4.3.3.4 PATCH 89
5.4.3.3.5 DELETE 90
5.4.4 Resource: Instantiate VNF task 90
5.4.4.1 Description 90
5.4.4.2 Resource definition 90
5.4.4.3 Resource methods 91
5.4.4.3.1 POST 91
5.4.4.3.2 GET 91
5.4.4.3.3 PUT 91
5.4.4.3.4 PATCH 91
5.4.4.3.5 DELETE 92
5.4.5 Resource: Scale VNF task 92
5.4.5.1 Description 92
5.4.5.2 Resource definition 92
5.4.5.3 Resource methods 92
5.4.5.3.1 POST 92
5.4.5.3.2 GET 93
5.4.5.3.3 PUT 93
5.4.5.3.4 PATCH 93
5.4.5.3.5 DELETE 94
5.4.6 Resource: Scale VNF to Level task 94
5.4.6.1 Description 94
5.4.6.2 Resource definition 94
5.4.6.3 Resource methods 94
5.4.6.3.1 POST 94
5.4.6.3.2 GET 95
5.4.6.3.3 PUT 95
5.4.6.3.4 PATCH 95
5.4.6.3.5 DELETE 96
5.4.7 Resource: Change VNF Flavour task 96
5.4.7.1 Description 96
5.4.7.2 Resource definition 96
5.4.7.3 Resource methods 96
5.4.7.3.1 POST 96
5.4.7.3.2 GET 97
5.4.7.3.3 PUT 97
5.4.7.3.4 PATCH 98
5.4.7.3.5 DELETE 98
5.4.8 Resource: Terminate VNF task 98
5.4.8.1 Description 98
5.4.8.2 Resource definition 98
5.4.8.3 Resource methods 98
5.4.8.3.1 POST 98
5.4.8.3.2 GET 99
5.4.8.3.3 PUT 99

5.4.8.3.4 PATCH ... 99
5.4.8.3.5 DELETE ... 99
5.4.9 Resource: Heal VNF task ... 99
5.4.9.1 Description ... 99
5.4.9.2 Resource definition ... 100
5.4.9.3 Resource methods ... 100
5.4.9.3.1 POST ... 100
5.4.9.3.2 GET ... 101
5.4.9.3.3 PUT ... 101
5.4.9.3.4 PATCH ... 101
5.4.9.3.5 DELETE ... 102
5.4.10 Resource: Operate VNF task ... 102
5.4.10.1 Description ... 102
5.4.10.2 Resource definition ... 102
5.4.10.3 Resource methods ... 102
5.4.10.3.1 POST ... 102
5.4.10.3.2 GET ... 103
5.4.10.3.3 PUT ... 104
5.4.10.3.4 PATCH ... 104
5.4.10.3.5 DELETE ... 104
5.4.11 Resource: Change external VNF connectivity task ... 104
5.4.11.1 Description ... 104
5.4.11.2 Resource definition ... 104
5.4.11.3 Resource methods ... 104
5.4.11.3.1 POST ... 104
5.4.11.3.2 GET ... 105
5.4.11.3.3 PUT ... 105
5.4.11.3.4 PATCH ... 105
5.4.11.3.5 DELETE ... 106
5.4.12 Resource: VNF LCM operation occurrences ... 106
5.4.12.1 Description ... 106
5.4.12.2 Resource definition ... 106
5.4.12.3 Resource methods ... 106
5.4.12.3.1 POST ... 106
5.4.12.3.2 GET ... 106
5.4.12.3.3 PUT ... 108
5.4.12.3.4 PATCH ... 108
5.4.12.3.5 DELETE ... 108
5.4.13 Resource: Individual VNF LCM operation occurrence ... 108
5.4.13.1 Description ... 108
5.4.13.2 Resource definition ... 109
5.4.13.3 Resource methods ... 109
5.4.13.3.1 POST ... 109
5.4.13.3.2 GET ... 109
5.4.13.3.3 PUT ... 109
5.4.13.3.4 PATCH ... 110
5.4.13.3.5 DELETE ... 110
5.4.14 Resource: Retry operation task ... 110
5.4.14.1 Description ... 110
5.4.14.2 Resource definition ... 110
5.4.14.3 Resource methods ... 110
5.4.14.3.1 POST ... 110
5.4.14.3.2 GET ... 111
5.4.14.3.3 PUT ... 111
5.4.14.3.4 PATCH ... 111
5.4.14.3.5 DELETE ... 112
5.4.15 Resource: Rollback operation task ... 112
5.4.15.1 Description ... 112
5.4.15.2 Resource definition ... 112
5.4.15.3 Resource methods ... 112
5.4.15.3.1 POST ... 112
5.4.15.3.2 GET ... 113

5.4.15.3.3 PUT ... 113
5.4.15.3.4 PATCH ... 113
5.4.15.3.5 DELETE ... 113
5.4.16 Resource: Fail operation task ... 114
5.4.16.1 Description ... 114
5.4.16.2 Resource definition ... 114
5.4.16.3 Resource methods ... 114
5.4.16.3.1 POST ... 114
5.4.16.3.2 GET ... 115
5.4.16.3.3 PUT ... 115
5.4.16.3.4 PATCH ... 115
5.4.16.3.5 DELETE ... 116
5.4.17 Resource: Cancel operation task ... 116
5.4.17.1 Description ... 116
5.4.17.2 Resource definition ... 116
5.4.17.3 Resource methods ... 116
5.4.17.3.1 POST ... 116
5.4.17.3.2 GET ... 117
5.4.17.3.3 PUT ... 117
5.4.17.3.4 PATCH ... 117
5.4.17.3.5 DELETE ... 118
5.4.18 Resource: Subscriptions ... 118
5.4.18.1 Description ... 118
5.4.18.2 Resource definition ... 118
5.4.18.3 Resource methods ... 118
5.4.18.3.1 POST ... 118
5.4.18.3.2 GET ... 119
5.4.18.3.3 PUT ... 120
5.4.18.3.4 PATCH ... 120
5.4.18.3.5 DELETE ... 120
5.4.19 Resource: Individual subscription ... 120
5.4.19.1 Description ... 120
5.4.19.2 Resource definition ... 120
5.4.19.3 Resource methods ... 121
5.4.19.3.1 POST ... 121
5.4.19.3.2 GET ... 121
5.4.19.3.3 PUT ... 121
5.4.19.3.4 PATCH ... 121
5.4.19.3.5 DELETE ... 121
5.4.20 Resource: Notification endpoint ... 122
5.4.20.1 Description ... 122
5.4.20.2 Resource definition ... 122
5.4.20.3 Resource methods ... 122
5.4.20.3.1 POST ... 122
5.4.20.3.2 GET ... 123
5.4.20.3.3 PUT ... 123
5.4.20.3.4 PATCH ... 123
5.4.20.3.5 DELETE ... 124
5.5 Data model ... 124
5.5.1 Introduction ... 124
5.5.2 Resource and notification data types ... 124
5.5.2.1 Introduction ... 124
5.5.2.2 Type: VnfInstance ... 124
5.5.2.3 Type: CreateVnfRequest ... 127
5.5.2.4 Type: InstantiateVnfRequest ... 128
5.5.2.5 Type: ScaleVnfRequest ... 128
5.5.2.6 Type: ScaleVnfToLevelRequest ... 129
5.5.2.7 Type: ChangeVnfFlavourRequest ... 129
5.5.2.8 Type: TerminateVnfRequest ... 130
5.5.2.9 Type: HealVnfRequest ... 130
5.5.2.10 Type: OperateVnfRequest ... 131
5.5.2.11 Type: ChangeExtVnfConnectivityRequest ... 131

5.5.2.12 Type: VnfInfoModificationRequest ... 132
5.5.2.12a Type: VnfInfoModifications ... 133
5.5.2.13 Type: VnfLcmOpOcc ... 133
5.5.2.14 Type: CancelMode ... 135
5.5.2.15 Type: LccnSubscriptionRequest ... 135
5.5.2.16 Type: LccnSubscription ... 136
5.5.2.17 Type: VnfLcmOperationOccurrenceNotification ... 136
5.5.2.18 Type: VnfIdentifierCreationNotification ... 137
5.5.2.19 Type: VnfIdentifierDeletionNotification ... 138
5.5.3 Referenced structured data types ... 138
5.5.3.1 Introduction ... 138
5.5.3.2 Type: ExtVirtualLinkInfo ... 138
5.5.3.3 Type: ExtManagedVirtualLinkInfo ... 139
5.5.3.4 Type: ScaleInfo ... 139
5.5.3.5 Type: VnfcResourceInfo ... 139
5.5.3.6 Type: VnfVirtualLinkResourceInfo ... 140
5.5.3.7 Type: VirtualStorageResourceInfo ... 140
5.5.3.8 Type: VnfLinkPortInfo ... 140
5.5.3.9 Type: ExtLinkPortInfo ... 141
5.5.3.9a Type: ExtLinkPortData ... 141
5.5.3.9b Type: CpProtocolInfo ... 142
5.5.3.10 Type: IpOverEthernetAddressInfo ... 142
5.5.3.11 Type: MonitoringParameter ... 143
5.5.3.12 Type: LifecycleChangeNotificationsFilter ... 143
5.5.3.13 Type: AffectedVnfc ... 144
5.5.3.14 Type: AffectedVirtualLink ... 145
5.5.3.15 Type: AffectedVirtualStorage ... 145
5.5.3.16 Type: LccnLinks ... 146
5.5.3.17 Type: VnfExtCpInfo ... 146
5.5.4 Referenced simple data types and enumerations ... 147
5.5.4.1 Introduction ... 147
5.5.4.2 Simple data types ... 147
5.5.4.3 Enumeration: VnfOperationalStateType ... 147
5.5.4.4 Enumeration: StopType ... 147
5.5.4.5 Enumeration: LcmOperationStateType ... 148
5.5.4.6 Enumeration: CancelModeType ... 148
5.5.4.7 Enumeration: LcmOperationType ... 148
5.6 Handling of errors during VNF lifecycle management operations ... 149
5.6.1 Basic concepts (informative) ... 149
5.6.1.1 Motivation ... 149
5.6.1.2 Failure resolution strategies: Retry and Rollback ... 149
5.6.1.3 Error handling at VNFM and NFVO ... 150
5.6.2 States and state transitions of a VNF lifecycle management operation occurrence ... 151
5.6.2.1 General ... 151
5.6.2.2 States of a VNF lifecycle management operation occurrence ... 151
5.6.2.3 Error handling operations that change the state of a VNF lifecycle management operation occurrence ... 154
5.6.3 Detailed flows ... 155
5.6.3.1 Immediate failure ... 155
5.6.3.2 Failure in "STARTING" state ... 155
5.6.3.3 Failure during actual LCM operation execution ... 156
5.6.3.4 LCM operation cancellation ... 158

6 VNF Performance Management interface ... 158
6.1 Description ... 158
6.1a API version ... 158
6.2 Resource structure and methods ... 158
6.3 Sequence diagrams (informative) ... 160
6.3.1 Flow of creating a PM job ... 160
6.3.2 Flow of querying/reading PM jobs ... 160
6.3.3 Flow of deleting a PM job ... 161
6.3.4 Flow of obtaining performance reports ... 162

6.3.5 Flow of creating a threshold ... 163
6.3.6 Flow of querying/reading thresholds ... 163
6.3.7 Flow of deleting thresholds ... 164
6.3.8 Flow of managing subscriptions ... 165
6.3.9 Flow of sending notifications ... 166
6.4 Resources ... 167
6.4.1 Introduction ... 167
6.4.1a Resource: API versions ... 167
6.4.2 Resource: PM jobs ... 167
6.4.2.1 Description ... 167
6.4.2.2 Resource definition ... 167
6.4.2.3 Resource methods ... 167
6.4.2.3.1 POST ... 167
6.4.2.3.2 GET ... 168
6.4.2.3.3 PUT ... 169
6.4.2.3.4 PATCH ... 169
6.4.2.3.5 DELETE ... 169
6.4.3 Resource: Individual PM job ... 169
6.4.3.1 Description ... 169
6.4.3.2 Resource definition ... 170
6.4.3.3 Resource methods ... 170
6.4.3.3.1 POST ... 170
6.4.3.3.2 GET ... 170
6.4.3.3.3 PUT ... 170
6.4.3.3.4 PATCH ... 171
6.4.3.3.5 DELETE ... 171
6.4.4 Resource: Individual performance report ... 171
6.4.4.1 Description ... 171
6.4.4.2 Resource definition ... 171
6.4.4.3 Resource methods ... 172
6.4.4.3.1 POST ... 172
6.4.4.3.2 GET ... 172
6.4.4.3.3 PUT ... 172
6.4.4.3.4 PATCH ... 172
6.4.4.3.5 DELETE ... 172
6.4.5 Resource: Thresholds ... 172
6.4.5.1 Description ... 172
6.4.5.2 Resource definition ... 173
6.4.5.3 Resource methods ... 173
6.4.5.3.1 POST ... 173
6.4.5.3.2 GET ... 173
6.4.5.3.3 PUT ... 174
6.4.5.3.4 PATCH ... 174
6.4.5.3.5 DELETE ... 175
6.4.6 Resource: Individual threshold ... 175
6.4.6.1 Description ... 175
6.4.6.2 Resource definition ... 175
6.4.6.3 Resource methods ... 175
6.4.6.3.1 POST ... 175
6.4.6.3.2 GET ... 175
6.4.6.3.3 PUT ... 176
6.4.6.3.4 PATCH ... 176
6.4.6.3.5 DELETE ... 176
6.4.7 Resource: Subscriptions ... 176
6.4.7.1 Description ... 176
6.4.7.2 Resource definition ... 177
6.4.7.3 Resource methods ... 177
6.4.7.3.1 POST ... 177
6.4.7.3.2 GET ... 178
6.4.7.3.3 PUT ... 179
6.4.7.3.4 PATCH ... 179
6.4.7.3.5 DELETE ... 179

6.4.8 Resource: Individual subscription 179
6.4.8.1 Description 179
6.4.8.2 Resource definition 179
6.4.8.3 Resource methods 179
6.4.8.3.1 POST 179
6.4.8.3.2 GET 179
6.4.8.3.3 PUT 180
6.4.8.3.4 PATCH 180
6.4.8.3.5 DELETE 180
6.4.9 Resource: Notification endpoint 181
6.4.9.1 Description 181
6.4.9.2 Resource definition 181
6.4.9.3 Resource methods 181
6.4.9.3.1 POST 181
6.4.9.3.2 GET 181
6.4.9.3.3 PUT 182
6.4.9.3.4 PATCH 182
6.4.9.3.5 DELETE 182
6.5 Data Model 182
6.5.1 Introduction 182
6.5.2 Resource and notification data types 182
6.5.2.1 Introduction 182
6.5.2.2 Type: PmSubscriptionRequest 182
6.5.2.3 Type: PmSubscription 183
6.5.2.4 Type: ThresholdCrossedNotification 183
6.5.2.5 Type: PerformanceInformationAvailableNotification 184
6.5.2.6 Type: CreatePmJobRequest 185
6.5.2.7 Type: PmJob 185
6.5.2.8 Type: CreateThresholdRequest 185
6.5.2.9 Type: Threshold 185
6.5.2.10 Type: PerformanceReport 186
6.5.3 Referenced structured data types 186
6.5.3.1 Introduction 186
6.5.3.2 Type: PmNotificationsFilter 186
6.5.3.3 Type: PmJobCriteria 187
6.5.3.4 Type: ThresholdCriteria 188
6.5.4 Referenced simple data types and enumerations 188
6.5.4.1 Introduction 188
6.5.4.2 Simple data types 188
6.5.4.3 Enumeration: CrossingDirectionType 188
7 VNF Fault Management interface 189
7.1 Description 189
7.1a API version 189
7.2 Resource structure and methods 189
7.3 Sequence diagrams (informative) 190
7.3.1 Flow of the Get Alarm List operation 190
7.3.2 Flow of acknowledging alarm 191
7.3.3 Flow of managing subscriptions 191
7.3.4 Flow of sending notifications 193
7.4 Resources 194
7.4.1 Introduction 194
7.4.2 Resource: Alarms 194
7.4.2.1 Description 194
7.4.2.2 Resource definition 194
7.4.2.3 Resource methods 194
7.4.2.3.1 POST 194
7.4.2.3.2 GET 194
7.4.2.3.3 PUT 195
7.4.2.3.4 PATCH 195
7.4.2.3.5 DELETE 195
7.4.3 Resource: Individual alarm 195

7.4.3.1 Description ... 195
7.4.3.2 Resource definition ... 195
7.4.3.3 Resource methods ... 196
7.4.3.3.1 POST ... 196
7.4.3.3.2 GET ... 196
7.4.3.3.3 PUT ... 196
7.4.3.3.4 PATCH ... 196
7.4.3.3.5 DELETE ... 197
7.4.4 Resource: Subscriptions ... 197
7.4.4.1 Description ... 197
7.4.4.2 Resource definition ... 198
7.4.4.3 Resource methods ... 198
7.4.4.3.1 POST ... 198
7.4.4.3.2 GET ... 199
7.4.4.3.3 PUT ... 200
7.4.4.3.4 PATCH ... 200
7.4.4.3.5 DELETE ... 200
7.4.5 Resource: Individual subscription ... 200
7.4.5.1 Description ... 200
7.4.5.2 Resource definition ... 200
7.4.5.3 Resource methods ... 200
7.4.5.3.1 POST ... 200
7.4.5.3.2 GET ... 200
7.4.5.3.3 PUT ... 201
7.4.5.3.4 PATCH ... 201
7.4.5.3.5 DELETE ... 201
7.4.6 Resource: Notification endpoint ... 201
7.4.6.1 Description ... 201
7.4.6.2 Resource definition ... 202
7.4.6.3 Resource methods ... 202
7.4.6.3.1 POST ... 202
7.4.6.3.2 GET ... 202
7.4.6.3.3 PUT ... 203
7.4.6.3.4 PATCH ... 203
7.4.6.3.5 DELETE ... 203
7.5 Data Model ... 203
7.5.1 Introduction ... 203
7.5.2 Resource and notification data types ... 203
7.5.2.1 Introduction ... 203
7.5.2.2 Type: FmSubscriptionRequest ... 203
7.5.2.3 Type: FmSubscription ... 204
7.5.2.4 Type: Alarm ... 204
7.5.2.5 Type: AlarmNotification ... 205
7.5.2.6 Type: AlarmClearedNotification ... 206
7.5.2.7 Type: AlarmListRebuiltNotification ... 206
7.5.2.8 Type: AlarmModifications ... 206
7.5.3 Referenced structured data types ... 207
7.5.3.1 Introduction ... 207
7.5.3.2 Type: FmNotificationsFilter ... 207
7.5.3.3 Type: FaultyResourceInfo ... 207
7.5.4 Referenced simple data types and enumerations ... 207
7.5.4.1 Introduction ... 207
7.5.4.2 Simple data types ... 208
7.5.4.3 Enumeration: PerceivedSeverityType ... 208
7.5.4.4 Enumeration: EventType ... 208
7.5.4.5 Enumeration: FaultyResourceType ... 208

8 VNF Indicator interface ... 209
8.1 Description ... 209
8.1a API version ... 209
8.2 Resource structure and methods ... 209
8.3 Sequence diagrams (informative) ... 210

8.3.1 Flow of querying VNF indicators ... 210
8.3.2 Flow of reading a VNF indicator ... 211
8.3.3 Flow of managing subscriptions ... 212
8.3.4 Flow of sending notifications ... 213
8.4 Resources ... 214
8.4.1 Introduction ... 214
8.4.1a Resource: API versions ... 214
8.4.2 Resource: VNF indicators ... 214
8.4.2.1 Description ... 214
8.4.2.2 Resource definition ... 214
8.4.2.3 Resource methods ... 214
8.4.2.3.1 POST ... 214
8.4.2.3.2 GET ... 214
8.4.2.3.3 PUT ... 215
8.4.2.3.4 PATCH ... 215
8.4.2.3.5 DELETE ... 215
8.4.3 Resource: VNF indicators related to a VNF instance ... 215
8.4.3.1 Description ... 215
8.4.3.2 Resource definition ... 215
8.4.3.3 Resource methods ... 216
8.4.3.3.1 POST ... 216
8.4.3.3.2 GET ... 216
8.4.3.3.3 PUT ... 217
8.4.3.3.4 PATCH ... 217
8.4.3.3.5 DELETE ... 217
8.4.4 Resource: Individual VNF indicator ... 217
8.4.4.1 Description ... 217
8.4.4.2 Resource definition ... 217
8.4.4.3 Resource methods ... 218
8.4.4.3.1 POST ... 218
8.4.4.3.2 GET ... 218
8.4.4.3.3 PUT ... 218
8.4.4.3.4 PATCH ... 218
8.4.4.3.5 DELETE ... 219
8.4.5 Resource: Subscriptions ... 219
8.4.5.1 Description ... 219
8.4.5.2 Resource definition ... 219
8.4.5.3 Resource methods ... 219
8.4.5.3.1 POST ... 219
8.4.5.3.2 GET ... 220
8.4.5.3.3 PUT ... 221
8.4.5.3.4 PATCH ... 221
8.4.5.3.5 DELETE ... 221
8.4.6 Resource: Individual subscription ... 221
8.4.6.1 Description ... 221
8.4.6.2 Resource definition ... 221
8.4.6.3 Resource methods ... 222
8.4.6.3.1 POST ... 222
8.4.6.3.2 GET ... 222
8.4.6.3.3 PUT ... 222
8.4.6.3.4 PATCH ... 222
8.4.6.3.5 DELETE ... 223
8.4.7 Resource: Notification endpoint ... 223
8.4.7.1 Description ... 223
8.4.7.2 Resource definition ... 223
8.4.7.3 Resource methods ... 223
8.4.7.3.1 POST ... 223
8.4.7.3.2 GET ... 224
8.4.7.3.3 PUT ... 224
8.4.7.3.4 PATCH ... 225
8.4.7.3.5 DELETE ... 225
8.5 Data model ... 225

8.5.1 Introduction ... 225
8.5.2 Resource and notification data types ... 225
8.5.2.1 Introduction ... 225
8.5.2.2 Type: VnfIndicator ... 225
8.5.2.3 Type: VnfIndicatorSubscriptionRequest ... 225
8.5.2.4 Type: VnfIndicatorSubscription ... 226
8.5.2.5 Type: VnfIndicatorValueChangeNotification ... 226
8.5.3 Referenced structured data types ... 227
8.5.3.1 Introduction ... 227
8.5.3.2 Type: VnfIndicatorNotificationsFilter ... 227
8.5.4 Referenced simple data types and enumerations ... 227

9 VNF Lifecycle Operation Granting interface ... 227
9.1 Description ... 227
9.1a API version ... 228
9.2 Resource structure and methods ... 228
9.3 Sequence diagrams (informative) ... 229
9.3.1 Flow of grant request with synchronous response ... 229
9.3.2 Flow of grant request with asynchronous response ... 230
9.4 Resources ... 230
9.4.1 Introduction ... 230
9.4.1a Resource: API versions ... 231
9.4.2 Resource: Grants ... 231
9.4.2.1 Description ... 231
9.4.2.2 Resource definition ... 231
9.4.2.3 Resource methods ... 231
9.4.2.3.1 POST ... 231
9.4.2.3.2 GET ... 232
9.4.2.3.3 PUT ... 232
9.4.2.3.4 PATCH ... 232
9.4.2.3.5 DELETE ... 232
9.4.3 Resource: Individual grant ... 232
9.4.3.1 Description ... 232
9.4.3.2 Resource definition ... 233
9.4.3.3 Resource methods ... 233
9.4.3.3.1 POST ... 233
9.4.3.3.2 GET ... 233
9.4.3.3.3 PUT ... 234
9.4.3.3.4 PATCH ... 234
9.4.3.3.5 DELETE ... 234
9.5 Data model ... 234
9.5.1 Introduction ... 234
9.5.2 Resource and notification data types ... 234
9.5.2.1 Introduction ... 234
9.5.2.2 Type: GrantRequest ... 235
9.5.2.3 Type: Grant ... 236
9.5.3 Referenced structured data types ... 238
9.5.3.1 Introduction ... 238
9.5.3.2 Type: ResourceDefinition ... 238
9.5.3.3 Type: GrantInfo ... 239
9.5.3.4 Type: ZoneInfo ... 240
9.5.3.5 Type: ZoneGroupInfo ... 241
9.5.3.6 Type: PlacementConstraint ... 241
9.5.3.7 Type: VimConstraint ... 241
9.5.3.8 Type: ConstraintResourceRef ... 242
9.5.3.9 Type: VimComputeResourceFlavour ... 243
9.5.3.10 Type: VimSoftwareImage ... 243
9.5.4 Referenced simple data types and enumerations ... 244
9.5.4.1 Introduction ... 244
9.5.4.2 Simple data types ... 244
9.5.4.3 Enumeration: GrantedLcmOperationType ... 244

10 VNF Package Management interface 244
10.1 Description 244
10.1a API version 245
10.2 Resource structure and methods 245
10.3 Sequence diagrams (informative) 246
10.3.1 Flow of querying/reading VNF package information 246
10.3.2 Flow of reading the VNFD of an on-boarded VNF package 247
10.3.3 Flow of fetching an on-boarded VNF package 247
10.3.4 Flow of fetching a VNF package artifact 248
10.3.5 Flow of managing subscriptions 249
10.3.6 Flow of sending notifications 251
10.4 Resources 251
10.4.1 Introduction 251
10.4.1a Resource: API versions 252
10.4.2 Resource: VNF packages 252
10.4.2.1 Description 252
10.4.2.2 Resource definition 252
10.4.2.3 Resource methods 252
10.4.2.3.1 POST 252
10.4.2.3.2 GET 252
10.4.2.3.3 PUT 253
10.4.2.3.4 PATCH 254
10.4.2.3.5 DELETE 254
10.4.3 Resource: Individual VNF package 254
10.4.3.1 Description 254
10.4.3.2 Resource definition 254
10.4.3.3 Resource methods 254
10.4.3.3.1 POST 254
10.4.3.3.2 GET 254
10.4.3.3.3 PUT 255
10.4.3.3.4 PATCH 255
10.4.3.3.5 DELETE 255
10.4.4 Resource: VNFD in an individual VNF package 255
10.4.4.1 Description 255
10.4.4.2 Resource definition 255
10.4.4.3 Resource methods 256
10.4.4.3.1 POST 256
10.4.4.3.2 GET 256
10.4.4.3.3 PUT 257
10.4.4.3.4 PATCH 257
10.4.4.3.5 DELETE 257
10.4.5 Resource: VNF package content 257
10.4.5.1 Description 257
10.4.5.2 Resource definition 257
10.4.5.3 Resource methods 257
10.4.5.3.1 POST 257
10.4.5.3.2 GET 258
10.4.5.3.3 PUT 259
10.4.5.3.4 PATCH 259
10.4.5.3.5 DELETE 259
10.4.6 Resource: Individual VNF package artifact 259
10.4.6.1 Description 259
10.4.6.2 Resource definition 259
10.4.6.3 Resource methods 259
10.4.6.3.1 POST 259
10.4.6.3.2 GET 260
10.4.6.3.3 PUT 261
10.4.6.3.4 PATCH 261
10.4.6.3.5 DELETE 261
10.4.7 Resource: Subscriptions 261
10.4.7.1 Description 261
10.4.7.2 Resource definition 261

10.4.7.3 Resource methods 261
10.4.7.3.1 POST 261
10.4.7.3.2 GET 262
10.4.7.3.3 PUT 263
10.4.7.3.4 PATCH 263
10.4.7.3.5 DELETE 263
10.4.8 Resource: Individual subscription 263
10.4.8.1 Description 263
10.4.8.2 Resource definition 263
10.4.8.3 Resource methods 264
10.4.8.3.1 POST 264
10.4.8.3.2 GET 264
10.4.8.3.3 PUT 264
10.4.8.3.4 PATCH 264
10.4.8.3.5 DELETE 264
10.4.9 Resource: Notification endpoint 265
10.4.9.1 Description 265
10.4.9.2 Resource definition 265
10.4.9.3 Resource methods 265
10.4.9.3.1 POST 265
10.4.9.3.2 GET 266
10.4.9.3.3 PUT 266
10.4.9.3.4 PATCH 266
10.4.9.3.5 DELETE 266
10.5 Data model 267
10.5.1 Introduction 267
10.5.2 Resource and notification data types 267
10.5.2.1 Introduction 267
10.5.2.2 Type: VnfPkgInfo 267
10.5.2.3 Type: PkgmSubscriptionRequest 268
10.5.2.4 Type: PkgmSubscription 268
10.5.2.5 Type: VnfPackageOnboardingNotification 269
10.5.2.6 Type: VnfPackageChangeNotification 269
10.5.3 Referenced structured data types 270
10.5.3.1 Introduction 270
10.5.3.2 Type: VnfPackageSoftwareImageInfo 270
10.5.3.3 Type: VnfPackageArtifactInfo 271
10.5.3.4 Type: PkgmNotificationsFilter 271
10.5.3.5 Type: PkgmLinks 272
10.5.3.6 Type: Checksum 273
10.5.4 Referenced simple data types and enumerations 273
10.5.4.1 Introduction 273
10.5.4.2 Simple data types 273
10.5.4.3 Enumeration: PackageOperationalStateType 273
10.5.4.4 Enumeration: PackageUsageStateType 273
10.5.4.5 Enumeration: PackageChangeType 273
10.5.4.6 Enumeration: PackageOnboardingStateType 274

11 Virtualised Resources Quota Available Notification interface 274
11.1 Description 274
11.1a API version 274
11.2 Resource structure and methods 274
11.3 Sequence diagrams (informative) 275
11.3.1 Flow of managing subscriptions 275
11.3.2 Flow of sending notifications 277
11.4 Resources 278
11.4.1 Introduction 278
11.4.1a Resource: API versions 278
11.4.2 Resource: Subscriptions 278
11.4.2.1 Description 278
11.4.2.2 Resource definition 278
11.4.2.3 Resource methods 278

11.4.2.3.1 POST ... 278
11.4.2.3.2 GET ... 279
11.4.2.3.3 PUT ... 280
11.4.2.3.4 PATCH ... 280
11.4.2.3.5 DELETE ... 280
11.4.3 Resource: Individual subscription ... 280
11.4.3.1 Description ... 280
11.4.3.2 Resource definition ... 281
11.4.3.3 Resource methods ... 281
11.4.3.3.1 POST ... 281
11.4.3.3.2 GET ... 281
11.4.3.3.3 PUT ... 281
11.4.3.3.4 PATCH ... 281
11.4.3.3.5 DELETE ... 282
11.4.4 Resource: Notification endpoint ... 282
11.4.4.1 Description ... 282
11.4.4.2 Resource definition ... 282
11.4.4.3 Resource methods ... 282
11.4.4.3.1 POST ... 282
11.4.4.3.2 GET ... 283
11.4.4.3.3 PUT ... 283
11.4.4.3.4 PATCH ... 283
11.4.4.3.5 DELETE ... 284
11.5 Data model ... 284
11.5.1 Introduction ... 284
11.5.2 Resource and notification data types ... 284
11.5.2.1 Introduction ... 284
11.5.2.2 Type: VrQuotaAvailSubscriptionRequest ... 284
11.5.2.3 Type: VrQuotaAvailSubscription ... 284
11.5.2.4 Type: VrQuotaAvailNotification ... 285
11.5.3 Referenced structured data types ... 285
11.5.3.1 Introduction ... 285
11.5.3.2 Type: VrQuotaAvailNotificationsFilter ... 285
11.5.3.3 Type: QuotaAvailLinks ... 286

Annex A (informative): Mapping operations to protocol elements ... 287

A.1 Overview ... 287

A.2 VNF Package Management interface ... 287

A.3 VNF Lifecycle Operation Granting interface ... 287

A.4 Virtualised Resources Management interfaces in indirect mode ... 287

A.5 Virtualised Resources Quota Available Notification interface ... 288

A.6 VNF Lifecycle Management interface ... 288

A.7 VNF Performance Management interface ... 289

A.8 VNF Fault Management interface ... 289

A.9 VNF Indicator interface ... 289

Annex B (informative): Explanations ... 290

B.1 Introduction ... 290

B.2 Scaling of a VNF instance ... 290

B.3 Examples of VNF connectivity patterns ... 292
B.3.1 Overview ... 292
B.3.2 Example of a VNF instance with two different types of external connections points ... 292
B.3.3 Example of changing VNF connectivity ... 293

Annex C (normative): VimConnectionInfo registry ... 294

C.1 Purpose ... 294
C.2 Registry content ... 294
C.3 Structure of the vimType identifier ... 295
C.4 Initial registration ... 295
C.4.1 Instructions for data structure definition ... 295
C.4.2 Template ... 295
C.5 Registration update ... 297
C.6 Initial registry content ... 297
C.6.1 Registration for ETSINFV.OPENSTACK_KEYSTONE.V_2 ... 297
C.6.2 Registration for ETSINFV.OPENSTACK_KEYSTONE.V_3 ... 299
Annex D (informative): Complementary material for API utilization ... 302
Annex E (informative): Authors & contributors ... 303
Annex F (informative): Change History ... 304
History ... 315

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies a set of RESTful protocols and data models fulfilling the requirements specified in ETSI GS NFV-IFA 007 [1] for the interfaces used over the Or-Vnfm reference point, except for the "Virtualised Resources Management interfaces in indirect mode" as defined in clause 6.4 of ETSI GS NFV-IFA 007 [1].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NFV-IFA 007: "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".

[2] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".

[3] IETF RFC 2818: "HTTP Over TLS".

NOTE: Available at https://tools.ietf.org/html/rfc2818.

[4] IETF RFC 3339: "Date and Time on the Internet: Timestamps".

NOTE: Available at https://tools.ietf.org/html/rfc3339.

[5] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

NOTE: Available at https://tools.ietf.org/html/rfc3986.

[6] IETF RFC 4918: "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)".

NOTE: Available at https://tools.ietf.org/html/rfc4918.

[7] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

NOTE: Available at https://tools.ietf.org/html/rfc5246.

[8] IETF RFC 5646: "Tags for Identifying Languages".

NOTE: Available at https://tools.ietf.org/html/rfc5646.

[9] IETF RFC 6585: "Hypertext Transfer Protocol (HTTP/1.1): Additional HTTP Status Codes".

NOTE: Available at https://tools.ietf.org/html/rfc6585.

[10] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

NOTE: Available from https://tools.ietf.org/html/rfc6749.

[11] IETF RFC 6750: "The OAuth 2.0 Authorization Framework: Bearer Token Usage".

NOTE: Available from https://tools.ietf.org/html/rfc6750.

[12] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at https://tools.ietf.org/html/rfc8259.

[13] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".

NOTE: Available at https://tools.ietf.org/html/rfc7231.

[14] IETF RFC 7232: "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

NOTE: Available at https://tools.ietf.org/html/rfc7232.

[15] IETF RFC 7233: "Hypertext Transfer Protocol (HTTP/1.1): Range Requests".

NOTE: Available at https://tools.ietf.org/html/rfc7233.

[16] IETF RFC 7235: "Hypertext Transfer Protocol (HTTP/1.1): Authentication".

NOTE: Available at https://tools.ietf.org/html/rfc7235.

[17] IETF RFC 7396: "JSON Merge Patch".

NOTE: Available at https://tools.ietf.org/html/rfc7396.

[18] IETF RFC 7617: "The 'Basic' HTTP Authentication Scheme".

NOTE: Available from https://tools.ietf.org/html/rfc7617.

[19] IETF RFC 7807: "Problem Details for HTTP APIs".

NOTE: Available at https://tools.ietf.org/html/rfc7807.

[20] ETSI GS NFV-IFA 027: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Performance Measurements Specification".

[21] IETF RFC 6901: "JavaScript Object Notation (JSON) Pointer".

NOTE: Available at https://tools.ietf.org/html/rfc6901.

[22] IETF RFC 8288: "Web Linking".

NOTE: Available at https://tools.ietf.org/html/rfc8288.

[23] Semantic Versioning 2.0.0.

NOTE: Available at https://semver.org/.

[24] IETF RFC 4229: "HTTP Header Field Registrations".

NOTE: Available at https://tools.ietf.org/html/rfc4229.

[25] Recommendation ITU-T X.733: "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] ETSI GS NFV-SOL 002: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point".

[i.3] ETSI TS 133 310: "Universal Mobile Telecommunications System (UMTS); LTE; Network Domain Security (NDS); Authentication Framework (AF)".

[i.4] Hypertext Transfer Protocol (HTTP) Status Code Registry at IANA.

NOTE: Available at http://www.iana.org/assignments/http-status-codes.

[i.5] ETSI NFV registry of VimConnectionInfo information.

NOTE: Available at http://register.etsi.org/NFV.

[i.6] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[i.7] OpenStack® documentation: "Disk and container formats for images".

NOTE: Available at http://docs.openstack.org/image-guide/image-formats.html.

[i.8] JSON Schema: "Core definitions and terminology", Version draft-07, November 19, 2017.

NOTE 1: JSON schema is documented at http://json-schema.org/.

NOTE 2: The specification is available as Internet Draft at https://tools.ietf.org/html/draft-handrews-json-schema-01.

[i.9] OpenAPI Specification.

NOTE: Available at https://github.com/OAI/OpenAPI-Specification.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI GS NFV 003 [i.1] and the following apply:

LCM workflow: set of operations, including resource management operations towards the VIM, that are executed by the VNFM to perform a lifecycle management operation

NOTE: Examples for LCM workflows are VNFM-internal procedures associated with an LCM operation, and LCM scripts contained in the VNF package.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CP	Connection Point
CPD	CP Descriptor
EM	Element Manager
ETSI	European Telecommunications Standards Institute
FM	Fault Management
GMT	Greenwich Mean Time

GS	Group Specification
GUI	Graphical User Interface
HATEOAS	Hypermedia As The Engine Of Application State
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IETF	Internet Engineering Task Force
IFA	Interfaces and Architecture
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
LCCN	Life Cycle Change Notifications
LCM	Lifecycle Management
MAC	Medium Access Control
MANO	Management and Orchestration
MIME	Multipurpose Internet Mail Extensions
NFV	Network Functions Virtualisation
NFVO	NFV Orchestrator
NS	Network Service
PM	Performance Management
RAM	Random Access Memory
REST	Representational State Transfer
RFC	Request For Comments
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
URI	Uniform Resource Identifier
VDU	Virtualisation Deployment Unit
VIM	Virtualised Infrastructure Manager
VL	Virtual Link
VLD	VL Descriptor
VNF	Virtualised Network Function
VNFC	VNF Component
VNFD	VNF Descriptor
VNFM	VNF Manager
YAML	YAML Ain't Markup Language

4 General aspects

4.1 Overview

The present document defines the protocol and data model for the following interfaces used over the Or-Vnfm reference point, in the form of RESTful Application Programming Interface (APIs) specifications:

- VNF Lifecycle Management interface (as produced by the VNFM towards the NFVO).
- VNF Performance Management interface (as produced by the VNFM towards the NFVO).
- VNF Fault Management interface (as produced by the VNFM towards the NFVO).
- VNF Indicator interface (as produced by the VNFM towards the NFVO).
- VNF Lifecycle Operation Granting interface (as produced by the NFVO towards the VNFM).
- VNF Package Management interface (as produced by the NFVO towards the VNFM).
- Virtualised Resources Quota Available Notification interface (as produced by the NFVO towards the VNFM).

The design of the protocol and data model for the above interfaces is based on the information model and requirements defined in ETSI GS NFV-IFA 007 [1]. In clause 4, general aspects such as URI structure and supported content formats, general procedures, common data types and authorization of API requests and notifications are specified.

In the subsequent clauses, the protocol and data model for the individual interfaces are specified. Per interface, the resource structure with associated HTTP methods is defined and applicable flows are provided. Further, the resources and the data model are specified in detail.

Annex A provides the mapping of the combination of resources and methods defined in the present document to the operations defined in ETSI GS NFV-IFA 007 [1]. Annex B contains explanations of key concepts. Annex C defines the structure of the VimConnectionInfo registry.

Even though the different interfaces defined in the present document (apart from the Virtualised Resources Quota Available Notification Interface) are related, implementations shall not assume a particular order of messages that arrive via different interfaces.

4.2 URI structure and supported content formats

This clause specifies the URI prefix and the supported formats applicable to the APIs defined in the present document.

All resource URIs of the APIs shall have the following prefix, except the "API versions" resource which shall follow the rules specified in clause 4.6.3:

{apiRoot}/{apiName}/{apiMajorVersion}/

where:

{apiRoot}	indicates the scheme ("http" or "https"), the host name and optional port, and an optional sequence of path segments that together represent a prefix path.

EXAMPLE: http://nfvo.example.com/nfv_apis/abc

{apiName}	indicates the interface name in an abbreviated form. The {apiName} of each interface is defined in the clause specifying the corresponding interface.
{apiMajorVersion}	indicates the current major version (see clause 4.6.1) of the API and is defined in the clause specifying the corresponding interface.

For HTTP requests and responses that have a body, the content format JSON (see IETF RFC 8259 [12]) shall be supported. The JSON format shall be signalled by the content type "application/json".

All APIs shall support and use HTTP over TLS (also known as HTTPS) (see IETF RFC 2818 [3]). TLS version 1.2 as defined by IETF RFC 5246 [7] shall be supported.

NOTE 1: The HTTP protocol elements mentioned in the present document originate from the HTTP specification; HTTPS runs the HTTP protocol in a TLS layer. The present document therefore uses the statement above to mention "HTTP request", "HTTP header", etc., without explicitly calling out whether or not these are run over TLS.

NOTE 2: There are a number of best practices and guidelines how to configure and implement TLS 1.2 in a secure manner, as security threats evolve. A detailed specification of those is beyond the scope of the present document; the reader is referred to external documentation such as annex E of ETSI TS 133 310 [i.3].

All resource URIs of the API shall comply with the URI syntax as defined in IETF RFC 3986 [5]. An implementation that dynamically generates resource URI parts (individual path segments, sequences of path segments that are separated by "/", query parameter values) shall ensure that these parts only use the character set that is allowed by IETF RFC 3986 [5] for these parts.

NOTE 3: This means that characters not part of this allowed set are escaped using percent-encoding as defined by IETF RFC 3986 [5].

Unless otherwise specified explicitly, all request URI parameters that are part of the path of the resource URI shall be individual path segments, i.e. shall not contain the "/" character.

NOTE 4: A request URI parameter is denoted by a string in curly brackets, e.g. {subscriptionId}.

4.3 Common procedures

4.3.1 Introduction

This clause specifies procedures applicable to all interfaces.

4.3.2 Attribute-based filtering

4.3.2.1 Overview and example (informative)

Attribute-based filtering allows to reduce the number of objects returned by a query operation. Typically, attribute-based filtering is applied to a GET request that reads a resource which represents a list of objects (e.g. child resources). Only those objects that match the filter are returned as part of the resource representation in the payload body of the GET response.

Attribute-based filtering can test a simple (scalar) attribute of the resource representation against a constant value, for instance for equality, inequality, greater or smaller than, etc. Attribute-based filtering is requested by adding a set of URI query parameters, the "attribute-based filtering parameters" or "filter" for short, to a resource URI.

The following example illustrates the principle. Assume a resource "container" with the following objects:

EXAMPLE 1: Objects

```
obj1: {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]}
obj2: {"id":456, "weight":500, "parts":[{"id":3, "color":"green"}, {"id":4, "color":"blue"}]}
```

A GET request on the "container" resource would deliver the following response:

EXAMPLE 2: Unfiltered GET

```
Request:
GET …/container

Response:
[
    {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]},
    {"id":456, "weight":500, "parts":[{"id":3, "color":"green"}, {"id":4, "color":"blue"}]}
]
```

A GET request with a filter on the "container" resource would deliver the following response:

EXAMPLE 3: GET with filter

```
Request:
GET …/container?filter=(eq,weight,100)

Response:
[
    {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]}

]
```

For hierarchically-structured data, filters can also be applied to attributes deeper in the hierarchy. In case of arrays, a filter matches if any of the elements of the array matches. In other words, when applying the filter "(eq,parts/color,green)" to the objects in Example 1, the filter matches obj1 when evaluating the second entry in the "parts" array of obj1 and matches obj2 already when evaluating the first entry in the "parts" array of obj2. As the result, both obj1 and obj2 match the filter.

If a filter contains multiple sub-parts that only differ in the leaf attribute (i.e. they share the same attribute prefix), they are evaluated together per array entry when traversing an array. As an example, the two expressions in the filter "(eq,parts/color,green);(eq,parts/id,3)" would be evaluated together for each entry in the array "parts". As the result, obj2 matches the filter.

4.3.2.2 Specification

An attribute-based filter shall be represented by a URI query parameter named "filter". The value of this parameter shall consist of one or more strings formatted according to "simpleFilterExpr", concatenated using the ";" character:

```
simpleFilterExprOne     := <opOne>","<attrName>["/"<attrName>]*","<value>
simpleFilterExprMulti   := <opMulti>","<attrName>["/"<attrName>]*","<value>[","<value>]*
simpleFilterExpr        := "("<simpleFilterExprOne>")" | "("<simpleFilterExprMulti>")"
filterExpr              := <simpleFilterExpr>[";"<simpleFilterExpr>]*
filter                  := "filter"=<filterExpr>
opOne                   := "eq" | "neq" | "gt" | "lt" | "gte" | "lte"
opMulti                 := "in" | "nin" | "cont" | "ncont"
attrName                := string
value                   := string
```

where:

```
*   zero or more occurrences
[]  grouping of expressions to be used with *
""  quotation marks for marking string constants
<>  name separator
|   separator of alternatives
```

"AttrName" is the name of one attribute in the data type that defines the representation of the resource. The slash ("/") character in "simpleFilterExprOne" and " simpleFilterExprMulti" allows concatenation of <attrName> entries to filter by attributes deeper in the hierarchy of a structured document. The elements "opOne" and "opMulti" stand for the comparison operators (accepting one comparison value or a list of such values). If the expression has concatenated <attrName> entries, it means that the operator is applied to the attribute addressed by the last <attrName> entry included in the concatenation. All simple filter expressions are combined by the "AND" logical operator, denoted by ";".

In a concatenation of <attrName> entries in a <simpleFilterExprOne> or <simpleFilterExprMulti>, the rightmost <attrName> entry is called "leaf attribute". The concatenation of all "attrName" entries except the leaf attribute is called the "attribute prefix". If an attribute referenced in an expression is an array, an object that contains a corresponding array shall be considered to match the expression if any of the elements in the array matches all expressions that have the same attribute prefix.

The leaf attribute of a <simpleFilterExprOne> or <simpleFilterExprMulti> shall not be structured but shall be of a simple (scalar) type such as String, Number, Boolean or DateTime, or shall be an array of simple (scalar) values. Attempting to apply a filter with a structured leaf attribute shall be rejected with "400 Bad request". A <filterExpr> shall not contain any invalid <simpleFilterExpr> entry.

The operators listed in table 4.3.2.2-1 shall be supported.

Table 4.3.2.2-1: Operators for attribute-based filtering

Operator with parameters	Meaning
eq,<attrName>,<value>	Attribute **equal** to <value>
neq,<attrName>,<value>	Attribute **not equal** to <value>
in,<attrName>,<value>[,<value>]*	Attribute equal to one of the values in the list ("**in set**" relationship)
nin,<attrName>,<value>[,<value>]*	Attribute not equal to any of the values in the list ("**not in set**" relationship)
gt,<attrName>,<value>	Attribute **greater than** <value>
gte,<attrName>,<value>	Attribute **greater than or equal** to <value>
lt,<attrName>,<value>	Attribute **less than** <value>
lte,<attrName>,<value>	Attribute **less than or equal** to <value>
cont,<attrName>,<value>[,<value>]*	String attribute **contains** (at least) one of the values in the list
ncont,<attrName>,<value>[,<value>]*	String attribute **does not contain** any of the values in the list

Table 4.3.2.2-2: Applicability of the operators to data types

Operator	String	Number	DateTime	Enumeration	Boolean
eq	x	x	-	x	x
neq	x	x	-	x	x
in	x	x	-	x	-
nin	x	x	-	x	-
gt	x	x	x	-	-
gte	x	x	x	-	-
lt	x	x	x	-	-
lte	x	x	x	-	-
cont	x	-	-	-	-
ncont	x	-	-	-	-

Table 4.3.2.2-2 defines which operators are applicable for which data types. All combinations marked with a "x" shall be supported.

All objects that match the filter shall be returned as response to a GET request that contains a filter.

A <value> entry shall contain a scalar value of type Number, String, Boolean, Enum or DateTime. The content of a <value> entry shall be formatted the same way as the representation of the related attribute in the resource representation: The syntax of DateTime <value> entries shall follow the "date-time" production of IETF RFC 3339 [4]. The syntax of Boolean and Number <value> entries shall follow IETF RFC 8259 [12].

A <value> entry of type String shall be enclosed in single quotes (') if it contains any of the characters ")", "'" or ",", and may be enclosed in single quotes otherwise. Any single quote (') character contained in a <value> entry shall be represented as a sequence of two single quote characters.

The "/" and "~" characters in <attrName> shall be escaped according to the rules defined in section 3 of IETF RFC 6901 [21]. The "," character in <attrName> shall be escaped by replacing it with "~a".

In the resulting <filterExpr>, percent-encoding as defined in IETF RFC 3986 [5] shall be applied to the characters that are not allowed in a URI query part according to Appendix A of IETF RFC 3986 [5], and to the ampersand "&" character.

NOTE: In addition to the statement on percent-encoding above, it is reminded that the percent "%" character is always percent-encoded when used in parts of a URI, according to IETF RFC 3986 [5].

Attribute-based filters are supported for certain resources. Details are defined in the clauses specifying the actual resources.

4.3.3 Attribute selectors

4.3.3.1 Overview and example (informative)

Certain resource representations can become quite big, in particular, if the resource is a container for multiple sub-resources, or if the resource representation itself contains a deeply-nested structure. In these cases, it can be desired to reduce the amount of data exchanged over the interface and processed by the API consumer application. On the other hand, it can also be desirable that a "drill-deep" for selected parts of the omitted data can be initiated quickly.

An attribute selector allows the API consumer to choose which attributes it wants to be contained in the response. Only attributes that are not required to be present, i.e. those with a lower bound of zero on their cardinality (e.g. 0..1, 0..N) and that are not conditionally mandatory, are allowed to be omitted as part of the selection process. Attributes can be marked for inclusion or exclusion.

If an attribute is omitted, a link to a resource may be added where the information of that attribute can be fetched. Such approach is known as HATEOAS which is a common pattern in REST, and enables drilling down on selected issues without having to repeat a request that may create a potentially big response.

4.3.3.2 Specification

4.3.3.2.1 GET request

The URI query parameters for attribute selection are defined in table 4.3.3.2.1-1.

In the provisions below, "complex attributes" are assumed to be those attributes that are structured, or that are arrays.

Table 4.3.3.2.1-1: Attribute selector parameters

Parameter	Definition
all_fields	This URI query parameter requests that all complex attributes are included in the response, including those suppressed by exclude_default. It is inverse to the "exclude_default" parameter. The API producer shall support this parameter for certain resources. Details are defined in the clauses specifying the actual resources.
fields	This URI query parameter requests that only the listed complex attributes are included in the response. The parameter shall be formatted as a list of attribute names. An attribute name shall either be the name of an attribute, or a path consisting of the names of multiple attributes with parent-child relationship, separated by "/". Attribute names in the list shall be separated by comma (","). Valid attribute names for a particular GET request are the names of all complex attributes in the expected response that have a lower cardinality bound of 0 and that are not conditionally mandatory. The API producer should support this parameter for certain resources. Details are defined in the clauses specifying the actual resources.
exclude_fields	This URI query parameter requests that the listed complex attributes are excluded from the response. For the format, eligible attributes and support by the API producer, the provisions defined for the "fields" parameter shall apply.
exclude_default	Presence of this URI query parameter requests that a default set of complex attributes shall be excluded from the response. The default set is defined per resource in the present document. Not every resource will necessarily have such a default set. Only complex attributes with a lower cardinality bound of zero that are not conditionally mandatory can be included in the set. The API producer shall support this parameter for certain resources. Details are defined in the clauses specifying the actual resources. This parameter is a flag, i.e. it has no value. If a resource supports attribute selectors and none of the attribute selector parameters is specified in a GET request, the "exclude_default" parameter shall be assumed as the default.

The "/" and "~" characters in attribute names in an attribute selector shall be escaped according to the rules defined in section 3 of IETF RFC 6901 [21]. The "," character in attribute names in an attribute selector shall be escaped by replacing it with "~a". Further, percent-encoding as defined in IETF RFC 3986 [5] shall be applied to the characters that are not allowed in a URI query part according to Appendix A of IETF RFC 3986 [5], and to the ampersand "&" character.

4.3.3.2.2 GET response

Table 4.3.3.2.2-1 defines the valid parameter combinations in a GET request and their effect on the GET response.

Table 4.3.3.2.2-1: Valid combinations of attribute selector parameters

Parameter combination	The GET response shall include…
(none)	… same as "exclude_default".
all_fields	… all attributes.
fields=<list>	… all attributes except all complex attributes with minimum cardinality of zero that are not conditionally mandatory, and that are not provided in <list>.
exclude_fields=<list>	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory, and that are provided in <list>.
exclude_default	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory, and that are part of the "default exclude set" defined in the present document for the particular resource.
exclude_default and fields=<list>	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory and that are part of the "default exclude set" defined in the present document for the particular resource, but that are not part of <list>.

If complex attributes were omitted in a GET response, the response may contain a number of links that allow to obtain directly the content of the omitted attributes. Such links shall be embedded into a structure named "_links" at the same level as the omitted attribute. That structure shall contain one entry for each link, named as the omitted attribute, and containing an "href" attribute that contains the URI of a resource that can be read with GET to obtain the content of the omitted attribute. A link shall not be present if the attribute is not present in the underlying resource representation. The resource URI structure of such links is not standardized but may be chosen by the VNFM implementation. Performing a GET request on such a link shall return a representation that contains the content of the omitted attribute.

EXAMPLE:

```
"_links" : {
      "vnfcs" : {"href" : ".../vnflcm/v1/vnf_instances/1234/vnfcs"},
      "extVirtualLinks" : {"href" : ".../vnflcm/v1/_dynamic/7d6bef4e-d86b-4abc-97ed-9dc9b951f206"}
}
```

4.3.4 Usage of HTTP header fields

4.3.4.1 Introduction

HTTP headers are components of the header section of the HTTP request and response messages. They contain the information about the server/client and metadata of the transaction. The use of HTTP header fields shall comply with the provisions defined for those header fields in the specifications referenced from tables 4.3.4.2-1 and 4.3.4.3-1. The following clauses describe the HTTP header fields that are explicitly mentioned in the present document.

4.3.4.2 Request header fields

This clause describes the usage of HTTP header fields of the request messages applicable to the APIs defined in the present document. The HTTP header fields used in the request messages are specified in table 4.3.4.2-1.

Table 4.3.4.2-1: Header fields supported in the request message

Header field name	Reference	Example	Descriptions
Accept	IETF RFC 7231 [13]	application/json	Content-Types that are acceptable for the response. This header field shall be present if the response is expected to have a non-empty message body.
Content-Type	IETF RFC 7231 [13]	application/json	The MIME type of the body of the request. This header field shall be present if the request has a non-empty message body.
Authorization	IETF RFC 7235 [16]	Bearer mF_9.B5f-4.1JqM	The authorization token for the request. Details are specified in clause 4.5.3.
Range	IETF RFC 7233 [15]	1 000-2 000	Requested range of bytes from a file.
Version	IETF RFC 4229 [24]	1.2.0 or 1.2.0-impl:example.com:myVNFM:4	Version of the API requested to use when responding to this request.

4.3.4.3 Response header fields

This clause describes the usage of HTTP header fields of the response messages applicable to the APIs defined in the present document. The HTTP header fields used in the response messages are specified in table 4.3.4.3-1.

Table 4.3.4.3-1: Header fields supported in the response message

Header field name	Reference	Example	Descriptions
Content-Type	IETF RFC 7231 [13]	application/json	The MIME type of the body of the response. This header field shall be present if the response has a non-empty message body.
Location	IETF RFC 7231 [13]	http://www.example.com/vnflcm/v1/vnf_instances/123	Used in redirection, or when a new resource has been created. This header field shall be present if the response status code is 201 or 3xx. In the present document this header field is also used if the response status code is 202 and a new resource was created.
WWW-Authenticate	IETF RFC 7235 [16]	Bearer realm="example"	Challenge if the corresponding HTTP request has not provided authorization, or error details if the corresponding HTTP request has provided an invalid authorization token.

Header field name	Reference	Example	Descriptions
Accept-Ranges	IETF RFC 7233 [15]	bytes	Used by the server to signal whether or not it supports ranges for certain resources.
Content-Range	IETF RFC 7233 [15]	bytes 21 010 - 47 021/47 022	Signals the byte range that is contained in the response, and the total length of the file.
Retry-After	IETF RFC 7231 [13]	Fri, 31 Dec 1999 23:59:59 GMT or 120	Used to indicate how long the user agent ought to wait before making a follow-up request. It can be used with 503 responses. The value of this field can be an HTTP-date or a number of seconds to delay after the response is received.
Link	IETF RFC 8288 [22]	<http://example.com/resources?nextpage_opaque_marker=abc123>; rel="next"	Reference to other resources. Used for paging in the present document, see clause 4.7.2.1.
Version	IETF RFC 4229 [24]	1.2.0 or 1.2.0-impl:example.com:myVNFM:4	Version of the API requested to use when responding to this request.

4.3.5 Error reporting

4.3.5.1 Introduction

In RESTful interfaces, application errors are mapped to HTTP errors. Since HTTP error information is generally not enough to discover the root cause of the error, additional application specific error information is typically delivered. The following clauses define such a mechanism to be used by the interfaces specified in the present document.

4.3.5.2 General mechanism

When an error occurs that prevents the API producer from successfully fulfilling the request, the HTTP response shall include in the response a status code in the range 400..499 (client error) or 500..599 (server error) as defined by the HTTP specification (see IETF RFC 7231 [13], IETF RFC 7232 [14], IETF RFC 7233 [15] and IETF RFC 7235 [16], as well as by IETF RFC 6585 [9]). In addition, the response body should contain a JSON representation of a "ProblemDetails" data structure according to IETF RFC 7807 [19] that provides additional details of the error. In that case, as defined by IETF RFC 7807 [19], the "Content-Type" HTTP header shall be set to "application/problem+json".

4.3.5.3 Type: ProblemDetails

The definition of the general "ProblemDetails" data structure from IETF RFC 7807 [19] is reproduced in table 4.3.5.3-1. Compared to the general framework defined in IETF RFC 7807 [19], the "status" and "detail" attributes are mandated to be included by the present document, to ensure that the response contains additional textual information about an error. IETF RFC 7807 [19] foresees extensibility of the "ProblemDetails" type. It is possible that particular APIs in the present document, or particular implementations, define extensions to define additional attributes that provide more information about the error.

The description column only provides some explanation of the meaning to facilitate understanding of the design. For a full description, see IETF RFC 7807 [19].

Table 4.3.5.3-1: Definition of the ProblemDetails data type

Attribute name	Data type	Cardinality	Description
type	URI	0..1	A URI reference according to IETF RFC 3986 [5] that identifies the problem type. It is encouraged that the URI provides human-readable documentation for the problem (e.g. using HTML) when dereferenced. When this member is not present, its value is assumed to be "about:blank".
title	String	0..1	A short, human-readable summary of the problem type. It should not change from occurrence to occurrence of the problem, except for purposes of localization. If type is given and other than "about:blank", this attribute shall also be provided.
status	Integer	1	The HTTP status code for this occurrence of the problem.
detail	String	1	A human-readable explanation specific to this occurrence of the problem.
instance	URI	0..1	A URI reference that identifies the specific occurrence of the problem. It may yield further information if dereferenced.
(additional attributes)	Not specified.	0..N	Any number of additional attributes, as defined in a specification or by an implementation.
NOTE: It is expected that the minimum set of information returned in ProblemDetails consists of "status" and "detail". For the definition of specific "type" values as well as extension attributes by implementations, guidance can be found in IETF RFC 7807 [19].			

4.3.5.4 Common error situations

The following common error situations are applicable on all REST resources and related HTTP methods specified in the present document, and shall be handled as defined in the present clause.

NOTE 1: The error handling defined in this clause only applies to REST resources defined in the present document. For the token endpoint defined in IETF RFC 6749 [10] and re-used in the present document as defined in clause 4.5.3, the error handling provisions are defined in clause 4.5.3.

400 Bad Request: If the request is malformed or syntactically incorrect (e.g. if the request URI contains incorrect query parameters or the payload body contains a syntactically incorrect data structure), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and should include in the "detail" attribute more information about the source of the problem.

400 Bad Request: If the response to a GET request which queries a container resource would be so big that the performance of the API producer is adversely affected, and the API producer does not support paging for the affected resource, it shall respond with this response code. The "ProblemDetails" structure shall be provided, and should include in the "detail" attribute more information about the source of the problem.

400 Bad Request:	If there is an application error related to the client's input that cannot be easily mapped to any other HTTP response code ("catch all error"), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and shall include in the "detail" attribute more information about the source of the problem.

NOTE 2: It is by design to represent these application error situations with the same HTTP error response code 400.

400 Bad Request:	If the request contains a malformed access token, the API producer should respond with this response. The details of the error shall be returned in the WWW-Authenticate HTTP header, as defined in IETF RFC 6750 [11]. The ProblemDetails structure may be provided.

NOTE 3: The use of this HTTP error response code described above is applicable to the use of the OAuth 2.0 for the authorization of API requests and notifications, as defined in clauses 4.5.3.3 and 4.5.3.4.

401 Unauthorized:	If the request contains no access token even though one is required, or if the request contains an authorization token that is invalid (e.g. expired or revoked), the API producer should respond with this response. The details of the error shall be returned in the WWW-Authenticate HTTP header, as defined in IETF RFC 6750 [11] and IETF RFC 7235 [16]. The ProblemDetails structure may be provided.
403 Forbidden:	If the API consumer is not allowed to perform a particular request to a particular resource, the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided. It should include in the "detail" attribute information about the source of the problem, and may indicate how to solve it.
404 Not Found:	If the API producer did not find a current representation for the resource addressed by the URI passed in the request, or is not willing to disclose that one exists, it shall respond with this response code. The "ProblemDetails" structure may be provided, including in the "detail" attribute information about the source of the problem, e.g. a wrong resource URI variable.

NOTE 4: This response code is not appropriate in case the resource addressed by the URI is a container resource which is designed to contain child resources, but does not contain any child resource at the time the request is received. For a GET request to an existing empty container resource, a typical response contains a 200 OK response code and a payload body with an empty array.

405 Method Not Allowed:	If a particular HTTP method is not supported for a particular resource, the API producer shall respond with this response code. The "ProblemDetails" structure may be omitted.
406 Not Acceptable:	If the "Accept" HTTP header does not contain at least one name of a content type that is acceptable to the API producer, the API producer shall respond with this response code. The "ProblemDetails" structure may be omitted.
413 Payload Too Large:	If the payload body of a request is larger than the amount of data the API producer is willing or able to process, it shall respond with this response code, following the provisions in IETF RFC 7231 [13] for the use of the "Retry-After" HTTP header and for closing the connection. The "ProblemDetails" structure may be omitted.
414 URI Too Long:	If the request URI of a request is longer than the API producer is willing or able to process, it shall respond with this response code. This condition can e.g. be caused by passing long queries in the request URI of a GET request. The "ProblemDetails" structure may be omitted.
422 Unprocessable Entity:	If the payload body of a request contains syntactically correct data (e.g. well-formed JSON) but the data cannot be processed (e.g. because it fails validation against a schema), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and should include in the "detail" attribute more information about the source of the problem.

NOTE 5: This error response code is only applicable for methods that have a request body.

429 Too Many Requests:	If the API consumer has sent too many requests in a defined period of time and the API producer is able to detect that condition ("rate limiting"), the API producer shall respond with this response code, following the provisions in IETF RFC 6585 [9] for the use of the "Retry-After" HTTP header. The "ProblemDetails" structure shall be provided, and shall include in the "detail" attribute more information about the source of the problem.

NOTE 6: The period of time and allowed number of requests are configured within the API producer by means outside the scope of the present document.

500 Internal Server Error:	If there is an application error not related to the client's input that cannot be easily mapped to any other HTTP response code ("catch all error"), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and shall include in the "detail" attribute more information about the source of the problem.
503 Service Unavailable:	If the API producer encounters an internal overload situation of itself or of a system it relies on, it should respond with this response code, following the provisions in IETF RFC 7231 [13] for the use of the "Retry-After" HTTP header and for the alternative to refuse the connection. The "ProblemDetails" structure may be omitted.
504 Gateway Timeout:	If the API producer encounters a timeout while waiting for a response from an upstream server (i.e. a server that the API producer communicates with when fulfilling a request), it should respond with this response code.

4.3.5.5 Overview of HTTP error status codes

Table 4.3.5.5-1 lists the HTTP error status codes that are explicitly mentioned in the present document. The full definition of each error code can be obtained from the referenced specification.

Table 4.3.5.5-1: HTTP error status codes used in the present document

Code	Status text	Reference	Explanation
400	Bad Request	IETF RFC 7231 [13] IETF RFC 6750 [11] IETF RFC 7235 [16]	Required information for the request was missing, the request had syntactical errors, or the request contains a malformed access token or malformed credentials. In the present document, this code is also used as "catch-all" code for client errors.
401	Unauthorized	IETF RFC 7235 [16]	Client is required to include valid credentials in the request. See clause 4.5.3.
403	Forbidden	IETF RFC 7231 [13]	The client is not allowed to perform the request on that resource.
404	Not Found	IETF RFC 7231 [13]	The requested URI was not found. A reason can e.g. be that resource URI variables were set wrongly.
405	Method Not Allowed	IETF RFC 7231 [13]	See clause 4.3.5.4.
406	Not Acceptable	IETF RFC 7231 [13]	See clause 4.3.5.4.
409	Conflict	IETF RFC 7231 [13]	Another request is in progress that prohibits the fulfilment of the current request, or the current resource state is inconsistent with the request.
412	Precondition Failed	IETF RFC 7232 [14]	This code is used in conjunction with conditional requests (typically used to protect resources consistency when using PUT or PATCH in a multi-client scenario) to indicate that a precondition has failed.
413	Payload Too Large	IETF RFC 7231 [13]	The server is refusing to process a request because the request payload is larger than the server is willing or able to process.
414	URI Too Long	IETF RFC 7231 [13]	The server is refusing to process a request because the request URI is longer than the server is willing or able to process.
416	Range Not Satisfiable	IETF RFC 7233 [15]	This code is returned if the requested byte range in the Range HTTP header is not present in the requested resource.
422	Unprocessable Entity	IETF RFC 4918 [6]	The server understands the content type of the request entity and the syntax of the request entity is correct but was unable to process the contained instructions.
429	Too Many Requests	IETF RFC 6585 [9]	The server is refusing to process a request because the client has sent too many requests in a given period of time (rate limiting).
500	Internal Server Error	IETF RFC 7231 [13]	Server is unable to process the request. Retrying the same request later might eventually succeed. In the present document, this code is also used as "catch-all" code for server errors.
503	Service Unavailable	IETF RFC 7231 [13]	Server is unable to process the request due to internal overload.
504	Gateway Timeout	IETF RFC 7231 [13]	The server did not receive a timely response from an upstream server it needed to access in order to complete the request.

In general, error response codes used for application errors should be mapped to the most similar HTTP error status code. If no such code is applicable, one of the codes 400 (Bad Request, for client errors) or 500 (Internal Server Error, for server errors) should be used. Implementations may use additional error response codes on top of the ones listed in table 4.3.5.5-1, as long as they are valid HTTP response codes; and should include a ProblemDetails structure in the payload body as defined in clause 4.3.5.2. A list of all valid HTTP response codes and their specification documents can be obtained from the HTTP status code registry [i.4].

4.4 Common data types

4.4.1 Structured data types

4.4.1.1 Introduction

This clause defines data structures that are referenced from data structures in multiple interfaces.

4.4.1.2 Type: Object

An object contains structured data, and shall comply with the provisions of clause 4 of IETF RFC 8259 [12].

4.4.1.3 Type: Link

This type represents a link to a resource using an absolute URI. It shall comply with the provisions defined in table 4.4.1.3-1.

Table 4.4.1.3-1: Definition of the Link data type

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI of another resource referenced from a resource. Shall be an absolute URI (i.e. a URI that contains {apiRoot}).

4.4.1.3a Type: NotificationLink

This type represents a link to a resource in a notification, using an absolute or relative URI. It shall comply with the provisions defined in table 4.4.1.3a-1.

Table 4.4.1.3a-1: Definition of the NotificationLink data type

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI of a resource referenced from a notification. Should be an absolute URI (i.e. a URI that contains {apiRoot}), however, may be a relative URI (i.e. a URI where the {apiRoot} part is omitted) if the {apiRoot} information is not available.

4.4.1.4 Type: KeyValuePairs

This type represents a list of key-value pairs. The order of the pairs in the list is not significant. In JSON, a set of key-value pairs is represented as an object. It shall comply with the provisions defined in clause 4 of IETF RFC 8259 [12]. In the following example, a list of key-value pairs with four keys ("aString", "aNumber", "anArray" and "anObject") is provided to illustrate that the values associated with different keys can be of different type.

EXAMPLE:

```
{
    "aString" : "ETSI NFV SOL",
    "aNumber" : 0.03,
    "anArray" : [1,2,3],
    "anObject" : {"organization" : "ETSI", "isg" : "NFV", workingGroup" : "SOL"}
}
```

4.4.1.5 Type: VnfInstanceSubscriptionFilter

This type represents subscription filter criteria to match VNF instances. It shall comply with the provisions defined in table 4.4.1.5-1.

Table 4.4.1.5-1: Definition of the VnfInstanceSubscriptionFilter data type

Attribute name	Data type	Cardinality	Description
vnfdIds	Identifier	0..N	If present, match VNF instances that were created based on a VNFD identified by one of the vnfdId values listed in this attribute. See note 1.
vnfProductsFromProviders	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products from certain providers. See note 1.
>vnfProvider	String	1	Name of the VNF provider to match.
>vnfProducts	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products with certain product names, from one particular provider.
>>vnfProductName	String	1	Name of the VNF product to match.
>>versions	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products with certain versions and a certain product name, from one particular provider.
>>>vnfSoftwareVersion	Version	1	Software version to match.
>>>vnfdVersions	Version	0..N	If present, match VNF instances that belong to VNF products with certain VNFD versions, a certain software version and a certain product name, from one particular provider.
vnfInstanceIds	Identifier	0..N	If present, match VNF instances with an instance identifier listed in this attribute. See note 2.
vnfInstanceNames	String	0..N	If present, match VNF instances with a VNF Instance Name listed in this attribute. See note 2.
NOTE 1: The attributes "vnfdIds" and "vnfProductsFromProviders" are alternatives to reference to VNF instances that are based on certain VNFDs in a filter. They should not be used both in the same filter instance, but one alternative should be chosen. NOTE 2: The attributes "vnfInstanceIds" and "vnfInstanceNames" are alternatives to reference to particular VNF Instances in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

4.4.1.6 Type: VimConnectionInfo

This type represents parameters to connect to a VIM for managing the resources of a VNF instance. It shall comply with the provisions defined in table 4.4.1.6-1.

This structure is used to convey VIM-related parameters over the Or-Vnfm interface. Additional parameters for a VIM may be configured into the VNFM by means outside the scope of the present document, and bound to the identifier of that VIM.

Table 4.4.1.6-1: Definition of the VimConnectionInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	The identifier of the VIM Connection. This identifier is managed by the NFVO.
vimId	Identifier	0..1	The identifier of the VIM instance. This identifier is managed by the NFVO. Shall be present to address additional information about the VIM if such information has been configured into the VNFM by means outside the scope of the present document, and should be absent otherwise.
vimType	String	1	Discriminator for the different types of the VIM information. The value of this attribute determines the structure of the "interfaceInfo" and "accessInfo" attributes, based on the type of the VIM. The set of permitted values is expected to change over time as new types or versions of VIMs become available. The ETSI NFV registry of VIM-related information [i.5] provides access to information about VimConnectionInfo definitions for various VIM types. The structure of the registry is defined in annex C.
interfaceInfo	KeyValuePairs	0..1	Information about the interface or interfaces to the VIM, if applicable, such as the URI of an interface endpoint to communicate with the VIM. The applicable keys are dependent on the content of vimType. Alternatively, such information may have been configured into the VNFM and bound to the vimId.
accessInfo	KeyValuePairs	0..1	Authentication credentials for accessing the VIM, and other access-related information such as tenants or infrastructure resource groups (see note). The applicable keys are dependent on the content of vimType. If the VimConnectionInfo structure is part of an HTTP response payload body, sensitive attributes that are children of this attributes (such as passwords) shall not be included. If the VimConnectionInfo structure is part of an HTTP request payload body, sensitive attributes that are children of this attribute (such as passwords) shall be present if they have not been provisioned out of band.
extra	KeyValuePairs	0..1	VIM type specific additional information. The applicable structure, and whether or not this attribute is available, is dependent on the content of vimType.
NOTE: If applicable, this attribute also provides information about the resourceGroupIds that are accessible using a particular set of credentials. See definition of "resourceGroupId" in clause 9.5.3.3.			

4.4.1.7 Type: ResourceHandle

This type represents the information that allows addressing a virtualised resource that is used by a VNF instance. Information about the resource is available from the VIM. The ResourceHandle type shall comply with the provisions defined in table 4.4.1.7-1.

Table 4.4.1.7-1: Definition of the ResourceHandle data type

Attribute name	Data type	Cardinality	Description
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage the resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the " vimConnectionInfo" attribute of the "VnfInstance" structure.
resourceProviderId	Identifier	0..1	Identifier of the entity responsible for the management of the resource. This attribute shall only be supported and present when VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	Identifier of the resource in the scope of the VIM or the resource provider.
vimLevelResourceType	String	0..1	Type of the resource in the scope of the VIM or the resource provider. See note.
NOTE: The value set of the "vimLevelResourceType" attribute is within the scope of the VIM or the resource provider and can be used as information that complements the ResourceHandle. This value set is different from the value set of the "type" attribute in the ResourceDefinition (refer to clause 9.5.3.2).			

4.4.1.8 void

4.4.1.9 void

4.4.1.10 Type: VnfExtCpData

This type represents configuration information for external CPs created from a CPD. It shall comply with the provisions defined in table 4.4.1.10-1.

Table 4.4.1.10-1: Definition of the VnfExtCpData data type

Attribute name	Data type	Cardinality	Description
cpdId	IdentifierInVnfd	1	The identifier of the CPD in the VNFD.
cpConfig	VnfExtCpConfig	1..N	List of instance data that need to be configured on the CP instances created from the respective CPD.

4.4.1.10a Type: VnfExtCpConfig

This type represents an externally provided link port or network address information per instance of an external connection point. In case a link port is provided, the VNFM shall use that link port when connecting the external CP to the external VL. In case a link port is not provided, the VNFM shall create a link port on the external VL, and use that link port to connect the external CP to the external VL.

This type shall comply with the provisions defined in table 4.4.1.10a-1.

Table 4.4.1.10a-1: Definition of the VnfExtCpConfig data type

Attribute name	Data type	Cardinality	Description
cpInstanceId	IdentifierInVnf	0..1	Identifier of the external CP instance to which this set of configuration parameters is requested to be applied. Shall be present if this instance has already been created.
linkPortId	Identifier	0..1	Identifier of a pre-configured link port to which the external CP will be associated. See note.
cpProtocolData	CpProtocolData	0..N	Parameters for configuring the network protocols on the link port that connects the CP to a VL. See note.
NOTE: The following conditions apply to the attributes "linkPortId" and " cpProtocolData": 1) The "linkPortId" and "cpProtocolData" attributes shall both be absent for the deletion of an existing external CP instance addressed by cpInstanceId. 2) At least one of these attributes shall be present for a to-be-created external CP instance or an existing external CP instance. 3) If the "linkPortId" attribute is absent, the VNFM shall create a link port. 4) If the "cpProtocolData" attribute is absent, the "linkPortId" attribute shall be provided referencing a pre-created link port, and the VNFM can use means outside the scope of the present document to obtain the pre-configured address information for the connection point from the resource representing the link port. 5) If both "cpProtocolData" and "linkportId" are provided, the API consumer shall ensure that the cpProtocolData can be used with the pre-created link port referenced by "linkPortId".			

4.4.1.10b Type: CpProtocolData

This type represents network protocol data. It shall comply with the provisions defined in table 4.4.1.10b-1.

Table 4.4.1.10b-1: Definition of the CpProtocolData data type

Attribute name	Data type	Cardinality	Description
layerProtocol	Enum (inlined)	1	Identifier of layer(s) and protocol(s). Permitted values: IP_OVER_ETHERNET See note.
ipOverEthernet	IpOverEthernetAddressData	0..1	Network address data for IP over Ethernet to assign to the extCP instance. Shall be present if layerProtocol is equal to "IP_OVER_ETHERNET", and shall be absent otherwise.
NOTE: This attribute allows to signal the addition of further types of layer and protocol in future versions of the present document in a backwards-compatible way. In the current version of the present document, only IP over Ethernet is supported.			

4.4.1.10c Type: IpOverEthernetAddressData

This type represents network address data for IP over Ethernet. It shall comply with the provisions defined in table 4.4.1.10c-1.

Table 4.4.1.10c-1: Definition of the IpOverEthernetAddressData data type

Attribute name	Data type	Cardinality	Description
macAddress	MacAddress	0..1	MAC address. If this attribute is not present, it shall be chosen by the VIM. See note 1.
ipAddresses	Structure (inlined)	0..N	List of IP addresses to assign to the CP instance. Each entry represents IP address data for fixed or dynamic IP address assignment per subnet. If this attribute is not present, no IP address shall be assigned. See note 1.
>type	Enum (inlined)	1	The type of the IP addresses. Permitted values: IPV4, IPV6.
>fixedAddresses	IpAddress	0..N	Fixed addresses to assign (from the subnet defined by "subnetId" if provided). See note 2.
>numDynamicAddresses	Integer	0..1	Number of dynamic addresses to assign (from the subnet defined by "subnetId" if provided). See note 2.
>addressRange	Structure (inlined)	0..1	An IP address range to be used, e.g. in case of egress connections. In case this attribute is present, IP addresses from the range will be used. See note 2.
>>minAddress	IpAddress	1	Lowest IP address belonging to the range.
>>maxAddress	IpAddress	1	Highest IP address belonging to the range.
>subnetId	IdentifierInVim	0..1	Subnet defined by the identifier of the subnet resource in the VIM. In case this attribute is present, IP addresses from that subnet will be assigned; otherwise, IP addresses not bound to a subnet will be assigned.
NOTE 1: At least one of "macAddress" or "ipAddresses" shall be present. NOTE 2: Exactly one of "fixedAddresses", "numDynamicAddresses" or "ipAddressRange" shall be present.			

4.4.1.11 Type: ExtVirtualLinkData

This type represents an external VL. It shall comply with the provisions defined in table 4.4.1.11-1.

Table 4.4.1.11-1: Definition of the ExtVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	The identifier of the external VL instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of this resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	The identifier of the resource in the scope of the VIM or the resource provider.
extCps	VnfExtCpData	1..N	External CPs of the VNF to be connected to this external VL.
extLinkPorts	ExtLinkPortData	0..N	Externally provided link ports to be used to connect external connection points to this external VL. If this attribute is not present, the VNFM shall create the link ports on the external VL.

4.4.1.12 Type: ExtManagedVirtualLinkData

This type represents an externally-managed internal VL. It shall comply with the provisions defined in table 4.4.1.12-1.

Table 4.4.1.12-1: Definition of the ExtManagedVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	The identifier of the externally-managed internal VL instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
vnfVirtualLinkDescId	IdentifierInVnfd	1	The identifier of the VLD in the VNFD for this VL.
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of this resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	The identifier of the resource in the scope of the VIM or the resource provider.

4.4.1.13 Type: ApiVersionInformation

This type represents API version information. It shall comply with the provisions defined in table 4.4.1.13-1.

Table 4.4.1.13-1: ApiVersionInformation data type

Attribute name	Data type	Cardinality	Description
uriPrefix	String	1	Specifies the URI prefix for the API, in the following form {apiRoot}/{apiName}/{apiMajorVersion}/.
apiVersions	Structure (inlined)	1..N	Version(s) supported for the API signaled by the uriPrefix attribute.
>version	String	1	Identifies a supported version. The value of the version attribute shall be a version identifier as specified in clause 4.6.1.
>isDeprecated	Boolean	0..1	If such information is available, this attribute indicates whether use of the version signaled by the version attribute is deprecated (true) or not (false). See note.
>retirementDate	DateTime	0..1	The date and time after which the API version will no longer be supported. This attribute may be included if the value of the isDeprecated attribute is set to true and shall be absent otherwise.
NOTE: A deprecated version is still supported by the API producer but is recommended not to be used any longer. When a version is no longer supported, it does not appear in the response body.			

4.4.2 Simple data types and enumerations

4.4.2.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in multiple interfaces.

4.4.2.2 Simple data types

Table 4.4.2.2-1: Simple data types

Type name	Description
Identifier	An identifier with the intention of being globally unique. Representation: string of variable length.
IdentifierInVnfd	An identifier that is unique within a VNF descriptor. Representation: string of variable length.
IdentifierInVim	An identifier maintained by the VIM or other resource provider. It is expected to be unique within the VIM instance. Representation: string of variable length.
IdentifierInVnf	An identifier that is unique for the respective type within a VNF instance, but that need not be globally unique. Representation: string of variable length.
IdentifierLocal	An identifier that is unique within a limited local scope other than above listed identifiers, such as within a complex data structure or within a request-response pair. Representation: string of variable length.
DateTime	Date-time stamp. Representation: String formatted as defined by the date-time production in IETF RFC 3339 [4].
Uri	String formatted according to IETF RFC 3986 [5].
Boolean	The Boolean is a data type having two values (true and false).
MacAddress	A MAC address. Representation: string that consists of groups of two hexadecimal digits, separated by hyphens or colons.
IpAddress	An IPV4 or IPV6 address. Representation: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.
Version	A Version. Representation: string of variable length.
String	A string as defined in IETF RFC 8259 [12].
Number	A number as defined in IETF RFC 8259 [12].

4.4.2.3 Void

4.5 Authorization of API requests and notifications

4.5.1 Introduction

The ETSI NFV MANO APIs are only allowed to be accessed by authorized consumers. Handling of authorization differs between making an API call and sending a notification. In the former case, OAuth 2.0 is used. In the latter case, OAuth 2.0 or HTTP Basic authentication is used, and the flows differ from those used in the former case. Alternatively, a solution based on public/private key pairs as authentication alternative to client identifier/password is also allowed.

The following terms (set in italics below) are used as defined by IETF RFC 6749 [10]: *client, resource server, authorization server, token endpoint, access token*. The description below is based on the "client credentials" grant type as defined by IETF RFC 6749 [10].

For API calls, the producer functional block of an API in NFV terms corresponds to the "*resource server*", and the consumer functional block of an API corresponds to the "*client*" as defined by IETF RFC 6749 [10]. For sending a notification, these roles are reversed: The producer (notification sender) corresponds to the "*client*", and the consumer (notification receiver) corresponds to the "*resource server*".

Before invoking an HTTP method on a REST resource provided by a *resource server*, a functional block (referred to as "*client*" from now on) first obtains authorization from another functional block fulfilling the role of the "*authorization server*". The present document makes no assumption about which functional block in the architecture plays the role of the *authorization server*. It is however assumed that the address of the *token endpoint* exposed by the *authorization server* and further specified in the clauses below is provisioned to the *client* together with additional authorization-related configuration parameters, such as valid client credentials. The *client* requests an *access token* from the *token endpoint*. As part of the request, it authenticates towards the *authorization server* by presenting its client credentials, consisting of client identifier and client password. The *authorization server* responds with an *access token* which the *client* will present to the *resource server* with every HTTP method invocation. An *access token* represents a particular access right (defining the particular set of protected resources to access in a particular manner) with a defined duration. The token is opaque to the *client*, and can typically be used by the *authorization server* and the *resource server* as an identifier to retrieve authorization information, such as information that identifies the client, its role and access rights. An *access token* expires after a certain time, or can be revoked. If that happens, the *client* can try to obtain a new *access token* from the *authorization server*.

In order to ensure that no third party can eavesdrop on sensitive information such as client credentials or access tokens, HTTP over TLS is used to protect the transport. If mutual authentication using TLS protocol is used, then the producer/server is authenticated to the consumer/client, but also the consumer/client is authenticated by the producer/server at the same time. To facilitate this mutual authentication, the server shall request a client certificate. This can be done as described in IETF RFC 5246 [7], including the optional CertificateRequest from server to client.

HTTP over TLS enables authorization on TLS certificates as an alternative to a token-based approach.

4.5.2 Flows (informative)

4.5.2.0 General

This clause outlines several alternative methods for authentication and authorization. Clause 4.5.2.1 presents an approach for authorizing API requests using OAuth 2.0 access tokens. Clause 4.5.2.1a describes an alternative method for authorization of API requests using TLS certificates. Clauses 4.5.2.2 and 4.5.2.3 outline a method to authorize notifications using basic authentication and OAuth2.0based approaches respectively. Finally, authorization of notifications using TLS certificates is presented in clause 4.5.2.4.

4.5.2.1 Authorization of API requests using OAuth 2.0 access tokens

The flow below illustrates the authorization of API requests that the API consumer sends to the API producer.

NOTE 1: Typical choices for the implementation of the authorization server include the authorization server as a component of the API producer, or as an external component.

Preconditions:

- Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.1-1.
- Authorization server is configured with the authorization policy and access rights against the client credentials.

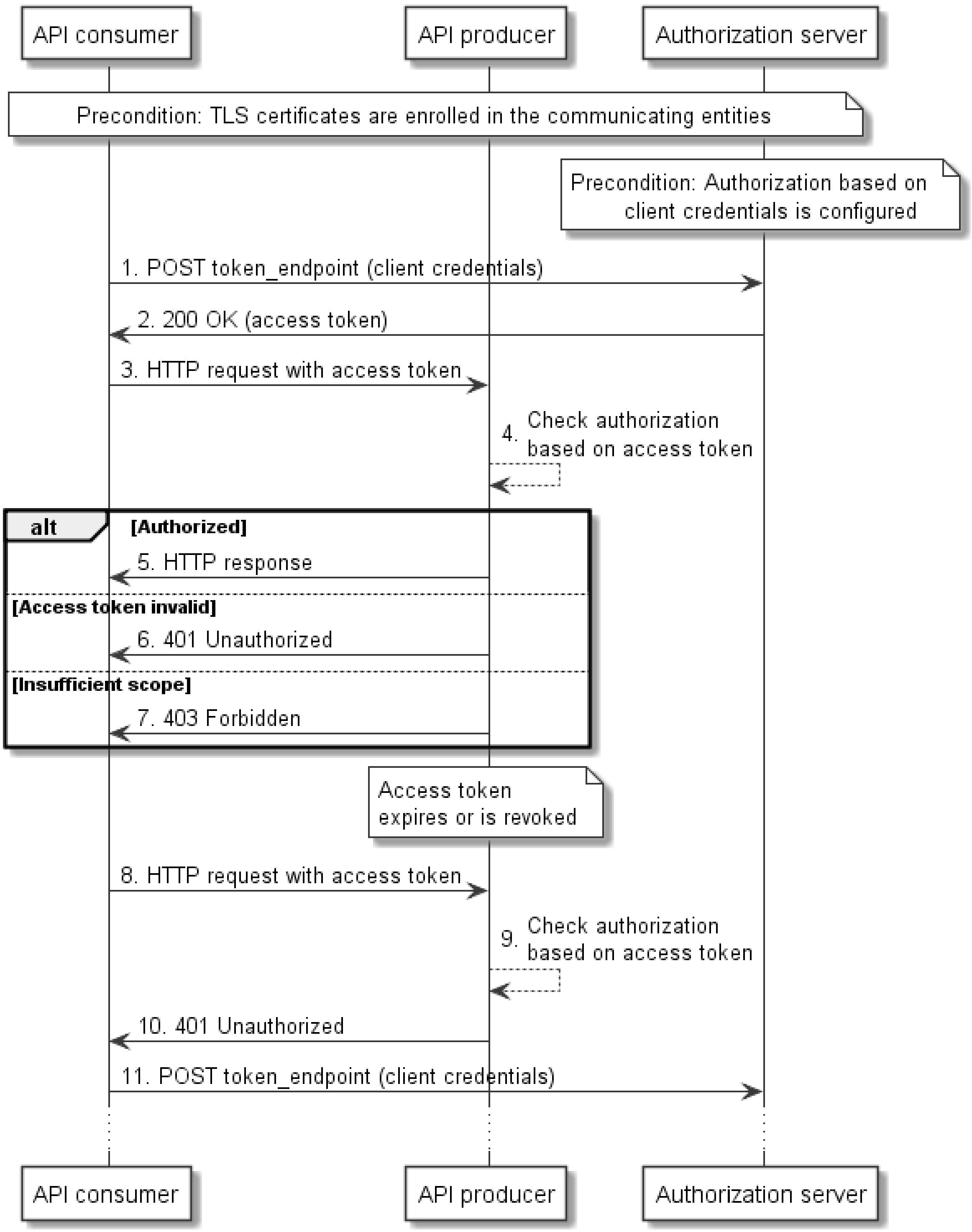
API consumer
API producer
Authorization server
Precondition: TLS certificates are enrolled in the communicating entities
Precondition: Authorization based on client credentials is configured
1. POST token_endpoint (client credentials)
2. 200 OK (access token)
3. HTTP request with access token
4. Check authorization based on access token
alt
[Authorized]
5. HTTP response
[Access token invalid]
6. 401 Unauthorized
[Insufficient scope]
7. 403 Forbidden
Access token expires or is revoked
8. HTTP request with access token
9. Check authorization based on access token
10. 401 Unauthorized
11. POST token_endpoint (client credentials)
API consumer
API producer
Authorization server

Figure 4.5.2.1-1: Authorization of API requests using OAuth 2.0 access tokens

The flow consists of the following steps:

1) To obtain an access token, the API consumer sends a POST request to the token endpoint of the authorization server and includes its client credentials.

2) The authorization server responds to the API consumer with an access token, and possibly additional information such as expiry time.

3) The API consumer sends an HTTP request to a resource provided by the API producer and includes the received access token.

4) The API producer checks the token for validity. This assumes that it has received information about the valid access tokens, and additional related information (e.g. time of validity, client identity, client access rights) from the authorization server. Such exchange is outside the scope of the present document, and assumed to be trivial if deployments choose to include the authorization server as a component into the API producer.

5) In case the token is valid and refers to access rights that allow accessing the actual resource with the actual request and its parameters, the API producer returns the HTTP response.

6) In case the token is invalid or expired, the API producer returns a "401 Unauthorized" response.

7) In case the access rights are insufficient to access the resource or to use the parameters, the API producer returns a "403 Forbidden" response.

8) The API consumer sends an HTTP request to the API producer and includes in the request the access token.

9) The API producer checks the token for validity, and establishes that it has expired, or has been revoked by the authorization server using means outside the scope of the present document.

10) The API producer responds with a "401 Unauthorized" response, indicating that the access token is invalid.

11) The API consumer attempts to obtain a new access token, as defined in step 3. This may eventually succeed or fail, depending on whether access is allowed for that API consumer any longer.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.1a Authorization of API requests using TLS certificates

As an alternative to the authorization using OAuth 2.0 access tokens, authentication and authorization is defined herein based on TLS certificates, applying the IETF RFC 5246 [7]. To facilitate mutual authentication during TLS tunnel setup process, the server requests a client certificate as described in section 7.4.4 in IETF RFC 5246 [7].

Preconditions:

- Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.1a-1.
- Authorization server is configured with the authorization policy and access rights against the certificates.

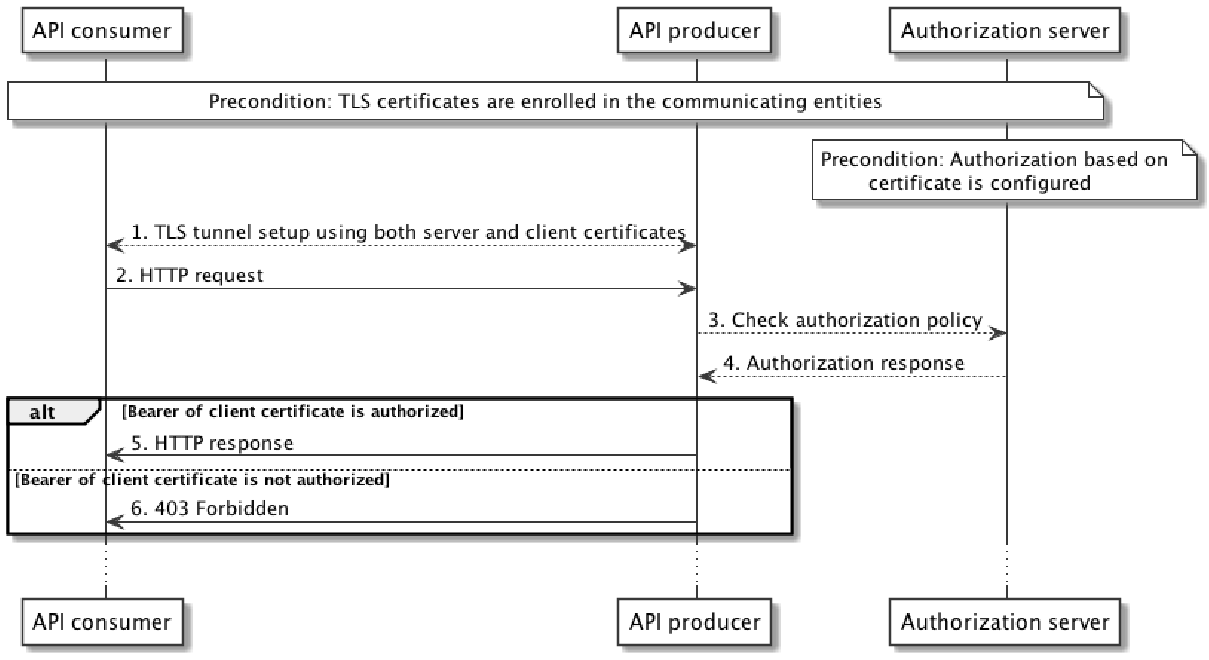
API consumer
API producer
Authorization server
Precondition: TLS certificates are enrolled in the communicating entities
Precondition: Authorization based on certificate is configured
1. TLS tunnel setup using both server and client certificates
2. HTTP request
3. Check authorization policy
4. Authorization response
alt
[Bearer of client certificate is authorized]
5. HTTP response
[Bearer of client certificate is not authorized]
6. 403 Forbidden

Figure 4.5.2.1a-1: Authorization of API requests using TLS certificates

The flow consists of the following steps:

1) The API consumer initiates the TLS tunnel setup process with the API producer. During the tunnel setup process the API producer sends its certificate to API consumer and obtains the certificate from the API consumer by including the CertificateRequest message specified in IETF RFC 5246 [7]. This ensures the mutual authentication between the consumer and the producer.

2) API consumer further sends the HTTP request for a resource over the TLS tunnel.

3) API producer now checks for the authorization information from the authorization server based on the API consumer client certificate.

4) Authorization server checks its policy and sends the response to the API producer.

5) If the API consumer is authorized, then the API producer sends the response related to the requested resource.

6) If the API consumer is unauthorized, then the API producer sends "403 Forbidden" response to the API consumer.

NOTE 1: Step 3 and 4 are outside the scope of the present document. However, typical implementations can use the certificates in such a way that the API producer verifies the certificate of the API consumer and extracts the subject name from the certificate. This information will be sent to the authorization server in order to check the authorization. In a response, the authorization server will send the associated client profile that contains the access rights.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

NOTE 3: Authorization based on TLS certificates assumes the existence of a trust relationship between the API producer and the authorization server. The authorization server has no direct communications with the API consumer and thus cannot authenticate it but relies on the API producer to perform this authentication.

4.5.2.2 Authorization of notifications using the HTTP Basic authentication scheme

Figure 4.5.2.2-1 illustrates the authorization of notifications that the API producer sends to the API consumer based on the HTTP Basic authentication scheme (see IETF RFC 7617 [18]). In this flow, no authorization server is needed.

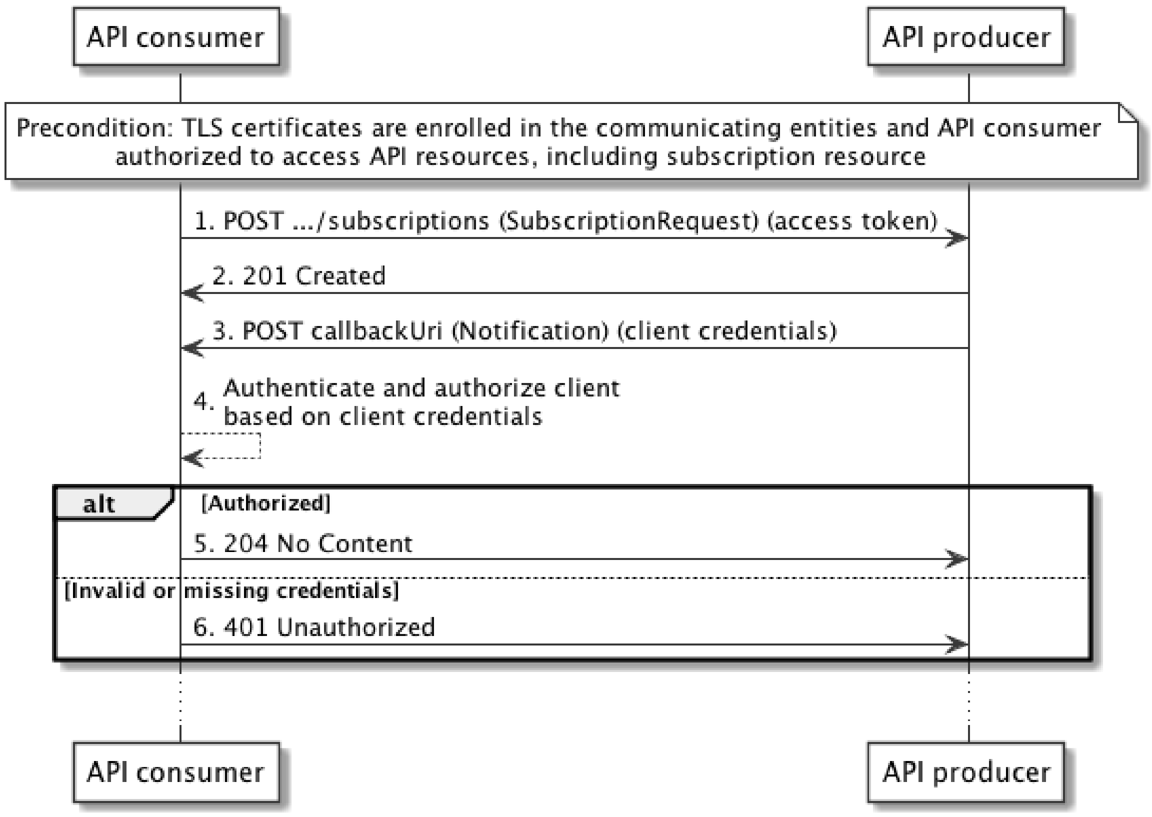
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities and API consumer authorized to access API resources, including subscription resource
1. POST .../subscriptions (SubscriptionRequest) (access token)
2. 201 Created
3. POST callbackUri (Notification) (client credentials)
4. Authenticate and authorize client based on client credentials
alt
[Authorized]
5. 204 No Content
[Invalid or missing credentials]
6. 401 Unauthorized
API consumer
API producer

Figure 4.5.2.2-1: Authorization of notifications using the HTTP Basic authentication scheme

It is a precondition for this flow that the API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1. Additionally, to ensure secure communication, it is a precondition that the TLS certificates are enrolled in the communicating entities.

The flow consists of the following steps:

1) The API consumer sends a request to create a new subscription resource to the API producer and includes in the request a valid access token to prove that it is authorized to access the API. Also, it includes in the subscription client credentials that the API producer can use to authenticate towards the API consumer when subsequently sending notifications. Note that these credentials are typically different from the client credentials used in the flow in clause 4.5.2.1.

2) The API producer creates the subscription resource and responds with "201 Created".

3) The API producer sends an HTTP POST request with a notification to the callback URI registered by the API consumer during subscription, and includes the client credential in the request to authenticate.

4) The API consumer checks the credentials against the information it has sent in step 1.

5) In case the credentials are valid, the API consumer returns a "204 No Content" HTTP response to indicate successful delivery of the notification.

6) In case the credentials are invalid, the API consumer returns a "401 Unauthorized" response.

NOTE: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.3 Authorization of notifications using OAuth 2.0 access tokens

The flow below illustrates the authorization of notifications that the API producer sends to the API consumer using OAuth 2.0. In this flow, the authorization server can be a different entity than the authorization server in clause 4.5.2.1.

NOTE 1: Typical choices for the implementation of the authorization server include the authorization server as a component of the API consumer, or as an external component.

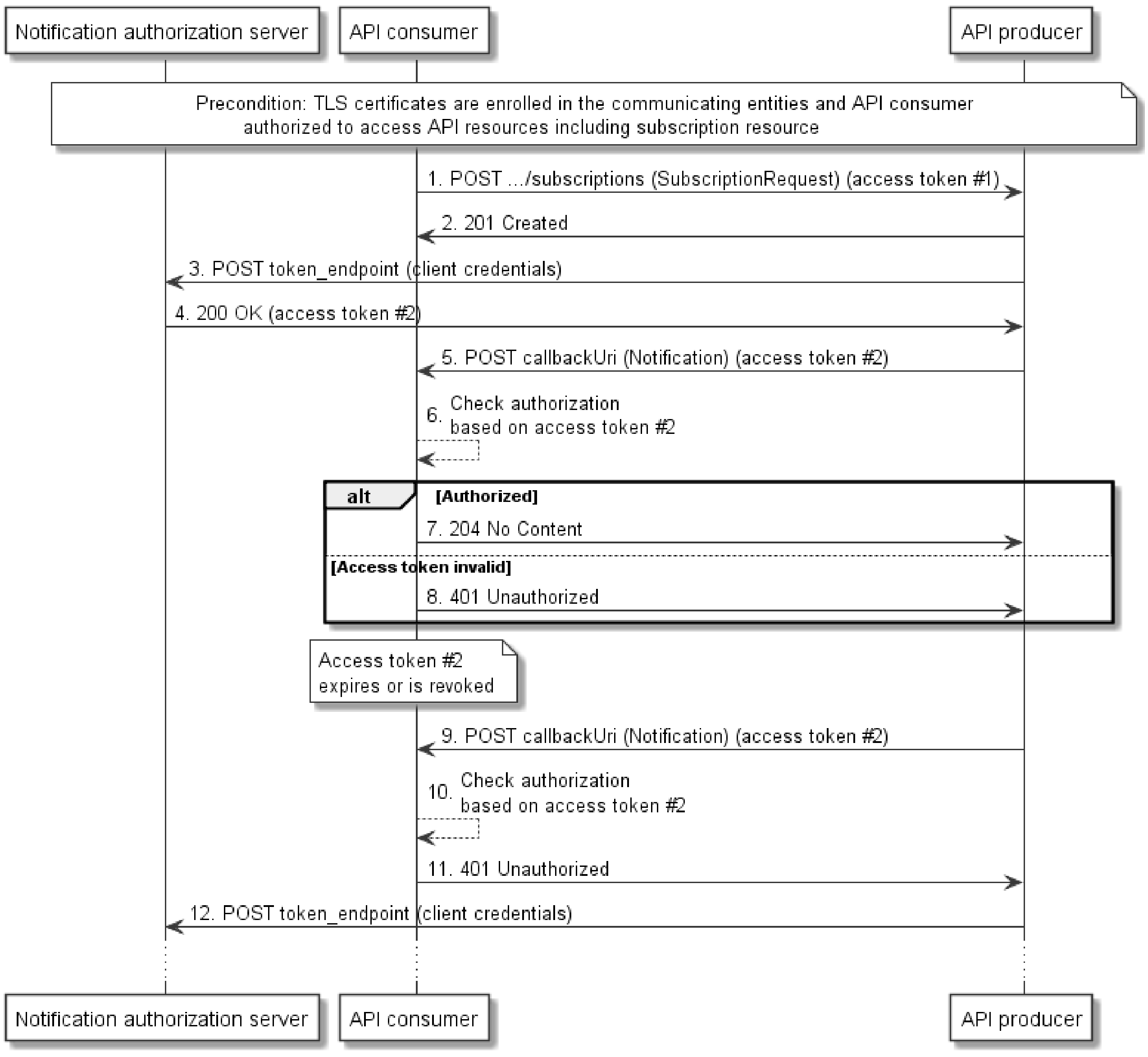
Notification authorization server
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities and API consumer authorized to access API resources including subscription resource
1. POST .../subscriptions (SubscriptionRequest) (access token #1)
2. 201 Created
3. POST token_endpoint (client credentials)
4. 200 OK (access token #2)
5. POST callbackUri (Notification) (access token #2)
6. Check authorization based on access token #2
alt
[Authorized]
7. 204 No Content
[Access token invalid]
8. 401 Unauthorized
Access token #2 expires or is revoked
9. POST callbackUri (Notification) (access token #2)
10. Check authorization based on access token #2
11. 401 Unauthorized
12. POST token_endpoint (client credentials)

Figure 4.5.2.3-1: Authorization of notifications using OAuth 2.0

It is a precondition for this flow that the API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1. Additionally, to ensure secure communication, it is a precondition that the TLS certificates are enrolled in the communicating entities.

The flow consists of the following steps:

1) The API consumer sends a request to create a new subscription resource to the API producer and includes in the request a valid access token #1 to prove that it is authorized to access the API. Also, it includes in the subscription request parameters that the API producer can use to obtain authorization to send notifications to the API consumer, such as client credentials and a token endpoint. Note that these are typically different from the credentials and token endpoint used in the flow in clause 4.5.2.1.

2) The API producer creates the subscription resource and responds with "201 Created".

3) Subsequently, and prior to sending any notification to the API consumer, the API producer obtains authorization to do so by requesting an access token from the authorization server, using the end point and notification client credentials that were sent in the subscription request, or provisioned otherwise.

4) The authorization server responds to the API producer with an access token, hereafter called access token #2, and possibly additional information such as expiry time.

5) The API producer sends an HTTP POST request with a notification to the callback URI registered by the API consumer during subscription, and includes the received access token #2.

6) The API consumer checks the token for validity. This assumes that it has received information about the valid access tokens, and additional related information (e.g. time of validity, client identity, client access rights) from the authorization server. Such exchange is outside the scope of the present document, and assumed to be trivial if deployments choose to include the authorization server as a component into the API consumer.

7) In case the token #2 is valid, the API consumer returns a "204 No Content" HTTP response to indicate successful delivery of the notification.

8) In case the token #2 is invalid or expired, the API consumer returns a "401 Unauthorized" response.

9) The API producer sends another notification in an HTTP POST request to the API consumer and includes in the request the access token #2.

10) The API consumer checks the token #2 for validity, and establishes that it has expired, or has been revoked by the authorization server using means outside the scope of the present document.

11) The API consumer responds with a "401 Unauthorized" response, indicating that the access token #2 is invalid.

12) The API producer attempts to obtain a new access token. This may eventually succeed or fail, depending on whether access is allowed for that API producer any longer.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.4 Authorization of notifications using TLS certificates

The flow in figure 4.5.2.4-1 illustrates the authorization of notifications that the API producer sends to the API consumer using TLS certificates.

Preconditions:

- Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.4-1.
- The API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1 or 4.5.2.1a.

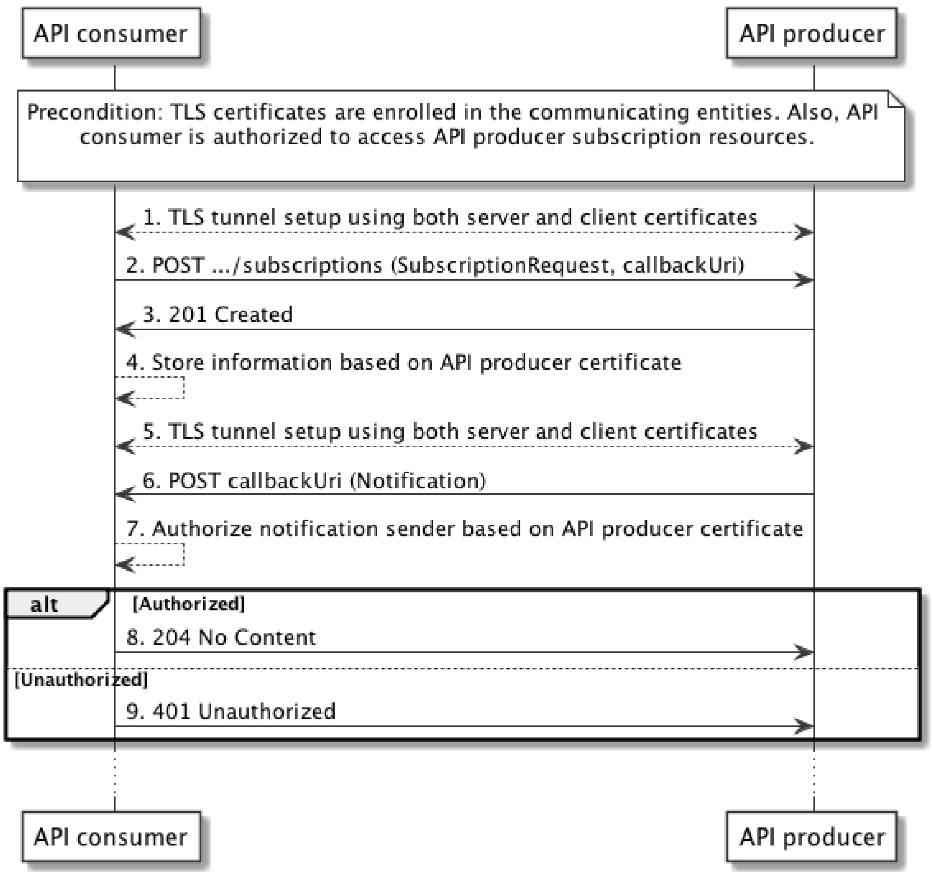
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities. Also, API consumer is authorized to access API producer subscription resources.
1. TLS tunnel setup using both server and client certificates
2. POST .../subscriptions (SubscriptionRequest, callbackUri)
3. 201 Created
4. Store information based on API producer certificate
5. TLS tunnel setup using both server and client certificates
6. POST callbackUri (Notification)
7. Authorize notification sender based on API producer certificate
alt
[Authorized]
8. 204 No Content
[Unauthorized]
9. 401 Unauthorized
API consumer
API producer

Figure 4.5.2.4-1: Authorization of notifications using TLS certificates

The flow consists of the following steps:

1) The API consumer initiates the TLS tunnel setup process with the API producer. During the tunnel setup process the API producer obtains the certificate from the API consumer. This ensures the mutual authentication between the consumer and the producer.

2) The API consumer sends a request to create a new subscription resource to the API producer. The API producer can authenticate and authorize this request based on the API consumer certificate as illustrated in clause 4.5.2.1a. The request also includes the callbackURI where the notification will be sent in future.

3) The API producer creates the subscription resource and responds with "201 Created".

4) The API consumer now stores the relevant information of the API producer's certificate in association with the requested notification subscription.

5) The API producer initiates the TLS tunnel with the API consumer whenever there is a notification to send. During the tunnel setup process the API consumer sends its certificate to API producer and obtains the client certificate from the API producer. This ensures the mutual authentication between the consumer and the producer.

6) The API producer sends the notification over the established TLS tunnel.

7) API consumer can now verify whether this sender is allowed to send this notification by matching the sender's certificate information with the previously stored information at step 4.

8) In case is the API producer is authorized to send a notification, then the API consumer sends a "204 No Content" response to indicate successful delivery of the notification.

9) In case if the API producer is not authorized to send a notification, the API consumer returns a "403 Forbidden" response.

NOTE 1: Step 4 and 7 are outside the scope of the present document. However, typical implementation can use the certificates in such a way that the API consumer verifies the certificate of the API producer and extract subject name from the certificate. This information is used in order to check the authorization at the API consumer.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

NOTE 3: It is assumed that the API producer uses the same certificate for both the client and server role.

4.5.3 Specification

4.5.3.1 Introduction

OAuth 2.0 provides a framework for authorization of web applications that has multiple modes and options. This clause profiles the framework for use in the context of the Or-Vnfm reference point. Clause 4.5.3.2 specifies the general mechanism. Two different uses of the general mechanism, actually for API requests and for sending notifications, are defined in clauses 4.5.3.3 and 4.5.3.4.

4.5.3.2 General mechanism

For all requests to an API defined in the present document, and for all notifications sent via such an API, authorization as defined below shall be used. Requests and notifications without authorization credentials shall be rejected.

To allow the *client* to obtain an access token, the *authorization server* shall expose a *token endpoint* that shall comply with the provisions defined by the OAuth 2.0 specification for the *client credentials* grant type (see IETF RFC 6749 [10]). A *client* shall use the access token request and response according to this grant type, as defined by IETF RFC 6749 [10], to obtain an *access token* for access to the REST resources defined by the present document. The content of the *access token* is out of the scope of the present document; however, it shall not be possible for an attacker to easily guess it. The *access token* shall be a string. The set of allowed characters is defined in IETF RFC 6749 [10].

A *client* that invokes an HTTP request towards a resource defined by one of the APIs of the present document shall include the *access token* as a bearer token in every HTTP method in the "Authorization" HTTP header, as defined by IETF RFC 6750 [11]. A *resource server* that receives an HTTP request with an invalid *access token*, or without an *access token,* shall reject the request, and shall signal the error in the HTTP response according to the provisions for the error codes and the "WWW-Authenticate" response HTTP header as defined by IETF RFC 6750 [11].

A *client* that receives a rejection of an *access token* may obtain a new *access token* from the *token endpoint* of the *authorization server*, and retry the request.

As an alternative to OAuth 2.0 access tokens, certificates, as defined by TLS 1.2 in IETF RFC 5246 [7], can be used to facilitate the authentication and authorization between client and the server.

4.5.3.3 Authorizing API requests

A consumer of an API that wishes to issue HTTP requests towards resources provided by that API shall act as a *client* according to clause 4.5.3.2 to obtain an access token, and shall include this access token in every HTTP request, as defined in clause 4.5.3.2. The respective API producer shall act as a *resource server* as defined in clause 4.5.3.2.

Alternatively, API requests can be authorized based on TLS certificates.

These two different alternatives are listed in the following:

1) API consumer passes access token when accessing a resource provided by API producer. API producer checks authorization based on access token. Access token can be obtained from the authorization server based on client ID and password.

2) API consumer accesses a resource provided by API producer using TLS tunnel where both server and client certificates are used to establish the secure tunnel. API producer checks authorization based on client's TLS certificate. The client's TLS certificate is obtained during the TLS handshake.

4.5.3.4 Authorizing the sending of notifications

The procedure defined in clause 4.5.2 allows an API consumer to obtain authorization to perform API requests towards the API producer, including subscription requests. For sending the actual notifications matching a subscription, the API producer needs to obtain separate authorization to actually *send* the notification to the API consumer.

If an API consumer requires the API producer to authorize for sending notifications to that API consumer, it shall include in the subscription request a data structure that defines the authorization requirements, as defined in table 4.5.3.4-1.

Table 4.5.3.4-1: Definition of the SubscriptionAuthentication data type

Attribute name	Data type	Cardinality	Description
authType	Enum (inlined)	1..N	Defines the types of Authentication / Authorization which the API consumer is willing to accept when receiving a notification. Permitted values: - BASIC: In every HTTP request to the notification endpoint, use HTTP Basic authentication with the client credentials. - OAUTH2_CLIENT_CREDENTIALS: In every HTTP request to the notification endpoint, use an OAuth 2.0 Bearer token, obtained using the client credentials grant type. - TLS_CERT: Every HTTP request to the notification endpoint is sent over a mutually authenticated TLS session, i.e. not only the server is authenticated, but also the client is authenticated during the TLS tunnel setup.
paramsBasic	Structure (inlined)	0..1	Parameters for authentication/authorization using BASIC. Shall be present if authType is "BASIC" and the contained information has not been provisioned out of band. Shall be absent otherwise.
>userName	String	0..1	Username to be used in HTTP Basic authentication. Shall be present if it has not been provisioned out of band.
>password	String	0..1	Password to be used in HTTP Basic authentication. Shall be present if it has not been provisioned out of band.
paramsOauth2ClientCredentials	Structure (inlined)	0..1	Parameters for authentication/authorization using OAUTH2_CLIENT_CREDENTIALS. Shall be present if authType is "OAUTH2_CLIENT_CREDENTIALS" and the contained information has not been provisioned out of band. Shall be absent otherwise.
>clientId	String	0..1	Client identifier to be used in the access token request of the OAuth 2.0 client credentials grant type. Shall be present if it has not been provisioned out of band. See note.

Attribute name	Data type	Cardinality	Description
>clientPassword	String	0..1	Client password to be used in the access token request of the OAuth 2.0 client credentials grant type. Shall be present if it has not been provisioned out of band. See note.
>tokenEndpoint	Uri	0..1	The *token endpoint* from which the access token can be obtained. Shall be present if it has not been provisioned out of band.
NOTE: The clientId and clientPassword passed in a subscription shall not be the same as the clientId and clientPassword that are used to obtain authorization for API requests. Client credentials may differ between subscriptions. The value of clientPassword should be generated by a random process.			

If the value of "authType" is "OAUTH2_CLIENT_CREDENTIALS":

- The API producer shall, prior to sending any notification, obtain an *access token* from the *token endpoint* using the OAuth 2.0 client credentials grant type as defined in IETF RFC 6749 [10]. The API consumer should include expiry information with the token response.
- The API producer shall include that *access token* as a Bearer token in every POST request that sends a notification (according to IETF RFC 6750 [11]).
- If the *access token* is expired, the API consumer shall reject the notification. In that case, the API producer shall obtain a new *access token*, and repeat sending the notification.
- If the token expiry time is known to the API producer, it may obtain proactively a new access token.

If the value of "authType" is "BASIC":

- The API producer shall pass its client credentials in every POST request that sends a notification, as defined in IETF RFC 7617 [18].

If the value of "authType" is "TLS_CERT":

- The API producer (client) shall use its TLS certificate to create a mutually authenticated TLS session with the API consumer (server) and further the API consumer will do the authorization based on the API producer's certificate.

4.5.3.5 Client roles

An *access token* allows the API producer to identify information about the *client* that has obtained the access token, such as client identity, client role or client access rights. By having this property, *access tokens* can be used as a means to distinguish between different roles (and consequently different access rights) to the same set of resources.

The mechanism for this works as follows: By means out of scope of the present document, the role of the client identified by a particular client identifier is provisioned to the authorization server. When that client obtains an access token, it sends its client identifier and client password to the authorization server. The authorization sever can obtain the role of the client by evaluating the data that were provisioned for the client identifier, and associate that information to the access token. By means out of scope of the present document, that association is shared with the API producer. This enables the API producer to detect the role based on the access token.

In ETSI NFV, certain interfaces are exposed on multiple different reference points, i.e. the same interface is exposed to different consumer functional blocks. Depending on the consumer block that originates an HTTP request, not all resources / HTTP methods / request and parameters might be available. From the point of view of the producer functional block, this can be seen as consumers acting in different roles when accessing a particular interface, such as the VNF LCM interface.

Implementations may use the OAuth *access token* to differentiate between these cases, assuming that an *access token* can determine whether a consumer functional block acts in the role of the VNFM or the NFVO. This assumes that the role of the consumer functional block is bound to its client credentials. The means of creating this binding is out of scope of the present document (e.g. a configuration step or policy).

As an alternative mechanism, the client role can be bound to its certificate. The mechanism for this works as follows: By means out of scope of the present document, the client is identified by a particular client subject name that is extracted from its certificate. This subject name is then provided to the authorization server in order to get the associated role of that particular client. By means out of scope of the present document, the authorization server is preconfigured to have this association between the client subject name and the role.

4.5.3.6 Negotiation of the authorization method

4.5.3.6.1 Authorization of API requests

The following provisions apply to the support of the authorization methods defined in the present document for the authorization of API requests:

- The API producer shall support checking the authorization of API requests it receives based on an OAuth 2.0 access token, and should support checking the authorization of API requests it receives based on TLS certificates, as defined in clause 4.5.3.3.
- The API consumer shall support the authorization of API requests it sends by including an OAuth 2.0 bearer token in the request, and should support the authorization of API requests it sends by providing its client certificate to the API producer during TLS tunnel setup as defined in clause 4.5.3.3.

When performing and authorizing an API request, API consumer and API producer shall use the following procedure, illustrated in figure 4.5.3.6.1-1, to negotiate the authorization method to use if the API consumer supports both the authorization based on OAuth 2.0 and the authorization based on TLS certificates, and the API consumer leaves the choice of OAuth or TLS to the API producer.

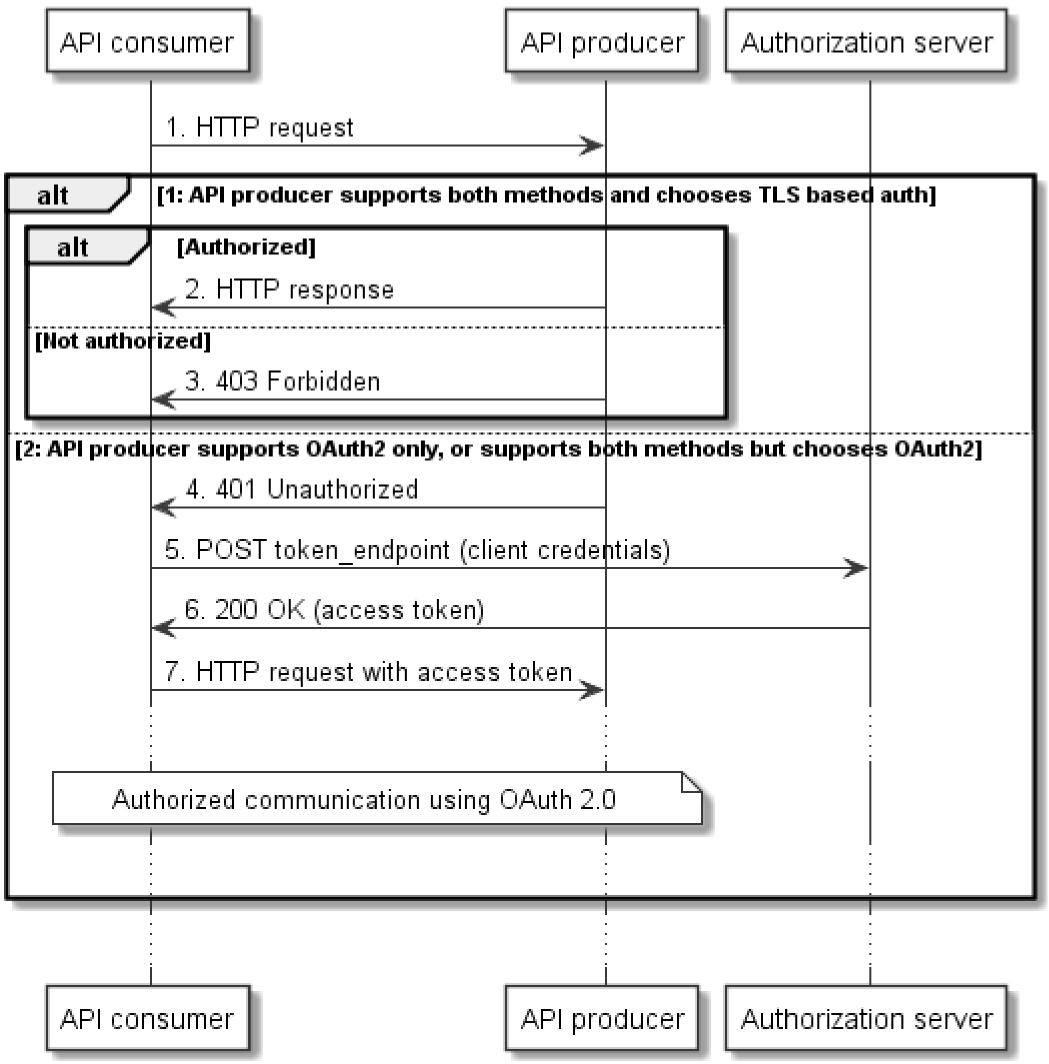
API consumer
API producer
Authorization server
1. HTTP request
alt [1: API producer supports both methods and chooses TLS based auth]
alt [Authorized]
2. HTTP response
[Not authorized]
3. 403 Forbidden
[2: API producer supports OAuth2 only, or supports both methods but chooses OAuth2]
4. 401 Unauthorized
5. POST token_endpoint (client credentials)
6. 200 OK (access token)
7. HTTP request with access token
Authorized communication using OAuth 2.0
API consumer
API producer
Authorization server

Figure 4.5.3.6.1-1: Negotiation of the authorization method to use for API requests

1) The API consumer shall send an HTTP request to the API producer without an access token.

2) If the API producer supports both authorization methods, chooses to use the method based on TLS certificates and the API consumer is authorized, it shall return the HTTP response to fulfil the request. Subsequent communication between API consumer and API producer shall use the authorization based on TLS credentials.

3) If the API producer supports both authorization methods, chooses to use the method based on TLS certificates and the API consumer is not authorized, it shall return a 403 Forbidden response.

4) If the API producer does not support the authorization based on TLS certificates, or chooses to use OAuth 2.0 for authorization, it shall return a 401 Unauthorized response to challenge the API consumer to use OAuth 2.0.

5) Once it has received the 401 Unauthorized response, the API consumer shall subsequently request an access token from the authorization server, according to clause 4.5.3.2.

6) The authorization server shall respond with an access token according to clause 4.5.3.2.

7) The API consumer shall subsequently retry the HTTP request with the access token included as a bearer token according to clause 4.5.3.2.

Subsequent authorized communication between API consumer and API producer shall take place as defined in clause 4.5.3.2 (see also the flow in clause 4.5.2.1, starting at step 4).

When performing and authorizing an API request and the API consumer does not support the method based on TLS certificates, or supports both methods but decides to use OAuth 2.0, no negotiation takes place, and the method defined in clause 4.5.3.2 shall be used (see also the flow in clause 4.5.2.1).

Table 4.5.3.6.1-1 illustrates the alternatives.

Table 4.5.3.6.1-1: Illustration of the alternatives

Consumer supports	Producer supports	Consumer request	Producer reaction
OAuth2	OAuth2	Consumer sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
OAuth2+TLS	OAuth2	If consumer intends to use OAuth2, it sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
		Otherwise, consumer sends the 1st HTTP request without access token.	Producer sends a 401 challenge to initiate use of OAuth2 (see note 2).
OAuth2	OAuth2+TLS	Consumer sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
OAuth2+TLS	OAuth2+TLS	If consumer intends to use OAuth2, it sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
		Otherwise, consumer sends the 1st HTTP request without access token.	If producer chooses OAuth2, it sends a 401 challenge (see note 2).
			Otherwise, if producer chooses TLS, it sends a success HTTP response if consumer is authorized (see note 3).
NOTE 1: This flow (OAuth2 method chosen by API consumer) is illustrated in figure 4.5.2.1-1. NOTE 2: This flow (OAuth2 method chosen by API producer) is illustrated in figure 4.5.3.6.1-1 as alternative 2. NOTE 3: This flow (TLS method chosen by API producer) is illustrated in figure 4.5.3.6.1-1 as alternative 1.			

4.5.3.6.2 Authorization of notification requests

The following provisions apply to the support of the authorization methods defined in the present document for the authorization of notification requests:

- The API consumer shall support checking the authorization of notification requests it receives based on an OAuth 2.0 access token as defined in clause 4.5.3.4. Further, the API producer should support checking the authorization of notification requests it receives based on HTTP Basic authentication, and based on TLS certificates as defined in clause 4.5.3.4.
- The API producer shall support the authorization of notification requests it sends by including an OAuth 2.0 bearer token in the request as defined in clause 4.5.3.4. Further, the API producer should support the authorization of notification requests it sends by providing credentials based on HTTP Basic authentication, and by providing its client certificate to the API producer during TLS tunnel setup as defined in clause 4.5.3.4.

When performing and authorizing a notification request, API consumer and API producer shall use the following procedure to negotiate the authorization method to use.

1) The API consumer shall signal in the subscription the authorization methods it accepts for notifications related to that particular subscription.

2) If none of the methods signalled is supported by the API producer, the API producer shall reject the subscription with "422 Unprocessable Entity", and shall include in the payload body a ProblemDetails structure which shall provide the reason for the rejection in the "details" attribute.

3) Otherwise, the API producer shall select one of the authorization methods that was signalled in the subscription, and shall use that method for the authorization of notifications it sends based on that subscription.

4.6 Version management

4.6.1 Version identifiers and parameters

4.6.1.1 Version identifiers

API version identifiers shall consist of 3 numerical fields, following a MAJOR.MINOR.PATCH pattern and the rules for Semantic Versioning [23] with the additional clarifications defined in clause 4.6.2. The fields in an API version identifier shall be separated by dots ".". The last field may be followed by one or more version parameters.

The MAJOR, MINOR and PATCH fields are defined in ETSI GS NFV-IFA 027 [20] for Semantic Versioning.

The {apiMajorVersion} segment of the URIs used by an API shall be set to the character "v" followed by value of the MAJOR field of the API version identifier.

EXAMPLE: ".../vnflcm/v1/

The full version identifier (including parameters) is used in ApiVersionInformation (see clause 4.4.1.13) and in version signalling (see clause 4.6.4). Furthermore, it also appears in the corresponding OpenAPI file (see annex D).

4.6.1.2 Version parameters

Version parameters are separated from the version identifier by a dash "-". Version parameters are separated by semicolons ";".

The present document defines the following version parameters:

- impl

The optional "impl" parameter identifies an implementation and a version of this implementation (e.g. implementation delivered by an open source community or a vendor). The OpenAPI specification referenced in annex D is also considered an implementation under this scheme. The "impl" parameter shall have the following structure: "impl:" <vendor>":"<product>":"<impl_version>, where:

- the <vendor> field shall be a string that contains either an IANA Enterprise Number assigned to that vendor, or an Internet domain name owned by that vendor;
- the <product> field shall contain a string identifying the product, chosen by the vendor;
- the <impl_version> field shall contain a number that defines the version of the implementation. Version numbers of subsequent implementations shall be monotonically increasing.

In case of the OpenAPI files provided by ETSI (see annex D), <vendor> shall be set to "etsi.org" and "product" shall be set to "ETSI_NFV_OpenAPI".

4.6.2 Rules for incrementing version identifier fields

4.6.2.1 General

In a REST API, versioning applies to the resources structure (URI structure, URI query parameters, and supported HTTP methods) and the payload body. Different criteria are applied to increment MAJOR, MINOR, and PATCH version fields for changes that affect the URI compared to changes that affect the payload body.

The fields of an API version identifier are incremented from a previous version to the current version according to the following rules:

- 1st field (MAJOR): This field is always incremented when one or more changes made to the resources structure defined in the present document break backward compatibility. This field is also incremented if one or more changes to at least one payload body defined in the present document break backward compatibility, unless that change is correcting an error.

NOTE 1: A change that corrects an error that would lead the API producer to always send an error response if a certain valid condition is met is not considered a non-backward compatible change, irrespective of the type of change. Indeed, compatibility between a new version and a previous version can only be assessed for a feature that is properly supported in the previous version.

NOTE 2: The 1st field (MAJOR) is kept equal to "1" in version 2.5.1 of the present document, recognizing that existing and emerging commercial implementations could be adversely affected by a change to the MAJOR version at this time.

- 2nd field (MINOR): This field is incremented if one or more technical changes (at least one of which is not an error correction) are made to the API specification in the present document API but none of them (apart from error corrections to the payload body) breaks backward compatibility. It is reset to zero if the MAJOR version identifier is changed.

- 3rd field (PATCH): This field is incremented if one or more error corrections that are visible in communication between API producer and API consumer are made on the API specification in the present document but none of them (apart from error corrections to the payload body) breaks backward compatibility. It is reset to zero if the MINOR version identifier is changed.

NOTE 3: All the aforementioned types of changes affect the corresponding OpenAPI specification (see annex D).

4.6.2.2 Examples of backward and non-backward compatible changes

Examples of backward compatible changes include:

- Adding a new resource
- Adding a new URI
- Supporting a new HTTP method for an existing resource
- Adding new optional URI query parameters
- Adding new optional attributes to a resource representation in a request
- Adding new attributes to a resource representation in a response or to a notification message
- Responding with a new status code of an error class
- Certain cardinality changes (see note 2)

NOTE 1: Whether attribute cardinality changes are backward compatible depends on the type of change. An example of a backward-compatible cardinality change include making an attribute in a response required (e.g. changing cardinality from 0..1 to 1).

Examples of non-backward compatible changes to the resources structure include:

- Removing a resource / URI
- Removing support for an HTTP method
- Changing a resource URI
- Adding new mandatory URI query parameters

Examples of non-backward compatible changes to the payload body include:

- Renaming an attribute in a resource representation
- Adding new mandatory attributes to a resource representation in a request
- Changing the data type of an attribute
- Certain cardinality changes (see note 2)

NOTE 2: Whether attribute cardinality changes are backward compatible depends on the type of change. Examples of non-backward compatible cardinality changes include decreasing the upper bound of a cardinality range for attributes sent by the client, changing the meaning of the default behavior associated to the absence of an attribute of cardinality 0..N, etc.

4.6.3 Version information retrieval

4.6.3.1 General

The API producer shall support the following dedicated URIs to enable API consumers to retrieve information about API versions supported by an API producer:

```
1. {apiRoot}/{apiName}/api_versions
2. {apiRoot}/{apiName}/{apiMajorVersion}/api_versions
```

To obtain information about the supported API versions, the API consumer shall send a GET request to a URI of one of above forms. The information contained in the GET response depends on the form of URI used in the GET request, as follows:

- If the first form is used, the GET response shall provide the list of supported versions for the API corresponding to the apiName indicated in the GET Request URI.
- If the second form is used, the GET response shall provide the list of supported versions for the API corresponding to the {apiName} and the {apiMajorVersion} indicated in the GET Request URI.

If the API producer receives a GET request:

- In case of success, the API producer shall return in the body of a 200 OK response a value of the ApiVersionInformation data type specified in clause 4.4.1.13.
- In case URI query parameters are provided, the API producer shall return a "400 Bad request" response as defined in clause 4.3.5.4.
- In other cases of failure, the API producer shall return appropriate error information as defined in clauses 4.3.5.4 and 4.3.5.5.

4.6.3.2 Resource structure and methods

Table 4.6.3.2-1 lists the individual resources defined for supporting API version information retrieval, and theapplicable HTTP method. The VNFM shall support responding to GET requests on the resources in table 4.6.3.2-1.

Table 4.6.3.2-1: Resources and methods overview for API version information retrieval

Resource name	Resource URI	HTTP Method	Meaning
API versions	/{apiName}/api_versions	GET	Version information associated to an API
API versions	/{apiName}/{apiMajorVersion}/api_versions	GET	Version information associated to a major version of an API

Figure 4.6.3.2-1 shows the "API versions" resources in the overall resource URI structure defined for all APIs.

The "API versions" resources, as defined in the present clause, are part of the overall resource URI structure of each API defined in the present document.

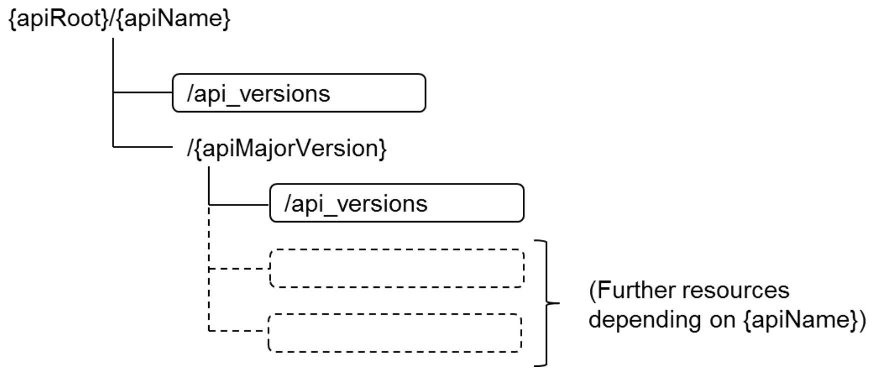

Figure 4.6.3.2-1: "API versions" resources

4.6.3.3 Resource: API versions

4.6.3.3.1 Description

There are two "API versions" resources defined for each API. The client can use these resources to obtain API version information.

4.6.3.3.2 Resource definition

The resource URI of each of the two "API versions" resources shall be of one of the following forms:

```
1. {apiRoot}/{apiName}/api_versions
2. {apiRoot}/{apiName}/{apiMajorVersion}/api_versions
```

These resources shall support the resource URI variables defined in table 4.6.3.3.2-1.

Table 4.6.3.3.2-1: Resource URI variables for these resources

Name	Definition
apiRoot	See clause 4.2
apiName	See clause 4.2
apiMajorVersion	See clause 4.2

4.6.3.3.3 Resource methods

4.6.3.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.2 GET

The GET method reads API version information. This method shall follow the provisions specified in table 4.6.3.3.3.2-1 for request and response data structures, and response codes. URI query parameters are not supported.

Table 4.6.3.3.3.2-1: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	ApiVersionInformation	1	200 OK	API version information was read successfully. The response body shall contain API version information, as defined in clause 4.4.1.13.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

4.6.3.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.4 Version signaling

The API consumer shall include the "Version" HTTP header (see IETF RFC 4229 [24]) in each HTTP request. The "Version" header shall contain the three version identifier fields (MAJOR.MINOR.PATCH) indicating the API version the API consumer intends to use. The "impl" version parameter may be provided, indicating the version of the API producer implementation that the API consumer intends to use.

The API producer shall support receiving and interpreting the "Version" HTTP header. The API producer shall include in the response the "Version" HTTP header signaling the used API version, including the "impl" version parameter if available. If the "impl" version parameter has been omitted in the request, the API producer shall use the combination of MAJOR, MINOR and PATCH as requested and the highest supported value for the "impl_version" field of the "impl" version parameter for that combination, if available.

NOTE: In case multiple versions and/or implementation versions are supported by an API producer, this allows the API consumer to request a particular version.

API consumers conforming to versions of the present document previous to version 2.5.1 omit this header. If the API producer receives a request without this header:

- If it supports the previous version 2.4.1 of the present document, it shall behave as defined in that document, and should indicate this by using MAJOR=1 and MINOR=1 and PATCH=0 in the "Version" HTTP header in the response.

- If it does not support any of the previous versions, it shall respond with a 400 Bad Request response and shall include in the response payload body a ProblemDetails structure providing more information on the cause of the error in the "detail" attribute.

If the API version signaled in the "Version" request header is not supported by the API producer, the API producer shall respond with a "406 Not Acceptable" error and shall include in the response payload body a ProblemDetails structure providing more information on the cause of the error in the "detail" attribute.

4.7 Handling of large query results

4.7.1 Description

If the response to a query to a container resource (i.e. a resource that contains child resources whose representations will be returned when responding to a GET request) will become so large that the response will adversely affect the performance of the server, the server either rejects the request with a 400 Bad Request response, or the server provides a paged response, i.e. it returns only a subset of the query result in the response, and also provides information how to obtain the remainder of the query result.

When returning a paged response, depending on the underlying storage organization, it might be problematic for the server to determine the actual size of the result; however, it is usually possible to determine whether there will be additional results returned when knowing, for the last entry in the returned page, the position in the overall query result or some other property that has ordering semantics. For example, the time of creation of a resource has such an ordering property. When using such an (implementation-specific) property, the API producer can correctly handle deletions of child resources that happen between sending the first page of the query result, and sending the next page. It cannot be guaranteed that child resources inserted between returning subsequent pages can be considered in the query result, however, it shall be guaranteed that this does not lead to skipping of entries that have existed prior to insertion.

At minimum, a paged response needs to contain information telling the API consumer that the response is paged, and how to obtain the next page of information. For that purpose, a link to obtain the next page is returned in an HTTP header, containing a parameter that is opaque to the API consumer, but that allows the API producer to determine the start of the next page.

NOTE: In the present document, this functionality is designed for overload protection only. Additional functionality, such as configuring the page size by the API consumer, determining the size of the overall query result or the number of pages, and determining the previous page, is left outside the scope of the present document.

4.7.2 Specification

4.7.2.1 Alternatives

For each container resource (i.e. a resource that contains child resources whose representations will be returned when responding to a GET request), the API producer shall support one of the following two behaviours specified below to handle the case that a response to a query (GET request) will become so large that the response will adversely affect performance:

1) Return an error response, as defined in clause 4.7.2.2.

2) Return the result in a paged manner, as defined in clause 4.7.2.3.

4.7.2.2 Error response

In this alternative, the server shall reject the request with a 400 Bad Request response, shall include the "ProblemDetails" payload body, and shall provide in the "detail" attribute more information about the error.

This error code indicates to the API consumer that with the given attribute-based filtering query (or absence thereof), the response would have been so big that performance is adversely affected. The client can obtain a query result by specifying a (more restrictive) attribute-based filtering query (see clause 4.3.2).

4.7.2.3 Paged response

In this alternative, the API producer shall provide a response that contains a first page (subset) of the results to the query, and shall include a LINK HTTP header (see IETF RFC 8288 [22]) with the "rel" attribute set to "next", which communicates a URI that allows to obtain the next page of results to the original query.

The API consumer can send a GET request to the URI communicated in the LINK header to obtain the next page of results. The response which returns that next page shall contain the LINK header to point to the next page, as specified above, unless there are no further pages available in which case the LINK header shall be omitted.

To allow the API producer to determine the start of the next page, the LINK header shall contain the URI query parameter "nextpage_opaque_marker" whose value is chosen by the API producer. This parameter has no meaning for the API consumer, but is echoed back by the API consumer to the API producer when requesting the next page. The URI in the link header may include further parameters, such as those passed in the original request.

The size of each page may be chosen by the API provider, and may vary from page to page. The maximum page size is determined by means outside the scope of the present document.

The response need not contain entries that correspond to child resources which were created after the original query was issued.

5 VNF Lifecycle Management interface

5.1 Description

This interface allows the NFVO to invoke VNF lifecycle management operations of VNF instances towards the VNFM, and to subscribe to notifications regarding VNF lifecycle changes provided by the VNFM.

The operations provided through this interface are:

- Create VNF Identifier
- Query VNF
- Modify VNF Information
- Delete VNF Identifier
- Instantiate VNF
- Scale VNF
- Scale VNF to Level
- Change VNF Flavour
- Terminate VNF
- Heal VNF
- Operate VNF
- Change external VNF connectivity
- Get Operation Status
- Subscribe
- Query Subscription Information
- Terminate Subscription

- Notify

This interface also enables to invoke error handling procedures (Retry, Rollback, Cancel, Fail) on the actual VNF lifecycle management operation occurrences, and API version information retrieval.

5.1a API version

For the VNF lifecycle management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

5.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnflcm" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 5.2-1 shows the overall resource URI structure defined for the VNF lifecycle management interface.

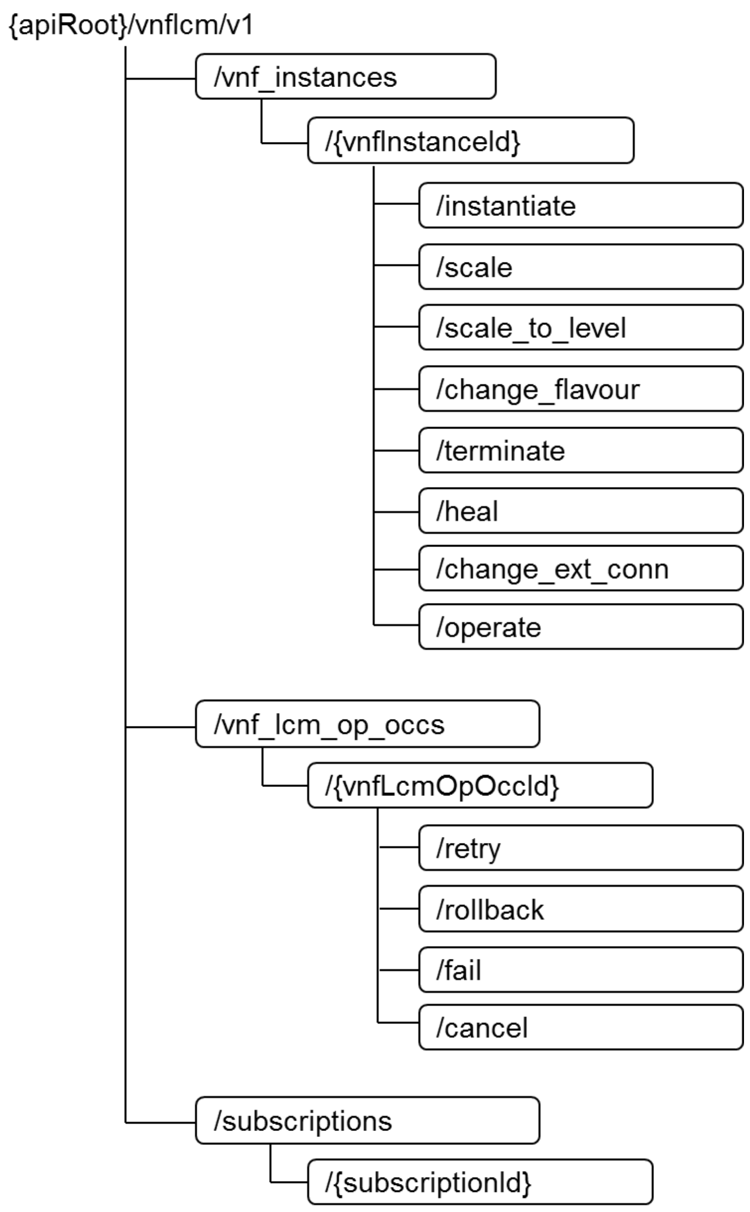

Figure 5.2-1: Resource URI structure of the VNF Lifecycle Management Interface

Table 5.2-1 lists the individual resources defined, and the applicable HTTP methods.

The VNFM shall support responding to requests for all HTTP methods on the resources in table 5.2-1 that are marked as "M" (mandatory) in the "Cat" column. The VNFM shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 5.2-1: Resources and methods overview of the VNF Lifecycle Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
VNF instances	/vnf_instances	GET	M	Query multiple VNF instances.
		POST	M	Create a VNF instance resource.
Individual VNF instance	/vnf_instances/{vnfInstanceId}	GET	M	Read an individual VNF instance resource.
		PATCH	M	Modify VNF instance information.
		DELETE	M	Delete VNF instance resource.
Instantiate VNF task	/vnf_instances/{vnfInstanceId}/instantiate	POST	M	Instantiate a VNF.
Scale VNF task	/vnf_instances/{vnfInstanceId}/scale	POST	M	Scale a VNF instance incrementally.
Scale VNF to Level task	/vnf_instances/{vnfInstanceId}/scale_to_level	POST	M	Scale a VNF instance to a target level.
Change VNF flavour task	/vnf_instances/{vnfInstanceId}/change_flavour	POST	M	Change the deployment flavour of a VNF instance.
Terminate VNF task	/vnf_instances/{vnfInstanceId}/terminate	POST	M	Terminate a VNF instance.
Heal VNF task	/vnf_instances/{vnfInstanceId}/heal	POST	M	Heal a VNF instance.
Operate VNF task	/vnf_instances/{vnfInstanceId}/operate	POST	M	Operate a VNF instance.
Change external VNF connectivity task	/vnf_instances/{vnfInstanceId}/change_ext_conn	POST	M	Change the external connectivity of a VNF instance.
VNF LCM operation occurrences	/vnf_lcm_op_occs	GET	M	Query multiple VNF lifecycle management operation occurrences.
Individual VNF LCM operation occurrence	/vnf_lcm_op_occs/{vnfLcmOpOccId}	GET	M	Read an individual VNF lifecycle management operation occurrence.
Retry operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/retry	POST	M	Retry a VNF lifecycle management operation occurrence.
Rollback operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback	POST	M	Rollback a VNF lifecycle management operation occurrence.
Fail operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail	POST	M	Mark a VNF lifecycle management operation occurrence as failed.
Cancel operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel	POST	M	Cancel a VNF lifecycle management operation occurrence.
Subscriptions	/subscriptions	POST	M	Subscribe to VNF lifecycle change notifications.
		GET	M	Query multiple subscriptions.
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription resource.
		DELETE	M	Terminate a subscription.
Notification endpoint	(client-provided)	POST	See note	Notify about VNF lifecycle change. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The VNFM shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the NFVO. If the NFVO supports invoking the POST method on the "Subscriptions" resource towards the VNFM, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

5.3 Sequence diagrams (informative)

5.3.1 Flow of the creation of a VNF instance resource

This clause describes the procedure for the creation of a VNF instance resource.

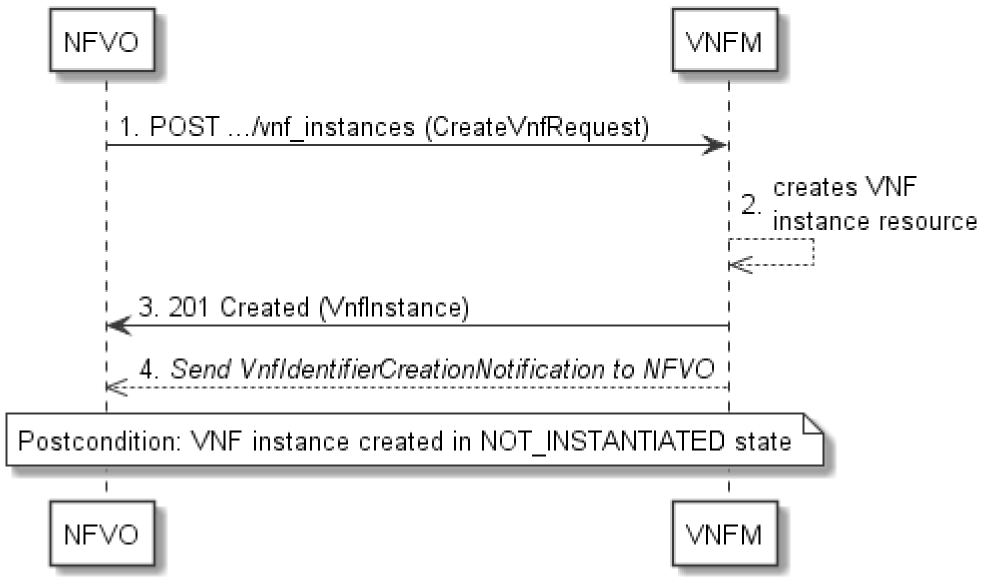
NFVO
VNFM
1. POST .../vnf_instances (CreateVnfRequest)
2. creates VNF instance resource
3. 201 Created (VnfInstance)
4. Send VnfIdentifierCreationNotification to NFVO
Postcondition: VNF instance created in NOT_INSTANTIATED state
NFVO
VNFM

Figure 5.3.1-1: Flow of the creation of a VNF instance resource

NOTE: Due to possible race conditions, the 201 response and the VnfIdentifierCreationNotification can arrive in any order at the NFVO.

The procedure consists of the following steps as illustrated in figure 5.3.1-1:

1) The NFVO sends a POST request to the "VNF Instances" resource including in the payload body a data structure of type "CreateVnfRequest".

2) The VNFM creates a new VNF instance resource in NOT_INSTANTIATED state, and the associated VNF instance identifier.

3) The VNFM returns a 201 Created response containing a representation of the VNF instance resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

4) The VNFM sends a VNF Identifier Creation Notification (see clause 5.3.9) to the NFVO to indicate the creation of the VNF instance resource and the associated VNF instance identifier.

Postcondition: Upon successful completion, the VNF instance resource has been created in "NOT_INSTANTIATED" state.

5.3.2 Flow of the deletion of a VNF instance resource

This clause describes the procedure for the deletion of a VNF instance resource.

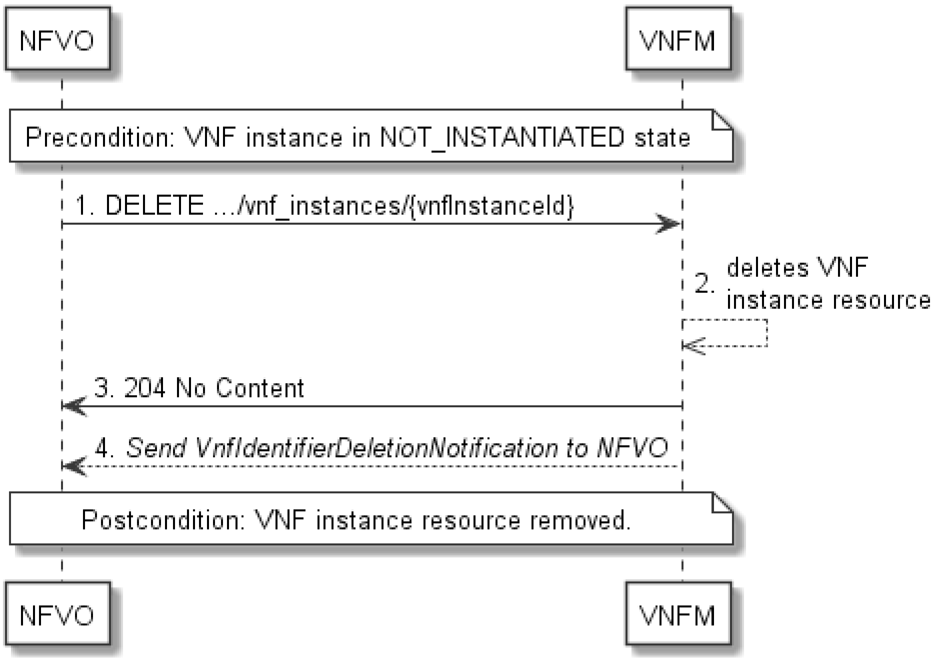
NFVO
VNFM
Precondition: VNF instance in NOT_INSTANTIATED state
1. DELETE .../vnf_instances/{vnfInstanceId}
2. deletes VNF instance resource
3. 204 No Content
4. Send VnfIdentifierDeletionNotification to NFVO
Postcondition: VNF instance resource removed.
NFVO
VNFM

Figure 5.3.2-1: Flow of the deletion of a VNF instance resource

NOTE: Due to possible race conditions, the 204 response and the VnfIdentifierDeletionNotification can arrive in any order at the NFVO.

Precondition: The resource representing the VNF instance to be deleted needs to be in NOT_INSTANTIATED state.

The procedure consists of the following steps as illustrated in figure 5.3.2-1:

1) NFVO sends a DELETE request to the "Individual VNF Instance" resource.
2) The VNFM deletes the VNF instance resource and the associated VNF instance identifier.
3) The VNFM returns a "204 No Content" response with an empty payload body.
4) The VNFM sends to the NFVO a VnfIdentifierDeletionNotification to indicate the deletion of the VNF instance resource and the associated VNF instance identifier.

Postcondition: The resource representing the VNF instance has been removed from the list of VNF instance resources.

Error handling: If the VNF instance is not in NOT_INSTANTIATED state, the VNFM rejects the deletion request.

5.3.3 Flow of VNF lifecycle management operations triggered by task resources

This clause describes the general sequence for VNF Lifecycle Management operations that operate on VNF instance resource and are triggered by task resources. The flows for these operations are very similar. The differences between the individual operations are covered in table 5.3.3-1.

This flow is applicable to the following operations:

- Instantiate VNF
- Scale VNF
- Scale VNF to Level

- Change VNF flavour
- Operate VNF
- Heal VNF
- Change external VNF connectivity
- Terminate VNF

Figure 5.3.3-1 illustrates the general lifecycle management flow. Placeholders in this flow allow for differentiating between the operations and are marked with double angular brackets "<<…>>".

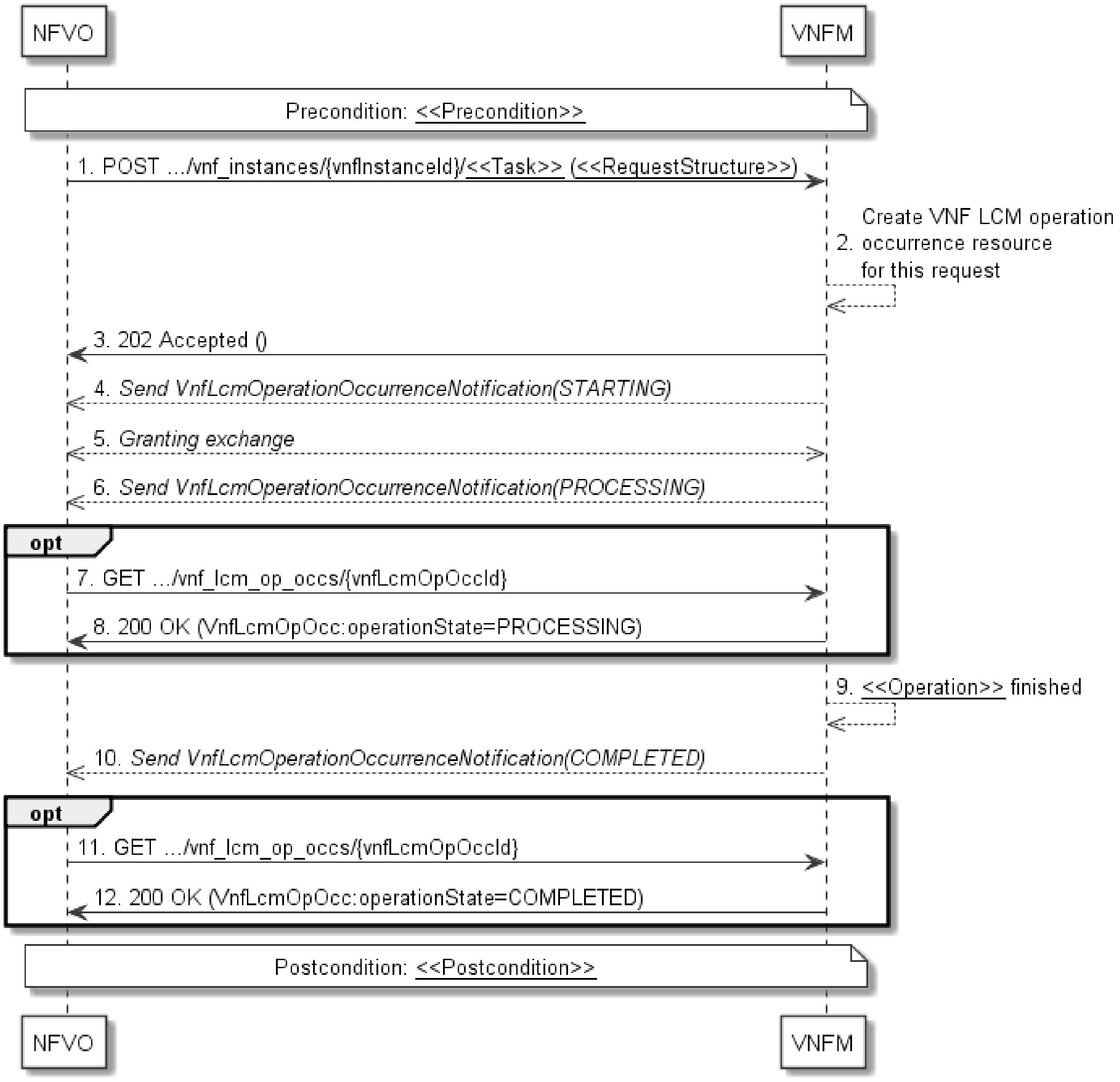
NFVO
VNFM
Precondition: <<Precondition>>
1. POST .../vnf_instances/{vnfInstanceId}/<<Task>> (<<RequestStructure>>)
2. Create VNF LCM operation occurrence resource for this request
3. 202 Accepted ()
4. Send VnfLcmOperationOccurrenceNotification(STARTING)
5. Granting exchange
6. Send VnfLcmOperationOccurrenceNotification(PROCESSING)
opt
7. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
8. 200 OK (VnfLcmOpOcc:operationState=PROCESSING)
9. <<Operation>> finished
10. Send VnfLcmOperationOccurrenceNotification(COMPLETED)
opt
11. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
12. 200 OK (VnfLcmOpOcc:operationState=COMPLETED)
Postcondition: <<Postcondition>>
NFVO
VNFM

Figure 5.3.3-1: General flow of VNF lifecycle management operations triggered by task resources

NOTE: Due to possible race conditions, the 202 response, the "STARTING" VnfLcmOperationOccurrenceNotification and the request of the Granting exchange can arrive in any order at the NFVO.

Precondition: The precondition depends on the actual operation and is described by the template parameter <<Precondition>> in table 5.3.3-1.

A VNF lifecycle operation, as illustrated in figure 5.3.3-1, consists of the following steps:

1) The NFVO sends a POST request to the <<Task>> resource that represents the lifecycle operation to be executed on the VNF instance, and includes in the payload body a data structure of type <<RequestStructure>>. The name <<Task>> of the task resource and the <<RequestStructure>> depend on the operation and are described in table 5.3.3-1.
2) The VNFM creates a "VNF LCM operation occurrence" resource for the request.
3) The VNFM returns a "202 Accepted" response with an empty payload body and a "Location" HTTP header that points to the new "VNF LCM operation occurrence" resource, i.e. it includes the URI of that resource which is "…/vnf_lcm_op_occs/{vnfLcmOpOccId}". See note.
4) The VNFM sends to the NFVO a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the start of the lifecycle management operation occurrence with the "STARTING" state. See note.
5) VNFM and NFVO exchange granting information (see VNF Lifecycle Operation Granting interface, clause 9.3). See note.
6) The VNFM sends to the NFVO a VNF lifecycle management operation occurrence notification (see clause 5.3.8) to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state.
7) If desired, the NFVO can poll the "VNF LCM operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF LCM operation occurrence.
8) In the response to that request, the VNFM returns to the NFVO information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".
9) The VNFM has finished the operation <<Operation>>.
10) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the completion of the lifecycle management operation occurrence with the success state "COMPLETED".
11) If desired, the NFVO can send a new GET request to the "VNF LCM operation occurrence" resource.
12) In the response to that request, the VNFM returns to the NFVO information about the result of the operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: The postcondition depends on the actual operation and is described by the template parameter <<Postcondition>> in table 5.3.3-1.

Error handling: If the VNF lifecycle management operation fails, error information is provided in the notification message that reports the erroneous completion of the procedure, and is also available in the resource that represents the actual VNF lifecycle management operation occurrence related to this VNF lifecycle management operation.

Table 5.3.3-1 defines how the flow described above is parameterized for the different VNF lifecycle management operations.

Table 5.3.3-1: Parameterization of the flow for different VNF lifecycle management operations

Operation	Precondition	Task	RequestStructure	Postcondition
Instantiate VNF	VNF instance created and in NOT_INSTANTIATED state	instantiate	InstantiateVnfRequest	VNF instance in INSTANTIATED state
Scale VNF	VNF instance in INSTANTIATED state	scale	ScaleVnfRequest	VNF instance still in INSTANTIATED state and VNF was scaled
Scale VNF to Level	VNF instance in INSTANTIATED state	scale_to_level	ScaleVnfToLevel Request	VNF instance still in INSTANTIATED state and VNF was scaled
Change VNF flavor	VNF instance in INSTANTIATED state	change_flavour	ChangeVnfFlavour Request	VNF instance still in INSTANTIATED state and VNF deployment flavour changed
Operate VNF	VNF instance in INSTANTIATED state	operate	OperateVnfRequest	VNF instance still in INSTANTIATED state and VNF operational state changed
Heal VNF	VNF instance in INSTANTIATED state	heal	HealVnfRequest	VNF instance still in INSTANTIATED state
Change external VNF connectivity	VNF instance in INSTANTIATED state	change_ext_conn	ChangeExtVnfConnectivityRequest	VNF instance still in INSTANTIATED state and external connectivity of the VNF is changed
Terminate VNF	VNF instance in INSTANTIATED state	terminate	TerminateVnfRequest	VNF instance in NOT_INSTANTIATED state

5.3.4 Flow of automatic invocation of VNF scaling and VNF healing

This clause describes the sequence for the automatic invocation of "Scale VNF", "Scale VNF to Level" and "Heal VNF" operations by the VNFM, also known as "auto-scale" and "auto-heal". The criteria based on which the VNFM decides when to invoke an automatic scaling or automatic healing are outside the scope of the present document and can include certain changes in monitoring parameters that are monitored by the VNFM by PM jobs or thresholds, changes in VNF indicator values that are polled by the VNFM or that are reported to the VNFM via "VnfIndicatorValueChangeNotification" messages. Auto-scaling and auto-healing can be enabled and disabled by the NFVO by modifying the appropriate "isAutoscaleEnabled" and "isAutohealEnabled" configurable properties of the VNF using the sequence flow according to clause 5.3.6.

This flow is applicable to the automatic invocation of the following operations:

- Scale VNF
- Scale VNF to Level
- Heal VNF

Figure 5.3.4-1 illustrates the flow.

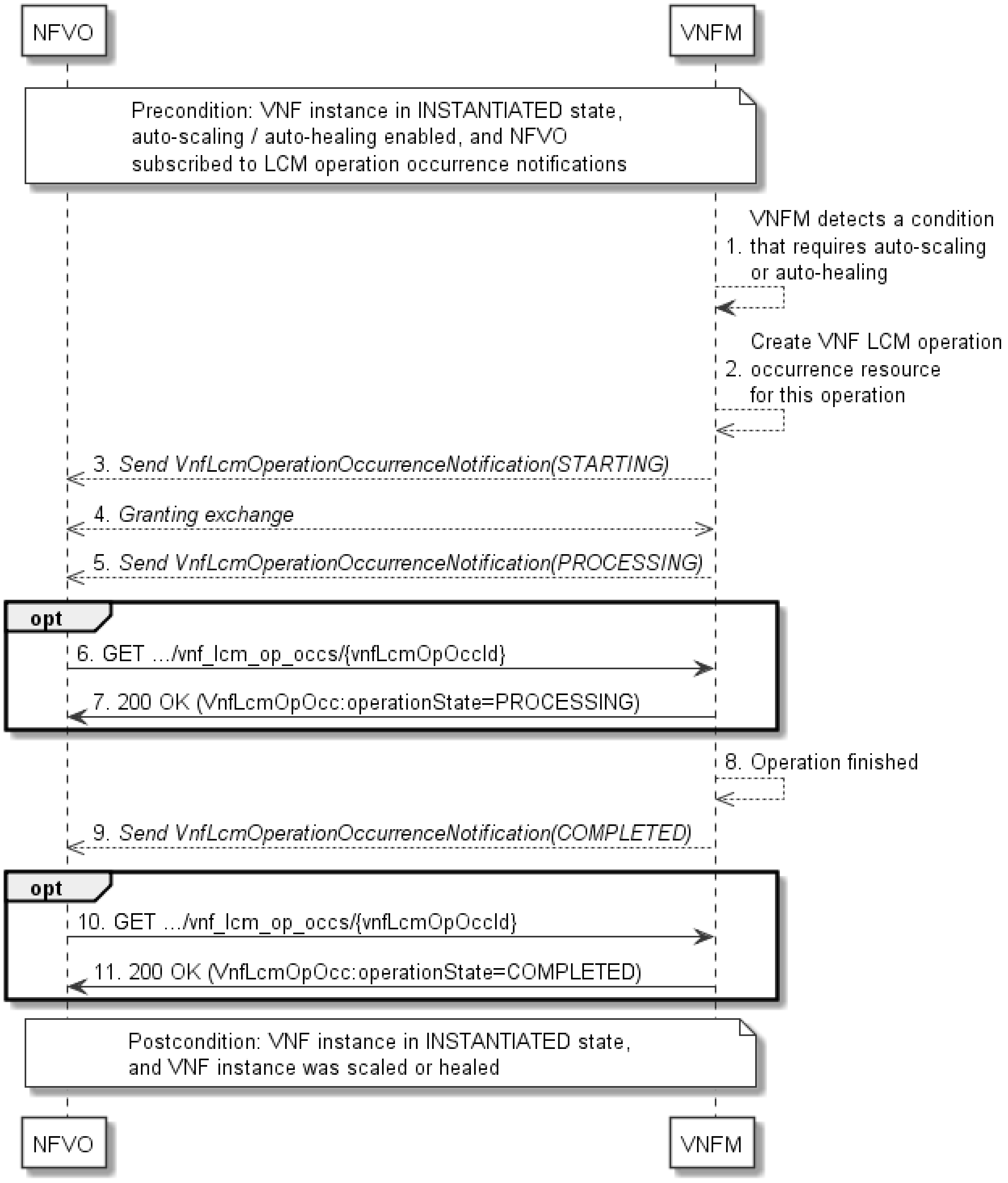
NFVO
VNFM
Precondition: VNF instance in INSTANTIATED state, auto-scaling / auto-healing enabled, and NFVO subscribed to LCM operation occurrence notifications
1. VNFM detects a condition that requires auto-scaling or auto-healing
2. Create VNF LCM operation occurrence resource for this operation
3. Send VnfLcmOperationOccurrenceNotification(STARTING)
4. Granting exchange
5. Send VnfLcmOperationOccurrenceNotification(PROCESSING)
opt
6. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
7. 200 OK (VnfLcmOpOcc:operationState=PROCESSING)
8. Operation finished
9. Send VnfLcmOperationOccurrenceNotification(COMPLETED)
opt
10. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
11. 200 OK (VnfLcmOpOcc:operationState=COMPLETED)
Postcondition: VNF instance in INSTANTIATED state, and VNF instance was scaled or healed
NFVO
VNFM

Figure 5.3.4-1: Flow of auto-scaling and auto-healing

NOTE: Due to possible race conditions, the "STARTING" VnfLcmOperationOccurrenceNotification and the request of the Granting exchange can arrive in any order at the NFVO.

Precondition: The VNF instance is in INSTANTIATED state, auto-scaling / auto-healing is enabled, and the NFVO is subscribed to VNF LCM operation occurrence notifications.

The automatic invocation of a VNF scaling or VNF healing operation, as illustrated in figure 5.3.4-1, consists of the following steps:

1) The VNFM detects a condition that triggers auto-scaling (Scale VNF or Scale VNF To Level) or auto-healing (Heal VNF) of the VNF, and selects the appropriate parameters for the operation.
2) The VNFM creates a "VNF LCM operation occurrence" resource for the operation.
3) The VNFM sends to the NFVO a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the start of the lifecycle management operation occurrence.
4) VNFM and NFVO exchange granting information (see VNF Lifecycle Operation Granting interface, clause 9.3).
5) The VNFM sends to the NFVO a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state
6) If desired, the NFVO can poll the "VNF lifecycle management operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF lifecycle management operation occurrence.
7) In the response to that request, the VNFM returns to the NFVO information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".
8) The VNFM has finished the operation.
9) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the completion of the lifecycle management operation occurrence.
10) If desired, the NFVO can send a new GET request to the "VNF lifecycle management operation occurrence" resource.
11) In the response to that request, the VNFM returns to the NFVO information about the result of the operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: The VNF instance in INSTANTIATED state, and the VNF instance was scaled or healed as appropriate.

Error handling: If the VNF lifecycle management operation fails, error information is provided in the notification message that reports the erroneous completion of the procedure, and is also available in the resource that represents the actual VNF lifecycle management operation occurrence related to this VNF lifecycle management operation.

5.3.5 Flow of the Query VNF operation

This clause describes a sequence for querying/reading information about a VNF instance.

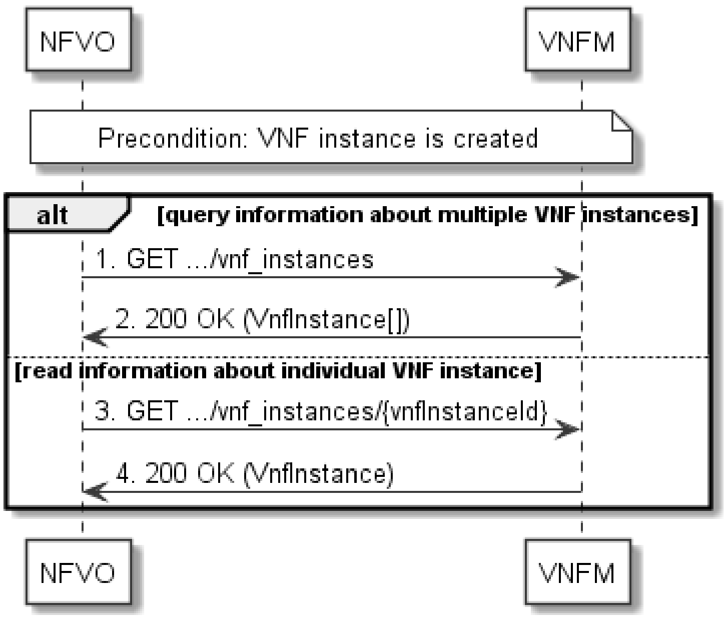
NFVO
VNFM
Precondition: VNF instance is created
alt
[query information about multiple VNF instances]
1. GET .../vnf_instances
2. 200 OK (VnfInstance[])
[read information about individual VNF instance]
3. GET .../vnf_instances/{vnfInstanceId}
4. 200 OK (VnfInstance)
NFVO
VNFM

Figure 5.3.5-1: Flow of VNF instance query/read

Precondition: The resource representing the VNF instance has been created.

VNF instance query, as illustrated in figure 5.3.5-1, consists of the following steps:

1) If the NFVO intends to query all VNF instances, it sends a GET request to the "VNF instances" resource.

2) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "VnfInstance" in the payload body.

3) If the NFVO intends to read information about a particular VNF instance, it sends a GET request to the "Individual VNF instance" resource, addressed by the appropriate VNF instance identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the NFVO, and includes one data structure of type "VnfInstance" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

5.3.6 Flow of the Modify VNF Information operation

This clause describes a sequence for updating information about a VNF instance.

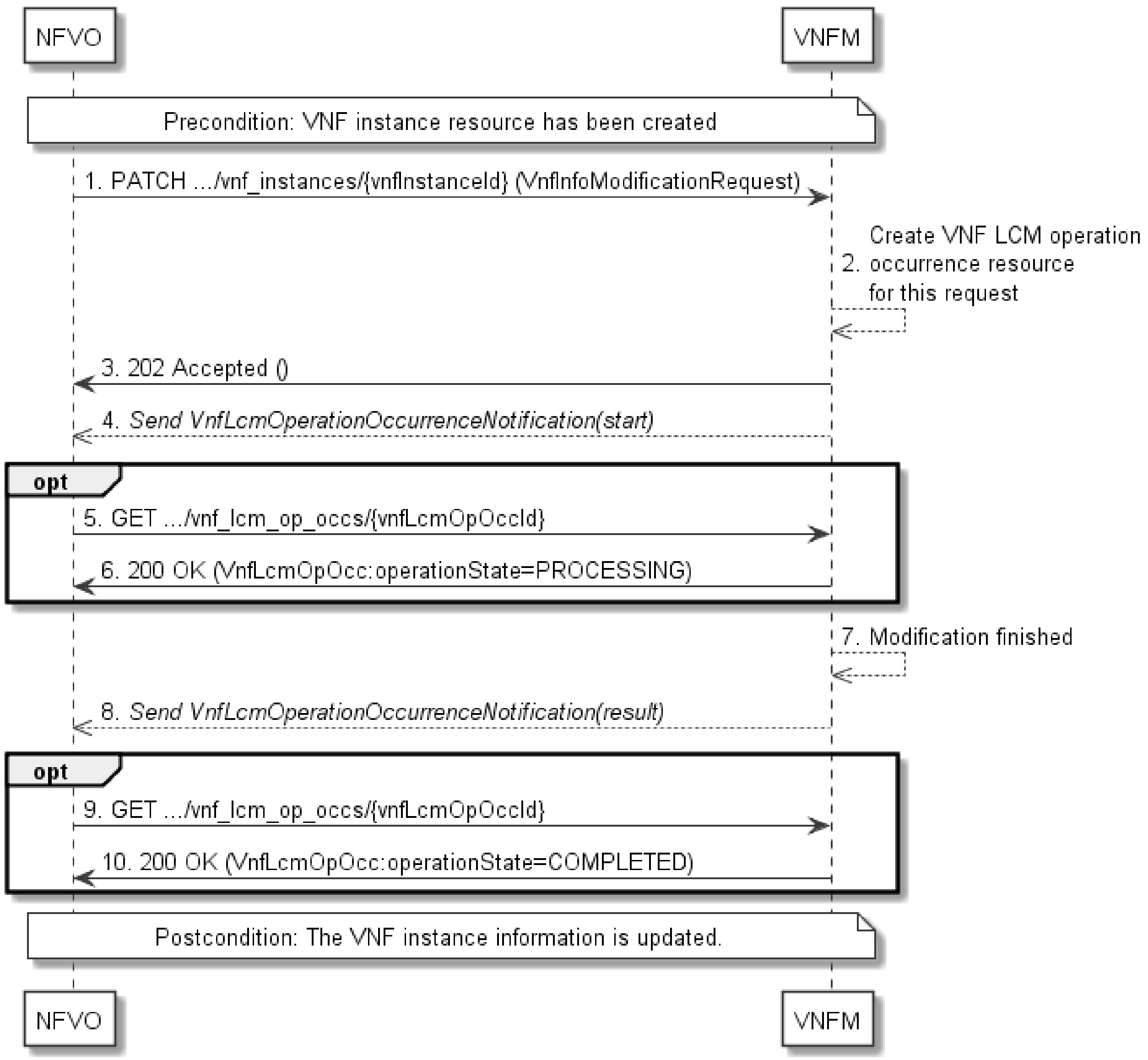
NFVO
VNFM
Precondition: VNF instance resource has been created
1. PATCH .../vnf_instances/{vnfInstanceId} (VnfInfoModificationRequest)
2. Create VNF LCM operation occurrence resource for this request
3. 202 Accepted ()
4. Send VnfLcmOperationOccurrenceNotification(start)
opt
5. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
6. 200 OK (VnfLcmOpOcc:operationState=PROCESSING)
7. Modification finished
8. Send VnfLcmOperationOccurrenceNotification(result)
opt
9. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
10. 200 OK (VnfLcmOpOcc:operationState=COMPLETED)
Postcondition: The VNF instance information is updated.
NFVO
VNFM

Figure 5.3.6-1: Flow of the modification of VNF instance information

NOTE: Due to possible race conditions, the 202 response and the VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The resource representing the VNF instance has been created.

Updating the VNF instance information, as illustrated in figure 5.3.6-1, consists of the following steps:

1) The NFVO sends a PATCH request to the "Individual VNF instance" resource of the VNF instance that is to be operated and includes in the payload body a data structure of type "VnfInfoModificationRequest".

2) The VNFM creates a "VNF LCM operation occurrence" resource for the request.

3) The VNFM returns a "202 Accepted" response with an empty payload body and a "Location" HTTP header that points to the new "VNF LCM operation occurrence" resource, i.e. it includes the URI of that resource which is "…/vnf_lcm_op_occs/{vnfLcmOpOccId}".

4) The VNFM sends to the NFVO a lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the start of the operation.

5) If desired, the NFVO can poll the "VNF LCM operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF lifecycle management operation occurrence.

6) In the response to that request, the VNFM returns to the NFVO information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".

7) The VNFM has finished the modification operation.

8) The VNFM sends a lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the completion of the operation, and the performed changes.

9) If desired, the NFVO can send a new GET request to the "VNF LCM operation occurrence" resource.

10) In the response to that request, the VNFM returns to the NFVO information about the result of the operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: Upon successful completion, information of the VNF instance is updated.

Error handling: If the updating of VNF instance information fails, error information is provided in the notification message that reports the erroneous completion of the procedure, and is also available in the resource that represents the actual VNF lifecycle management operation occurrence related to this VNF LCM operation.

5.3.7 Flow of the Get Operation Status operation

This clause describes a sequence for obtaining the status of a VNF lifecycle management operation occurrence.

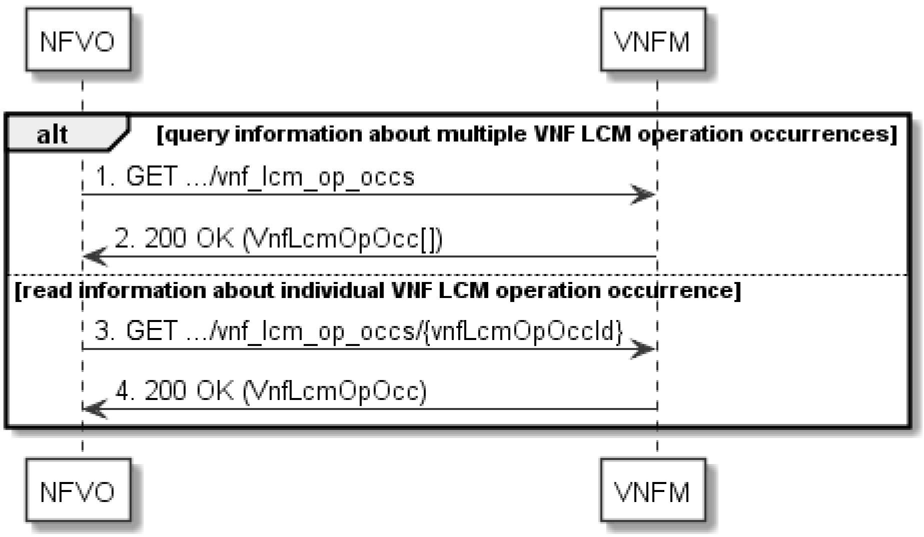
NFVO
VNFM
alt
[query information about multiple VNF LCM operation occurrences]
1. GET .../vnf_lcm_op_occs
2. 200 OK (VnfLcmOpOcc[])
[read information about individual VNF LCM operation occurrence]
3. GET .../vnf_lcm_op_occs/{vnfLcmOpOccId}
4. 200 OK (VnfLcmOpOcc)
NFVO
VNFM

Figure 5.3.7-1: Flow of Get VNF lifecycle operation status

Obtaining the VNF lifecycle operation status, as illustrated in figure 5.3.7-1, consists of the following steps:

1) If the NFVO intends to query all VNF lifecycle management operation occurrences, it sends a GET request to the "VNF LCM operation occurrences" resource.

2) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "VnfLcmOpOcc" in the payload body.

3) If the NFVO intends to read information about a particular VNF LCM operation occurrence, it sends a GET request to the "Individual VNF LCM operation occurrence" resource, addressed by the appropriate VNF LCM operation occurrence identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the NFVO, and includes one data structure of type "VnfLcmOpOcc" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

5.3.8 Flow of managing subscriptions

This clause describes the procedure for creating, querying/reading and terminating subscriptions to notifications related to VNF lifecycle management.

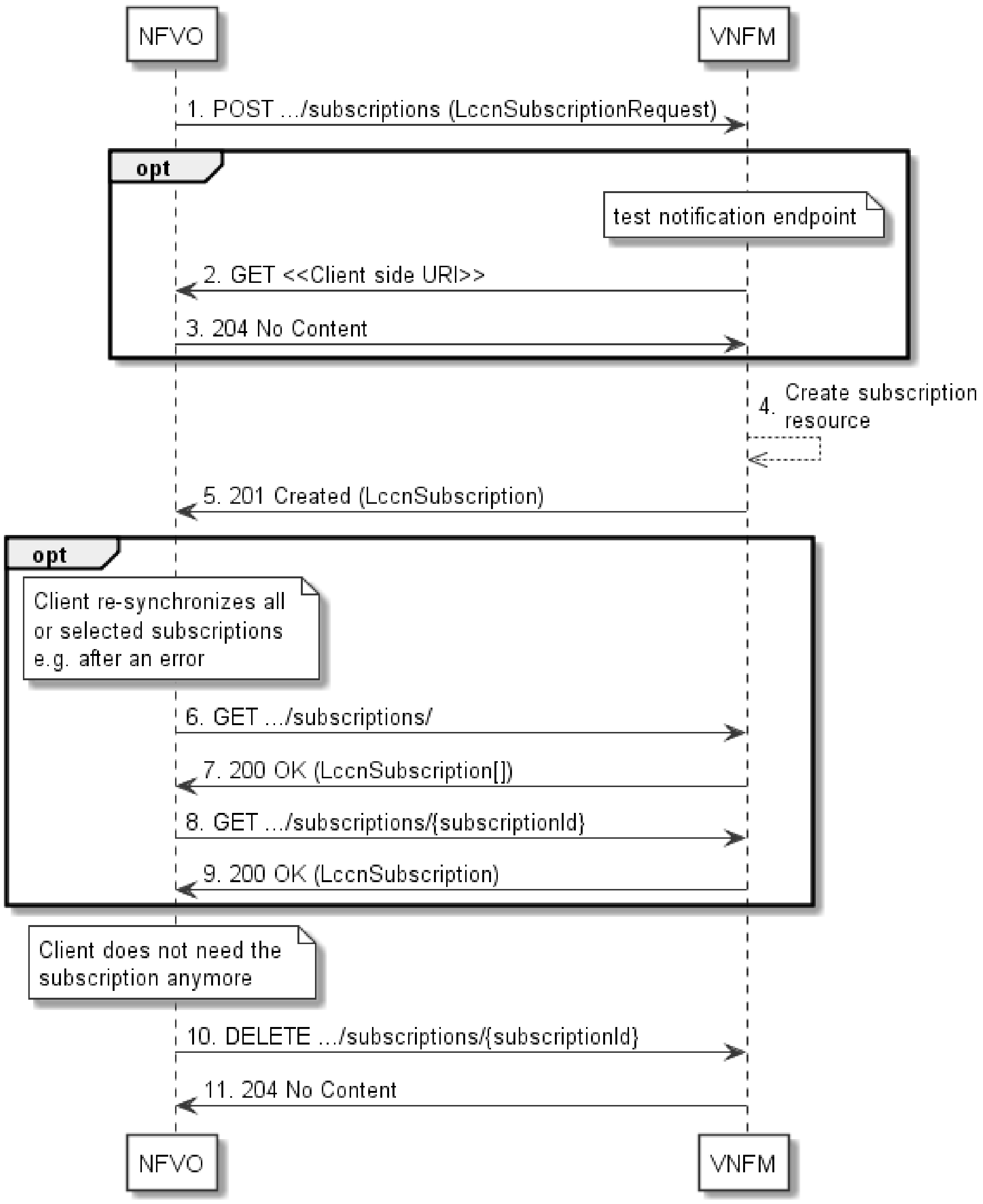
NFVO
VNFM
1. POST .../subscriptions (LccnSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (LccnSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (LccnSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (LccnSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
NFVO
VNFM

Figure 5.3.8-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 5.3.8-1:

1) The NFVO sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "LccnSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the NFVO as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3) In that case, the NFVO returns a "204 No Content" response to indicate success.

4) The VNFM creates a new subscription to notifications related to VNF lifecycle changes, and a resource that represents this subscription.

5) The VNFM returns a 201 Created response containing a data structure of type "LccnSubscription" representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) If desired, e.g. to recover from an error situation, the NFVO may query information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the VNFM returns a "200 OK" response that contains zero or more representations of all existing subscriptions that were created by the NFVO.

8) If desired, e.g. to recover from an error situation, the NFVO may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10) If the NFVO does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11) The NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

5.3.9 Flow of sending notifications

This clause describes the procedure for sending notifications.

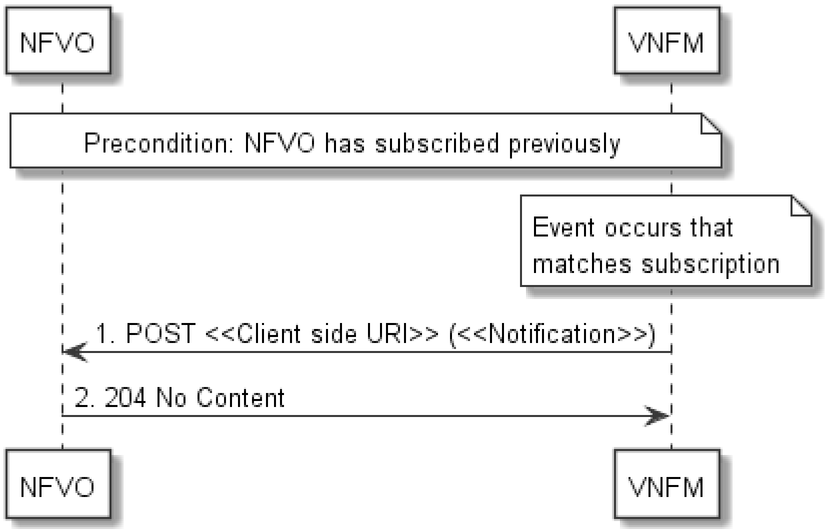
NFVO
VNFM
Precondition: NFVO has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
NFVO
VNFM

Figure 5.3.9-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in figure 5.3.9-1.

Precondition: The NFVO has subscribed previously to notifications related to VNF lifecycle management.

1) If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the NFVO has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API (see clauses 5.5.2.17 to 5.5.2.19).

2) The NFVO acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the NFVO, it can retry sending the notification.

5.3.10 Flow of retrying a VNF lifecycle management operation

This clause describes a sequence for retrying a VNF lifecycle management operation occurrence that is represented by a "VNF LCM operation occurrence" resource. Retry is used if an operation is in FAILED_TEMP state, and there is reason to believe that the operation will eventually succeed when retried, for instance because obstacle that led to an error during the execution of the LCM operation have been removed by an automated procedure, or by manual intervention. The "retry" operation is also called "idempotent retry" because it is possible to invoke retry multiple times, without side effects.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.

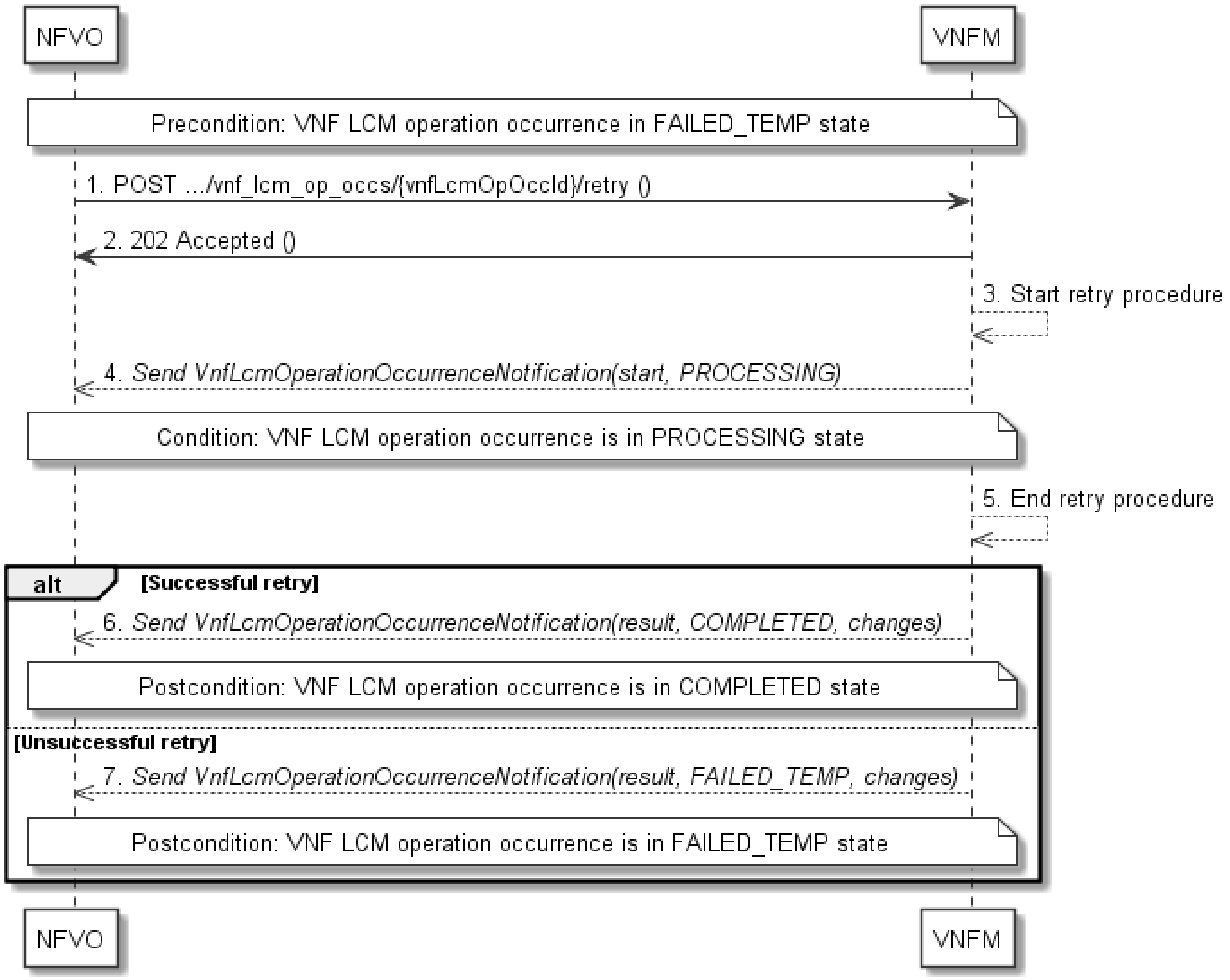
NFVO
VNFM
Precondition: VNF LCM operation occurrence in FAILED_TEMP state
1. POST .../vnf_lcm_op_occs/{vnfLcmOpOccId}/retry ()
2. 202 Accepted ()
3. Start retry procedure
4. Send VnfLcmOperationOccurrenceNotification(start, PROCESSING)
Condition: VNF LCM operation occurrence is in PROCESSING state
5. End retry procedure
alt
[Successful retry]
6. Send VnfLcmOperationOccurrenceNotification(result, COMPLETED, changes)
Postcondition: VNF LCM operation occurrence is in COMPLETED state
[Unsuccessful retry]
7. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: VNF LCM operation occurrence is in FAILED_TEMP state
NFVO
VNFM

Figure 5.3.10-1: Flow of retrying a VNF lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "PROCESSING" VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Retrying a VNF lifecycle operation, as illustrated in figure 5.3.10-1, consists of the following steps:

1) The NFVO sends a POST request with an empty body to the "Retry operation task" resource of the VNF LCM operation occurrence that is to be retried.

2) The VNFM returns a "202 Accepted" response.

3) The VNFM starts the retry procedure.

4) The VNFM sends a VNF lifecycle management operation occurrence notification of type "start" to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state.

5) The VNFM finishes the retry procedure.

6) On successful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate successful completion of the operation, and inform the NFVO about the virtualised resources changes.

7) On unsuccessful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (retry failed) of the operation, and to inform the NFVO about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in one of the following states: FAILED_TEMP, COMPLETED. COMPLETED is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the "VNF LCM operation occurrence" resource is in any other state than FAILED_TEMP, or in case Retry is not supported by for the particular VNF LCM operation for the particular VNF.

5.3.11 Flow of rolling back a VNF lifecycle management operation

This clause describes a sequence for rolling back a VNF lifecycle management operation occurrence that is represented by a "VNF LCM operation occurrence" resource. Rollback can be used for example if an operation is in FAILED_TEMP state, and there is no reason to believe that retrying the operation will eventually succeed.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.

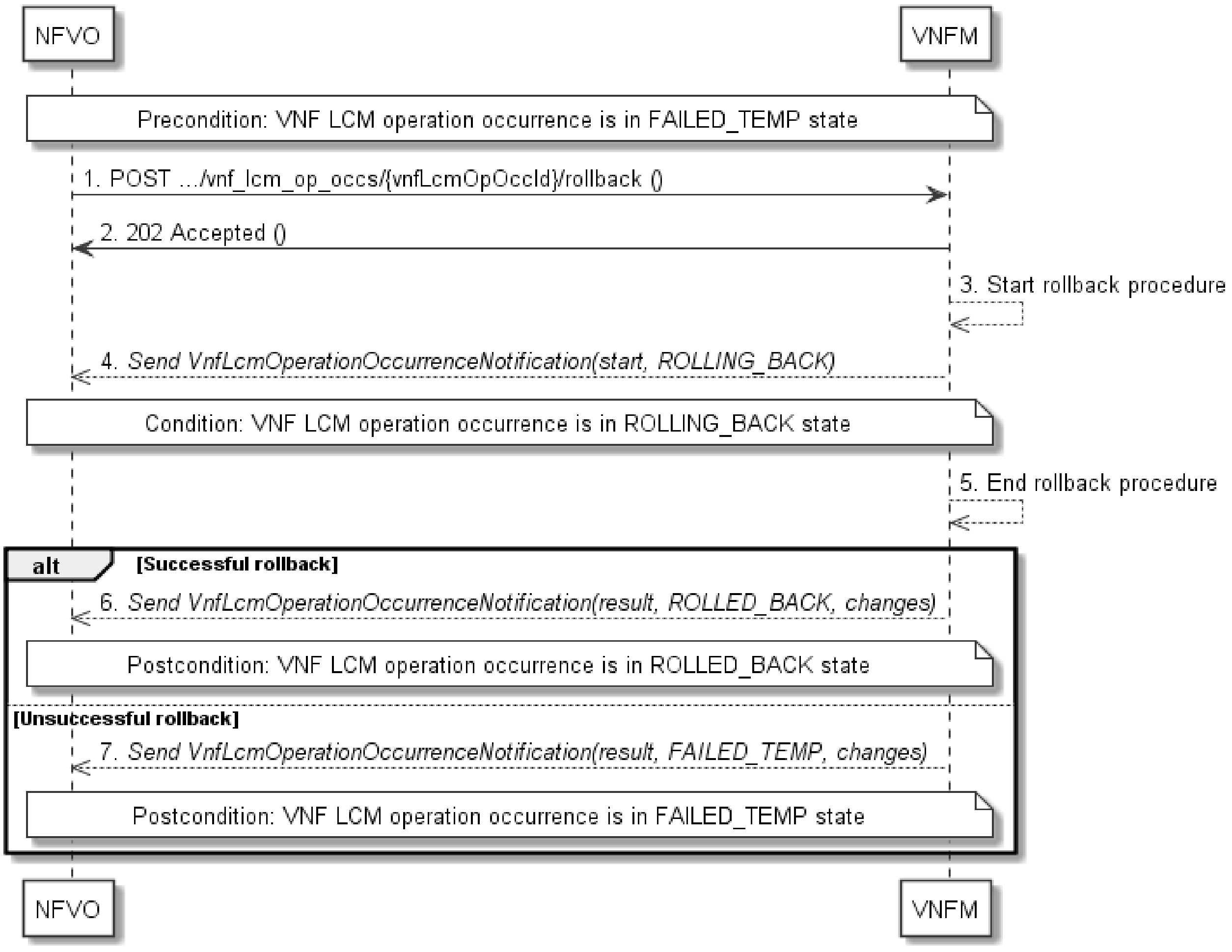
NFVO
VNFM
Precondition: VNF LCM operation occurrence is in FAILED_TEMP state
1. POST .../vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback ()
2. 202 Accepted ()
3. Start rollback procedure
4. Send VnfLcmOperationOccurrenceNotification(start, ROLLING_BACK)
Condition: VNF LCM operation occurrence is in ROLLING_BACK state
5. End rollback procedure
alt
[Successful rollback]
6. Send VnfLcmOperationOccurrenceNotification(result, ROLLED_BACK, changes)
Postcondition: VNF LCM operation occurrence is in ROLLED_BACK state
[Unsuccessful rollback]
7. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: VNF LCM operation occurrence is in FAILED_TEMP state
NFVO
VNFM

Figure 5.3.11-1: Flow of rolling back a VNF lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "PROCESSING" VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Initiating the rollback of a VNF lifecycle management operation, as illustrated in figure 5.3.11-1, consists of the following steps:

1) The NFVO sends a POST request with an empty body to the "Rollback operation task" resource of the VNF LCM operation occurrence that is to be rolled back.
2) The VNFM returns a "202 Accepted" response.
3) The VNFM starts the rollback procedure.
4) The VNFM sends a VNF lifecycle management operation occurrence notification of type "start" to indicate that the VNF LCM operation occurrence enters the "ROLLING_BACK" state.
5) The VNFM finishes the rollback procedure.
6) On successful rollback, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate successful completion of the operation, and inform the NFVO about the virtualised resources changes.
7) On unsuccessful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (rollback failed) of the operation, and to inform the NFVO about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in one of the following states: FAILED_TEMP, ROLLED_BACK. ROLLED_BACK is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than FAILED_TEMP, or in case Rollback is not supported for the particular VNF LCM operation for the particular VNF.

5.3.12 Flow of failing a VNF lifecycle management operation

This clause describes a sequence for declaring as "failed" a VNF lifecycle management operation occurrence that is represented by a "VNF LCM operation occurrence" resource. If there is neither an assumption that the operation can eventually succeed after further retries, nor that the operation can be successfully rolled back, the operation can be declared as "failed". This will unblock the invocation of other LCM operations, such as HealVnf, or non-graceful VNF termination, on the affected VNF instance.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.

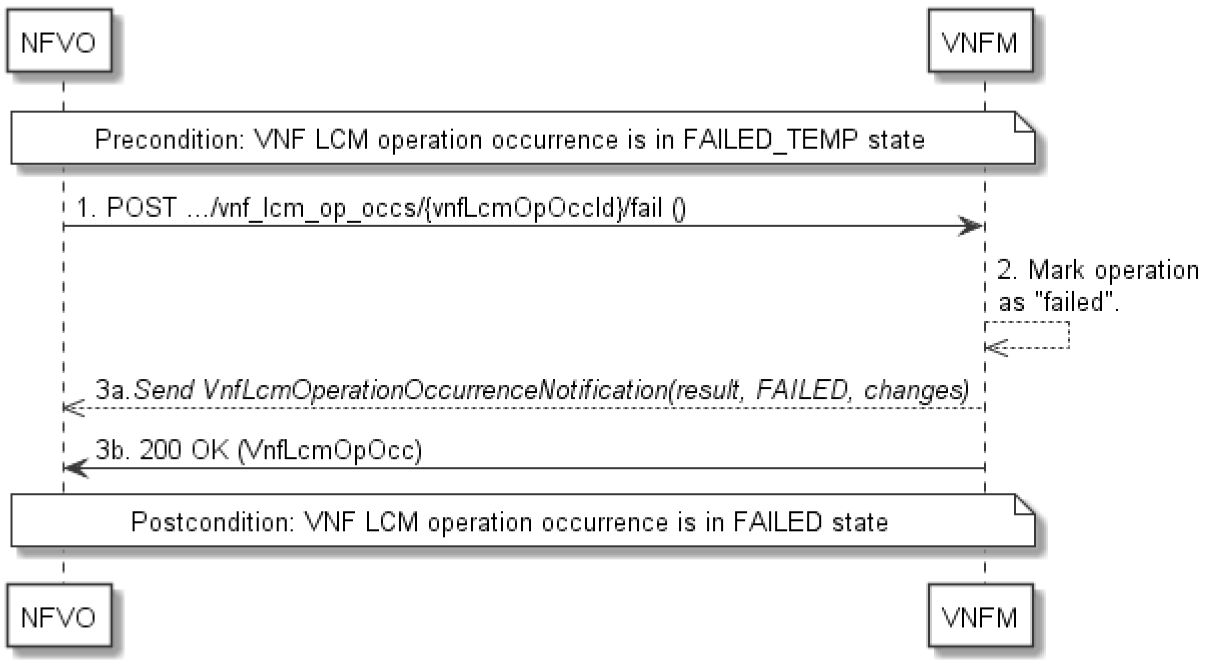
NFVO
VNFM
Precondition: VNF LCM operation occurrence is in FAILED_TEMP state
1. POST .../vnf_lcm_op_occs/{vnfLcmOpOccId}/fail ()
2. Mark operation as "failed".
3a. Send VnfLcmOperationOccurrenceNotification(result, FAILED, changes)
3b. 200 OK (VnfLcmOpOcc)
Postcondition: VNF LCM operation occurrence is in FAILED state
NFVO
VNFM

Figure 5.3.12-1: Flow of declaring a VNF lifecycle management operation as failed

NOTE: Due to possible race conditions, the 200 response and the "FAILED" VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Declaring a VNF lifecycle management operation as failed, as illustrated in figure 5.3.12-1, consists of the following steps:

1) The NFVO sends a POST request with an empty body to the "Fail operation task" resource of the VNF LCM operation occurrence that is to be marked as failed.

2) The VNFM marks the operation as failed.

3) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the final failure of the operation, and to inform the NFVO about the virtualised resources changes. Furthermore, it returns a "200 OK" response, and includes in the body a VnfLcmOpOcc structure. The order in which the response and the notification arrive at the NFVO is not defined.

Postcondition: The VNF lifecycle management operation occurrence is FAILED state. This is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than FAILED_TEMP.

5.3.13 Flow of cancelling a VNF lifecycle management operation

This clause describes a sequence for cancelling an ongoing VNF LCM operation occurrence, or a rollback of a VNF LCM operation occurrence. The possibility and timing of cancellation is dependent on the implementation of the underlying lifecycle management operation.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.

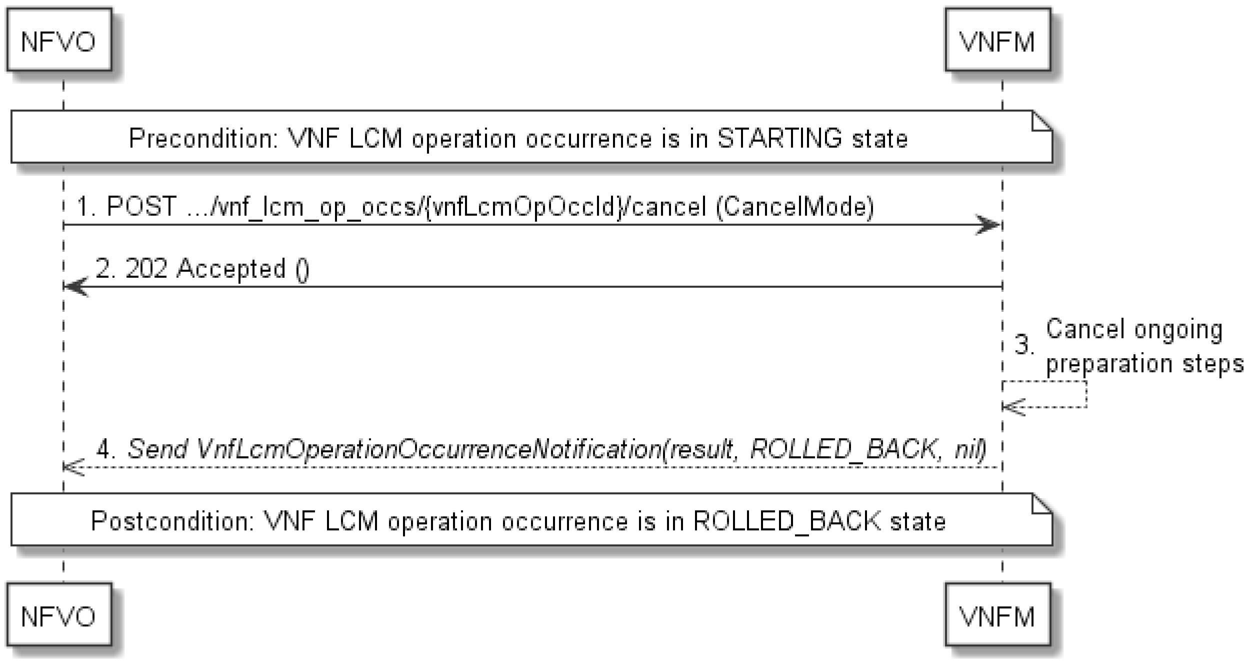
NFVO
VNFM
Precondition: VNF LCM operation occurrence is in STARTING state
1. POST .../vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel (CancelMode)
2. 202 Accepted ()
3. Cancel ongoing preparation steps
4. Send VnfLcmOperationOccurrenceNotification(result, ROLLED_BACK, nil)
Postcondition: VNF LCM operation occurrence is in ROLLED_BACK state
NFVO
VNFM

Figure 5.3.13-1: Flow of cancelling a VNF lifecycle management operation in "STARTING" state

NOTE 1: Due to possible race conditions, the 202 response and the "ROLLED_BACK" VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The VNF lifecycle management operation occurrence is in STARTING state.

Cancelling a VNF lifecycle operation when it is in STARTING state, as illustrated in figure 5.3.13-1, consists of the following steps:

1) The NFVO sends a POST request with "CancelMode" structure in the body to the "Cancel operation task" resource of the VNF LCM operation occurrence that is to be cancelled.

2) The VNFM returns a "202 Accepted" response.

3) The VNFM cancels the ongoing preparation operations.

4) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (cancelled) of the operation, and to inform the NFVO that there were no virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in ROLLED_BACK state.

Error handling: The operation is rejected in case the VNF lifecycle operation occurrence is in any other state than STARTING.

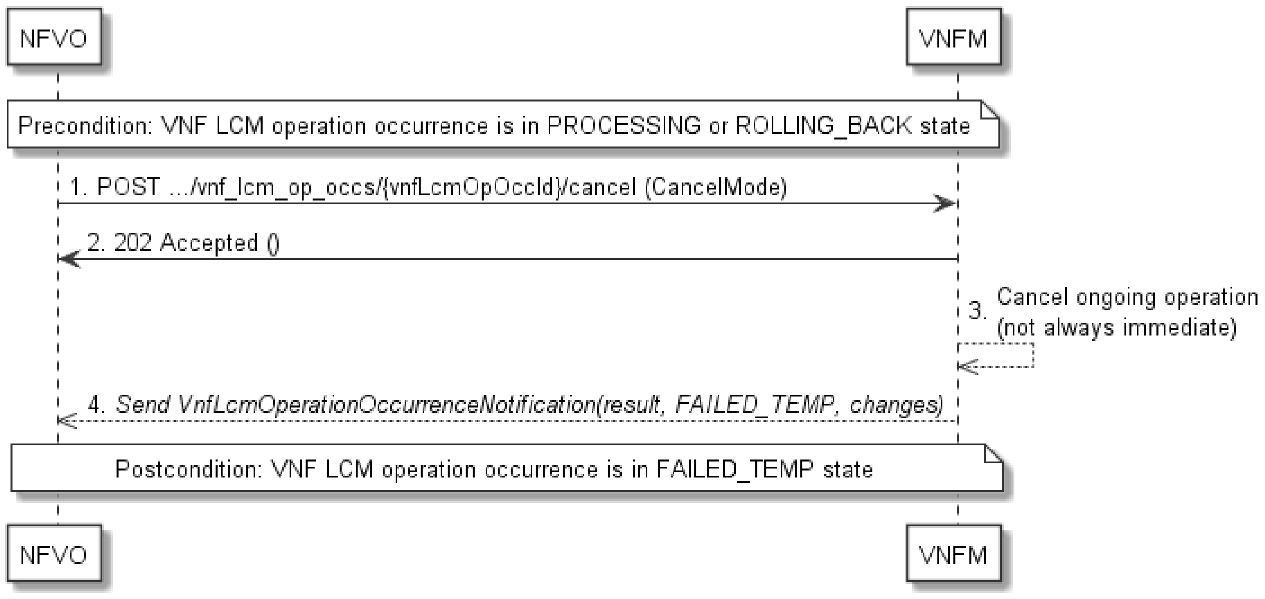
NFVO
VNFM
Precondition: VNF LCM operation occurrence is in PROCESSING or ROLLING_BACK state
1. POST .../vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel (CancelMode)
2. 202 Accepted ()
3. Cancel ongoing operation (not always immediate)
4. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: VNF LCM operation occurrence is in FAILED_TEMP state
NFVO
VNFM

Figure 5.3.13-2: Flow of cancelling a VNF lifecycle management operation in "PROCESSING" or "ROLLING_BACK" state

NOTE 2: Due to possible race conditions, the 202 response and the "FAILED_TEMP" VnfLcmOperationOccurrenceNotification can arrive in any order at the NFVO.

Precondition: The VNF lifecycle management operation occurrence is in PROCESSING or ROLLING_BACK state.

Cancelling a VNF lifecycle operation when it is in "PROCESSING" or "ROLLING_BACK" state, as illustrated in figure 5.3.13-2, consists of the following steps:

1) The NFVO sends a POST request with a "CancelMode" structure in the body to the "Cancel operation task" resource of the VNF LCM operation occurrence that is to be cancelled.

2) The VNFM returns a "202 Accepted" response.

3) The VNFM cancels the ongoing LCM operation. This can take some time.

4) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (cancelled) of the operation, and to inform the NFVO about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is FAILED_TEMP state.

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than PROCESSING or ROLLING_BACK, or in case Cancel is not supported for the particular VNF LCM operation for the particular VNF.

5.4 Resources

5.4.1 Introduction

This clause defines all the resources and methods provided by the VNF lifecycle management interface.

5.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF lifecycle management interface.

5.4.2 Resource: VNF instances

5.4.2.1 Description

This resource represents VNF instances. The client can use this resource to create individual VNF instance resources, and to query VNF instances.

5.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances

This resource shall support the resource URI variables defined in table 5.4.2.2-1.

Table 5.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.2.3 Resource methods

5.4.2.3.1 POST

The POST method creates a new VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.2.3.1-1 and 5.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.2.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	CreateVnfRequest	1	The VNF creation parameters, as defined in clause 5.5.2.3	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfInstance	1	201 Created	A VNF Instance identifier was created successfully. The response body shall contain a representation of the created VNF instance, as defined in clause 5.5.2.2. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created VNF instance.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.2.3.2 GET

The GET method queries information about multiple VNF instances.

This method shall follow the provisions specified in the tables 5.4.2.3.2-1 and 5.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the VnfInstance and in data types referenced from it shall be supported by the VNFM in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the VnfInstance structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - vnfConfigurableProperties - vimConnectionInfo - instantiatedVnfInfo - metadata - extensions
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 5.4.2.3.2-2: Details of the GET request/response on this resource

Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfInstance	0..N	200 OK	Information about zero or more VNF instances was queried successfully. The response body shall contain in an array the representations of zero or more VNF instances, as defined in clause 5.5.2.2. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3 Resource: Individual VNF instance

5.4.3.1 Description

This resource represents an individual VNF instance. The client can use this resource to modify and delete the underlying VNF instance, and to read information about the VNF instance.

5.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}

The base resource URI variables for this resource are defined in table 5.4.3.2-1.

Table 5.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
vnfInstanceId	Identifier of the VNF instance. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.3.3 Resource methods

5.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3.3.2 GET

The GET method retrieves information about a VNF instance by reading an individual VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.3.3.2-1 and 5.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response codes	Description
	VnfInstance	1	200 OK	Information about an individual VNF instance was read successfully. The response body shall contain a representation of the VNF instance, as defined in clause 5.5.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3.3.4 PATCH

This method modifies an individual VNF instance resource.

Changes to the VNF configurable properties are applied to the configuration in the VNF instance, and are reflected in the representation of this resource. Other changes are applied to the VNF instance information managed by the VNFM, and are reflected in the representation of this resource.

This method shall follow the provisions specified in the tables 5.4.3.3.4-1 and 5.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality	Description	
Request body	VnfInfoModificationRequest	1	Parameters for the VNF modification, as defined in clause 5.5.2.12. The Content-Type header shall be set to "application/merge-patch+json" according to IETF RFC 7396 [17].	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. On success, the HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation. The response body shall be empty.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF instance resource. Typically, this is due to the fact that another LCM operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	412 Precondition failed	Error: A precondition given in an HTTP request header is not fulfilled. Typically, this is due to an ETag mismatch, indicating that the resource was modified by another entity. The response body should contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.3.3.5 DELETE

This method deletes an individual VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.3.3.5-1 and 5.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The VNF instance resource and the associated VNF identifier were deleted successfully. The response body shall be empty.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in INSTANTIATED state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.4 Resource: Instantiate VNF task

5.4.4.1 Description

This task resource represents the "Instantiate VNF" operation. The client can use this resource to instantiate a VNF instance.

5.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/instantiate

This resource shall support the resource URI variables defined in table 5.4.4.2-1.

Table 5.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance to be instantiated. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.4.3 Resource methods

5.4.4.3.1 POST

The POST method instantiates a VNF instance.

This method shall follow the provisions specified in the tables 5.4.4.3.1-1 and 5.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.4.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	InstantiateVnfRequest	1	Parameters for the VNF instantiation, as defined in clause 5.5.2.4.	
Response body	Data type	Cardinality	Response Codes	Description
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in INSTANTIATED state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.4.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5 Resource: Scale VNF task

5.4.5.1 Description

This task resource represents the "Scale VNF" operation. The client can use this resource to request scaling a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

See clause B.2 in annex B for an explanation of VNF scaling.

5.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/scale

This resource shall support the resource URI variables defined in table 5.4.5.2-1.

Table 5.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
vnfInstanceId	Identifier of the VNF instance to be scaled. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.5.3 Resource methods

5.4.5.3.1 POST

The POST method requests to scale a VNF instance resource incrementally.

This method shall follow the provisions specified in the tables 5.4.5.3.1-1 and 5.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.5.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	ScaleVnfRequest	1		Parameters for the scale VNF operation, as defined in clause 5.5.2.5.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.5.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6 Resource: Scale VNF to Level task

5.4.6.1 Description

This task resource represents the "Scale VNF to Level" operation. The client can use this resource to request scaling of a VNF instance to a target level.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

See clause B.2 in annex B for an explanation of VNF scaling.

5.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/scale_to_level

This resource shall support the resource URI variables defined in table 5.4.6.2-1.

Table 5.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
vnfInstanceId	Identifier of the VNF instance to be scaled to a target level. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.6.3 Resource methods

5.4.6.3.1 POST

The POST method requests to scale a VNF instance resource to a target level.

This method shall follow the provisions specified in the tables 5.4.6.3.1-1 and 5.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.6.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.6.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**	**Description**	
	ScaleVnfToLevelRequest	1	Parameters for the scale VNF to Level operation, as defined in clause 5.5.2.6.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.6.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7 Resource: Change VNF Flavour task

5.4.7.1 Description

This task resource represents the "Change VNF Flavour" operation. The client can use this resource to change the deployment flavour for a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF. This operation may be service-disruptive.

5.4.7.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/change_flavour

This resource shall support the resource URI variables defined in table 5.4.7.2-1.

Table 5.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance of which the deployment flavour is requested to be changed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.7.3 Resource methods

5.4.7.3.1 POST

The POST method changes the deployment flavour of a VNF instance.

This method shall follow the provisions specified in the tables 5.4.7.3.1-1 and 5.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.7.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	ChangeVnfFlavourRequest	1		Parameters for the Change VNF Flavour operation, as defined in clause 5.5.2.7.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.7.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8 Resource: Terminate VNF task

5.4.8.1 Description

This task resource represents the "Terminate VNF" operation. The client can use this resource to terminate a VNF instance.

5.4.8.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/terminate

This resource shall support the resource URI variables defined in table 5.4.8.2-1.

Table 5.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance to be terminated. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.8.3 Resource methods

5.4.8.3.1 POST

The POST method terminates a VNF instance.

This method shall follow the provisions specified in the tables 5.4.8.3.1-1 and 5.4.8.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.8.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.8.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	TerminateVnfRequest	1		Parameters for the VNF termination, as defined in clause 5.5.2.8.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.8.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9 Resource: Heal VNF task

5.4.9.1 Description

This task resource represents the "Heal VNF" operation. The client can use this resource to request healing a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

5.4.9.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/heal

This resource shall support the resource URI variables defined in table 5.4.9.2-1.

Table 5.4.9.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be healed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.9.3 Resource methods

5.4.9.3.1 POST

The POST method requests to heal a VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.9.3.1-1 and 5.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	HealVnfRequest	1	Parameters for the Heal VNF operation, as defined in clause 5.5.2.9.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.9.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10 Resource: Operate VNF task

5.4.10.1 Description

This task resource represents the "Operate VNF" operation. The client can use this resource to operate a VNF.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

The "Operate VNF" operation enables requesting to change the operational state of a VNF instance, including starting and stopping the VNF instance.

NOTE 1: These operations are complementary to instantiating and terminating a VNF.

NOTE 2: In the present document, only starting and stopping the VNF instances is supported. Extension of this operation to support other VNF state changes is left for future specification.

A VNF instance can be in the following states:

STARTED: the VNF instance is up and running.

STOPPED: the VNF instance has been shut down. A VNF instance is stopped if all its VNFC instances are also stopped.

In the state STOPPED, the virtualisation containers, where the VNFC instances of the VNF run, are shut down but not deleted. In addition, if the workflow requires a graceful stop, as part of this process the VNFM (producer of the interface) will interact with VNF/EM to gracefully stop the VNF application. Once a VNF is instantiated, i.e. all instantiation steps have been completed, the VNF instance is in the state STARTED.

5.4.10.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/operate

This resource shall support the resource URI variables defined in table 5.4.10.2-1.

Table 5.4.10.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be operated. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.10.3 Resource methods

5.4.10.3.1 POST

The POST method changes the operational state of a VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.10.3.1-1 and 5.4.10.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.10.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.10.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	OperateVnfRequest	1		Parameters for the Operate VNF operation, as defined in clause 5.5.2.10.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.10.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.11 Resource: Change external VNF connectivity task

5.4.11.1 Description

This task resource represents the "Change external VNF connectivity" operation. The client can use this resource to change the external connectivity of a VNF instance. The types of changes that this operation supports are:

- Disconnect the external CPs that are connected to a particular external VL, and connect them to a different external VL.
- Change the connectivity parameters of the existing external CPs, including changing addresses.

NOTE: Depending on the capabilities of the underlying VIM resources, certain changes (e.g. modifying the IP address assignment) might not be supported without deleting the resource and creating another one with the modified configuration.

VNFs shall support this operation. This operation may be service-disruptive.

5.4.11.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/change_ext_conn

This resource shall support the resource URI variables defined in table 5.4.11.2-1.

Table 5.4.11.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance of which the external connectivity is requested to be changed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.11.3 Resource methods

5.4.11.3.1 POST

The POST method changes the external connectivity of a VNF instance.

This method shall follow the provisions specified in the tables 5.4.11.3.1-1 and 5.4.11.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.11.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.11.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	ChangeExtVnfConnectivityRequest	1	Parameters for the Change external VNF connectivity operation, as defined in clause 5.5.2.11.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the instantiation operation.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.11.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.11.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.11.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.11.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.12 Resource: VNF LCM operation occurrences

5.4.12.1 Description

This resource represents VNF lifecycle management operation occurrences. The client can use this resource to query status information about multiple VNF lifecycle management operation occurrences.

5.4.12.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs

The base resource URI variables for this resource are defined in table 5.4.12.2-1.

Table 5.4.12.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.12.3 Resource methods

5.4.12.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.12.3.2 GET

The client can use this method to query status information about multiple VNF lifecycle management operation occurrences.

This method shall follow the provisions specified in the tables 5.4.12.3.2-1 and 5.4.12.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.12.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the VnfLcmOpOcc and in data types referenced from it shall be supported by the VNFM in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the VnfLcmOpOcc structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - operationParams - error - resourceChanges - changedInfo - changedExtConnectivity.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 5.4.12.3.2-2: Details of the GET request/response on this resource

Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfLcmOpOcc	0..N	200 OK	Status information for zero or more VNF lifecycle management operation occurrences was queried successfully. The response body shall contain in an array the status information about zero or more VNF lifecycle operation occurrences, as defined in clause 5.5.2.13. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.12.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.12.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.12.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.13 Resource: Individual VNF LCM operation occurrence

5.4.13.1 Description

This resource represents a VNF lifecycle management operation occurrence. The client can use this resource to read status information about an individual VNF lifecycle management operation occurrence. Further, the client can use task resources which are children of this resource to request cancellation of an operation in progress, and to request the handling of operation errors via retrying the operation, rolling back the operation, or permanently failing the operation.

5.4.13.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}

The base resource URI variables for this resource are defined in table 5.4.13.2-1.

Table 5.4.13.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.13.3 Resource methods

5.4.13.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.13.3.2 GET

The client can use this method to retrieve status information about a VNF lifecycle management operation occurrence by reading an individual "VNF LCM operation occurrence" resource.

This method shall follow the provisions specified in the tables 5.4.13.3.2-1 and 5.4.13.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.13.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.13.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VnfLcmOpOcc	1	200 OK	Information about a VNF LCM operation occurrence was read successfully. The response body shall contain status information about a VNF lifecycle management operation occurrence (see clause 5.5.2.13).
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.13.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.13.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.13.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.14 Resource: Retry operation task

5.4.14.1 Description

This task resource represents the "Retry operation" operation. The client can use this resource to initiate retrying a VNF lifecycle operation.

5.4.14.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/ vnf_lcm_op_occs/{vnfLcmOpOccId}/retry

This resource shall support the resource URI variables defined in table 5.4.14.2-1.

Table 5.4.14.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be retried. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.14.3 Resource methods

5.4.14.3.1 POST

The POST method initiates retrying a VNF lifecycle operation if that operation has experienced a temporary failure, i.e. the related "VNF LCM operation occurrence" resource is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.14.3.1-1 and 5.4.14.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.14.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.14.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			The POST request to this resource has an empty payload body.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state, or another error handling action is starting, such as rollback or fail. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.14.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.14.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.14.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.14.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.15 Resource: Rollback operation task

5.4.15.1 Description

This task resource represents the "Rollback operation" operation. The client can use this resource to initiate rolling back a VNF lifecycle operation.

5.4.15.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback

This resource shall support the resource URI variables defined in table 5.4.15.2-1.

Table 5.4.15.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be rolled back. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.15.3 Resource methods

5.4.15.3.1 POST

The POST method initiates rolling back a VNF lifecycle operation if that operation has experienced a temporary failure, i.e. the related "VNF LCM operation occurrence" resource is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.15.3.1-1 and 5.4.15.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.15.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.15.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a		The POST request to this resource has an empty payload body.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state, or another error handling action is starting, such as retry or fail. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.15.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.15.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.15.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.15.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.16 Resource: Fail operation task

5.4.16.1 Description

This task resource represents the "Fail operation" operation. The client can use this resource to mark a VNF lifecycle management operation occurrence as "finally failed", i.e. change the state of the related VNF LCM operation occurrence resource to "FAILED", if it is not assumed that a subsequent retry or rollback will succeed. Once the operation is marked as "finally failed", it cannot be retried or rolled back anymore.

5.4.16.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail

This resource shall support the resource URI variables defined in table 5.4.16.2-1.

Table 5.4.16.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be marked as "failed". See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.16.3 Resource methods

5.4.16.3.1 POST

The POST method marks a VNF lifecycle management operation occurrence as "finally failed" if that operation occurrence is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.16.3.1-1 and 5.4.16.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.16.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.16.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			The POST request to this resource has an empty payload body.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfLcmOpOcc	1	200 OK	The state of the VNF lifecycle management operation occurrence was changed successfully The response shall include a representation of the VNF lifecycle operation occurrence resource.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state, or another error handling action is starting, such as retry or rollback. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.16.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.16.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.16.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.16.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.17 Resource: Cancel operation task

5.4.17.1 Description

This task resource represents the "Cancel operation" operation. The client can use this resource to cancel an ongoing VNF lifecycle operation.

5.4.17.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel

This resource shall support the resource URI variables defined in table 5.4.17.2-1.

Table 5.4.17.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be cancelled. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.17.3 Resource methods

5.4.17.3.1 POST

The POST method initiates cancelling an ongoing VNF lifecycle operation while it is being executed or rolled back, i.e. the related "VNF LCM operation occurrence" is either in "PROCESSING" or "ROLLING_BACK" state.

This method shall follow the provisions specified in the tables 5.4.17.3.1-1 and 5.4.17.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.17.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.17.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	CancelMode	1		The POST request to this resource shall include a CancelMode structure in the payload body to choose between "graceful" and "forceful" cancellation.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the operation occurrence is not in STARTING, PROCESSING or ROLLING_BACK state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.17.3.2 GET

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.17.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.17.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.17.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.18 Resource: Subscriptions

5.4.18.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF lifecycle management, and to query its subscriptions.

5.4.18.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/subscriptions

This resource shall support the resource URI variables defined in table 5.4.18.2-1.

Table 5.4.18.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.18.3 Resource methods

5.4.18.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 5.4.18.3.1-1 and 5.4.18.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the NFVO, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 5.4.18.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.18.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**	**Description**	
	LccnSubscriptionRequest	1	Details of the subscription to be created, as defined in clause 5.5.2.15.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	LccnSubscription	1	201 Created	The subscription was created successfully. The response body shall contain a representation of the created subscription resource. The HTTP response shall include a "Location" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.18.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 5.4.18.3.2-1 and 5.4.18.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.18.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the LccnSubscription and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 5.4.18.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	LccnSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of lifecycle change notification subscriptions as defined in clause 5.5.2.16. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.18.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.18.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.18.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.19 Resource: Individual subscription

5.4.19.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to VNF lifecycle management.

5.4.19.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 5.4.19.2-1.

Table 5.4.19.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.19.3 Resource methods

5.4.19.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.19.3.2 GET

The GET method retrieves information about a subscription by reading an individual subscription resource.

This method shall follow the provisions specified in the tables 5.4.19.3.2-1 and 5.4.19.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.19.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.19.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	LccnSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.19.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.19.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.19.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 5.4.19.3.5-1 and 5.4.19.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.19.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.19.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.20 Resource: Notification endpoint

5.4.20.1 Description

This resource represents a notification endpoint. The API producer can use this resource to send notifications related to VNF lifecycle changes to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

5.4.20.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 5.4.20.2-1.

Table 5.4.20.2-1: Resource URI variables for this resource

Name	Definition
none supported	

5.4.20.3 Resource methods

5.4.20.3.1 POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 5.4.20.3.1-1 and 5.4.20.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.20.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one of the alternatives defined in table 5.4.20.3.1-2.

Table 5.4.20.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	VnfLcmOperationOccurrenceNotification	1		A notification about lifecycle changes triggered by a VNF LCM operation occurrence.
	VnfIdentifierCreationNotification	1		A notification about the creation of a VNF identifier and the related VNF instance resource.
	VnfIdentifierDeletionNotification	1		A notification about the deletion of a VNF identifier and the related VNF instance resource.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.20.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 5.4.20.3.2-1 and 5.4.20.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.20.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.20.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.20.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.20.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.20.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.5 Data model

5.5.1 Introduction

This clause defines the request and response data structures of the VNF Lifecycle management interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

5.5.2 Resource and notification data types

5.5.2.1 Introduction

This clause defines the data structures to be used in resource representations and notifications.

5.5.2.2 Type: VnfInstance

This type represents a VNF instance. It shall comply with the provisions defined in table 5.5.2.2-1.

NOTE: Clause B.3.2 in annex B provides examples illustrating the relationship among the different run-time information elements (CP, VL and link ports) used to represent the connectivity of a VNF.

Table 5.5.2.2-1: Definition of the VnfInstance data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the VNF instance.
vnfInstanceName	String	0..1	Name of the VNF instance. This attribute can be modified with the PATCH method.
vnfInstanceDescription	String	0..1	Human-readable description of the VNF instance. This attribute can be modified with the PATCH method.
vnfdId	Identifier	1	Identifier of the VNFD on which the VNF instance is based.
vnfProvider	String	1	Provider of the VNF and the VNFD. The value is copied from the VNFD.
vnfProductName	String	1	Name to identify the VNF Product. The value is copied from the VNFD.
vnfSoftwareVersion	Version	1	Software version of the VNF. The value is copied from the VNFD.
vnfdVersion	Version	1	Identifies the version of the VNFD. The value is copied from the VNFD.
vnfPkgId	Identifier	1	Identifier of information held by the NFVO about the specific VNF package on which the VNF is based. This identifier was allocated by the NFVO. This attribute can be modified with the PATCH method. See note 1.

Attribute name	Data type	Cardinality	Description
vnfConfigurableProperties	KeyValuePairs	0..1	Current values of the configurable properties of the VNF instance. Configurable properties referred in this attribute are declared in the VNFD (see note 2 and note 3). These configurable properties include the following standard attributes, which are declared in the VNFD if auto-scaling and/or auto-healing are supported by the VNF: - isAutoscaleEnabled: If present, the VNF supports auto-scaling. If set to true, auto-scaling is currently enabled. If set to false, auto-scaling is currently disabled. - isAutohealEnabled: If present, the VNF supports auto-healing. If set to true, auto-healing is currently enabled. If set to false, auto-healing is currently disabled. This attribute can be modified with the PATCH method.
vimConnectionInfo	VimConnectionInfo	0..N	Information about VIM connections to be used for managing the resources for the VNF instance. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable. This attribute can be modified with the PATCH method.
instantiationState	Enum (inlined)	1	The instantiation state of the VNF. Permitted values: NOT_INSTANTIATED: The VNF instance is terminated or not instantiated. INSTANTIATED: The VNF instance is instantiated.
instantiatedVnfInfo	Structure (inlined)	0..1	Information specific to an instantiated VNF instance. This attribute shall be present if the instantiateState attribute value is INSTANTIATED.
>flavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour applied to this VNF instance.
>vnfState	VnfOperationalStateType	1	State of the VNF instance.
>scaleStatus	ScaleInfo	0..N	Scale status of the VNF, one entry per aspect. Represents for every scaling aspect how "big" the VNF has been scaled w.r.t. that aspect. This attribute shall be present if the VNF supports scaling. See clause B.2 in annex B for an explanation of VNF scaling.
>extCpInfo	VnfExtCpInfo	1..N	Information about the external CPs exposed by the VNF instance.
>extVirtualLinkInfo	ExtVirtualLinkInfo	0..N	Information about the external VLs the VNF instance is connected to.
>extManagedVirtualLinkInfo	ExtManagedVirtualLinkInfo	0..N	Information about the externally-managed internal VLs of the VNF instance.
>monitoringParameters	MonitoringParameter	0..N	Active monitoring parameters.

Attribute name	Data type	Cardinality	Description
>localizationLanguage	String	0..1	Information about localization language of the VNF (includes e.g. strings in the VNFD). The localization languages supported by a VNF can be declared in the VNFD, and localization language selection can take place at instantiation time. The value shall comply with the format defined in IETF RFC 5646 [8].
>vnfcResourceInfo	VnfcResourceInfo	0..N	Information about the virtualised compute and storage resources used by the VNFCs of the VNF instance.
>vnfVirtualLinkResourceInfo	VnfVirtualLinkResourceInfo	0..N	Information about the virtualised network resources used by the VLs of the VNF instance.
>virtualStorageResourceInfo	VirtualStorageResourceInfo	0..N	Information about the virtualised storage resources used as storage for the VNF instance.
metadata	KeyValuePairs	0..1	Additional VNF-specific metadata describing the VNF instance. Metadata that are writeable are declared in the VNFD (see note 2). This attribute can be modified with the PATCH method.
extensions	KeyValuePairs	0..1	VNF-specific attributes that affect the lifecycle management of this VNF instance by the VNFM, or the lifecycle management scripts. Extensions that are writeable are declared in the VNFD (see note 2). This attribute can be modified with the PATCH method.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>indicators	Link	0..1	Indicators related to this VNF instance, if applicable.
>instantiate	Link	0..1	Link to the "instantiate" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance in NOT_INSTANTIATED state).
>terminate	Link	0..1	Link to the "terminate" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>scale	Link	0..1	Link to the "scale" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>scaleToLevel	Link	0..1	Link to the "scale_to_level" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>changeFlavour	Link	0..1	Link to the "change_flavour" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>heal	Link	0..1	Link to the "heal" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).

Attribute name	Data type	Cardinality	Description
>operate	Link	0..1	Link to the "operate" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>changeExtConn	Link	0..1	Link to the "change_ext_conn" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
NOTE 1: Modifying the value of this attribute shall not be performed when conflicts exist between the previous and the newly referred VNF package, i.e. when the new VNFD is not changed with respect to the previous VNFD apart from referencing to other VNF software images. In order to avoid misalignment of the VnfInstance with the current VNF's on-boarded VNF Package, the values of attributes in the VnfInstance that have corresponding attributes in the VNFD shall be kept in sync with the values in the VNFD. NOTE 2: ETSI GS NFV-SOL 001 [i.6] specifies the structure and format of the VNFD based on TOSCA specifications. NOTE 3: VNF configurable properties are sometimes also referred to as configuration parameters applicable to a VNF. Some of these are set prior to instantiation and cannot be modified if the VNF is instantiated, some are set prior to instantiation (are part of initial configuration) and can be modified later, and others can be set only after instantiation. The applicability of certain configuration may depend on the VNF and the required operation of the VNF at a certain point in time.			

5.5.2.3 Type: CreateVnfRequest

This type represents request parameters for the "Create VNF identifier" operation. It shall comply with the provisions defined in table 5.5.2.3-1.

Table 5.5.2.3-1: Definition of the CreateVnfRequest data type

Attribute name	Data type	Cardinality	Description
vnfdId	Identifier	1	Identifier that identifies the VNFD which defines the VNF instance to be created.
vnfInstanceName	String	0..1	Human-readable name of the VNF instance to be created.
vnfInstanceDescription	String	0..1	Human-readable description of the VNF instance to be created.

5.5.2.4 Type: InstantiateVnfRequest

This type represents request parameters for the "Instantiate VNF" operation. It shall comply with the provisions defined in table 5.5.2.4-1.

Table 5.5.2.4-1: Definition of the InstantiateVnfRequest data type

Attribute name	Data type	Cardinality	Description
flavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour to be instantiated.
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated.
extVirtualLinks	ExtVirtualLinkData	0..N	Information about external VLs to connect the VNF to.
extManagedVirtualLinks	ExtManagedVirtualLinkData	0..N	Information about internal VLs that are managed by the NFVO.
vimConnectionInfo	VimConnectionInfo	0..N	Information about VIM connections to be used for managing the resources for the VNF instance, or refer to external / externally-managed virtual links. This attribute shall only be supported and may be present if VNF-related resource management in direct mode is applicable.
localizationLanguage	String	0..1	Localization language of the VNF to be instantiated. The value shall comply with the format defined in IETF RFC 5646 [8].
additionalParams	KeyValuePairs	0..1	Additional input parameters for the instantiation process, specific to the VNF being instantiated, as declared in the VNFD as part of "InstantiateVnfOpConfig".

5.5.2.5 Type: ScaleVnfRequest

This type represents request parameters for the "Scale VNF" operation. It shall comply with the provisions defined in table 5.5.2.5-1. See clause B.2 in annex B for an explanation of VNF scaling.

Table 5.5.2.5-1: Definition of the ScaleVnfRequest data type

Attribute name	Data type	Cardinality	Description
type	Enum (inlined)	1	Indicates the type of the scale operation requested. Permitted values: - SCALE_OUT: adding additional VNFC instances to the VNF to increase capacity. - SCALE_IN: removing VNFC instances from the VNF in order to release unused capacity.
aspectId	IdentifierInVnfd	1	Identifier of the scaling aspect.
numberOfSteps	Integer	0..1	Number of scaling steps to be executed as part of this Scale VNF operation. It shall be a positive number and the default value shall be 1.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the scaling process, specific to the VNF being scaled, as declared in the VNFD as part of "ScaleVnfOpConfig".

5.5.2.6 Type: ScaleVnfToLevelRequest

This type represents request parameters for the "Scale VNF to Level" operation. It shall comply with the provisions defined in table 5.5.2.6-1. See clause B.2 in annex B for an explanation of VNF scaling.

Table 5.5.2.6-1: Definition of the ScaleVnfToLevelRequest data type

Attribute name	Data type	Cardinality	Description
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the target instantiation level of the current deployment flavour to which the VNF is requested to be scaled. See note.
scaleInfo	ScaleInfo	0..N	For each scaling aspect of the current deployment flavour, indicates the target scale level to which the VNF is to be scaled. See note.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the scaling process, specific to the VNF being scaled, as declared in the VNFD as part of "ScaleVnfToLevelOpConfig".
NOTE: Either the instantiationLevelId attribute or the scaleInfo attribute shall be included.			

5.5.2.7 Type: ChangeVnfFlavourRequest

This type represents request parameters for the "Change VNF flavour" operation. It shall comply with the provisions defined in table 5.5.2.7-1.

Table 5.5.2.7-1: Definition of the ChangeVnfFlavourRequest data type

Attribute name	Data type	Cardinality	Description
newFlavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour to be instantiated.
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated.
extVirtualLinks	ExtVirtualLinkData	0..N	Information about external VLs to connect the VNF to.
extManagedVirtualLinks	ExtManagedVirtualLinkData	0..N	Information about internal VLs that are managed by the NFVO.
vimConnectionInfo	VimConnectionInfo	0..N	Information about VIM connections to be used for managing the resources for the VNF instance, or refer to external / externally-managed virtual links. This attribute shall only be supported and may be present if VNF-related resource management in direct mode is applicable.
additionalParams	KeyValuePairs	0..1	Additional input parameters for the flavour change process, specific to the VNF being modified, as declared in the VNFD as part of "ChangeVnfFlavourOpConfig".

5.5.2.8 Type: TerminateVnfRequest

This type represents request parameters for the "Terminate VNF" operation. It shall comply with the provisions defined in table 5.5.2.8-1.

Table 5.5.2.8-1: Definition of the TerminateVnfRequest data type

Attribute name	Data type	Cardinality	Description
terminationType	Enum (inlined)	1	Indicates whether forceful or graceful termination is requested. See note. Permitted values: - FORCEFUL: The VNFM will shut down the VNF and release the resources immediately after accepting the request. - GRACEFUL: The VNFM will first arrange to take the VNF out of service after accepting the request. Once the operation of taking the VNF out of service finishes (irrespective of whether it has succeeded or failed) or once the timer value specified in the "gracefulTerminationTimeout" attribute expires, the VNFM will shut down the VNF and release the resources.
gracefulTerminationTimeout	Integer	0..1	This attribute is only applicable in case of graceful termination. It defines the time to wait for the VNF to be taken out of service before shutting down the VNF and releasing the resources. The unit is seconds. If not given and the "terminationType" attribute is set to "GRACEFUL", it is expected that the VNFM waits for the successful taking out of service of the VNF, no matter how long it takes, before shutting down the VNF and releasing the resources.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the termination process, specific to the VNF being terminated, as declared in the VNFD as part of "TerminateVnfOpConfig".
NOTE: If the VNF is still in service, requesting forceful termination can adversely impact network service.			

5.5.2.9 Type: HealVnfRequest

This type represents request parameters for the "Heal VNF" operation. It shall comply with the provisions defined in table 5.5.2.9-1.

Table 5.5.2.9-1: Definition of the HealVnfRequest data type

Attribute name	Data type	Cardinality	Description
cause	String	0..1	Indicates the reason why a healing procedure is required.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the healing process, specific to the VNF being healed, as declared in the VNFD as part of "HealVnfOpConfig".

5.5.2.10 Type: OperateVnfRequest

This type represents request parameters for the "Operate VNF" operation. It shall comply with the provisions defined in table 5.5.2.10-1.

Table 5.5.2.10-1: Definition of the OperateVnfRequest data type

Attribute name	Data type	Cardinality	Description
changeStateTo	VnfOperationalStateType	1	The desired operational state (i.e. started or stopped) to change the VNF to.
stopType	StopType	0..1	It signals whether forceful or graceful stop is requested. See note.
gracefulStopTimeout	Integer	0..1	The time interval (in seconds) to wait for the VNF to be taken out of service during graceful stop, before stopping the VNF. See note.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the process, specific to the VNF of which the operation status is changed, as declared in the VNFD as part of "OperateVnfOpConfig".
NOTE: The "stopType" and "gracefulStopTimeout" attributes shall be absent, when the "changeStateTo" attribute is equal to "STARTED". The "gracefulStopTimeout" attribute shall be present, when the "changeStateTo" is equal to "STOPPED" and the "stopType" attribute is equal to "GRACEFUL". The "gracefulStopTimeout" attribute shall be absent, when the "changeStateTo" attribute is equal to "STOPPED" and the "stopType" attribute is equal to "FORCEFUL". The request shall be treated as if the "stopType" attribute was set to "FORCEFUL", when the "changeStateTo" attribute is equal to "STOPPED" and the "stopType" attribute is absent.			

5.5.2.11 Type: ChangeExtVnfConnectivityRequest

This type represents request parameters for the "Change external VNF connectivity" operation to modify the external connectivity of a VNF instance. It shall comply with the provisions defined in table 5.5.2.11-1.

Table 5.5.2.11-1: Definition of the ChangeExtVnfConnectivityRequest data type

Attribute name	Data type	Cardinality	Description
extVirtualLinks	ExtVirtualLinkData	1..N	Information about external VLs to change (e.g. connect the VNF to).
vimConnectionInfo	VimConnectionInfo	0..N	Information about VIM connections to be used for managing the resources for the VNF instance, or refer to external virtual links. This attribute shall only be supported and may be present if VNF-related resource management in direct mode is applicable.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the process, specific to the VNF of which the external connectivity is changed, as declared in the VNFD as part of " ChangeExtVnfConnectivityOpConfig".

The following behaviour applies for the changes that can be performed with this operation:

- To change the connection of external CP instances based on certain external CPDs from a "source" external VL to a different "target" external VL, the identifier of the "target" external VL shall be sent in the "extVirtualLinkId" attribute of the "extVirtualLinks" parameter, and the "extCps" attributes of that parameter shall refer via the "cpdId" attribute to the external CPDs of the corresponding external connection point instances that are to be reconnected to the target external VL.

NOTE: This means that all CP instances based on a given external CPD will be reconnected. See clause B.3.3 in annex B for an illustration.

- To change the connectivity parameters of the external CPs connected to a particular external VL, including changing addresses, the identifier of that external VL shall be sent in the "extVirtualLinkId" attribute of the "extVirtualLinks" parameter, and the "extCps" attribute of that parameter shall contain at least those entries with modified parameters.

5.5.2.12 Type: VnfInfoModificationRequest

This type represents attribute modifications for an "Individual VNF instance" resource, i.e. modifications to a resource representation based on the "VnfInstance" data type. The attributes of "VnfInstance" that can be modified according to the provisions in clause 5.5.2.2 are included in the "VnfInfoModificationRequest" data type.

The "VnfInfoModificationRequest" data type shall comply with the provisions defined in table 5.5.2.12-1.

Table 5.5.2.12-1: Definition of the VnfInfoModificationRequest data type

Attribute name	Data type	Cardinality	Description
vnfInstanceName	String	0..1	New value of the "vnfInstanceName" attribute in "VnfInstance", or "null" to remove the attribute.
vnfInstanceDescription	String	0..1	New value of the "vnfInstanceDescription" attribute in "VnfInstance", or "null" to remove the attribute.
vnfPkgId	Identifier	0..1	New value of the "vnfPkgId" attribute in "VnfInstance". The value "null" is not permitted.
vnfConfigurableProperties	KeyValuePairs	0..1	Modifications of the "vnfConfigurableProperties" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [17]).
metadata	KeyValuePairs	0..1	Modifications of the "metadata" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [17]).
extensions	KeyValuePairs	0..1	Modifications of the "extensions" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [17]).
vimConnectionInfo	VimConnectionInfo	0..N	New content of certain entries in the "vimConnectionInfo" attribute array in "VnfInstance", as defined below this table.
vimConnectionInfoDeleteIds	Identifier	0..N	List of identifiers entries to be deleted from the "vimConnectionInfo" attribute array in "VnfInstance", to be used as "deleteIdList" as defined below this table.

The following provisions shall apply when modifying an attribute that is an array of objects of type "VimConnectionInfo".

Assumptions:

1) "oldList" is the array to be modified, "newList" is the array that contains the changes and "deleteIdList" is the array that contains the identifiers of those "oldList" entries to be deleted.

2) "oldEntry" is an entry in "oldList" and "newEntry" is an entry in "newList".

3) A "newEntry" has a "corresponding entry" if there exists an "oldEntry" that has the same content of "id" attribute as the "newEntry"; a "newEntry" has no corresponding entry if no such "oldEntry" exists.

4) In any array of "VimConnectionInfo" structures, the content of "id" is unique (i.e. there are no two entries with the same content of "id").

Provisions:

1) For each "newEntry" in "newList" that has no corresponding entry in "oldList", the "oldList" array shall be modified by adding that "newEntry".

2) For each "newEntry" in "newList" that has a corresponding "oldEntry" in "oldList", the value of "oldEntry" shall be replaced by the value of "newEntry".

3) For each entry in "deleteIdList", delete the entry in "oldList" that has the same content of the "id" attribute as the entry in "deleteIdList".

5.5.2.12a Type: VnfInfoModifications

This type represents attribute modifications that were performed on an "Individual VNF instance" resource. The attributes that can be included consist of those requested to be modified explicitly in the "VnfInfoModificationRequest" data structure, and additional attributes of the "VnfInstance" data structure that were modified implicitly e.g. when modifying the referenced VNF package.

The "VnfInfoModifications" data type shall comply with the provisions defined in table 5.5.2.12a-1.

Table 5.5.2.12a-1: Definition of the VnfInfoModifications data type

Attribute name	Data type	Cardinality	Description
vnfInstanceName	String	0..1	If present, this attribute signals modifications of the "vnfInstanceName" attribute in "VnfInstance" as defined in clause 5.5.2.12.
vnfInstanceDescription	String	0..1	If present, this attribute signals modifications of the "vnfInstanceDescription" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vnfConfigurableProperties	KeyValuePairs	0..1	If present, this attribute signals modifications of the "vnfConfigurableProperties" attribute in "VnfInstance", as defined in clause 5.5.2.12.
metadata	KeyValuePairs	0..1	If present, this attribute signals modifications of the "metadata" attribute in "VnfInstance", as defined in clause 5.5.2.12.
extensions	KeyValuePairs	0..1	If present, this attribute signals modifications of the "extensions" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vimConnectionInfo	VimConnectionInfo	0..N	If present, this attribute signals modifications of certain entries in the "vimConnectionInfo" attribute array in "VnfInstance", as defined in clause 5.5.2.12.
vimConnectionInfoDeleteIds	Identifier	0..N	If present, this attribute signals the deletion of certain entries in the "vimConnectionInfo" attribute array in "VnfInstance", as defined in clause 5.5.2.12.
vnfPkgId	Identifier	0..1	If present, this attribute signals modifications of the "vnfPkgId" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vnfdId	Identifier	0..1	If present, this attribute signals modifications of the "vnfdId" attribute in "VnfInstance". See note.
vnfProvider	String	0..1	If present, this attribute signals modifications of the "vnfProvider" attribute in "VnfInstance". See note.
vnfProductName	String	0..1	If present, this attribute signals modifications of the "vnfProductName" attribute in "VnfInstance". See note.
vnfSoftwareVersion	Version	0..1	If present, this attribute signals modifications of the "vnfSoftwareVersion" attribute in "VnfInstance". See note.
vnfdVersion	Version	0..1	If present, this attribute signals modifications of the "vnfdVersion" attribute in "VnfInstance". See note.
NOTE: If present, this attribute (which depends on the value of the "vnfPkgId" attribute) was modified implicitly following a request to modify the "vnfPkgId" attribute, by copying the value of this attribute from the VNFD in the VNF Package identified by the "vnfPkgId" attribute.			

5.5.2.13 Type: VnfLcmOpOcc

This type represents a VNF lifecycle management operation occurrence. It shall comply with the provisions defined in table 5.5.2.13-1.

Table 5.5.2.13-1: Definition of the VnfLcmOpOcc data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this VNF lifecycle management operation occurrence.
operationState	LcmOperationStateType	1	The state of the LCM operation.
stateEnteredTime	DateTime	1	Date-time when the current state was entered.
startTime	DateTime	1	Date-time of the start of the operation.

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance to which the operation applies.
grantId	Identifier	0..1	Identifier of the grant related to this VNF LCM operation occurrence, if such grant exists.
operation	LcmOperationType	1	Type of the actual LCM operation represented by this VNF LCM operation occurrence.
isAutomaticInvocation	Boolean	1	Set to true if this VNF LCM operation occurrence has been triggered by an automated procedure inside the VNFM (i.e. ScaleVnf / ScaleVnfToLevel triggered by auto-scale, or HealVnf triggered by auto-heal). Set to false otherwise.
operationParams	Object	0..1	Input parameters of the LCM operation. This attribute shall be formatted according to the request data type of the related LCM operation. The following mapping between operationType and the data type of this attribute shall apply: • INSTANTIATE: InstantiateVnfRequest • SCALE: ScaleVnfRequest • SCALE_TO_LEVEL: ScaleVnfToLevelRequest • CHANGE_FLAVOUR: ChangeVnfFlavourRequest • OPERATE: OperateVnfRequest • HEAL: HealVnfRequest • CHANGE_EXT_CONN: ChangeExtVnfConnectivityRequest • TERMINATE: TerminateVnfRequest • MODIFY_INFO: VnfInfoModificationRequest This attribute shall be present if this data type is returned in a response to reading an individual resource, and may be present according to the chosen attribute selector parameter if this data type is returned in a response to a query of a container resource.
isCancelPending	Boolean	1	If the VNF LCM operation occurrence is in "STARTING", "PROCESSING" or "ROLLING_BACK" state and the operation is being cancelled, this attribute shall be set to true. Otherwise, it shall be set to false.
cancelMode	CancelModeType	0..1	The mode of an ongoing cancellation. Shall be present when isCancelPending=true, and shall be absent otherwise.
error	ProblemDetails	0..1	If "operationState" is "FAILED_TEMP" or "FAILED" or "operationState" is "PROCESSING" or "ROLLING_BACK" and previous value of "operationState" was "FAILED_TEMP", this attribute shall be present and contain error information, unless it has been requested to be excluded via an attribute selector.
resourceChanges	Structure (inlined)	0..1	This attribute contains information about the cumulative changes to virtualised resources that were performed so far by the LCM operation since its start, if applicable.
>affectedVnfcs	AffectedVnfc	0..N	Information about VNFC instances that were affected during the lifecycle operation. See note.
>affectedVirtualLinks	AffectedVirtualLink	0..N	Information about VL instances that were affected during the lifecycle operation. See note.
>affectedVirtualStorages	AffectedVirtualStorage	0..N	Information about virtualised storage instances that were affected during the lifecycle operation. See note.

Attribute name	Data type	Cardinality	Description
changedInfo	VnfInfoModifications	0..1	Information about the changed VNF instance information, including VNF configurable properties, if applicable. See note.
changedExtConnectivity	ExtVirtualLinkInfo	0..N	Information about changed external connectivity, if applicable. See note.
_links	Structure (inline)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>vnfInstance	Link	1	Link to the VNF instance that the operation applies to.
>grant	Link	0..1	Link to the grant for this operation, if one exists.
>cancel	Link	0..1	Link to the task resource that represents the "cancel" operation for this VNF LCM operation occurrence, if cancelling is currently allowed.
>retry	Link	0..1	Link to the task resource that represents the "retry" operation for this VNF LCM operation occurrence, if retrying is currently allowed.
>rollback	Link	0..1	Link to the task resource that represents the "rollback" operation for this VNF LCM operation occurrence, if rolling back is currently allowed.
>fail	Link	0..1	Link to the task resource that represents the "fail" operation for this VNF LCM operation occurrence, if declaring as failed is currently allowed.
NOTE: This allows the NFVO to obtain the information contained in the latest "result" notification if it has not received it due to an error or a wrongly configured subscription filter.			

5.5.2.14 Type: CancelMode

This type represents a parameter to select the mode of cancelling an ongoing VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.2.14-1.

Table 5.5.2.14-1: Definition of the CancelMode data type

Attribute name	Data type	Cardinality	Description
cancelMode	CancelModeType	1	Cancellation mode to apply.

5.5.2.15 Type: LccnSubscriptionRequest

This type represents a subscription request related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.2.15-1.

Table 5.5.2.15-1: Definition of the LccnSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	LifecycleChangeNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

5.5.2.16 Type: LccnSubscription

This type represents a subscription related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.2.16-1.

Table 5.5.2.16-1: Definition of the LccnSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	LifecycleChangeNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.

5.5.2.17 Type: VnfLcmOperationOccurrenceNotification

This type represents a VNF lifecycle management operation occurrence notification, which informs the receiver of changes in the VNF lifecycle caused by a VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.2.17-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM when there is a change in the VNF lifecycle caused by a VNF LCM operation occurrence, including:

- Instantiation of the VNF
- Scaling of the VNF instance (including auto-scaling)
- Healing of the VNF instance (including auto-healing)
- Change of the state of the VNF instance (i.e. Operate VNF)
- Change of the deployment flavour of the VNF instance
- Change of the external connectivity of the VNF instance
- Termination of the VNF instance
- Modification of VNF instance information and/or VNF configurable properties through the "PATCH" method on the "Individual VNF instance" resource

If this is the initial notification about the start of a VNF LCM operation occurrence, it is assumed that the notification is sent by the VNFM before any action (including sending the grant request) is taken as part of the LCM operation. Due to possible race conditions, the "start" notification, the grant request and the LCM operation acknowledgment can arrive in any order at the NFVO, and the NFVO shall be able to handle such a situation.

If this is a notification about a final or intermediate result state of a VNF LCM operation occurrence, the notification shall be sent after all related actions of the LCM operation that led to this state have been executed.

The new state shall be set in the VnfLcmOpOcc resource before the notification about the state change is sent.

See clause 5.6.2.2 for further provisions regarding sending this notification in case of handling LCM operation errors.

Table 5.5.2.17-1: Definition of the VnfLcmOperationOccurrenceNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to " VnfLcmOperationOccurrenceNotification " for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
notificationStatus	Enum (inlined)	1	Indicates whether this notification reports about the start of a lifecycle operation or the result of a lifecycle operation. Permitted values: - START: Informs about the start of the VNF LCM operation occurrence. - RESULT: Informs about the final or intermediate result of the VNF LCM operation occurrence.
operationState	LcmOperationStateType	1	The state of the VNF LCM operation occurrence.
vnfInstanceId	Identifier	1	The identifier of the VNF instance affected.
operation	LcmOperationType	1	The lifecycle management operation.
isAutomaticInvocation	Boolean	1	Set to true if this VNF LCM operation occurrence has been triggered by an automated procedure inside the VNFM (i.e. ScaleVnf / ScaleVnfToLevel triggered by auto-scale, or HealVnf triggered by auto-heal). Set to false otherwise.
vnfLcmOpOccId	Identifier	1	The identifier of the VNF lifecycle management operation occurrence associated to the notification.
affectedVnfcs	AffectedVnfc	0..N	Information about VNFC instances that were affected during the lifecycle operation. See note.
affectedVirtualLinks	AffectedVirtualLink	0..N	Information about VL instances that were affected during the lifecycle operation. See note.
affectedVirtualStorages	AffectedVirtualStorage	0..N	Information about virtualised storage instances that were affected during the lifecycle operation. See note.
changedInfo	VnfInfoModifications	0..1	Information about the changed VNF instance information, including changed VNF configurable properties. Shall be present if the "notificationStatus" is set to "RESULT" and the operation has performed any changes to VNF instance information, including VNF configurable properties. Shall be absent otherwise.
changedExtConnectivity	ExtVirtualLinkInfo	0..N	Information about changed external connectivity, if this notification represents the result of a lifecycle operation occurrence. Shall be present if the "notificationStatus" is set to "RESULT" and the "operation" is set to "CHANGE_EXT_CONN". Shall be absent otherwise.
error	ProblemDetails	0..1	Details of the latest error, if one has occurred during executing the LCM operation (see clause 4.3.5). Shall be present if the "operationState" attribute is "FAILED_TEMP" or "FAILED", and shall be absent otherwise.
_links	LccnLinks	1	Links to resources related to this notification.
NOTE: Shall be present if the "notificationStatus" is set to "RESULT" and the operation has performed any resource modification. Shall be absent otherwise. This attribute contains information about the cumulative changes to virtualised resources that were performed so far by the VNF LCM operation occurrence and by any of the error handling procedures for that operation occurrence.			

5.5.2.18 Type: VnfIdentifierCreationNotification

This type represents a VNF identifier creation notification, which informs the receiver of the creation of a new VNF instance resource and the associated VNF instance identifier. It shall comply with the provisions defined in table 5.5.2.18-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM when it has created a VNF instance resource and the associated VNF instance identifier.

Table 5.5.2.18-1: Definition of the VnfIdentifierCreationNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIdentifierCreationNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfInstanceId	Identifier	1	The created VNF instance identifier.
links	LccnLinks	1	Links to resources related to this notification.

5.5.2.19 Type: VnfIdentifierDeletionNotification

This type represents a VNF identifier deletion notification, which informs the receiver of the deletion of a new VNF instance resource and the associated VNF instance identifier. It shall comply with the provisions defined in table 5.5.2.19-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM when it has deleted a VNF instance resource and the associated VNF instance identifier.

Table 5.5.2.19-1: Definition of the VnfIdentifierDeletionNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIdentifierDeletionNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfInstanceId	Identifier	1	The deleted VNF instance identifier.
links	LccnLinks	1	Links to resources related to this notification.

5.5.3 Referenced structured data types

5.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

5.5.3.2 Type: ExtVirtualLinkInfo

This type represents information about an external VL. It shall comply with the provisions defined in table 5.5.3.2-1.

Table 5.5.3.2-1: Definition of the ExtVirtualLinkInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the external VL and the related external VL information instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
resourceHandle	ResourceHandle	1	Reference to the resource realizing this VL.
extLinkPorts	ExtLinkPortInfo	0..N	Link ports of this VL.

5.5.3.3 Type: ExtManagedVirtualLinkInfo

This type provides information about an externally-managed virtual link. It shall comply with the provisions defined in table 5.5.3.3-1.

Table 5.5.3.3-1: Definition of the ExtManagedVirtualLinkInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the externally-managed internal VL and the related externally-managed VL information instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
vnfVirtualLinkDescId	IdentifierInVnfd	1	Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource.
vnfLinkPorts	VnfLinkPortInfo	0..N	Link ports of this VL.

5.5.3.4 Type: ScaleInfo

This type represents the scale level of a VNF instance related to a scaling aspect. It shall comply with the provisions defined in table 5.5.3.4-1.

Table 5.5.3.4-1: Definition of the ScaleInfo data type

Attribute name	Data type	Cardinality	Description
aspectId	IdentifierInVnfd	1	Identifier of the scaling aspect.
scaleLevel	Integer	1	Indicates the scale level. The minimum value shall be 0 and the maximum value shall be <= maxScaleLevel as described in the VNFD.

5.5.3.5 Type: VnfcResourceInfo

This type represents the information on virtualised compute and storage resources used by a VNFC in a VNF instance. It shall comply with the provisions defined in table 5.5.3.5-1.

Table 5.5.3.5-1: Definition of the VnfcResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VnfcResourceInfo instance.
vduId	IdentifierInVnfd	1	Reference to the applicable VDU in the VNFD. See note.
computeResource	ResourceHandle	1	Reference to the VirtualCompute resource.
storageResourceIds	IdentifierInVnf	0..N	References to the VirtualStorage resources. The value refers to a VirtualStorageResourceInfo item in the VnfInstance.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
vnfcCpInfo	Structure (inlined)	0..N	CPs of the VNFC instance. Shall be present when that particular CP of the VNFC instance is associated to an external CP of the VNF instance. May be present otherwise.
>id	IdentifierInVnf	1	Identifier of this VNFC CP instance and the associated array entry.
>cpdId	IdentifierInVnfd	1	Identifier of the VDU CPD, cpdId, in the VNFD. See note.
>vnfExtCpId	IdentifierInVnf	0..1	When the VNFC CP is exposed as external CP of the VNF, the identifier of this external VNF CP.
>cpProtocolInfo	CpProtocolInfo	0..N	Network protocol information for this CP.
>vnfLinkPortId	IdentifierInVnf	0..1	Identifier of the "vnfLinkPorts" structure in the "VnfVirtualLinkResourceInfo" structure. Shall be present if the CP is associated to a link port.
>metadata	KeyValuePairs	0..1	Metadata about this CP.

Attribute name	Data type	Cardinality	Description
metadata	KeyValuePairs	0..1	Metadata about this resource.
NOTE: ETSI GS NFV-SOL 001 [i.6] specifies the structure and format of the VNFD based on TOSCA specifications.			

5.5.3.6 Type: VnfVirtualLinkResourceInfo

This type represents the information that allows addressing a virtualised resource that is used by an internal VL instance in a VNF instance. It shall comply with the provisions defined in table 5.5.3.6-1.

Table 5.5.3.6-1: Definition of the VnfVirtualLinkResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VnfVirtualLinkResourceInfo instance.
vnfVirtualLinkDescId	IdentifierInVnfd	1	Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
vnfLinkPorts	VnfLinkPortInfo	0..N	Links ports of this VL. Shall be present when the linkPort is used for external connectivity by the VNF (refer to VnfLinkPortInfo). May be present otherwise.
metadata	KeyValuePairs	0..1	Metadata about this resource.

5.5.3.7 Type: VirtualStorageResourceInfo

This type represents the information that allows addressing a virtualised resource that is used by a VNF instance. It shall comply with the provisions defined in table 5.5.3.7-1.

Table 5.5.3.7-1: Definition of the VirtualStorageResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VirtualStorageResourceInfo instance.
virtualStorageDescId	IdentifierInVnfd	1	Identifier of the VirtualStorageDesc in the VNFD.
storageResource	ResourceHandle	1	Reference to the VirtualStorage resource.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
metadata	KeyValuePairs	0..1	Metadata about this resource.

5.5.3.8 Type: VnfLinkPortInfo

This type represents a link port of an internal VL of a VNF. It shall comply with the provisions defined in table 5.5.3.8-1.

Table 5.5.3.8-1: Definition of the VnfLinkPortInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised network resource realizing this link port.
cpInstanceId	IdentifierInVnf	0..1	When the link port is used for external connectivity by the VNF, this attribute represents the identifier associated with this link port. When the link port is used for internal connectivity in the VNF, this attribute represents the VNFC CP to be connected to this link port. Shall be present when the link port is used for external connectivity by the VNF. May be present if used to reference a VNFC CP instance. There shall be at most one link port associated with any external connection point instance or internal connection point (i.e. VNFC CP) instance. The value refers to an "extCpInfo" item in the VnfInstance or a "vnfcCpInfo" item of a "vnfcResouceInfo" item in the VnfInstance.
cpInstanceType	0..1	Enum (inlined)	Type of the CP instance that is identified by cpInstanceId. Shall be present if "cpInstanceId" is present, and shall be absent otherwise. Permitted values: VNFC_CP: The link port is connected to a VNFC CP. EXT_CP: The link port is associated to an external CP.

5.5.3.9 Type: ExtLinkPortInfo

This type represents information about a link port of an external VL, i.e. a port providing connectivity for the VNF to an NS VL. It shall comply with the provisions defined in table 5.5.3.9-1.

Table 5.5.3.9-1: Definition of the ExtLinkPortInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised resource realizing this link port.
cpInstanceId	IdentifierInVnf	0..1	Identifier of the external CP of the VNF connected to this link port. There shall be at most one link port associated with any external connection point instance. The value refers to an "extCpInfo" item in the VnfInstance.

5.5.3.9a Type: ExtLinkPortData

This type represents an externally provided link port to be used to connect an external connection point to an external VL. It shall comply with the provisions defined in table 5.5.3.9a-1.

Table 5.5.3.9a-1: Definition of the ExtLinkPortData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised resource realizing this link port.

5.5.3.9b Type: CpProtocolInfo

This type describes the protocol layer(s) that a CP uses together with protocol-related information, like addresses. It shall comply with the provisions defined in table 5.5.3.9b-1.

Table 5.5.3.9b-1: Definition of the CpProtocolInfo data type

Attribute name	Data type	Cardinality	Description
layerProtocol	Enum (inlined)	1	The identifier of layer(s) and protocol(s) associated to the network address information. Permitted values: IP_OVER_ETHERNET See note.
ipOverEthernet	IpOverEthernetAddressInfo	0..1	IP addresses over Ethernet to assign to the extCP instance. Shall be present if layerProtocol is equal to " IP_OVER_ETHERNET", and shall be absent otherwise.
NOTE: This attribute allows to signal the addition of further types of layer and protocol in future versions of the present document in a backwards-compatible way. In the current version of the present document, only IP over Ethernet is supported.			

5.5.3.10 Type: IpOverEthernetAddressInfo

This type represents information about a network address that has been assigned. It shall comply with the provisions defined in table 5.5.3.10-1.

Table 5.5.3.10-1: Definition of the IpOverEthernetAddressInfo data type

Attribute name	Data type	Cardinality	Description
macAddress	MacAddress	0..1	MAC address, if assigned. See note 1.
ipAddresses	Structure (inlined)	0..N	Addresses assigned to the CP instance. Each entry represents IP addresses assigned by fixed or dynamic IP address assignment per subnet.
>type	Enum (inlined)	1	The type of the IP addresses. Permitted values: IPV4, IPV6.
>addresses	IpAddress	0..N	Fixed addresses assigned (from the subnet defined by "subnetId" if provided). See note 2.
>isDynamic	Boolean	0..1	Indicates whether this set of addresses was assigned dynamically (true) or based on address information provided as input from the API consumer (false). Shall be present if "addresses" is present and shall be absent otherwise.
>addressRange	Structure (inlined)	0..1	An IP address range used, e.g. in case of egress connections. See note 2.
>>minAddress	IpAddress	1	Lowest IP address belonging to the range
>>maxAddress	IpAddress	1	Highest IP address belonging to the range

Attribute name	Data type	Cardinality	Description
>subnetId	IdentifierInVim	0..1	Subnet defined by the identifier of the subnet resource in the VIM. In case this attribute is present, IP addresses are bound to that subnet.
NOTE 1: At least one of "macAddress" or "ipAddresses" shall be present. NOTE 2: Exactly one of "addresses" or "addressRange" shall be present.			

5.5.3.11 Type: MonitoringParameter

This type represents a monitoring parameter that is tracked by the VNFM, e.g. for auto-scaling purposes. It shall comply with the provisions defined in table 5.5.3.11-1.

Valid monitoring parameters of a VNF are defined in the VNFD.

NOTE: ETSI GS NFV-SOL 001 [i.6] specifies the structure and format of the VNFD based on TOSCA specifications.

Table 5.5.3.11-1: Definition of the MonitoringParameter data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of the monitoring parameter defined in the VNFD.
name	String	0..1	Human readable name of the monitoring parameter, as defined in the VNFD.
performanceMetric	String	1	Performance metric that is monitored. This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [20].

5.5.3.12 Type: LifecycleChangeNotificationsFilter

This type represents a subscription filter related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.3.12-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 5.5.3.12-1: Definition of the LifecycleChangeNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: - VnfLcmOperationOccurrenceNotification - VnfIdentifierCreationNotification - VnfIdentifierDeletionNotification See note.
operationTypes	LcmOperationType	0..N	Match particular VNF lifecycle operation types for the notification of type VnfLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "VnfLcmOperationOccurrenceNotification", and shall be absent otherwise.

Attribute name	Data type	Cardinality	Description
operationStates	LcmOperationStateType	0..N	Match particular LCM operation state values as reported in notifications of type VnfLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "VnfLcmOperationOccurrenceNotification", and shall be absent otherwise.
NOTE: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.			

5.5.3.13 Type: AffectedVnfc

This type provides information about added, deleted, modified and temporary VNFCs. It shall comply with the provisions in table 5.5.3.13-1.

Table 5.5.3.13-1: Definition of the AffectedVnfc data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the Vnfc instance, identifying the applicable "vnfcResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
vduId	IdentifierInVnfd	1	Identifier of the related VDU in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change Permitted values: - ADDED - REMOVED - MODIFIED - TEMPORARY For a temporary resource, an AffectedVnfc structure exists as long as the temporary resource exists.
computeResource	ResourceHandle	1	Reference to the VirtualCompute resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VnfcResourceInfo structure.
affectedVnfcCpIds	IdentifierInVnf	0..N	Identifiers of CP(s) of the VNFC instance that were affected by the change. Shall be present for those affected CPs of the VNFC instance that are associated to an external CP of the VNF instance. May be present for further affected CPs of the VNFC instance.
addedStorageResourceIds	IdentifierInVnf	0..N	References to VirtualStorage resources that have been added. Each value refers to a VirtualStorageResourceInfo item in the VnfInstance that was added to the VNFC. It shall be provided if at least one storage resource was added to the VNFC.

Attribute name	Data type	Cardinality	Description
removedStorageResourceIds	IdentifierInVnf	0..N	References to VirtualStorage resources that have been removed. The value contains the identifier of a VirtualStorageResourceInfo item that has been removed from the VNFC, and might no longer exist in the VnfInstance. It shall be provided if at least one storage resource was removed from the VNFC.

5.5.3.14 Type: AffectedVirtualLink

This type provides information about added, deleted, modified and temporary VLs. It shall comply with the provisions in table 5.5.3.14-1.

Table 5.5.3.14-1: Definition of the AffectedVirtualLink data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the virtual link instance, identifying the applicable "vnfVirtualLinkResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
vnfVirtualLinkDescId	IdentifierInVnfd	1	Identifier of the related VLD in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change. Permitted values: - ADDED - REMOVED - MODIFIED - TEMPORARY - LINK_PORT_ADDED - LINK_PORT_REMOVED For a temporary resource, an AffectedVirtualLink structure exists as long as the temporary resource exists.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VnfVirtualLinkResourceInfo structure.

5.5.3.15 Type: AffectedVirtualStorage

This type provides information about added, deleted, modified and temporary virtual storage resources. It shall comply with the provisions in table 5.5.3.15-1.

Table 5.5.3.15-1: Definition of the AffectedVirtualStorage data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the storage instance, identifying the applicable "virtualStorageResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
virtualStorageDescId	IdentifierInVnfd	1	Identifier of the related VirtualStorage descriptor in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change. Permitted values: - ADDED - REMOVED - MODIFIED - TEMPORARY For a temporary resource, an AffectedVirtualStorage structure exists as long as the temporary resource exists.
storageResource	ResourceHandle	1	Reference to the VirtualStorage resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VirtualStorageResourceInfo structure.

5.5.3.16 Type: LccnLinks

This type represents the links to resources that a notification can contain. It shall comply with the provisions defined in table 5.5.3.16-1.

Table 5.5.3.16-1: Definition of the LccnLinks data type

Attribute name	Data type	Cardinality	Description
vnfInstance	NotificationLink	1	Link to the resource representing the VNF instance to which the notified change applies.
subscription	NotificationLink	1	Link to the related subscription.
vnfLcmOpOcc	NotificationLink	0..1	Link to the VNF lifecycle management operation occurrence that this notification is related to. Shall be present if there is a related lifecycle operation occurrence.

5.5.3.17 Type: VnfExtCpInfo

This type represents information about an external CP of a VNF. It shall comply with the provisions defined in table 5.5.3.17-1.

Table 5.5.3.17-1: Definition of the VnfExtCpInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the external CP instance and the related information instance.
cpdId	IdentifierInVnfd	1	Identifier of the external CPD, VnfExtCpd, in the VNFD.
cpProtocolInfo	CpProtocolInfo	1..N	Network protocol information for this CP.
extLinkPortId	Identifier	0..1	Identifier of the "extLinkPortInfo" structure inside the "extVirtualLinkInfo" structure. Shall be present if the CP is associated to a link port.
metadata	KeyValuePairs	0..1	Metadata about this external CP.
associatedVnfcCpId	Identifier	0..1	Identifier of the "vnfcCpInfo" structure in "VnfcResourceInfo" structure that represents the VNFC CP which is exposed by this external CP instance. Shall be present in case this CP instance maps to a VNFC CP. See note.
associatedVnfVirtualLinkId	Identifier	0..1	Identifier of the "VnfVirtualLinkResourceInfo" structure that represents the internal VL which is exposed by this external CP instance. Shall be present in case this CP instance maps to an internal VL. See note.
NOTE: The attributes "associatedVnfcCpId" and "associatedVnfVirtualLinkId" are mutually exclusive. One and only one shall be present.			

5.5.4 Referenced simple data types and enumerations

5.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

5.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

5.5.4.3 Enumeration: VnfOperationalStateType

The enumeration VnfOperationalStateType shall comply with the provisions defined in table 5.5.4.3-1.

Table 5.5.4.3-1: Enumeration VnfOperationalStateType

Enumeration value	Description
STARTED	The VNF instance is up and running.
STOPPED	The VNF instance has been shut down.

5.5.4.4 Enumeration: StopType

The enumeration StopType shall comply with the provisions defined in table 5.5.4.4-1.

Table 5.5.4.4-1: Enumeration StopType

Enumeration value	Description
FORCEFUL	The VNFM will stop the VNF immediately after accepting the request.
GRACEFUL	The VNFM will first arrange to take the VNF out of service after accepting the request. Once that operation is successful or once the timer value specified in the "gracefulStopTimeout" attribute expires, the VNFM will stop the VNF.

5.5.4.5 Enumeration: LcmOperationStateType

The enumeration LcmOperationStateType shall comply with the provisions defined in table 5.5.4.5-1. More information of the meaning of the states can be found in clause 5.6.2.2.

Table 5.5.4.5-1: Enumeration LcmOperationStateType

Enumeration value	Description
STARTING	The LCM operation is starting.
PROCESSING	The LCM operation is currently in execution.
COMPLETED	The LCM operation has been completed successfully.
FAILED_TEMP	The LCM operation has failed and execution has stopped, but the execution of the operation is not considered to be closed.
FAILED	The LCM operation has failed and it cannot be retried or rolled back, as it is determined that such action will not succeed.
ROLLING_BACK	The LCM operation is currently being rolled back.
ROLLED_BACK	The LCM operation has been successfully rolled back, i.e. The state of the VNF prior to the original operation invocation has been restored as closely as possible.

5.5.4.6 Enumeration: CancelModeType

The enumeration CancelModeType defines the valid modes of cancelling a VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.4.6-1.

Table 5.5.4.6-1: Enumeration CancelModeType

Enumeration value	Description
GRACEFUL	If the VNF LCM operation occurrence is in "PROCESSING" or "ROLLING_BACK" state, the VNFM shall not start any new resource management operation and shall wait for the ongoing resource management operations in the underlying system, typically the VIM, to finish execution or to time out. After that, the VNFM shall put the operation occurrence into the FAILED_TEMP state. If the VNF LCM operation occurrence is in "STARTING" state, the VNFM shall not start any resource management operation and shall wait for the granting request to finish execution or time out. After that, the VNFM shall put the operation occurrence into the ROLLED_BACK state.
FORCEFUL	If the VNF LCM operation occurrence is in "PROCESSING" or "ROLLING_BACK" state, the VNFM shall not start any new resource management operation, shall cancel the ongoing resource management operations in the underlying system, typically the VIM, and shall wait for the cancellation to finish or to time out. After that, the VNFM shall put the operation occurrence into the FAILED_TEMP state. If the VNF LCM operation occurrence is in "STARTING" state, the VNFM shall not start any resource management operation and put the operation occurrence into the ROLLED_BACK state.

5.5.4.7 Enumeration: LcmOperationType

The enumeration LcmOperationType defines the permitted values to represent VNF lifecycle operation types in VNF lifecycle management operation occurrence resources and VNF lifecycle management operation occurrence notifications. It shall comply with the provisions defined in table 5.5.4.7-1.

Table 5.5.4.7-1: Enumeration LcmOperationType

Enumeration value	Description
INSTANTIATE	Represents the "Instantiate VNF" LCM operation.
SCALE	Represents the "Scale VNF" LCM operation.
SCALE_TO_LEVEL	Represents the "Scale VNF to Level" LCM operation.
CHANGE_FLAVOUR	Represents the "Change VNF Flavour" LCM operation.
TERMINATE	Represents the "Terminate VNF" LCM operation.
HEAL	Represents the "Heal VNF" LCM operation.
OPERATE	Represents the "Operate VNF" LCM operation.
CHANGE_EXT_CONN	Represents the "Change external VNF connectivity" LCM operation.
MODIFY_INFO	Represents the "Modify VNF Information" LCM operation.

5.6 Handling of errors during VNF lifecycle management operations

5.6.1 Basic concepts (informative)

5.6.1.1 Motivation

VNF lifecycle management operation occurrences can fail. Failure can be caused by multiple reasons, which generally fall into the following categories:

- Transient errors which do not require intervention from a human operator or a higher-layer management entity for resolution, e.g. momentary network outage.
- "Permanent" errors which require such intervention.

It is unreasonable to expect that all errors can be resolved automatically, therefore the possibility of intervention will usually be incorporated in the system design as acknowledged means of error resolution.

5.6.1.2 Failure resolution strategies: Retry and Rollback

Most transient errors are handled best with a retry mechanism. Retry might happen automatically at the point of failure within the same LCM workflow (where it makes sense to limit the number of automatic retries). It is important to strive for designing retry operations that have no unintended side effects from the original invocation of the operation. This is called *idempotent retry*. Idempotent retry can also be used as an on-demand error resolution mechanism (see below) if the original operation failed because of a condition that has been resolved manually by the human operator or by a higher-level management entity, so idempotent retry is suitable for general error resolution in most cases.

However, even if a system is designed with idempotent retry capabilities, eventual success of the operation cannot be guaranteed. In this case, the resolution of the inconsistent state can be attempted by requesting to roll back the changes made by the operation. Therefore, rollback as an error handling strategy is also desired to be allowed in the system design.

In many cases, idempotent retry can resolve transient errors and lead to success eventually. Depending on the situation, rollback followed by a repetition of the operation could take longer than a successful retry, as rollback first removes allocated resources and then the repetition of the operation allocates them again, which costs time.

Therefore, it often makes sense to perform first idempotent retry, which is followed by rollback if the retry has failed. Idempotent retry is meaningful and useful for all operation types, but for some operations rollback is better suited and has a better chance of success. In general, rollback is well-suited for additive operations such as InstantiateVnf or scale out, while ill-suited for subtractive ones such as scale in or TerminateVnf, or for HealVnf.

Both rollback and idempotent retry can fail. In that case, the system can be left in an inconsistent state after a failed operation, which requires resolution by a higher-level entity such as NFVO or human operator.

5.6.1.3 Error handling at VNFM and NFVO

If the VNFM executes an LCM workflow and encounters a problem, the following options are possible:

- Stop on first error:
  - Once the VNFM encounters an error, the normal execution of the LCM workflow is interrupted, and an error handling procedure is triggered (automatic retry, automatic rollback, automatic fail, escalate). See the paragraphs below for description of error handling procedures.
  - It is assumed that all VNFs and all VNFMs support "stop on first error".

EXAMPLE 1: NFVO is attempting to instantiate a VNF with 100 VNFCs. The first 97 VNFCs are instantiated successfully, however, an error occurs when attempting to instantiate VNFC #98. The VNFM stops execution and chooses which of the error handling options it invokes (note that it even could try multiple options after each other).

- Best Effort:
  - Each time the VNFM encounters an error, it is decided whether the execution of a part or all of the remaining steps of the LCM workflow is performed, or whether the execution is interrupted and an error handling procedure is triggered (automatic retry, automatic rollback, automatic fail, escalate). See the paragraphs below for description of error handling procedures.
  - Support of "best effort" requires a suitable workflow design.
  - It is therefore assumed that not all VNFs and not all VNFMs support "best effort".

EXAMPLE 2: Same example as above. After the error occurs attempting to instantiate VNFC #98, the VNFM continues by creating #99 and #100, and then chooses which error handling options it invokes.

The VNFM has the following error handling procedures to react to errors (see clause 5.6.1.2 for general elaboration regarding retry and rollback):

- Automatic Retry: The VNFM retries (once or more) to continue the execution of the workflow without involving an external entity. Automatic retry of failed parts of the workflow might even be built into the workflow itself. Retry can eventually succeed or fail. Successful retry leads to the LCM operation to be reported as successful. Failed retry is typically escalated.
- Automatic Rollback: The VNFM rolls back the VNF to the state prior to starting the LCM operation without involving an external entity. Rollback can eventually succeed or can fail, preventing the VNF from reaching that previous state. Successful rollback leads to the LCM operation to be reported as rolled back. Failed rollback is typically escalated.
- Escalate: After failed automatic retry/retries, automatic rollback is typically not the first option in most situations, but the error is preferably reported to the NFVO for further resolution. The same applies if no automatic error resolution was attempted by the VNFM, or if automated rollback has failed. This is done by sending a VNF LCM operation occurrence notification.
- Unresolvable Error: The VNFM determines that the operation has failed and definitely cannot be recovered (e.g. if no retry and no rollback is possible), and that escalating the error to the NFVO will have no chance to lead to a resolution either. In this case, the VNFM would report that the operation has terminally failed. After that, other means of resolution can be attempted, such as the invocation of HealVnf, or manual procedures using the GUI of the VNFM or VIM to release stranded resources.

The NFVO has the following error handling procedures to react to error reports from the VNFM:

- On-demand retry: After the VNFM has reported the error to the NFVO, the NFVO or the human operator takes steps to resolve the situation that has led to the occurrence of the error. Subsequently, the retry of the operation is triggered towards the VNFM by the NFVO via the VNF LCM interface.
- On-demand rollback: After the VNFM has reported the error to the NFVO, and after the NFVO or the human operator has decided to roll back the operation, the rollback of the operation is triggered towards the VNFM by the NFVO via the VNF LCM interface.

- Fail: After the VNFM has reported the error to the NFVO, and after the NFVO or the human operator has determined that neither on-demand retry nor on-demand rollback will fix the error, the LCM operation can be declared as terminally failed towards the VNFM by the NFVO via the VNF LCM interface. After that, other means of resolution can be attempted, such as the invocation of HealVnf, or manual procedures using the GUI of the VNFM or VIM to release stranded resources.

5.6.2 States and state transitions of a VNF lifecycle management operation occurrence

5.6.2.1 General

A VNF lifecycle management operation occurrence supports a number of states and error handling operations. The states and state transitions that shall be supported are shown in figure 5.6.2.1-1. Transitions labelled with underlined text represent error handling operations; other transitions represent conditions.

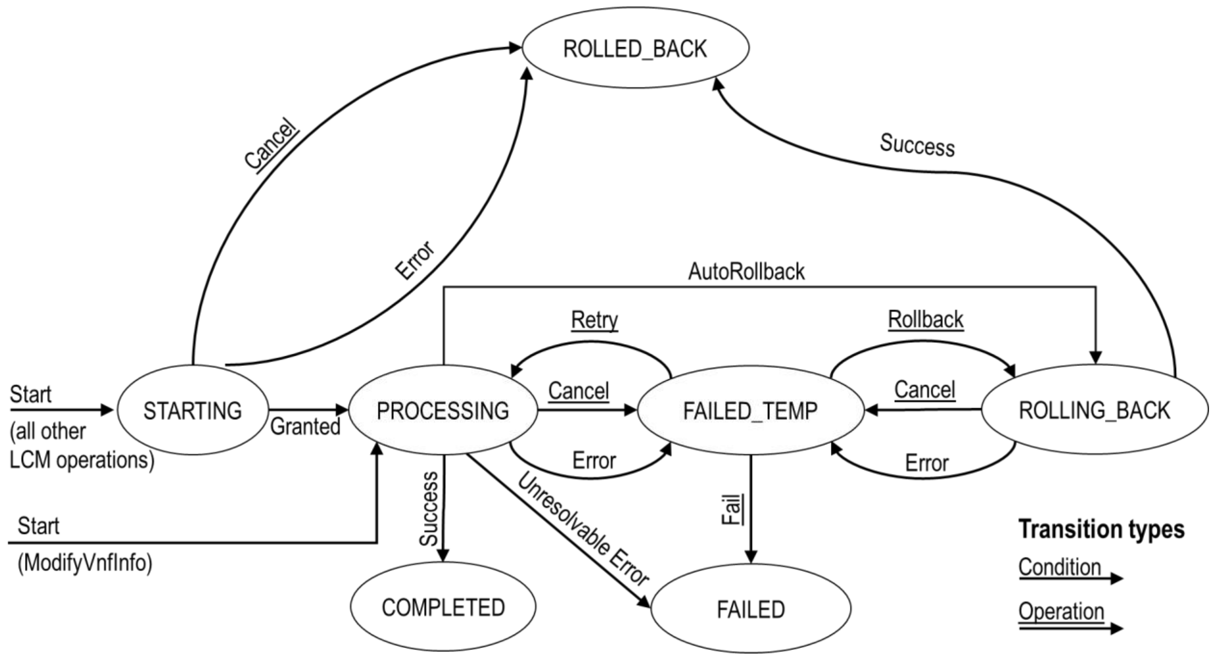

Figure 5.6.2.1-1: States of a VNF lifecycle management operation occurrence

5.6.2.2 States of a VNF lifecycle management operation occurrence

At each time, a VNF lifecycle management operation occurrence is in one of the following states. There are transitional states (states from which a different state can be reached) and terminal states (states from which no other state can be reached; i.e. the state of a VNF lifecycle management operation occurrence in a terminal state cannot change anymore).

STARTING: The operation is starting. This state represents the preparation phase of the operation, including invoking Grant Lifecycle Operation. This state has the following characteristics:

- This is the initial state for any LCM operation except ModifyVnfInformation.
- This is a transient state.
- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).
- In this state, the VNF lifecycle management operation occurrence does not perform any changes to the VNF instance or to resources.
- Once the VNF lifecycle operation has been granted, the VNF lifecycle management operation occurrence transits into the PROCESSING state.

- If the LCM operation is cancelled in the "STARTING" state, the VNF lifecycle management operation occurrence shall transit to the "ROLLED_BACK" state. The NFVO shall be prepared to receive the notification about the cancellation of the operation before and after having provided the grant. This is necessary to address possible race conditions.
- If an error occurs before the VNFM receives the grant response, or the grant is rejected, as no changes to the underlying VNF or resources were done, the VNF lifecycle management operation occurrence shall transit into the "ROLLED_BACK" state.

COMPLETED: The operation has completed successfully. This is a terminal state.

FAILED_TEMP: The operation has failed and execution has stopped, but the execution of the operation is not considered to be closed. This state has the following characteristics:

- This is a transient state.
- The grant received for the operation is still valid, and the granted resource changes are still foreseen for the VNF.
- This state may block other LCM operations from being executed on the same VNF instance (enforced by the VNFM, and up to VNF and VNFM capabilities).
- Retry and/or rollback and/or fail may be invoked for the operation.
- If the VNF LCM operation is retried, the VNF lifecycle management operation occurrence shall transit into the "PROCESSING" state.
- If the VNF LCM operation is rolled back, the VNF lifecycle management operation occurrence shall transit into the "ROLLING_BACK" state.
- If the VNF LCM operation is marked as "failed", the VNF lifecycle management operation occurrence shall transit into the "FAILED" state.
- Operation cancellation and failure to roll back should result in FAILED_TEMP.

FAILED: The operation has failed and it cannot be retried or rolled back, as it is determined that such action will not succeed. This state has the following characteristics:

- This is a terminal state.
- Such an operation state is typically the result of a decision of a higher layer management entity (NFVO) or its human operator that an operation in "FAILED_TEMP" state cannot be retried or rolled back ("Fail").
- Such an operation state can also be reached immediately in case of failure of an operation in "PROCESSING" state that can neither be retried nor rolled back ("Unresolvable Error").
- The result of the LCM operation (the actual resource changes) can show an inconsistent state of the VNF, and can reflect partial resource changes compared to the granted changes. Nevertheless, these changes shall be synchronized between the VNFM and NFVO (by reporting them in the LCCN, and by allowing the NFVO to obtain them on request) in order for other VNF LCM operations (e.g. Heal, Terminate) to be guaranteed to work on resources that are known to the NFVO.
- The fact that a LCM operation is in "FAILED" state shall not block other operations from execution on the VNF instance by the VNFM. However, the VNF instance may itself be in a state that disallows certain operations.

ROLLED_BACK: The state of the VNF prior to the original operation invocation has been restored as closely as possible. This state has the following characteristics:

- This is a terminal state.
- This may involve recreating some resources that have been deleted by the operation, the recreated resources should be as similar as possible to the deleted ones. Differences between original resources and re-created ones may include a different resource identity, but also different dynamic attributes such as an IP address.

PROCESSING: The LCM operation is currently in execution. This state has the following characteristics:

- This is the initial state for the ModifyVnfInformation operation.
- This is a transient state.
- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).
- The operations "Retry" and "Rollback" shall not be permitted to be invoked for an operation that is in this state.
- All failures of procedures executed by the VNFM as part of the LCM operation while in "PROCESSING" state should result by default in transiting to FAILED_TEMP, with the following two alternative options:
    - If a failure occurs in the "PROCESSING" state from which the VNFM knows that the VNF instance can be brought into a consistent state by immediately rolling back the operation, the VNF lifecycle management operation occurrence may transit directly into the "ROLLING_BACK" state ("Autorollback"). For the ModifyVnfInformation operation, Autorollback is the typical error handling method.
    - If a failure occurs in the "PROCESSING" state from which the VNFM knows that it can neither be fixed by retrying nor be rolled back, the VNF lifecycle management operation occurrence may transit directly into the "FAILED" state ("Unresolvable Error").
- If a "cancel" request was issued during the operation is in "PROCESSING" state, processing will be cancelled but this might not be immediate. This is represented by a flag in the data model that indicates there is a pending "cancel" request for this state. Upon successful cancellation, the VNF lifecycle management operation occurrence shall transit into the "FAILED_TEMP" state.

ROLLING_BACK: The LCM operation is currently being rolled back. This state has the following characteristics:

- This is a transient state.
- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).
- The operations "Retry" and "Rollback" shall not be permitted to be invoked for an operation that is in this state.
- If a "Cancel" request was issued during the operation is in "ROLLING_BACK" state, rolling back will be cancelled but this might not be immediate. This is represented by a flag in the data model that indicates there is a pending "Cancel" request for this state. Upon successful cancellation, the VNF lifecycle management operation occurrence shall transit into the "FAILED_TEMP" state.
- If a failure occurs during rolling back, the operation should transition to the "FAILED_TEMP" state.
- Upon successful rollback, the VNF lifecycle management operation occurrence shall transit into the "ROLLED_BACK" state.

In addition, the following provisions apply to VNF lifecycle management operation occurrence notifications:

- The "start" notification (i.e. notificationStatus="START") shall be sent when the operation enters one of states "STARTING", "PROCESSING" and "ROLLING_BACK" from another state, indicating the state entered.
- The "result" notification (i.e. notificationStatus="RESULT") shall be sent when the VNF LCM operation occurrence enters one of the error states "FAILED_TEMP", "FAILED", "ROLLED_BACK", indicating the state entered, the error cause and the changes to the VNF's resources since the operation was initially started.
- The "result" notification (i.e. notificationStatus="RESULT") shall be sent when the operation enters the success state "COMPLETED", indicating the state entered and the changes to the VNF's resources.

Such a notification scheme allows the NFVO to keep in sync with changes to the VNF's resources by an ongoing LCM operation. If the notification relates to a transient state, further changes can be expected. If the notification relates to a terminal state, no further changes to the VNF's resources will be performed by the related VNF lifecycle management operation occurrence, and the NFVO can use the information in the notification to synchronize its internal state with the result of the LCM operation. In case of loss of notifications, a the NFVO can read the resource that represents the VNF lifecycle management operation occurrence to obtain the same information.

5.6.2.3 Error handling operations that change the state of a VNF lifecycle management operation occurrence

Retry: This operation retries a VNF lifecycle operation. It has the following characteristics:

- Execution of "Retry" for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.
- "Retry" shall operate within the bounds of the Grant for the LCM operation.
- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Rollback: This operation rolls back a VNF lifecycle operation. It has the following characteristics:

- Execution of "Rollback" for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.
- "Rollback" shall operate within the bounds of the Grant for the LCM operation, an additionally may execute the inverse of granted LCM operations (e.g. if a resource deletion was granted, rollback might re-create the deleted resource or a similar resource).
- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Fail: This operation transits the VNF lifecycle management operation occurrence into the terminal "FAILED" state. It has the following characteristics:

- Execution of "Fail" shall be supported for an LCM operation on a particular VNF if at least one of Retry, Rollback, Cancel is supported for this operation.
- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Cancel: This operation cancels an ongoing VNF lifecycle management operation, its Retry or Rollback. It has the following characteristics:

- Execution of Cancel for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.
- The "Cancel" operation need not have immediate effect, depending on the capabilities of the underlying systems, and the currently executed resource management operation.
- Two modes of cancellation are supported: graceful and forceful.
  - When executing the *graceful* "Cancel" operation, the VNFM will not initiate any new operation towards the underlying systems, will wait until the currently executed operations finish, fail or time out in the VNFM, and will then put the VNF lifecycle management operation occurrence into the "FAILED_TEMP" state.
  - When executing the *forceful* "Cancel" operation, the VNFM will cancel all ongoing operations in the underlying systems for which cancellation is supported, will not initiate any new operation towards the underlying systems, will wait for the requested cancellations to finish, fail or time out in the VNFM, and will then put the VNF lifecycle management operation occurrence into the "FAILED_TEMP" state.

NOTE: In both modes, the time-out is determined by means outside the scope of the present document.

  - In "STARTING" state, there is no difference between the graceful and the forceful cancellation mode.

- Executing "Cancel" can lead to inconsistencies between the information that the VNFM has about the state of the resources of the VNF, and their actual state. The probability of such inconsistencies is bigger when using the *forceful* cancellation mode.

5.6.3 Detailed flows

5.6.3.1 Immediate failure

If the VNF LCM operation fails immediately, i.e. it returns an HTTP error, then the operation has not started, and no "VNF LCM operation occurrence" resource has been created. Also, neither a "start" VNF lifecycle management operation occurrence notification nor a Grant request has been sent. The operation cannot be retried, but the same operation may be invoked again from the API. The VNF instance is not changed by a synchronous failure, so no special error handling is required.

Figure 5.6.3.1-1 illustrates the flow.

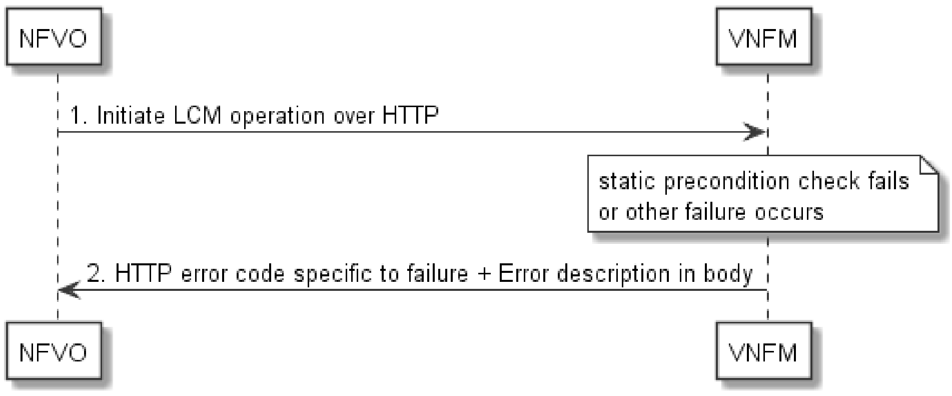
NFVO
VNFM
1. Initiate LCM operation over HTTP
static precondition check fails
or other failure occurs
2. HTTP error code specific to failure + Error description in body
NFVO
VNFM

Figure 5.6.3.1-1: Immediate failure of a VNF LCM operation

5.6.3.2 Failure in "STARTING" state

This error scenario assumes that the "VNF LCM operation occurrence" resource has been created and the "start" VNF lifecycle management operation occurrence notification has been sent.

If the operation fails before the VNFM receives the Grant response, or the Grant is rejected, persistent change to the state of the VNF cannot have happened. Therefore, it is assumed that this operation enters the ROLLED_BACK state immediately. Figure 5.6.3.2-1 illustrates the flow.

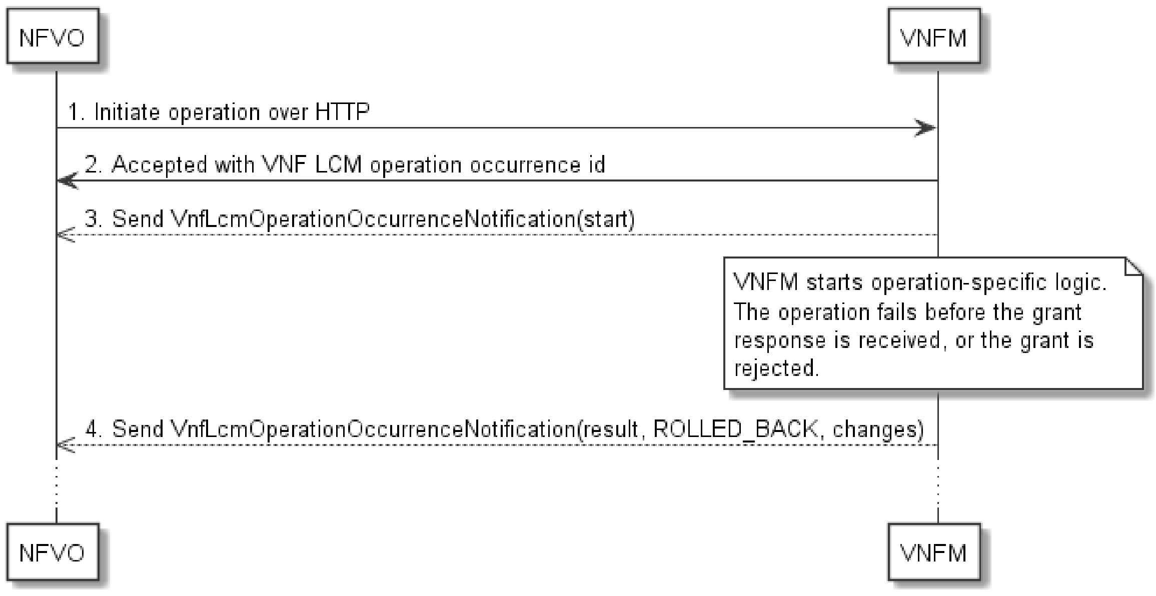
NFVO
VNFM
1. Initiate operation over HTTP
2. Accepted with VNF LCM operation occurrence id
3. Send VnfLcmOperationOccurrenceNotification(start)
VNFM starts operation-specific logic. The operation fails before the grant response is received, or the grant is rejected.
4. Send VnfLcmOperationOccurrenceNotification(result, ROLLED_BACK, changes)
NFVO
VNFM

Figure 5.6.3.2-1: Failure of a VNF LCM operation before applying any change to the VNF instance

5.6.3.3 Failure during actual LCM operation execution

After a failed resource management operation, automatic retry can be invoked by the VNFM itself. These invocations are not visible outside of the VNFM, as the VNF LCM operation occurrence stays in "PROCESSING" state during these automatic retries. If these do not resolve the issue, intervention (typically by a human operator) is necessary. For that purpose, the LCM operation is set into a temporary failure state, and the NFVO is notified. The human operator performs a root cause analysis and eventually resolves the obstacle. Subsequently, and if supported, the operation can be retried, rolled-back or determined as permanently failed. Figure 5.6.3.3-1 illustrates the possible options.

NOTE 1: Excluding automated rollback which is seen as a rare option.

NOTE 2: Excluding "start" notifications (i.e. notificationStatus="START") for simplification purposes.

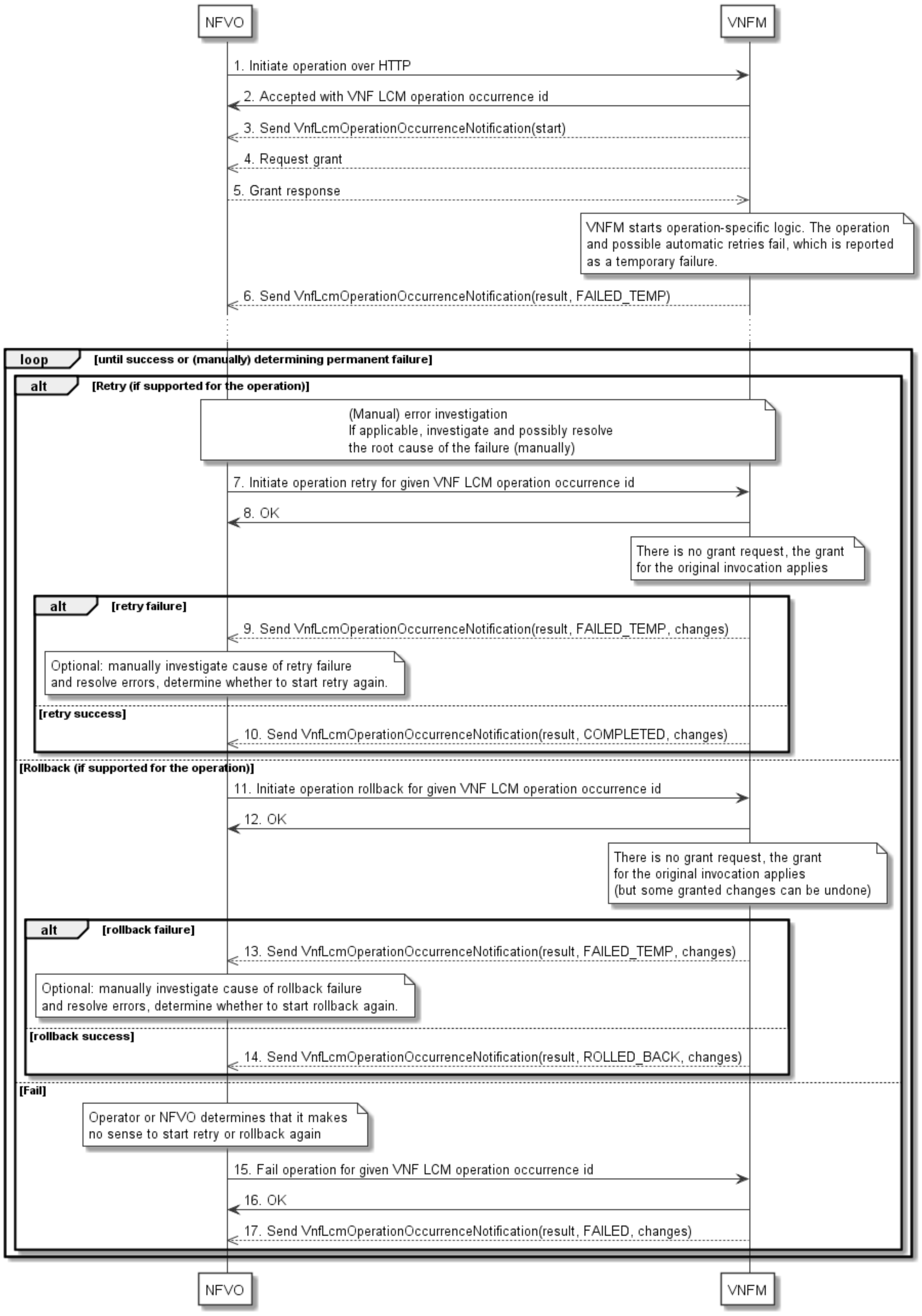
NFVO
VNFM
1. Initiate operation over HTTP
2. Accepted with VNF LCM operation occurrence id
3. Send VnfLcmOperationOccurrenceNotification(start)
4. Request grant
5. Grant response
VNFM starts operation-specific logic. The operation and possible automatic retries fail, which is reported as a temporary failure.
6. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP)
loop [until success or (manually) determining permanent failure]
alt [Retry (if supported for the operation)]
(Manual) error investigation If applicable, investigate and possibly resolve the root cause of the failure (manually)
7. Initiate operation retry for given VNF LCM operation occurrence id
8. OK
There is no grant request, the grant for the original invocation applies
alt [retry failure]
9. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Optional: manually investigate cause of retry failure and resolve errors, determine whether to start retry again.
[retry success]
10. Send VnfLcmOperationOccurrenceNotification(result, COMPLETED, changes)
[Rollback (if supported for the operation)]
11. Initiate operation rollback for given VNF LCM operation occurrence id
12. OK
There is no grant request, the grant for the original invocation applies (but some granted changes can be undone)
alt [rollback failure]
13. Send VnfLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Optional: manually investigate cause of rollback failure and resolve errors, determine whether to start rollback again.
[rollback success]
14. Send VnfLcmOperationOccurrenceNotification(result, ROLLED_BACK, changes)
[Fail]
Operator or NFVO determines that it makes no sense to start retry or rollback again
15. Fail operation for given VNF LCM operation occurrence id
16. OK
17. Send VnfLcmOperationOccurrenceNotification(result, FAILED, changes)
NFVO
VNFM

Figure 5.6.3.3-1: Handling failures during the actual execution of a VNF LCM operation

5.6.3.4 LCM operation cancellation

The cancellation of an LCM operation that is in PROCESSING or ROLLING_BACK state is handled like any other error that leads to stopping the execution of the VNF LCM workflow before it can be successfully completed. The VNF LCM operation transits into the FAILED_TEMP state which allows root cause analysis, possible fixing of the root cause, followed by retrying, rolling back, or finally failing of the operation.

The cancellation of an operation in STARTING state (i.e. until the Grant is received) transits the operation into the ROLLED_BACK state, as no changes to the resources or VNF instance have been performed.

6 VNF Performance Management interface

6.1 Description

This interface allows providing performance management (measurement results collection and notifications) related to VNFs. Performance information on a given VNF instance results from performance information of the virtualised resources that is collected from the VIM and mapped to this VNF instance. Collection and reporting of performance information is controlled by a PM job that groups details of performance collection and reporting information. Further, this interface allows API version information retrieval.

When new performance information is available, the consumer is notified using the notification PerformanceInformationAvailableNotification.

The operations provided through this interface are:

- Create PM Job
- Query PM Job
- Delete PM Job
- Create Threshold
- Query Threshold
- Delete Threshold
- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

6.1a API version

For the VNF performance management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

6.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2.

The string "vnfpm" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 6.2-1 shows the overall resource URI structure defined for the performance management API.

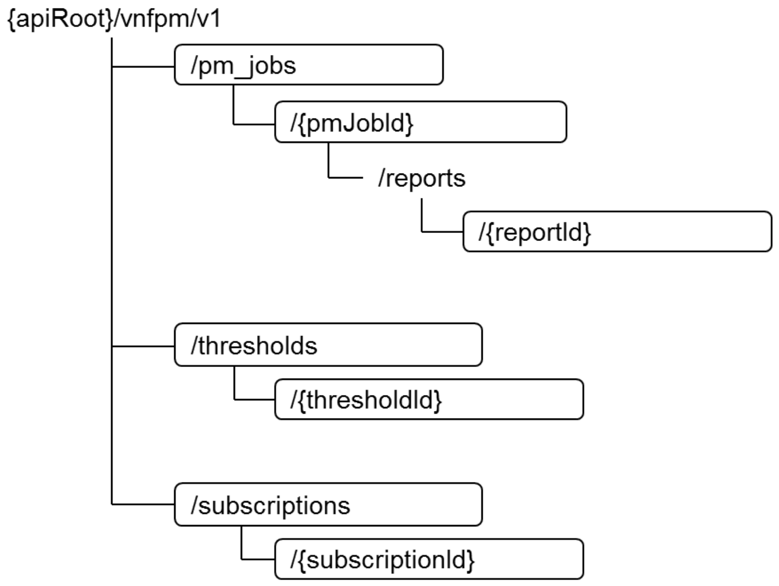

Figure 6.2-1: Resource URI structure of the VNF Performance Management interface

Table 6.2-1 lists the individual resources defined, and the applicable HTTP methods.

The VNFM shall support responding to requests for all HTTP methods on the resources in table 6.2-1 that are marked as "M" (mandatory) in the "Cat" column. The VNFM shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 6.2-1: Resources and methods overview of the VNF Performance Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
PM jobs	/pm_jobs	POST	M	Create a PM job
		GET	M	Query PM jobs
Individual PM job	/pm_jobs/{pmJobId}	GET	M	Read a single PM job
		DELETE	M	Delete a PM job
Individual performance report	/pm_jobs/{pmJobId}/reports/{reportId}	GET	M	Read an individual performance report
Thresholds	/thresholds	POST	M	Create a threshold
		GET	M	Query thresholds
Individual threshold	/thresholds/{thresholdId}	GET	M	Query a single threshold
		DELETE	M	Delete a threshold
Subscriptions	/subscriptions	POST	M	Subscribe to PM notifications
		GET	M	Query PM related subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Query a single PM related subscription
		DELETE	M	Terminate a subscription
Notification endpoint	(client-defined)	POST	See note	Notify about PM related events. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The VNFM shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the NFVO. If the NFVO supports invoking the POST method on the "Subscription" resource towards the VNFM, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

6.3 Sequence diagrams (informative)

6.3.1 Flow of creating a PM job

This clause describes a sequence for creating a performance management jobs.

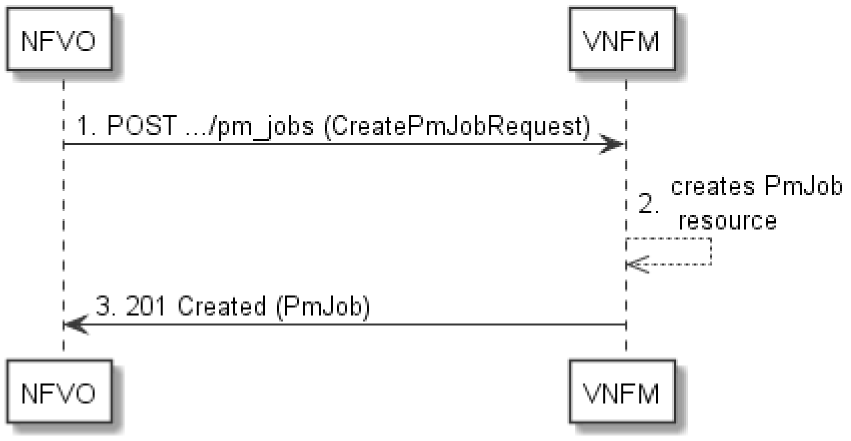
NFVO
VNFM
1. POST .../pm_jobs (CreatePmJobRequest)
2. creates PmJob resource
3. 201 Created (PmJob)
NFVO
VNFM

Figure 6.3.1-1: Flow of PM job creation

PM job creation, as illustrated in figure 6.3.1-1, consists of the following steps:

1) If the NFVO intends to create a PM job, it sends a POST request to the "PM jobs" resource, including one data structure of type "CreatePmJobRequest" in the payload body.

2) The VNFM creates a PM job instance.

3) The VNFM returns a "201 Created" response to the NFVO, and includes in the payload body a representation of the PM job just created.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.2 Flow of querying/reading PM jobs

This clause describes a sequence for querying/reading performance management jobs.

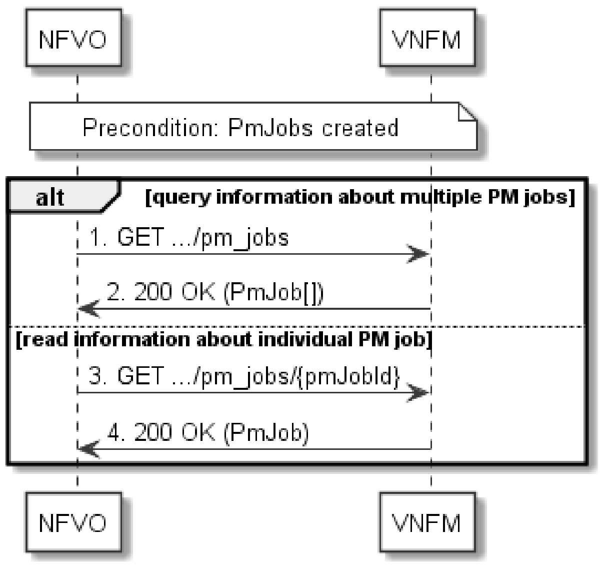
NFVO
VNFM
Precondition: PmJobs created
alt [query information about multiple PM jobs]
1. GET .../pm_jobs
2. 200 OK (PmJob[])
[read information about individual PM job]
3. GET .../pm_jobs/{pmJobId}
4. 200 OK (PmJob)
NFVO
VNFM

Figure 6.3.2-1: Flow of PM jobs query/read

PM jobs query/read, as illustrated in figure 6.3.2-1, consists of the following steps:

1) If the NFVO intends to query all PM jobs, it sends a GET request to the "PM jobs" resource.
2) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "PmJob" in the payload body.
3) If the NFVO intends to read information about a particular PM job, it sends a GET request to the "Individual PM job" resource, addressed by the appropriate PM job identifier in its resource URI.
4) The VNFM returns a "200 OK" response to the NFVO, and includes one data structure of type "PmJob" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.3 Flow of deleting a PM job

This clause describes a sequence for deleting a performance management jobs.

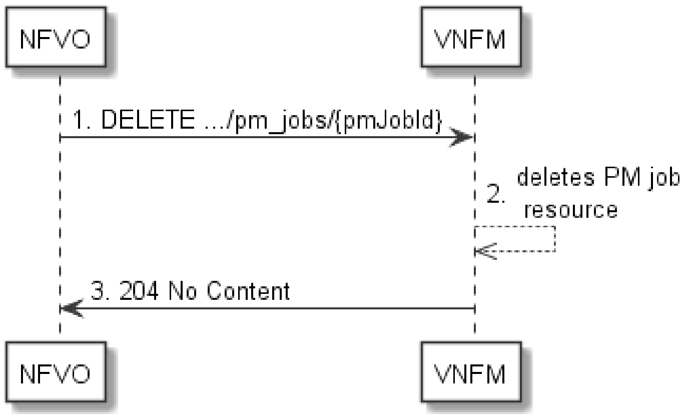

Figure 6.3.3-1: Flow of PM job deletion

PM job deletion, as illustrated in figure 6.3.3-1, consists of the following steps:

1) If the NFVO intends to delete a PM job, it sends a DELETE request to the "Individual PM job" resource, addressed by the appropriate PM job identifier in its resource URI.
2) The VNFM returns a response with a "204 No Content" response code and an empty payload body to the NFVO.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.4 Flow of obtaining performance reports

This clause describes a sequence for obtaining performance reports.

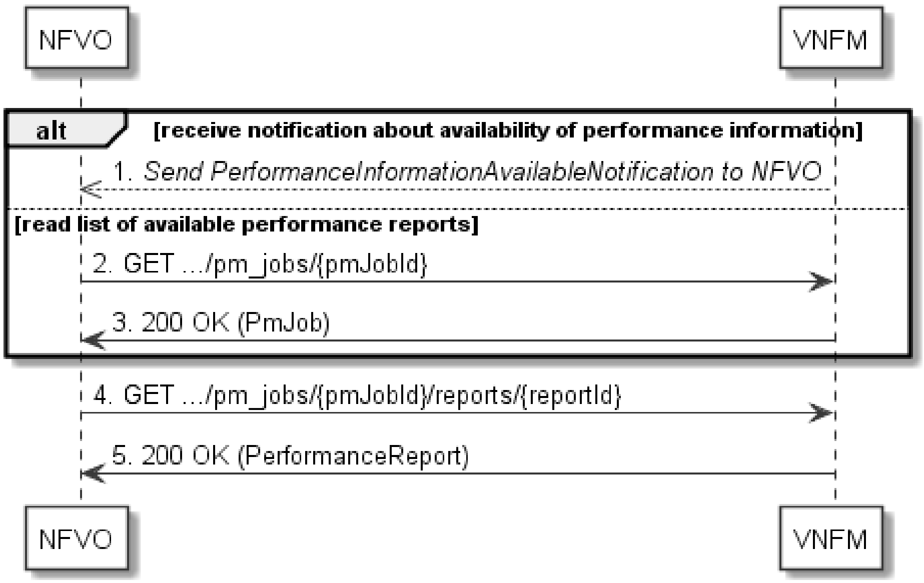
NFVO
VNFM
alt
[receive notification about availability of performance information]
1. Send PerformanceInformationAvailableNotification to NFVO
[read list of available performance reports]
2. GET .../pm_jobs/{pmJobId}
3. 200 OK (PmJob)
4. GET .../pm_jobs/{pmJobId}/reports/{reportId}
5. 200 OK (PerformanceReport)
NFVO
VNFM

Figure 6.3.4-1: Flow of obtaining performance reports

Obtaining a performance report, as illustrated in figure 6.3.4-1, consists of the following steps:

1) The VNFM sends to the NFVO a PerformanceInformationAvailableNotification (see clause 6.3.9) that indicates the availability of a new performance report, including a link from which the report can be obtained.

2) Alternatively, the NFVO sends a GET request to the "Individual PM job" resource, to obtain a representation of the PM job resource including information about performance reports that are available for this PM job, including their URIs.

3) In that case, the VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "PmJob" in the payload body.

4) The NFVO sends to the VNFM a GET request to the URI obtained either in step (1) or step (3), in order to read a performance report resource.

5) The VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "PerformanceReport" in the payload body.

6.3.5 Flow of creating a threshold

This clause describes a sequence for creating a performance management threshold.

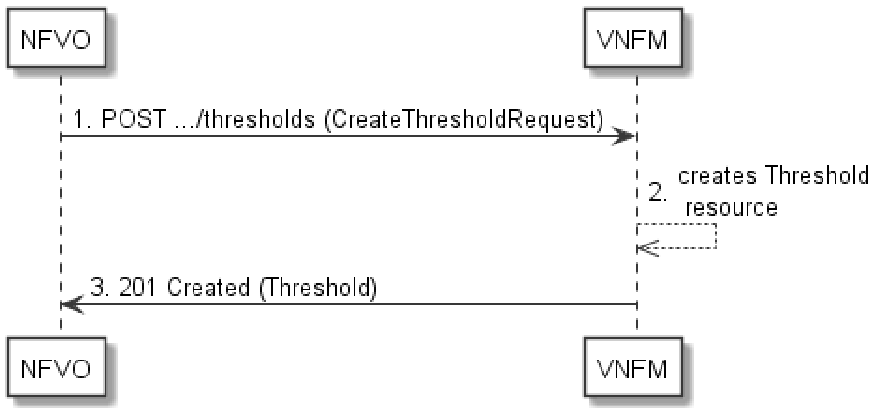
NFVO
VNFM
1. POST .../thresholds (CreateThresholdRequest)
2. creates Threshold resource
3. 201 Created (Threshold)
NFVO
VNFM

Figure 6.3.5-1: Flow of threshold creation

Threshold creation, as illustrated in figure 6.3.5-1, consists of the following steps:

1) If the NFVO intends to create a threshold, it sends a POST request to the "Thresholds" resource, including a data structure of type "CreateThresholdRequest" in the payload body.

2) The VNFM creates a threshold instance.

3) The VNFM returns a "201 Created" response to the NFVO, and includes in the payload body a representation of the threshold just created.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.6 Flow of querying/reading thresholds

This clause describes a sequence for querying/reading performance management thresholds.

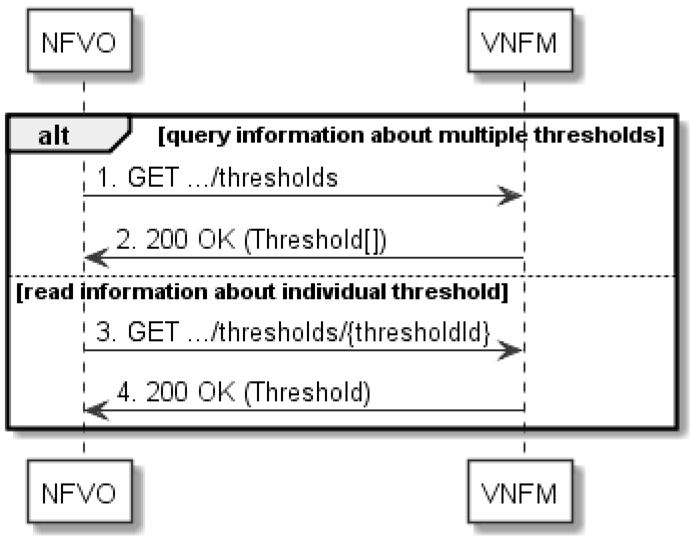
NFVO
VNFM
alt
[query information about multiple thresholds]
1. GET .../thresholds
2. 200 OK (Threshold[])
[read information about individual threshold]
3. GET .../thresholds/{thresholdId}
4. 200 OK (Threshold)
NFVO
VNFM

Figure 6.3.6-1: Flow of thresholds query/read

Threshold query/read, as illustrated in figure 6.3.6-1, consists of the following steps:

1) If the NFVO intends to query all thresholds, it sends a GET request to the "Thresholds" resource.

2) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "Threshold" in the payload body.

3) If the NFVO intends to read information about a particular threshold, it sends a GET request to the "Individual threshold" resource addressed by the appropriate threshold identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "Threshold" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.7 Flow of deleting thresholds

This clause describes a sequence for deleting performance management thresholds.

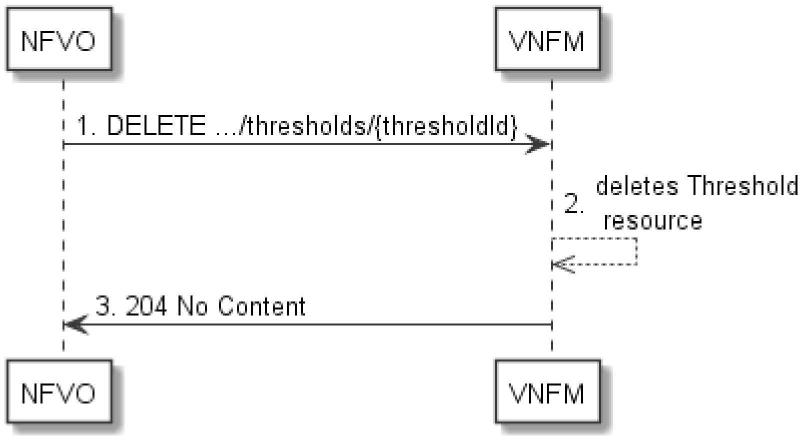
NFVO
VNFM
1. DELETE .../thresholds/{thresholdId}
2. deletes Threshold resource
3. 204 No Content
NFVO
VNFM

Figure 6.3.7-1: Flow of threshold deletion

Threshold deletion, as illustrated in figure 6.3.7-1, consists of the following steps:

1) If the NFVO intends to delete a particular threshold, it sends a DELETE request to the "Individual treshold" resource, addressed by the appropriate threshold identifier in its resource URI.

2) The VNFM returns a "204 No Content" response code to the NFVO. The response body shall be empty.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.8 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF performance management.

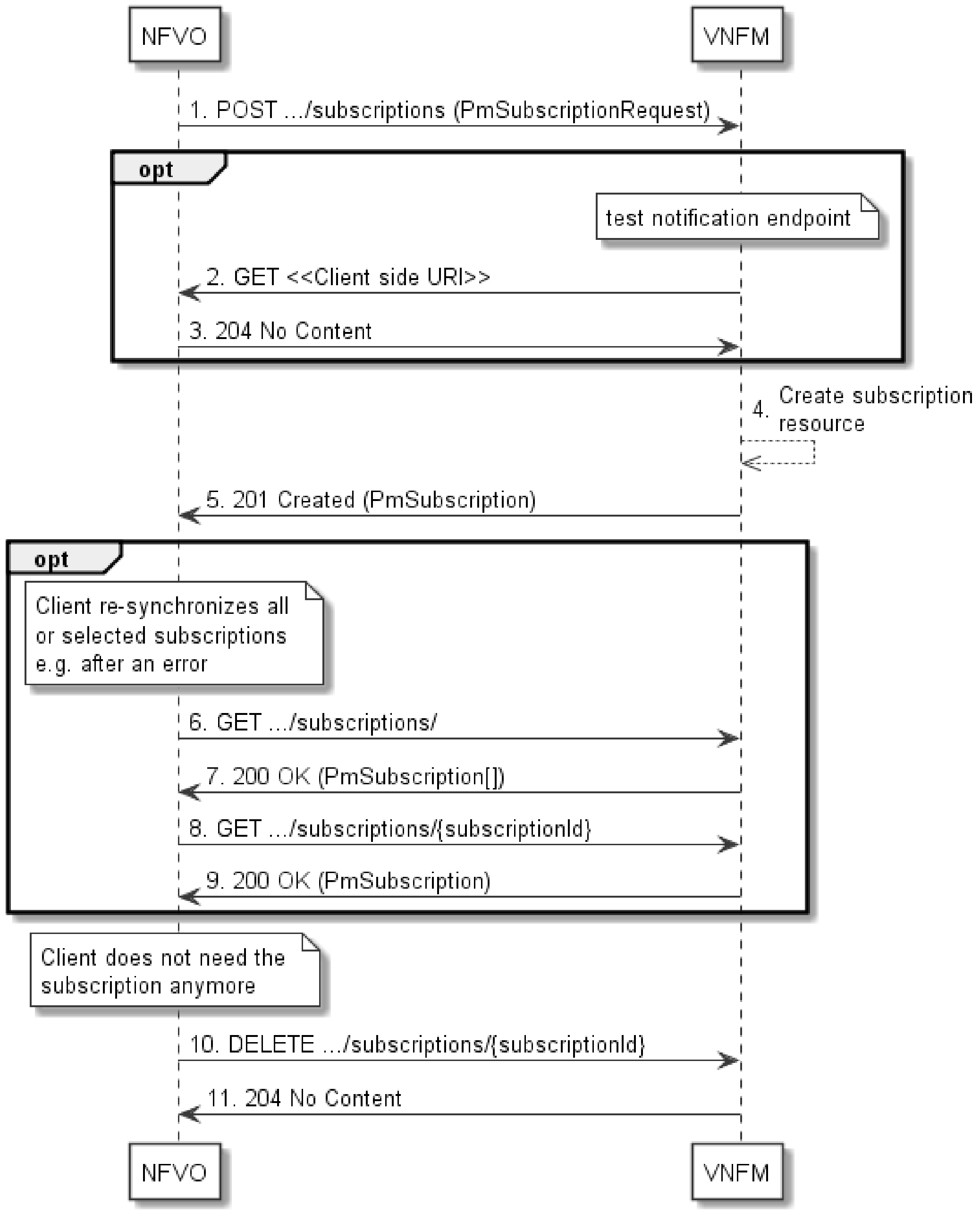
NFVO
VNFM
1. POST .../subscriptions (PmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (PmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (PmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (PmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
NFVO
VNFM

Figure 6.3.8-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 6.3.8-1:

1) The NFVO sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "PmSubscriptionRequest". This data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the NFVO as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3) In that case, the NFVO returns a "204 No Content" response to indicate success.

4) The VNFM creates a new subscription to notifications related to VNF performance management, and a resource that represents this subscription.

5) The VNFM returns a "201 Created" response containing a data structure of type "PmSubscription," representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) Optionally, for example when trying to recover from an error situation, the NFVO may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the VNFM returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the NFVO.

8) Optionally, for example when trying to recover from an error situation, the NFVO may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10) When the NFVO does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription.

11) The VNFM acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

6.3.9 Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF performance management.

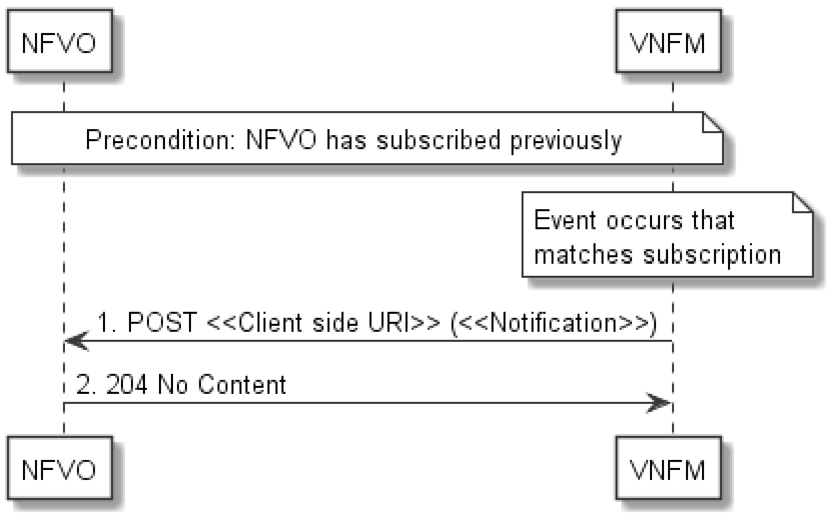
NFVO
VNFM
Precondition: NFVO has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
NFVO
VNFM

Figure 6.3.9-1: Flow of sending notifications

Precondition: The NFVO has subscribed previously for notifications related to VNF performance management.

The procedure consists of the following steps as illustrated in figure 6.3.9-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the NFVO has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API.

2) The NFVO acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the NVFO, it can retry sending the notification.

6.4 Resources

6.4.1 Introduction

This clause defines all the resources and methods provided by the performance management interface.

6.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF performance management interface.

6.4.2 Resource: PM jobs

6.4.2.1 Description

This resource represents PM jobs. The client can use this resource to create and query PM jobs.

6.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs

This resource shall support the resource URI variables defined in table 6.4.2.2-1.

Table 6.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.2.3 Resource methods

6.4.2.3.1 POST

The POST method creates a PM job.

This method shall follow the provisions specified in the tables 6.4.2.3.1-1 and 6.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.2.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**	**Description**	
	CreatePmJobRequest	1	PM job creation request	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmJob	1	201 Created	The PM job was created successfully. The response body shall contain a representation of the created PM job resource, as defined in clause 6.5.2.7. The HTTP response shall include a "Location" HTTP header that points to the created PM job resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.2.3.2 GET

The client can use this method to retrieve information about PM jobs.

This method shall follow the provisions specified in the tables 6.4.2.3.2-1 and 6.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the PmJob and in data types referenced from it shall be supported by the VNFM in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the PmJob structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - Reports
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 6.4.2.3.2-2: Details of the GET request/response on this resource

Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmJob	0..N	200 OK	Information about zero or more PM jobs was queried successfully. The response body shall contain in an array the representations of zero or more PM jobs, as defined in clause 6.5.2.7. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3 Resource: Individual PM job

6.4.3.1 Description

This resource represents an individual PM job. The client can use this resource to delete and read the underlying PM job.

6.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs/{pmJobId}

This resource shall support the resource URI variables defined in table 6.4.3.2-1.

Table 6.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
pmJobId	Identifier of the PM job. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new PM job resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

6.4.3.3 Resource methods

6.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.2 GET

The client can use this method for reading an individual PM job.

This method shall follow the provisions specified in the tables 6.4.3.3.2-1 and 6.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	PmJob	1	200 OK	Information about an individual PM job was read successfully. The response body shall contain a representation of the PM job resource, as defined in clause 6.5.2.7.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.5 DELETE

This method terminates an individual PM job.

This method shall follow the provisions specified in the tables 6.4.3.3.5-1 and 6.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.3.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The PM job was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.4 Resource: Individual performance report

6.4.4.1 Description

This resource represents an individual performance report that was collected by a PM job. The client can use this resource to read the performance report. The URI of this report can be obtained from a PerformanceInformationAvailableNotification (see clause 6.5.2.5) or from the representation of the "Individual PM job" resource.

It is determined by means outside the scope of the present document, such as configuration or policy, how long an individual performance report is available.

6.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs/{pmJobId}/reports/{reportId}

This resource shall support the resource URI variables defined in table 6.4.4.2-1.

Table 6.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
pmJobId	Identifier of the PM job.
reportId	Identifier of the performance report.

6.4.4.3 Resource methods

6.4.4.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.2 GET

The client can use this method for reading an individual performance report.

This method shall follow the provisions specified in the tables 6.4.4.3.2-1 and 6.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.4.3.2-2: Details of the GET request/response on this resource

Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PerformanceReport	1	200 OK	Information of an individual performance report was read successfully. The response body shall contain a representation of the performance report resource, as defined in clause 6.5.2.10.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5 Resource: Thresholds

6.4.5.1 Description

This resource represents thresholds. The client can use this resource to create and query thresholds.

6.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/thresholds

This resource shall support the resource URI variables defined in table 6.4.5.2-1.

Table 6.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.5.3 Resource methods

6.4.5.3.1 POST

The POST method can be used by the client to create a threshold.

This method shall follow the provisions specified in the tables 6.4.5.3.1-1 and 6.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.5.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	CreateThresholdRequest	1		Request parameters to create a threshold resource.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	Threshold	1	201 Created	A threshold was created successfully. The response body shall contain a representation of the created threshold resource, as defined in clause 6.5.2.9. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created threshold resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.5.3.2 GET

The client can use this method to query information about thresholds.

This method shall follow the provisions specified in the tables 6.4.5.3.2-1 and 6.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the Thresholds data type and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

NOTE: There are no attribute selectors defined for this resource as the threshold attributes with cardinality 0..1 or 0..N are not structurally complex in nature.

Table 6.4.5.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Threshold	0..N	200 OK	Information about zero or more thresholds was queried successfully. The response body shall contain in an array the representations of zero or more thresholds, as defined in clause 6.5.2.9. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6 Resource: Individual threshold

6.4.6.1 Description

This resource represents an individual threshold.

6.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/thresholds/{thresholdId}

This resource shall support the resource URI variables defined in table 6.4.6.2-1.

Table 6.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
thresholdId	Identifier of the threshold. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new threshold resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

6.4.6.3 Resource methods

6.4.6.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.2 GET

The client can use this method for reading an individual threshold

This method shall follow the provisions specified in the tables 6.4.6.3.2-1 and 6.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.6.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Threshold	1	200 OK	Information about an individual threshold was read successfully. The response body shall contain a representation of the threshold, as defined in clause 6.5.2.9.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.5 DELETE

This method allows to delete a threshold.

This method shall follow the provisions specified in the tables 6.4.6.3.5-1 and 6.4.6.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.6.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The threshold was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.7 Resource: Subscriptions

6.4.7.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF performance management and to query its subscriptions.

6.4.7.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/subscriptions

This resource shall support the resource URI variables defined in table 6.4.7.2-1.

Table 6.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.7.3 Resource methods

6.4.7.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 6.4.7.3.1-1 and 6.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the NFVO, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 6.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	PmSubscriptionRequest	1	Details of the subscription to be created.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmSubscription	1	201 Created	The subscription was created successfully. A representation of the created subscription resource shall be returned in the response body, as defined in clause 6.5.2.3. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.

	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.7.3.2 GET

The client can use this method to query the list of active subscriptions to Performance management notifications subscribed by the client.

This method shall follow the provisions specified in the tables 6.4.7.3.2-1 and 6.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the PmSubscription and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 6.4.7.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	PmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of PM subscriptions as defined in clause 6.5.2.3. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8 Resource: Individual subscription

6.4.8.1 Description

This resource represents an individual subscription for notifications about performance management related events.

The client can use this resource to read and to terminate a subscription to notifications related to VNF performance management.

6.4.8.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 6.4.8.2-1.

Table 6.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of the subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

6.4.8.3 Resource methods

6.4.8.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.2 GET

The client can use this method for reading an individual subscription about Performance management notifications subscribed by the client.

This method shall follow the provisions specified in the tables 6.4.8.3.2-1 and 6.4.8.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.8.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	PmSubscription	1	200 OK	The subscription was read successfully. The response body shall contain a representation of the subscription resource, as defined in clause 6.5.2.3.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.5 DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the tables 6.4.8.3.5-1 and 6.4.8.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.8.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.8.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.9 Resource: Notification endpoint

6.4.9.1 Description

This resource represents a notification endpoint for VNF performance management.

The API producer can use this resource to send notifications related to performance management events to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

6.4.9.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 6.4.9.2-1.

Table 6.4.9.2-1: Resource URI variables for this resource

Name	Definition
n/a	

6.4.9.3 Resource methods

6.4.9.3.1 POST

The POST method delivers a notification regarding a performance management event from the server to the client.

This method shall follow the provisions specified in the tables 6.4.9.3.1-1 and 6.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	PerformanceInformationAvailableNotification	1	Notification about performance information availability	
	ThresholdCrossedNotification	1	Notification about threshold crossing	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.9.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 6.4.9.3.2-1 and 6.4.9.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.9.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.5 Data Model

6.5.1 Introduction

This clause defines the request and response data structures of the VNF Performance Management interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

6.5.2 Resource and notification data types

6.5.2.1 Introduction

This clause defines the data structures to be used in resource representations and notifications.

6.5.2.2 Type: PmSubscriptionRequest

This type represents a subscription request. It shall comply with the provisions defined in table 6.5.2.2-1.

Table 6.5.2.2-1: Definition of the PmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	PmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

6.5.2.3 Type: PmSubscription

This type represents a subscription. It shall comply with the provisions defined in table 6.5.2.3-1.

Table 6.5.2.3-1: Definition of the PmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier that identifies the subscription
filter	PmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter
callbackUri	Uri	1	The URI of the endpoint to send the notification to
_links	Structure (inlined)	1	Links to resources related to this resource
>self	Link	1	URI of this resource

6.5.2.4 Type: ThresholdCrossedNotification

This type represents a notification that is sent when a threshold has been crossed. It shall comply with the provisions defined in table 6.5.2.4-1.

The notification shall be triggered by the VNFM when a threshold has been crossed.

Table 6.5.2.4-1: Definition of the ThresholdCrossedNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "ThresholdCrossedNotification " for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date and time of the generation of the notification.
thresholdId	Identifier	1	Identifier of the threshold which has been crossed.
crossingDirection	CrossingDirectionType	1	An indication of whether the threshold was crossed in upward or downward direction.
objectInstanceId	Identifier	1	Identifier that identifies a VNF instance.
performanceMetric	String	1	Performance metric associated with the threshold.

Attribute name	Data type	Cardinality	Description
			This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [20].
performanceValue	(any type)	1	Value of the metric that resulted in threshold crossing. The type of this attribute shall correspond to the related "Measurement Unit" as defined in clause 7.2 of ETSI GS NFV-IFA 027 [20].
_links	Structure (inlined)	1	Links to resources related to this notification
>subscription	NotificationLink	1	Link to the related subscription.
>objectInstance	NotificationLink	0..1	Link to the resource representing the VNF instance to which the notified change applies. Shall be present if the VNF instance information is accessible as a resource.
>threshold	NotificationLink	1	Link to the resource that represents the threshold that was crossed.

6.5.2.5 Type: PerformanceInformationAvailableNotification

This notification informs the receiver that performance information is available. It shall comply with the provisions defined in table 6.5.2.5-1.

NOTE: The timing of sending this notification is determined by the capability of the producing entity to evaluate the threshold crossing condition.

The notification shall be triggered by the VNFM when new performance information collected by a PM job is available.

Table 6.5.2.5-1: Definition of the PerformanceInformationAvailableNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "PerformanceInformationAvailableNotification " for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date and time of the generation of the notification.
objectInstanceId	Identifier	1	Identifier that identifies a VNF instance.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.
>objectInstance	NotificationLink	0..1	Link to the resource representing the VNF instance to which the notified change applies. Shall be present if the VNF instance information is accessible as a resource.
>pmJob	NotificationLink	1	Link to the resource that represents the PM job for which performance information is available.
>performanceReport	NotificationLink	1	Link from which the available performance information of data type "PerformanceReport" (see clause 6.5.2.10) can be obtained. This link should point to an "Individual performance report" resource as defined in clause 6.4.4.

6.5.2.6 Type: CreatePmJobRequest

This type represents a request to create a PM job. It shall comply with the provisions defined in table 6.5.2.6-1.

Table 6.5.2.6-1: Definition of the CreatePmJobRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceIds	Identifier	1..N	Identifiers of the VNF instances for which performance information is requested to be collected.
criteria	PmJobCriteria	1	Criteria of the collection of performance information.

6.5.2.7 Type: PmJob

This type represents a PM job. It shall comply with the provisions defined in table 6.5.2.7-1.

Table 6.5.2.7-1: Definition of the PmJob data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this PM job.
objectInstanceIds	Identifier	1..N	Identifiers of the VNF instances for which performance information is collected.
criteria	PmJobCriteria	1	Criteria of the collection of performance information.
reports	Structure (inlined)	0..N	Information about available reports collected by this PM job.
>href	Uri	1	The Uri where the report can be obtained.
>readyTime	DateTime	1	The time when the report was made available.
>expiryTime	DateTime	0..1	The time when the report will expire.
>fileSize	UnsigendInt	0..1	The size of the report file in bytes, if known.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>objects	Link	0..N	Links to resources representing the VNF instances for which performance information is collected. Shall be present if the VNF instance information is accessible as a resource.

6.5.2.8 Type: CreateThresholdRequest

This type represents a request to create a threshold. It shall comply with the provisions defined in table 6.5.2.8-1.

Table 6.5.2.8-1: Definition of the CreateThresholdRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceId	Identifier	1	Identifier of the VNF instance associated with this threshold.
criteria	ThresholdCriteria	1	Criteria that define this threshold.

6.5.2.9 Type: Threshold

This type represents a threshold. It shall comply with the provisions defined in table 6.5.2.9-1.

Table 6.5.2.9-1: Definition of the Threshold data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this threshold resource.
objectInstanceId	Identifier	1	Identifier of the VNF instance associated with the threshold.
criteria	ThresholdCriteria	1	Criteria that define this threshold.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>object	Link	0..1	Link to a resource representing the VNF instance for which performance information is collected. Shall be present if the VNF instance information is accessible as a resource.

6.5.2.10 Type: PerformanceReport

This type defines the format of a performance report provided by the VNFM to the NFVO as a result of collecting performance information as part of a PM job. The type shall comply with the provisions defined in table 6.5.2.10-1.

Table 6.5.2.10-1: Definition of the PerformanceReport data type

Attribute name	Data type	Cardinality	Description
entries	Structure (inlined)	1..N	List of performance information entries. Each performance report entry is for a given metric of a given object (i.e. VNF instance), but can include multiple collected values.
>objectType	String	1	Defines the object type for which performance information is reported (i.e. VNF type). The string value shall be set to the vnfdId of the VNF instance to which the performance information relates.
>objectInstanceId	Identifier	1	The object instance (i.e. VNF instance) for which the performance metric is reported.
>performanceMetric	String	1	Name of the metric collected. This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [20].
>performanceValues	Structure (inlined)	1..N	List of performance values with associated timestamp.
>>timeStamp	DateTime	1	Time stamp indicating when the data was collected.
>>value	(any type)	1	Value of the metric collected. The type of this attribute shall correspond to the related "Measurement Unit" as defined in clause 7.2 of ETSI GS NFV-IFA 027 [20].

6.5.3 Referenced structured data types

6.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

6.5.3.2 Type: PmNotificationsFilter

This type represents a filter that can be used to subscribe for notifications related to performance management events. It shall comply with the provisions defined in table 6.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 6.5.3.2-1: Definition of the PmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: - ThresholdCrossedNotification - PerformanceInformationAvailable Notification See note.
NOTE: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.			

6.5.3.3 Type: PmJobCriteria

This type represents collection criteria for PM jobs. It shall comply with the provisions defined in table 6.5.3.3-1.

Table 6.5.3.3-1: Definition of the PmJobCriteria data type

Attribute name	Data type	Cardinality	Description
performanceMetric	String	0..N	This defines the types of performance metrics for the specified object instances. Valid values are specified as "Measurement Name" values in clause 7.2 of ETSI GS NFV-IFA 027 [20]. At least one of the two attributes (performance metric or group) shall be present.
performanceMetricGroup	String	0..N	Group of performance metrics. A metric group is a pre-defined list of metrics, known to the producer that it can decompose to individual metrics. Valid values are specified as "Measurement Group" values in clause 7.2 of ETSI GS NFV-IFA 027 [20]. At least one of the two attributes (performance metric or group) shall be present.
collectionPeriod	UnsignedInt	1	Specifies the periodicity at which the producer will collect performance information. The unit shall be seconds. See note 1 and note 2.
reportingPeriod	UnsignedInt	1	Specifies the periodicity at which the producer will report to the consumer. about performance information. The unit shall be seconds. See note 1 and note 2.
reportingBoundary	DateTime	0..1	Identifies a time boundary after which the reporting will stop. The boundary shall allow a single reporting as well as periodic reporting up to the boundary.
NOTE 1: At the end of each reportingPeriod, the producer will inform the consumer about availability of the performance data collected for each completed collection period during this reportingPeriod. The reportingPeriod should be equal to or a multiple of the collectionPeriod. In the latter case, the performance data for the collection periods within one reporting period are reported together. NOTE 2: In particular when choosing short collection and reporting periods, the number of PM jobs that can be supported depends on the capability of the producing entity.			

6.5.3.4 Type: ThresholdCriteria

This type represents criteria that define a threshold. It shall comply with the provisions defined in table 6.5.3.4-1.

Table 6.5.3.4-1: Definition of the ThresholdCriteria data type

Attribute name	Data type	Cardinality	Description
performanceMetric	String	1	Defines the performance metric associated with the threshold. Valid values are specified as "Measurement Name" values in clause 7.2 of ETSI GS NFV-IFA 027 [20].
thresholdType	Enum (inlined)	1	Type of threshold. This attribute determines which other attributes are present in the data structure. Permitted values: - SIMPLE: Single-valued static threshold See note 1.
simpleThresholdDetails	Structure (inlined)	0..1	Details of a simple threshold. Shall be present if thresholdType="SIMPLE".
>thresholdValue	Number	1	The threshold value. Shall be represented as a floating point number.
>hysteresis	Number	1	The hysteresis of the threshold. Shall be represented as a non-negative floating point number. A notification with crossing direction "UP" will be generated if the measured value reaches or exceeds "thresholdValue" + "hysteresis". A notification with crossing direction "DOWN" will be generated if the measured value reaches or undercuts "thresholdValue" - "hysteresis". See note 2.
NOTE 1: In the present document, simple thresholds are defined. The definition of additional threshold types is left for future specification. NOTE 2: The hysteresis is defined to prevent storms of threshold crossing notifications. When processing a request to create a threshold, implementations should enforce a suitable minimum value for this attribute (e.g. override the value or reject the request).			

6.5.4 Referenced simple data types and enumerations

6.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

6.5.4.3 Enumeration: CrossingDirectionType

The enumeration CrossingDirectionType shall comply with the provisions defined in table 6.5.4.3-1.

Table 6.5.4.3-1: Enumeration CrossingDirectionType

Enumeration value	Description
UP	The threshold was crossed in upward direction.
DOWN	The threshold was crossed in downward direction.

7 VNF Fault Management interface

7.1 Description

This interface allows the NFVO to subscribe to notifications regarding VNF alarms provided by the VNFM, and API version information retrieval.

Virtualised resource alarms collected by the VNFM are filtered, correlated and modified by the VNFM and mapped to the corresponding VNF instance, resulting in alarms on that VNF instance which contain information on the VNFC(s) affected by the fault.

The operations provided through this interface are:

- Get Alarm List
- Acknowledge Alarm
- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

7.1a API version

For the VNF fault management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

7.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnffm" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 7.2-1 shows the overall resource URI structure defined for the VNF fault management interface.

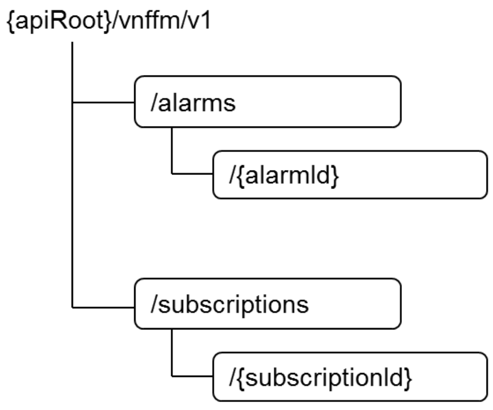

Figure 7.2-1: Resource URI structure of the VNF Fault Management interface

Table 7.2-1 lists the individual resources defined, and the applicable HTTP methods.

The VNFM shall support responding to requests for all HTTP methods on the resources in table 7.2-1 that are marked as "M" (mandatory) in the "Cat" column. The VNFM shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 7.2-1: Resources and methods overview of the VNF Fault Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
Alarms	/alarms	GET	M	Query alarms related to VNF instances
Individual alarm	/alarms/{alarmId}	GET	M	Read individual alarm
		PATCH	M	Acknowledge individual alarm
Subscriptions	/subscriptions	POST	M	Subscribe to VNF alarms
		GET	M	Query multiple subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription
		DELETE	M	Terminate a subscription
Notification endpoint	(client-provided)	POST	See note	Notify about VNF alarms. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The VNFM shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the NFVO. If the NFVO supports invoking the POST method on the "Subscription" resource towards the VNFM, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

7.3 Sequence diagrams (informative)

7.3.1 Flow of the Get Alarm List operation

This clause describes a sequence flow for querying one or multiple alarms.

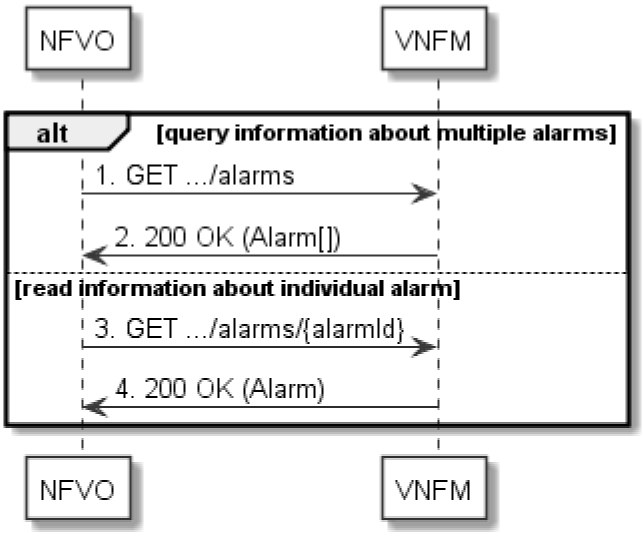
NFVO
VNFM
alt
[query information about multiple alarms]
1. GET .../alarms
2. 200 OK (Alarm[])
[read information about individual alarm]
3. GET .../alarms/{alarmId}
4. 200 OK (Alarm)
NFVO
VNFM

Figure 7.3.1-1: Flow of alarm query/read

Alarm query, as illustrated in figure 7.3.1-1, consists of the following steps:

1) If the NFVO intends to query all alarms, it sends a GET request to the "Alarms " resource.

2) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "Alarm" in the payload body.

3) If the NFVO intends to read a particular alarm, it sends a GET request to the "Individual alarm" resource, addressed by the appropriate alarm identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "Alarm" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

7.3.2 Flow of acknowledging alarm

This clause describes the procedure to acknowledge an individual alarm.

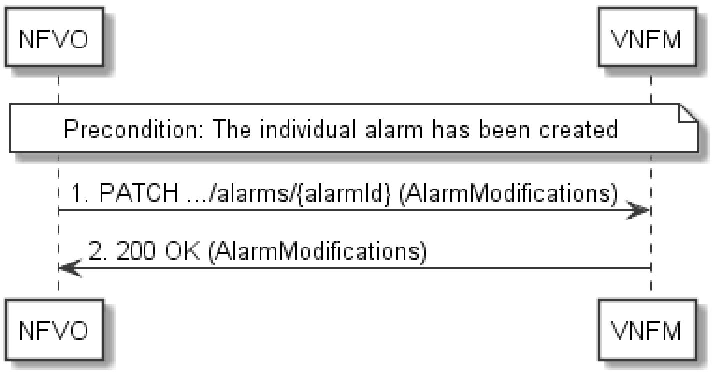
NFVO
VNFM
Precondition: The individual alarm has been created
1. PATCH .../alarms/{alarmId} (AlarmModifications)
2. 200 OK (AlarmModifications)
NFVO
VNFM

Figure 7.3.2-1: Flow of acknowledging alarm

Precondition: The resource representing the individual alarm has been created.

Acknowledge alarm, as illustrated in figure 7.3.2-1, consists of the following steps:

1) The NFVO sends a PATCH request to the individual alarm.

2) The VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "AlarmModifications" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

7.3.3 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF fault management.

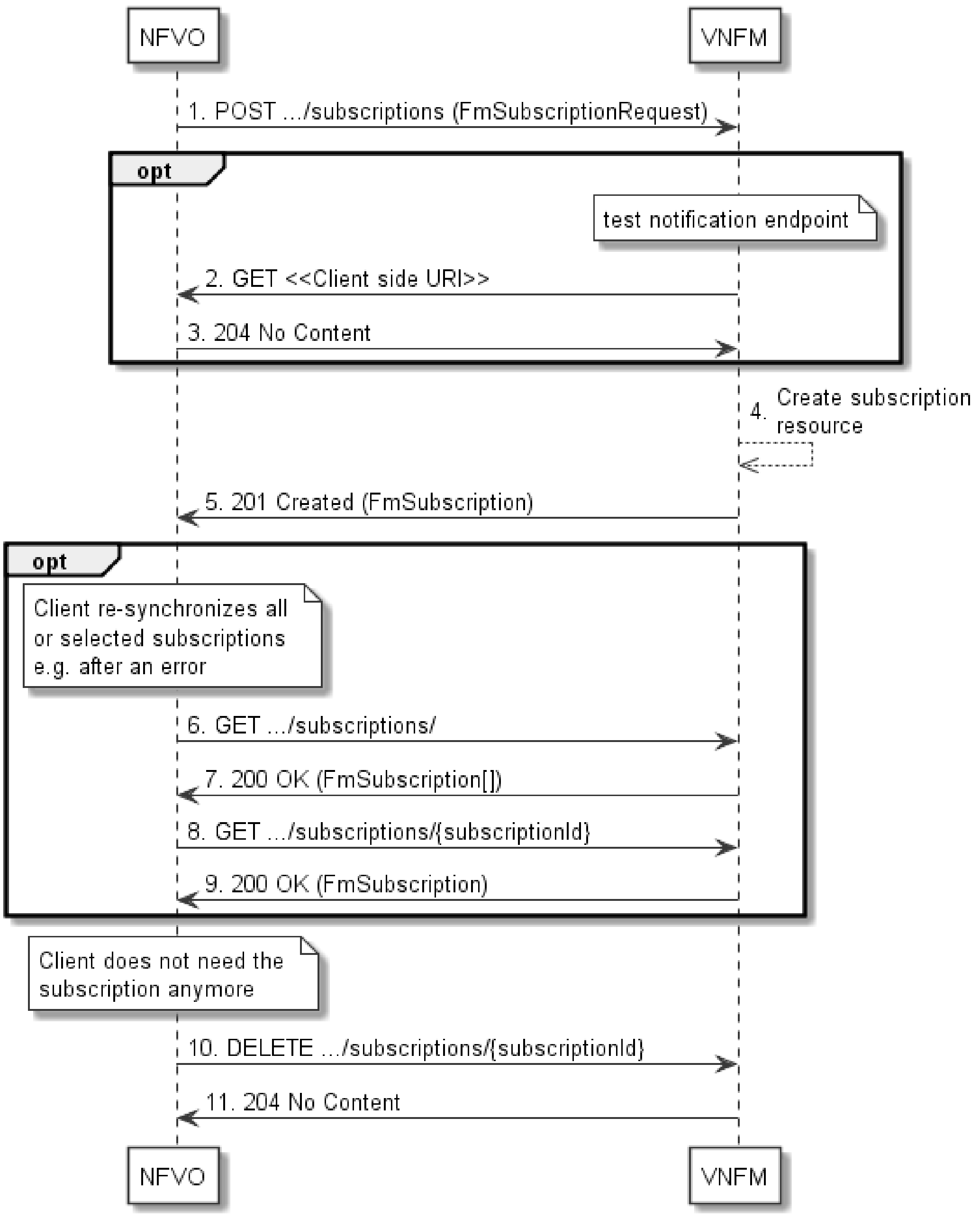
NFVO
VNFM
1. POST .../subscriptions (FmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (FmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (FmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (FmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
NFVO
VNFM

Figure 7.3.3-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 7.3.3-1:

1) The NFVO sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "FmSubscriptionRequest". This data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the NFVO as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3) In that case, the NFVO returns a "204 No Content" response to indicate success.

4) The VNFM creates a new subscription for notifications related to VNF fault management, and a resource that represents this subscription.

5) The VNFM returns a "201 Created" response containing a data structure of type "FmSubscription," representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) Optionally, for example when trying to recover from an error situation, the NFVO may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the VNFM returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the NFVO.

8) Optionally, for example when trying to recover from an error situation, the NFVO may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10) When the NFVO does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription.

11) The VNFM acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

7.3.4 Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF fault management.

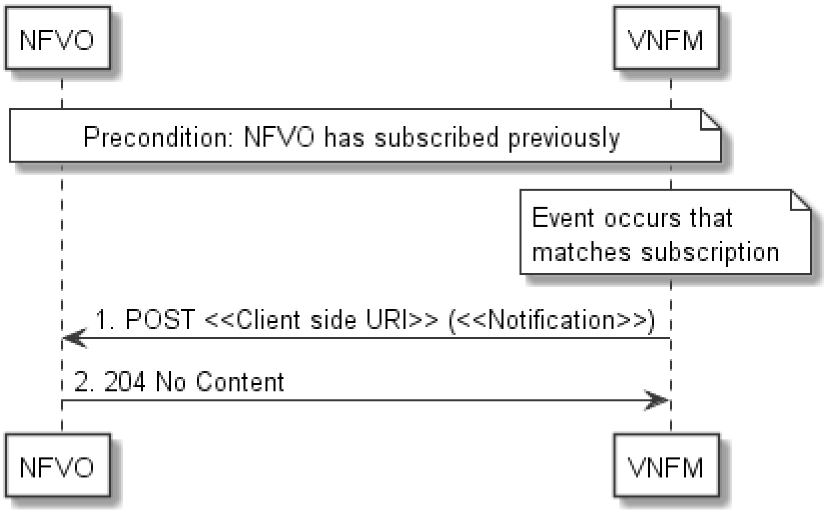
NFVO
VNFM
Precondition: NFVO has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
NFVO
VNFM

Figure 7.3.4-1: Flow of sending notifications

Precondition: The NFVO has subscribed previously for notifications related to VNF fault management.

The procedure consists of the following steps as illustrated in figure 7.3.4-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the NFVO has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API (see clauses 7.5.2.5, 7.5.2.6 and 7.5.2.7).

2) The NFVO acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the NVFO, it can retry sending the notification.

7.4 Resources

7.4.1 Introduction

This clause defines all the resources and methods provided by the VNF fault management interface.

7.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF fault management interface.

7.4.2 Resource: Alarms

7.4.2.1 Description

This resource represents a list of alarms related to VNF instances.

7.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/alarms

This resource shall support the resource URI variables defined in table 7.4.2.2-1.

Table 7.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

7.4.2.3 Resource methods

7.4.2.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.2.3.2 GET

The client can use this method to retrieve information about the alarm list.

This method shall follow the provisions specified in the tables 7.4.2.3.2-1 and 7.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. The following attribute names shall be supported by the VNFM in the attribute-based filtering expression: id, managedObjectId, rootCauseFaultyResource/faultyResourceType, eventType, perceivedSeverity, probableCause.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

NOTE: There are no attribute selectors defined for this resource as the Alarm attributes with cardinality 0..1 or 0..N are not structurally complex in nature.

Table 7.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	Alarm	0..N	200 OK	Information about zero or more alarms was queried successfully. The response body shall contain in an array the representations of zero or more alarms as defined in clause 7.5.2.4. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3 Resource: Individual alarm

7.4.3.1 Description

This resource represents an individual alarm.

7.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/alarms/{alarmId}

This resource shall support the resource URI variables defined in table 7.4.3.2-1.

Table 7.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
alarmId	Identifier of the alarm. See note.
NOTE: This identifier can be retrieved from the "id" attribute of the "alarm" attribute in the AlarmNotification or AlarmClearedNotification. It can also be retrieved from the "id" attribute of the applicable array element in the payload body of the response to a GET request to the "Alarms" resource.	

7.4.3.3 Resource methods

7.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3.3.2 GET

The client can use this method to read an individual alarm.

This method shall follow the provisions specified in the tables 7.4.3.3.2-1 and 7.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Alarm	1	200	Information about an individual alarm was read successfully. The response body shall contain a representation of the individual alarm.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3.3.4 PATCH

This method modifies an individual alarm resource.

This method shall follow the provisions specified in the tables 7.4.3.3.4-1 and 7.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.3.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality		Description
Request body	AlarmModifications	1		The parameter for the alarm modification, as defined in clause 7.5.2.8. The Content-Type header shall be set to "application/merge-patch+json" according to IETF RFC 7396 [17].
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	AlarmModifications	1	200 OK	The request was accepted and completed. The response body shall contain attribute modifications for an 'Individual alarm' resource (see clause 7.5.2.4).
	ProblemDetails	0..1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the "Individual alarm" resource. Typically, this is due to the fact that the alarm is already in the state that is requested to be set (such as trying to acknowledge an already-acknowledged alarm). The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	412 Precondition failed	Error: A precondition given in an HTTP request header is not fulfilled. Typically, this is due to an ETag mismatch, indicating that the resource was modified by another entity. The response body should contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in 4.3.5.5, may be returned.

7.4.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4 Resource: Subscriptions

7.4.4.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF alarms and to query its subscriptions.

7.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/subscriptions

This resource shall support the resource URI variables defined in table 7.4.4.2-1.

Table 7.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

7.4.4.3 Resource methods

7.4.4.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 7.4.4.3.1-1 and 7.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the NFVO, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 7.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
none supported		

Table 7.4.4.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	FmSubscriptionRequest	1		Details of the subscription to be created, as defined in clause 7.5.2.2.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	FmSubscription	1	201 Created	The subscription was created successfully. The response body shall contain a representation of the created subscription resource. The HTTP response shall include a "Location:" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.

	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.4.3.2 GET

The client can use this method to retrieve the list of active subscriptions for VNF alarms subscribed by the client. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 7.4.4.3.2-1 and 7.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the FmSubscription and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 7.4.4.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	FmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of FM subscriptions as defined in clause 7.5.2.3. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5 Resource: Individual subscription

7.4.5.1 Description

This resource represents an individual subscription for VNF alarms. The client can use this resource to read and to terminate a subscription to notifications related to VNF fault management.

7.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 7.4.5.2-1.

Table 7.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

7.4.5.3 Resource methods

7.4.5.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5.3.2 GET

The client can use this method for reading an individual subscription for VNF alarms subscribed by the client.

This method shall follow the provisions specified in the tables 7.4.5.3.2-1 and 7.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.5.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	FmSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5.3.5 DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the tables 7.4.5.3.5-1 and 7.4.5.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.5.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.5.3.5-2: Details of the DELETE request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.6 Resource: Notification endpoint

7.4.6.1 Description

This resource represents a notification endpoint for VNF alarms.

The API producer can use this resource to send notifications related to VNF alarms or about a rebuilt alarm list to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

7.4.6.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 7.4.6.2-1.

Table 7.4.6.2-1: Resource URI variables for this resource

Name	Definition
n/a	

7.4.6.3 Resource methods

7.4.6.3.1 POST

The POST method notifies a VNF alarm or that the alarm list has been rebuilt.

This method shall follow the provisions specified in the tables 7.4.6.3.1-1 and 7.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.6.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one of the alternatives defined in table 7.4.6.3.1-2.

Table 7.4.6.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	AlarmNotification	1	Information of a VNF alarm	
	AlarmClearedNotification	1	Information of the clearance of a VNF alarm	
	AlarmListRebuiltNotification	1	Information that the alarm list has been rebuilt by the VNFM	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification was delivered successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5 may be returned.

7.4.6.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 7.4.6.3.2-1 and 7.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.6.3.2-2: Details of the GET request/response on this resource

Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.5 Data Model

7.5.1 Introduction

This clause defines the request and response data structures of the VNF fault management interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

7.5.2 Resource and notification data types

7.5.2.1 Introduction

This clause defines the data structures to be used in the resource representations and notifications for the VNF fault management interface.

7.5.2.2 Type: FmSubscriptionRequest

This type represents a subscription request related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.2.2-1.

Table 7.5.2.2-1: Definition of the FmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	FmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

7.5.2.3 Type: FmSubscription

This type represents a subscription related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.2.3-1.

Table 7.5.2.3-1: Definition of the FmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	FmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.

7.5.2.4 Type: Alarm

The alarm data type encapsulates information about an alarm. It shall comply with the provisions defined in table 7.5.2.4-1.

Table 7.5.2.4-1: Definition of the Alarm data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this Alarm information element.
managedObjectId	Identifier	1	Identifier of the affected VNF instance.
rootCauseFaultyResource	FaultyResourceInfo	1	The virtualised resources that are causing the VNF fault.
alarmRaisedTime	DateTime	1	Time stamp indicating when the alarm is raised by the managed object.
alarmChangedTime	DateTime	0..1	Time stamp indicating when the alarm was last changed. It shall be present if the alarm has been updated.
alarmClearedTime	DateTime	0..1	Time stamp indicating when the alarm was cleared. It shall be present if the alarm has been cleared.
ackState	Enum (inlined)	1	Acknowledgement state of the alarm. Permitted values: - UNACKNOWLEDGED - ACKNOWLEDGED.
perceivedSeverity	PerceivedSeverityType	1	Perceived severity of the managed object failure.
eventTime	DateTime	1	Time stamp indicating when the fault was observed.
eventType	EventType	1	Type of event.
faultType	String	0..1	Additional information to clarify the type of the fault.
probableCause	String	1	Information about the probable cause of the fault.
isRootCause	Boolean	1	Attribute indicating if this fault is the root for other correlated alarms. If true, then the alarms listed in the attribute CorrelatedAlarmId are caused by this fault.
correlatedAlarmIds	Identifier	0..N	List of identifiers of other alarms correlated to this fault.
faultDetails	String	0..N	Provides additional information about the fault.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>objectInstance	Link	0..1	Link to the resource representing the VNF instance to which the notified alarm is correlated. Shall be present if the VNF instance information is accessible as a resource.

7.5.2.5 Type: AlarmNotification

This type represents an alarm notification about VNF faults. It shall comply with the provisions defined in table 7.5.2.5-1.

This notification shall be triggered by the VNFM when:

- An alarm has been created.
- An alarm has been updated, e.g. if the severity of the alarm has changed.

Table 7.5.2.5-1: Definition of the AlarmNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
alarm	Alarm	1	Information about an alarm including AlarmId, affected VNF identifier, and FaultDetails.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.

7.5.2.6 Type: AlarmClearedNotification

This type represents an alarm cleared notification about VNF faults. It shall comply with the provisions defined in table 7.5.2.6-1.

The notification shall be triggered by the VNFM when an alarm has been cleared.

Table 7.5.2.6-1: Definition of the AlarmClearedNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmClearedNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
alarmId	Identifier	1	Alarm identifier.
alarmClearedTime	DateTime	1	The time stamp indicating when the alarm was cleared.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.
>alarm	NotificationLink	1	Link to the resource that represents the related alarm.

7.5.2.7 Type: AlarmListRebuiltNotification

This type represents a notification that the alarm list has been rebuilt, e.g. if the VNFM detects its storage holding the alarm list is corrupted. It shall comply with the provisions defined in table 7.5.2.7-1.

The notification shall be triggered by the VNFM when the alarm list has been rebuilt.

Table 7.5.2.7-1: Definition of the AlarmListRebuiltNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmListRebuiltNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.
>alarms	NotificationLink	1	Link to the alarm list, i.e. the "Alarms" resource.

7.5.2.8 Type: AlarmModifications

This type represents attribute modifications for an "Individual alarm" resource, i.e. modifications to a resource representation based on the "Alarm" data type. The attributes of "Alarm" that can be modified according to the provisions in clause 7.5.2.4 are included in the "AlarmModifications" data type.

The "AlarmModifications" data type shall comply with the provisions defined in table 7.5.2.8-1.

Table 7.5.2.8-1: Definition of the AlarmModifications data type

Attribute name	Data type	Cardinality	Description
ackState	Enum (inlined)	1	New value of the "ackState" attribute in "Alarm". Permitted values: - ACKNOWLEDGED

7.5.3 Referenced structured data types

7.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

7.5.3.2 Type: FmNotificationsFilter

This type represents a subscription filter related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 7.5.3.2-1: Definition of the FmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: • AlarmNotification • AlarmClearedNotification • AlarmListRebuiltNotification See note.
faultyResourceTypes	FaultyResourceType	0..N	Match VNF alarms with a faulty resource type listed in this attribute.
perceivedSeverities	PerceivedSeverityType	0..N	Match VNF alarms with a perceived severity listed in this attribute.
eventTypes	EventType	0..N	Match VNF alarms with an event type listed in this attribute.
probableCauses	String	0..N	Match VNF alarms with a probable cause listed in this attribute.
NOTE: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.			

7.5.3.3 Type: FaultyResourceInfo

This type represents the faulty virtual resources that have a negative impact on a VNF. It shall comply with the provisions defined in table 7.5.3.3-1.

Table 7.5.3.3-1: Definition of the FaultyResourceInfo data type

Attribute name	Data type	Cardinality	Description
faultyResource	ResourceHandle	1	Information that identifies the faulty resource instance and its managing entity.
faultyResourceType	FaultyResourceType	1	Type of the faulty resource.

7.5.4 Referenced simple data types and enumerations

7.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

7.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

7.5.4.3 Enumeration: PerceivedSeverityType

The enumeration PerceivedSeverityType shall comply with the provisions defined in table 7.5.4.3-1. It indicates the relative level of urgency for operator attention.

Table 7.5.4.3-1: Enumeration PerceivedSeverityType

Enumeration value	Description
CRITICAL	The Critical severity level indicates that a service affecting condition has occurred and an immediate corrective action is required. Such a severity can be reported, for example, when a managed object becomes totally out of service and its capability needs to be restored (Recommendation ITU-T X.733 [25]).
MAJOR	The Major severity level indicates that a service affecting condition has developed and an urgent corrective action is required. Such a severity can be reported, for example, when there is a severe degradation in the capability of the managed object and its full capability needs to be restored (Recommendation ITU-T X.733 [25]).
MINOR	The Minor severity level indicates the existence of a non-service affecting fault condition and that corrective action should be taken in order to prevent a more serious (for example, service affecting) fault. Such a severity can be reported, for example, when the detected alarm condition is not currently degrading the capacity of the managed object (Recommendation ITU-T X.733 [25]).
WARNING	The Warning severity level indicates the detection of a potential or impending service affecting fault, before any significant effects have been felt. Action should be taken to further diagnose (if necessary) and correct the problem in order to prevent it from becoming a more serious service affecting fault (Recommendation ITU-T X.733 [25]).
INDETERMINATE	The Indeterminate severity level indicates that the severity level cannot be determined (Recommendation ITU-T X.733 [25]).
CLEARED	The Cleared severity level indicates the clearing of one or more previously reported alarms. This alarm clears all alarms for this managed object that have the same Alarm type, Probable cause and Specific problems (if given) (Recommendation ITU-T X.733 [25]).

7.5.4.4 Enumeration: EventType

The enumeration EventType represents those types of events that trigger an alarm. It shall comply with the provisions defined in table 7.5.4.4-1.

Table 7.5.4.4-1: Enumeration EventType

Enumeration value	Description
COMMUNICATIONS_ALARM	An alarm of this type is associated with the procedure and/or process required conveying information from one point to another (Recommendation ITU-T X.733 [25]).
PROCESSING_ERROR_ALARM	An alarm of this type is associated with a software or processing fault (Recommendation ITU-T X.733 [25]).
ENVIRONMENTAL_ALARM	An alarm of this type is associated with a condition related to an enclosure in which the equipment resides (Recommendation ITU-T X.733 [25]).
QOS_ALARM	An alarm of this type is associated with degradation in the quality of a service (Recommendation ITU-T X.733 [25]).
EQUIPMENT_ALARM	An alarm of this type is associated with an equipment fault (Recommendation ITU-T X.733 [25]).

7.5.4.5 Enumeration: FaultyResourceType

The enumeration FaultyResourceType represents those types of faulty resource. It shall comply with the provisions defined in table 7.5.4.5-1.

Table 7.5.4.5-1: Enumeration FaultyResourceType

Enumeration value	Description
COMPUTE	Virtual compute resource
STORAGE	Virtual storage resource
NETWORK	Virtual network resource

8 VNF Indicator interface

8.1 Description

This interface allows the VNFM to provide information on value changes of VNF related indicators, and API version information retrieval.

VNF related indicators are declared in the VNFD. This interface is originally produced by the EM and/or VNF on the Ve-Vnfm-em and/or Ve-Vnfm-vnf reference point respectively (see ETSI GS NFV-SOL 002 [i.2]) and is re-exposed by the VNFM towards the NFVO.

The operations provided through this interface are:

- Get Indicator Value
- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

8.1a API version

For the VNF indicator interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

8.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnfind" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 8.2-1 shows the overall resource URI structure defined for the VNF Indicator interface.

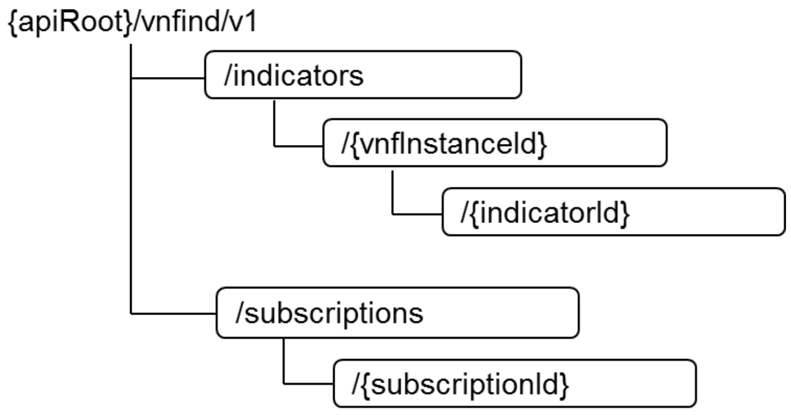

Figure 8.2-1: Resource URI structure of the VNF Indicator Interface

Table 8.2-1 lists the individual resources defined, and the applicable HTTP methods.

The VNFM shall support responding to requests for all HTTP methods on the resources in table 8.2-1 that are marked as "M" (mandatory) in the "Cat" column. The VNFM shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 8.2-1: Resources and methods overview of the VNF Indicator interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
VNF indicators	/indicators	GET	M	Query multiple VNF indicators. See note 1.
VNF indicators related to a VNF instance	/indicators/{vnfInstanceId}	GET	M	Query multiple VNF indicators related to one VNF instance. See note 1.
Individual VNF indicator	/indicators/{vnfInstanceId}/{indicatorId}	GET	M	Read an individual VNF indicator.
Subscriptions	/subscriptions	POST	M	Subscribe to VNF indicator change notifications.
		GET	M	Query multiple subscriptions.
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription.
		DELETE	M	Terminate a subscription.
Notification endpoint	(client-provided)	POST	See note 2	Notify about VNF indicator change. See note 2.
		GET	See note 2	Test the notification endpoint See note 2.
NOTE 1: This resource allows to query all VNF indicators that are known to the VNFM. NOTE 2: The VNFM shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the NFVO. If the NFVO supports invoking the POST method on the "Subscription" resource towards the VNFM, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

8.3 Sequence diagrams (informative)

8.3.1 Flow of querying VNF indicators

This clause describes a sequence for querying VNF indicators.

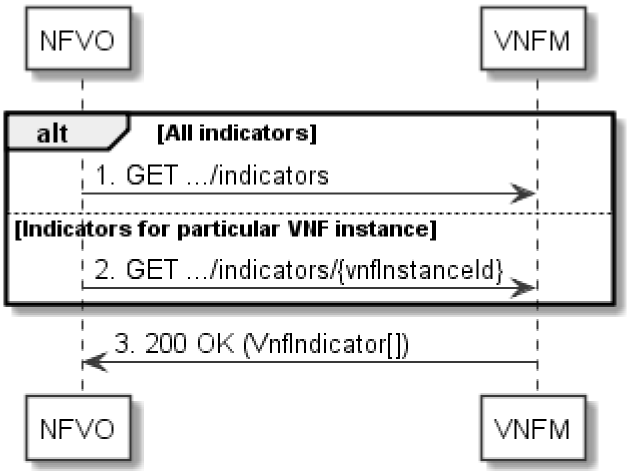
NFVO
VNFM
alt
[All indicators]
1. GET .../indicators
[Indicators for particular VNF instance]
2. GET .../indicators/{vnfInstanceId}
3. 200 OK (VnfIndicator[])
NFVO
VNFM

Figure 8.3.1-1: Flow of querying VNF indicators

VNF indicator query, as illustrated in figure 8.3.1-1, consists of the following steps:

1) If the NFVO intends to query all VNF indicators, it sends a GET request to the "VNF indicators" resource.

2) If the NFVO intends to query the VNF indicators of a particular VNF instance, it sends a GET request to the "VNF indicators related to a VNF instance" resource.

3) The VNFM returns a "200 OK" response to the NFVO, and includes zero or more data structures of type "VnfIndicator" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

8.3.2 Flow of reading a VNF indicator

This clause describes a sequence for reading a VNF indicator, i.e. for getting the indicator value.

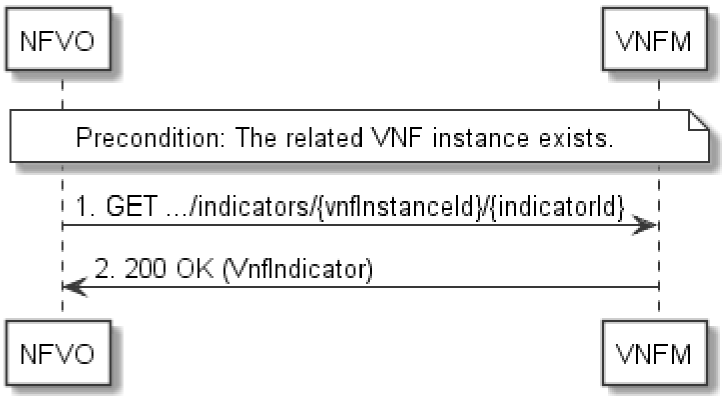
NFVO
VNFM
Precondition: The related VNF instance exists.
1. GET .../indicators/{vnfInstanceId}/{indicatorId}
2. 200 OK (VnfIndicator)
NFVO
VNFM

Figure 8.3.2-1: Flow of reading a VNF indicator

Precondition: The related VNF instance exists.

Reading a VNF indicator, as illustrated in figure 8.3.2-1, consists of the following steps:

1) The NFVO sends a GET request to the "Individual VNF indicator" resource that is to be read.

2) The VNFM returns a "200 OK" response to the NFVO, and includes a data structure of type "VnfIndicator" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

8.3.3 Flow of managing subscriptions

This clause describes the procedure for creating, querying/reading and terminating subscriptions to notifications related to VNF indicator value changes.

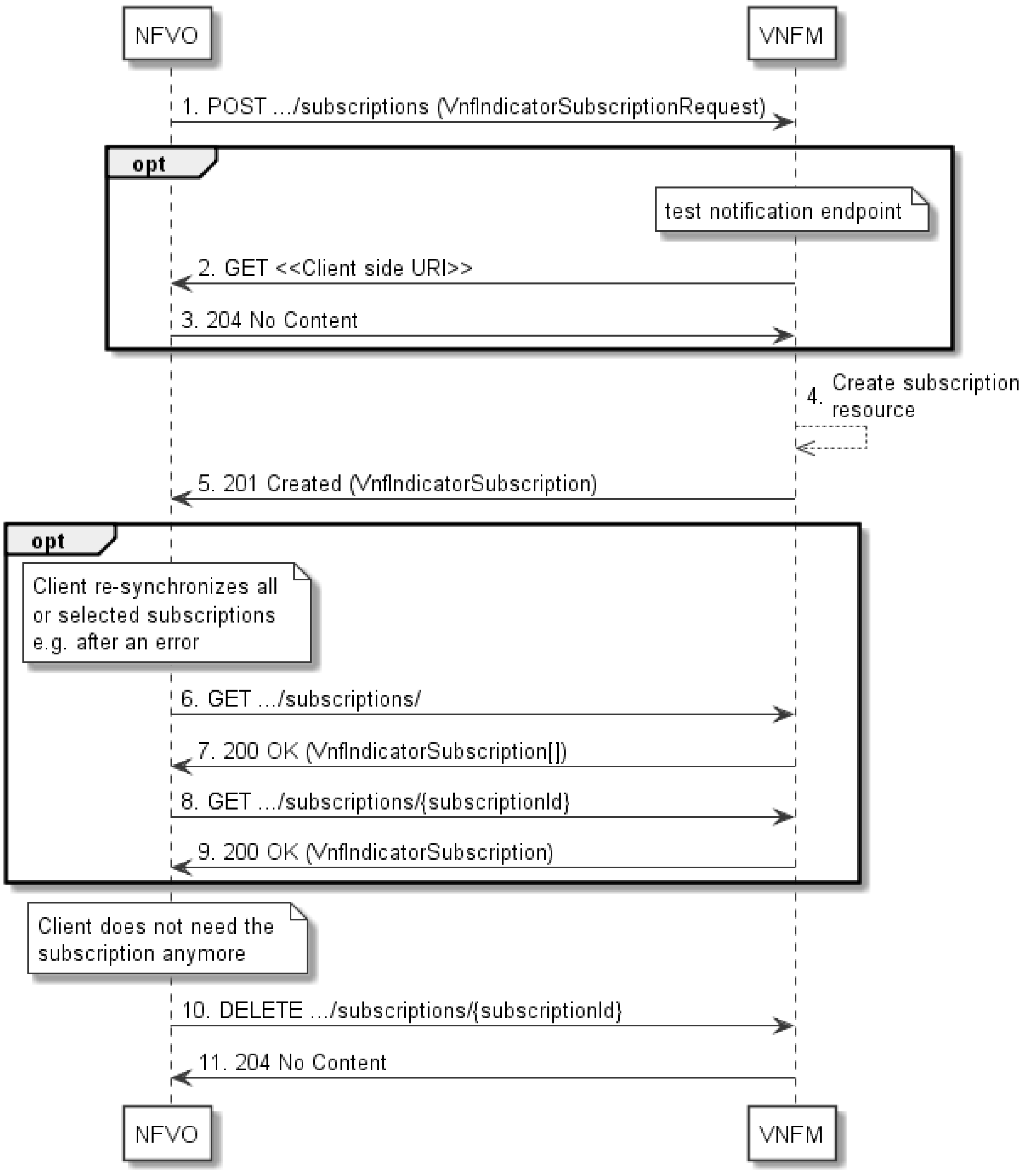
NFVO
VNFM
1. POST .../subscriptions (VnfIndicatorSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (VnfIndicatorSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (VnfIndicatorSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (VnfIndicatorSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
NFVO
VNFM

Figure 8.3.3-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 8.3.3-1:

1) The NFVO sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "VnfIndicatorSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the NFVO as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3) In that case, the NFVO returns a "204 No Content" response to indicate success.

4) The VNFM creates a new subscription to notifications related to VNF indicator value changes, and a resource that represents this subscription.

5) The VNFM returns a 201 Created response containing a data structure of type "VnfIndicatorSubscription" representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) If desired, e.g. to recover from an error situation, the NFVO may query information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the VNFM returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the NFVO.

8) If desired, e.g. to recover from an error situation, the NFVO may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10) If the NFVO does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11) The NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

8.3.4 Flow of sending notifications

This clause describes the procedure for sending notifications.

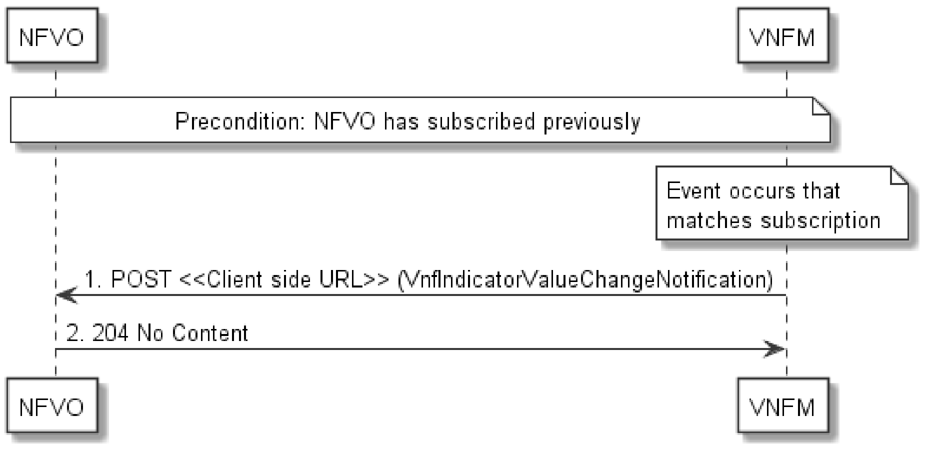
NFVO
VNFM
Precondition: NFVO has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URL>> (VnfIndicatorValueChangeNotification)
2. 204 No Content
NFVO
VNFM

Figure 8.3.4-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in figure 8.3.4-1.

Precondition: The NFVO has subscribed previously to notifications related to VNF indicator value changes.

1) If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates an VnfIndicatorValueChangeNotification that includes information about the event, and sends it in the body of a POST request to the client side URI which the NFVO has registered as part of the subscription request.

2) The NFVO acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the NVFO, it can retry sending the notification.

8.4 Resources

8.4.1 Introduction

This clause defines all the resources and methods provided by the VNF indicator interface.

8.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF indicator interface.

8.4.2 Resource: VNF indicators

8.4.2.1 Description

This resource represents VNF indicators. The client can use this resource to query multiple VNF indicators.

8.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators

This resource shall support the resource URI variables defined in table 8.4.2.2-1.

Table 8.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

8.4.2.3 Resource methods

8.4.2.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.2 GET

The GET method queries multiple VNF indicators.

This method shall follow the provisions specified in the tables 8.4.2.3.2-1 and 8.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the VnfIndicator data type and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 8.4.2.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfIndicator	0..N	200 OK	Information about zero or more VNF indicators was queried successfully. The response body shall contain in an array the representations of all VNF indicators that match the attribute filter, i.e. zero or more representations of VNF indicators as defined in clause 8.5.2.2. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3 Resource: VNF indicators related to a VNF instance

8.4.3.1 Description

This resource represents VNF indicators related to a VNF instance. The client can use this resource to query multiple VNF indicators that are related to a particular VNF instance.

8.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators/{vnfInstanceId}

This resource shall support the resource URI variables defined in table 8.4.3.2-1.

Table 8.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to which the VNF indicator applies. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

8.4.3.3 Resource methods

8.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3.3.2 GET

The GET method queries multiple VNF indicators related to a VNF instance.

This method shall follow the provisions specified in the tables 8.4.3.3.2-1 and 8.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the VnfIndicator data type and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 8.4.3.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	VnfIndicator	0..N	200 OK	Information about zero or more VNF indicators was queried successfully. The response body shall contain in an array the representations of all VNF indicators that are related to the particular VNF instance and that match the attribute filter, i.e. zero or more representations of VNF indicators as defined in clause 8.5.2.2. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4 Resource: Individual VNF indicator

8.4.4.1 Description

This resource represents an individual VNF indicator. The client can use this resource to read an individual VNF indicator.

8.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators/{vnfInstanceId}/{indicatorId}

This resource shall support the resource URI variables defined in table 8.4.4.2-1.

Table 8.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to which the VNF indicator applies. See note 1.
indicatorId	Identifier of the VNF indicator. See note 2.
NOTE 1: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response. NOTE 2: This identifier can be retrieved from the resource referenced by the payload body in the response to a POST request creating a new VNF instance resource.	

8.4.4.3 Resource methods

8.4.4.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4.3.2 GET

The GET method reads a VNF indicator.

This method shall follow the provisions specified in the tables 8.4.4.3.2-1 and 8.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.4.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VnfIndicator	1	200 OK	The VNF indicator was read successfully. The response body shall contain the representation of the VNF indicator.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5 Resource: Subscriptions

8.4.5.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF indicator value changes, and to query its subscriptions.

8.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/subscriptions

This resource shall support the resource URI variables defined in table 8.4.5.2-1.

Table 8.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.

8.4.5.3 Resource methods

8.4.5.3.1 POST

The POST method creates a new subscription.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the NFVO, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

This method shall follow the provisions specified in the tables 8.4.5.3.1-1 and 8.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.5.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	VnfIndicatorSubscriptionRequest	1		Details of the subscription to be created.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfIndicatorSubscription	1	201 Created	The subscription was created successfully. The response body shall contain a representation of the created subscription resource. The HTTP response shall include a "Location" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.5.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 8.4.5.3.2-1 and 8.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The VNFM shall support receiving this parameter as part of the URI query string. The NFVO may supply this parameter. All attribute names that appear in the VnfIndicatorSubscription data type and in data types referenced from it shall be supported by the VNFM in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 8.4.5.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfIndicatorSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method which match the attribute filter, i.e. zero or more representations of VNF indicator subscriptions as defined in clause 8.5.2.4. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6 Resource: Individual subscription

8.4.6.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to VNF indicator value changes.

8.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 8.4.6.2-1.

Table 8.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

8.4.6.3 Resource methods

8.4.6.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6.3.2 GET

The GET method reads an individual subscription.

This method shall follow the provisions specified in the tables 8.4.6.3.2-1 and 8.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.6.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VnfIndicatorSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 8.4.6.3.5-1 and 8.4.6.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.6.3.5-2: Details of the DELETE request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.7 Resource: Notification endpoint

8.4.7.1 Description

This resource represents a notification endpoint.

The API producer can use this resource to send notifications related to VNF indicator value changes to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

8.4.7.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 8.4.7.2-1.

Table 8.4.7.2-1: Resource URI variables for this resource

Name	Definition
n/a	

8.4.7.3 Resource methods

8.4.7.3.1 POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 8.4.7.3.1-1 and 8.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one instance of the VnfIndicatorValueChangeNotification structure.

Table 8.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	VnfIndicatorValueChangeNotification	1	A notification about VNF indicator value changes.	
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The notification was delivered successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.7.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 8.4.7.3.2-1 and 8.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.7.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.5 Data model

8.5.1 Introduction

This clause defines the request and response data structures of the VNF Indicator interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

8.5.2 Resource and notification data types

8.5.2.1 Introduction

This clause defines the data structures to be used in resource representations and notifications.

8.5.2.2 Type: VnfIndicator

This type represents a VNF indicator value. It shall comply with the provisions defined in table 8.5.2.2-1.

Table 8.5.2.2-1: Definition of the VnfIndicator data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of this VNF indicator.
name	String	0..1	Human readable name of the indicator. Shall be present if defined in the VNFD.
value	Object	1	Provides the value of the indicator. The value format is defined in the VNFD. See note.
vnfInstanceId	Identifier	1	Identifier of the VNF instance which provides the indicator value.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>vnfInstance	Link	1	Link to the related VNF instance resource.
NOTE: ETSI GS NFV-SOL 001 [i.6] specifies the structure and format of the VNFD based on TOSCA specifications.			

8.5.2.3 Type: VnfIndicatorSubscriptionRequest

This type represents a subscription request related to VNF indicator value change notifications. It shall comply with the provisions defined in table 8.5.2.3-1.

Table 8.5.2.3-1: Definition of the VnfIndicatorSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	VnfIndicatorNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

8.5.2.4 Type: VnfIndicatorSubscription

This type represents a subscription related to notifications about VNF indicator value changes. It shall comply with the provisions defined in table 8.5.2.4-1.

Table 8.5.2.4-1: Definition of the VnfIndicatorSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	VnfIndicatorNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.

8.5.2.5 Type: VnfIndicatorValueChangeNotification

This type represents a VNF indicator value change notification. It shall comply with the provisions defined in table 8.5.2.5-1.

The notification shall be triggered by the VNFM when the value of an indicator has changed.

Table 8.5.2.5-1: Definition of the VnfIndicatorValueChangeNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIndicatorValueChangeNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfIndicatorId	IdentifierInVnfd	1	Identifier of the VNF indicator whose value has changed.
name	String	0..1	Human readable name of the VNF indicator. Shall be present if defined in the VNFD.
value	Object	1	Provides the value of the VNF indicator. The value format is defined in the VNFD. See note.
vnfInstanceId	Identifier	1	Identifier of the VNF instance which provides the indicator value.
_links	Structure (inlined)	1	Links for this resource.

Attribute name	Data type	Cardinality	Description
>vnfInstance	NotificationLink	0..1	Link to the related VNF instance resource. Shall be present if the VNF instance information is accessible as a resource.
>subscription	NotificationLink	1	Link to the related subscription.
NOTE: ETSI GS NFV-SOL 001 [i.6] specifies the structure and format of the VNFD based on TOSCA specifications.			

8.5.3 Referenced structured data types

8.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

8.5.3.2 Type: VnfIndicatorNotificationsFilter

This type represents a subscription filter related to notifications about VNF indicator value changes. It shall comply with the provisions defined in table 8.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 8.5.3.2-1: Definition of the VnfIndicatorNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
indicatorIds	IdentifierInVnfd	0..N	Match particular VNF indicator identifiers.

8.5.4 Referenced simple data types and enumerations

No particular simple data types and enumerations are defined for this interface, in addition to those defined in clause 4.4.

9 VNF Lifecycle Operation Granting interface

9.1 Description

This interface allows the VNFM to obtain from the NFVO permission and configuration parameters for a VNF lifecycle operation. Further, this interface allows API version information retrieval.

The operation provided through this interface is:

- Grant Lifecycle Operation.

This operation allows the VNFM to request a grant for authorization of a VNF lifecycle operation. This interface supports multiple use cases, such as:

- The NFVO can approve or reject a request based on policies (e.g. dependencies between VNFs) and available capacity.
- When applicable, the NFVO can reserve resources based on the VNFM's virtualised resources request.
- The NFVO can provide to the VNFM information about the VIM where cloud resources are allocated. This can include additional information such as the resource zone.

When requesting resource creation or modification, the VNFM references the resource definitions that are available to the NFVO in the VNFD. When resources are to be released or modified, the VNFM provides references to the existing resources in the request.

Per each VNFM, one of the following operator policies can be selected as a configuration, by means outside the scope of the present document, to determine how the NFVO and the VNFM handle resource reservations in a grant request:

1) Policy GRANT_APPROVE: The NFVO approves the VIM resources to be allocated by the VNFM. In general, resource availability is not guaranteed. No explicit reservation identifier is returned to the VNFM. Optionally, to guarantee resource availability, the NFVO may do a reservation and use implicit reservation identification towards the VNFM, i.e. associate the reservation to the VIM access information.

2) Policy GRANT_RESERVE_MULTI: The NFVO guarantees the availability of the VIM resources to be allocated. The NFVO provides to the VNFM multiple reservation identifiers, one per granted resource requirement. Each such identifier identifies the reservation which is applicable to the resource requirements and which the VNFM shall use in the subsequent resource management operation.

3) Policy GRANT_RESERVE_SINGLE: The NFVO guarantees the availability of the VIM resources to be allocated. The NFVO provides to the VNFM a single reservation identifier per resource type (i.e. compute, network and storage). This identifier identifies the reservation which is applicable to all granted resource requirements of that type for the granted lifecycle operation.

These policies are used to configure the behaviour of both the NFVO and the VNFM identically, also considering the resource reservation capabilities of the VIM.

In the GrantVnfLifecycleOperation response, the NFVO can return information that allows to distribute the resources of a VNF over multiple resource zones. This decision is guided by affinity/anti-affinity rules in the VNFD as well as by placement constraints passed in the GrantVnfLifecycleOperation request. The NFVO can also return information that allows to manage the resources of a VNF using multiple VIMs, guided by VIM selection constraints passed in the GrantVnfLifecycleOperation request.

NOTE: In the present document, as part of that mechanism, attributes are defined for signalling the decision to use multiple VIMs per VNF. However, to actually support VNFs that include resources managed by multiple VIMs, additionally a mechanism is needed to manage the VNF-internal Virtual Link (VL) requirements across multiple VIMs. Such functionality is not specified; neither in the present document, nor in other documents referenced by the present document. Also, the current mechanism of signalling external and externally-managed VLs in the lifecycle management operations assumes single-VIM VNFs, and does not fulfil the requirements of multi-VIM scenarios.

9.1a API version

For the VNF lifecycle operation granting interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

9.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "grant" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 9.2-1 shows the overall resource URI structure defined for the VNF Lifecycle Operation Granting interface.

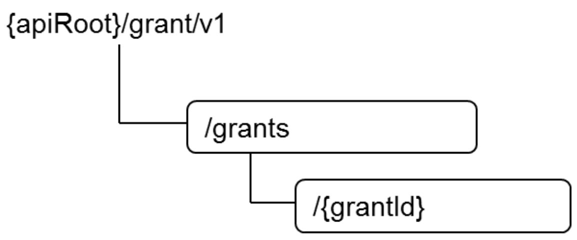

Figure 9.2-1: Resource URI structure of the VNF Lifecycle Operation Granting Interface

Table 9.2-1 lists the individual resources defined, and the applicable HTTP methods.

The NFVO shall support responding to requests for all HTTP methods on the resources in table 9.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 9.2-1: Resources and methods overview of the VNF Lifecycle Operation Granting interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
Grants	/grants	POST	M	Request a grant
Individual grant	/grants/{grantId}	GET	M	Read a grant

9.3 Sequence diagrams (informative)

9.3.1 Flow of grant request with synchronous response

This clause describes a sequence for a grant request with synchronous (i.e. immediate) response. If the NFVO can decide immediately what to respond to a grant request, it returns the response immediately.

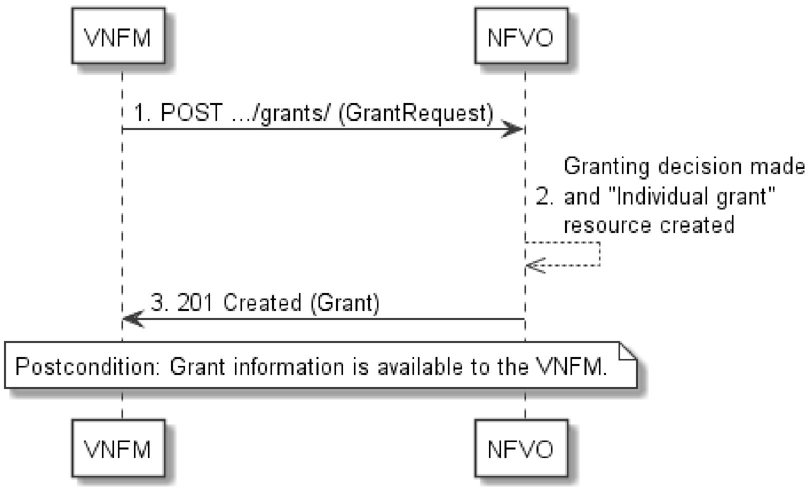
VNFM
NFVO
1. POST .../grants/ (GrantRequest)
2. Granting decision made and "Individual grant" resource created
3. 201 Created (Grant)
Postcondition: Grant information is available to the VNFM.
VNFM
NFVO

Figure 9.3.1-1: Flow of granting with synchronous response

Granting with synchronous response, as illustrated in figure 9.3.1-1, consists of the following steps:

1) The VNFM sends a POST request to the "Grants" resource with a "GrantRequest" data structure in the body.

2) The NFVO makes the granting decision, and creates a new "Individual grant" resource.

3) The NFVO returns to the VNFM a "201 Created" response with a "Grant" data structure in the body and a "Location" HTTP header that points to the new "Individual grant" resource.

Postcondition: The grant information is available to the VNFM.

9.3.2 Flow of grant request with asynchronous response

This clause describes a sequence for a grant request with asynchronous (i.e. delayed) response. If the NFVO can not decide immediately what to respond to a grant request, and therefore runs the risk of a timeout of the http connection while waiting for the completion of the decision, it returns the response in an asynchronous (delayed) fashion.

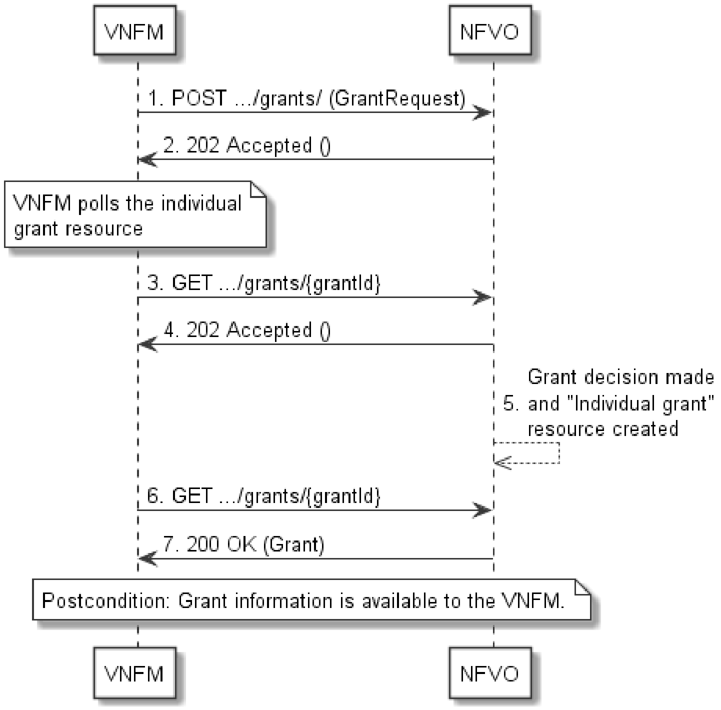
VNFM
NFVO
1. POST .../grants/ (GrantRequest)
2. 202 Accepted ()
VNFM polls the individual grant resource
3. GET .../grants/{grantId}
4. 202 Accepted ()
5. Grant decision made and "Individual grant" resource created
6. GET .../grants/{grantId}
7. 200 OK (Grant)
Postcondition: Grant information is available to the VNFM.
VNFM
NFVO

Figure 9.3.2-1: Flow of granting with asynchronous response

Granting with asynchronous response, as illustrated in figure 9.3.2-1, consists of the following steps:

1) The VNFM sends a POST request to the "Grants" resource with a "GrantRequest" data structure in the body.

2) The NFVO returns to the VNFM a "202 Accepted" response with an empty body and a "Location" HTTP header that indicates the URI of the "Individual grant" resource that will be created once the granting decision will have been made.

3) The VNFM tries to obtain the grant by sending a GET request to the NFVO, using the URI that was returned in step (2) in the "Location" header.

4) As there is no result of the granting decision available yet and consequently the "Individual grant" resource is still in the process of being created, the NFVO returns a "202 Accepted" response with an empty body.

5) The NFVO finalizes the granting decision and creates the "Individual grant" that contains the grant.

6) The VNFM tries to obtain the grant by sending a GET request to the NFVO, using the URI that was returned in step (2) in the "Location" header.

7) This time, the grant is available, and the NFVO returns a "200 OK" response with a "Grant" data structure in the body.

Postcondition: The grant information is available to the VNFM.

9.4 Resources

9.4.1 Introduction

This clause defines all the resources and methods provided by the VNF lifecycle operation granting interface.

9.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF lifecycle operation granting interface.

9.4.2 Resource: Grants

9.4.2.1 Description

This resource represents grants. The client can use this resource to obtain permission from the NFVO to perform a particular VNF lifecycle operation.

9.4.2.2 Resource definition

The resource URI is:

{apiRoot}/grant/v1/grants

This resource shall support the resource URI variables defined in table 9.4.2.2-1.

Table 9.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

9.4.2.3 Resource methods

9.4.2.3.1 POST

The POST method requests a grant for a particular VNF lifecycle operation.

This method shall follow the provisions specified in the tables 9.4.2.3.1-1 and 9.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.2.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	GrantRequest	1		
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Grant	1	201 Created	The grant was created successfully (synchronous mode). A representation of the created "Individual grant" resource shall be returned in the response body. The HTTP response shall include a "Location" HTTP header that indicates the URI of the "Individual grant" resource just created.
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. It is expected to take some time to create the grant (asynchronous mode). The response body shall be empty. The HTTP response shall include a "Location" HTTP header that indicates the URI of the "Individual grant" resource that will be created once the granting decision has been made.
	ProblemDetails	1	403 Forbidden	The grant was rejected. A ProblemDetails structure shall be included in the response to provide more details about the rejection in the "details" attribute.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

9.4.2.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3 Resource: Individual grant

9.4.3.1 Description

This resource represents an individual grant. The client can use this resource to read the grant.

It is determined by means outside the scope of the present document, such as configuration or policy, how long an individual grant is available.

9.4.3.2 Resource definition

The resource URI is:

{apiRoot}/grant/v1/grants/{grantId}

This resource shall support the resource URI variables defined in table 9.4.3.2-1.

Table 9.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
grantId	Identifier of the grant. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request granting a new VNF lifecycle operation. It can also be retrieved from the "id" attribute in the payload body of that response.	

9.4.3.3 Resource methods

9.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3.3.2 GET

The GET method reads a grant.

This method shall follow the provisions specified in the tables 9.4.3.3.2-1 and 9.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.3.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Grant	1	200 OK	The grant was read successfully. A representation of the "individual grant" resource shall be returned in the response body.
	n/a		202 Accepted	The process of creating the grant is ongoing, no grant is available yet. The response body shall be empty.
	ProblemDetails	1	403 Forbidden	The grant was rejected. A ProblemDetails structure shall be included in the response to provide more details about the rejection in the "details" attribute.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

9.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.5 Data model

9.5.1 Introduction

This clause defines the request and response data structures of the VNF Lifecycle Operation Granting interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

9.5.2 Resource and notification data types

9.5.2.1 Introduction

This clause defines data structures to be used in resource representations and notifications.

9.5.2.2 Type: GrantRequest

This type represents a grant request. It shall comply with the provisions defined in table 9.5.2.2-1.

Table 9.5.2.2-1: Definition of the GrantRequest data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance which this grant request is related to. Shall also be provided for VNFs that not yet exist but are planned to exist in the future, i.e. if the grant is requested for InstantiateVNF.
vnfLcmOpOccId	Identifier	1	The identifier of the VNF lifecycle management operation occurrence associated to the GrantRequest.
vnfdId	Identifier	1	Identifier of the VNFD that defines the VNF for which the LCM operation is to be granted.
flavourId	Identifier	0..1	Identifier of the VNF deployment flavour of the VNFD that defines the VNF for which the LCM operation is to be granted. Shall be provided when instantiating the VNF or changing the deployment flavour of the VNF instance.
operation	GrantedLcmOperationType	1	The lifecycle management operation for which granting is requested. See note 1.
isAutomaticInvocation	Boolean	1	Set to true if this VNF LCM operation occurrence has been triggered by an automated procedure inside the VNFM (i.e. ScaleVnf / ScaleVnfToLevel triggered by auto-scale, or HealVnf triggered by auto-heal). Set to false otherwise.
instantiationLevelId	Identifier	0..1	If operation=INSTANTIATE, the identifier of the instantiation level may be provided as an alternative way to define the resources to be added. This attribute shall only be used if operation=INSTANTIATE. See note 2.
addResources	ResourceDefinition	0..N	List of resource definitions in the VNFD for resources to be added by the LCM operation which is related to this grant request, with one entry per resource. See note 2.
tempResources	ResourceDefinition	0..N	List of resource definitions in the VNFD for resources to be temporarily instantiated during the runtime of the LCM operation which is related to this grant request, with one entry per resource. See note 3.
removeResources	ResourceDefinition	0..N	Provides the definitions of resources to be removed by the LCM operation which is related to this grant request, with one entry per resource.
updateResources	ResourceDefinition	0..N	Provides the definitions of resources to be modified by the LCM operation which is related to this grant request, with one entry per resource.

Attribute name	Data type	Cardinality	Description
placementConstraints	PlacementConstraint	0..N	Placement constraints that the VNFM may send to the NFVO in order to influence the resource placement decision. If sent, the NFVO shall take the constraints into consideration when making resource placement decisions, and shall reject the grant if they cannot be honoured. See note 4 and note 5.
vimConstraints	VimConstraint	0..N	Used by the VNFM to require that multiple resources are managed through the same VIM connection. If sent, the NFVO shall take the constraints into consideration when making VIM selection decisions, and shall reject the grant if they cannot be honoured. This attribute shall be supported if VNF-related Resource Management in direct mode is applicable. The applicability and further details of this attribute for indirect mode are left for future specification.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the VNFM, specific to the VNF and the LCM operation.
_links	Structure (inlined)	1	Links to resources related to this request.
>vnfLcmOpOcc	Link	1	Related lifecycle management operation occurrence.
>vnfInstance	Link	1	Related VNF instance.
NOTE 1: The VNF LCM operations CreateVnfIdentifier, DeleteVnfIdentifier, QueryVnf and ModifyVnfInformation can be executed by the VNFM without requesting granting. NOTE 2: If the granting request is for InstantiateVNF, either instantiationLevel or addResources shall be present. NOTE 3: The NFVO will assume that the VNFM will be responsible to both allocate and release the temporary resource during the runtime of the LCM operation. This means, the resource can be allocated and consumed after the "start" notification for the LCM operation is sent by the VNFM, and the resource will be-released before the "result" notification of the VNF LCM operation is sent by the VNFM. NOTE 4: The affinity/anti-affinity rules defined in the VNFD , and the placement constraints in the GrantVnfLifecycleOperation as defined in this clause should be conflict-free. In case of conflicts, the placement constraints in the GrantVnfLifecycleOperation shall take precedence. NOTE 5: Passing constraints allows the VNFM or the lifecycle management scripts to influence resource placement decisions by the NFVO to ensure VNF properties such as performance or fault tolerance.			

9.5.2.3 Type: Grant

This type represents a grant. It shall comply with the provisions defined in table 9.5.2.3-1.

Table 9.5.2.3-1: Definition of the Grant data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the grant.
vnfInstanceId	Identifier	1	Identifier of the related VNF instance.
vnfLcmOpOccId	Identifier	1	Identifier of the related VNF lifecycle management operation occurrence.

Attribute name	Data type	Cardinality	Description
vimConnections	VimConnectionInfo	0..N	Provides information regarding VIM connections that are approved to be used by the VNFM to allocate resources, and provides parameters of these VIM connections. The VNFM shall update the " vimConnectionInfo" attribute of the "VnfInstance" structure by adding unknown entries received in this attribute. This attribute is not intended for the modification of VimConnection entries passed earlier; for that, the VnfInfoModificationRequest structure shall be used. This attribute shall only be supported when VNF-related Resource Management in direct mode is applicable. In direct mode, this parameter shall be absent if the VIM information was configured to the VNFM in another way, present otherwise. See note 1.
zones	ZoneInfo	0..N	Identifies resource zones where the resources are approved to be allocated by the VNFM.
zoneGroups	ZoneGroupInfo	0..N	Information about groups of resource zones that are related and that the NFVO has chosen to fulfil a zoneGroup constraint in the GrantVnfLifecycleOperation request. This information confirms that the NFVO has honoured the zoneGroup constraints that were passed as part of "placementConstraints" in the GrantRequest.
computeReservationId	IdentifierInVim	0..1	Information that identifies a reservation applicable to the compute resource requirements of the corresponding grant request. See note 2.
networkReservationId	IdentifierInVim	0..1	Information that identifies a reservation applicable to the network resource requirements of the corresponding grant request. See note 2.
storageReservationId	IdentifierInVim	0..1	Information that identifies a reservation applicable to the storage resource requirements of the corresponding grant request. See note 2.
addResources	GrantInfo	0..N	List of resources that are approved to be added, with one entry per resource.
tempResources	GrantInfo	0..N	List of resources that are approved to be temporarily instantiated during the runtime of the lifecycle operation, with one entry per resource.
removeResources	GrantInfo	0..N	List of resources that are approved to be removed, with one entry per resource.

Attribute name	Data type	Cardinality	Description
updateResources	GrantInfo	0..N	List of resources that are approved to be modified, with one entry per resource.
vimAssets	Structure (inlined)	0..1	Information about assets for the VNF that are managed by the NFVO in the VIM, such as software images and virtualised compute resource flavours. This attribute is not intended for the modification of vimAssets entries passed earlier. See note 3.
>computeResourceFlavours	VimComputeResourceFlavour	0..N	Mappings between virtual compute descriptors defined in the VNFD and compute resource flavours managed in the VIM.
>softwareImages	VimSoftwareImage	0..N	Mappings between software images defined in the VNFD and software images managed in the VIM.
extVirtualLinks	ExtVirtualLinkData	0..N	Information about external VLs to connect the VNF to. See note 5.
extManagedVirtualLinks	ExtManagedVirtualLinkData	0..N	Information about internal VLs that are managed by other entities than the VNFM. See note 4 and note 5.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO, specific to the VNF and the LCM operation.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>vnfLcmOpOcc	Link	1	Related VNF lifecycle management operation occurrence.
>vnfInstance	Link	1	Related VNF instance.
NOTE 1: This interface allows to signal the use of multiple VIMs per VNF. However, due to the partial support of this feature in the present release, it is recommended in the present document that the number of entries in the "vims" attribute in the Grant is not greater than 1. NOTE 2: At least one of (computeReservationId, networkReservationId, storageReservationId) shall be present when policy is GRANT_RESERVE_SINGLE and an applicable reservation exists. None of these shall be present otherwise. NOTE 3: Modification of VIM assets during the lifetime of a VNF instance is not necessary, since it is expected that all applicable assets have been on boarded into the VIM before the VNF is instantiated. NOTE 4: The indication of externally-managed internal VLs is needed in case networks have been pre-configured for use with certain VNFs, for instance to ensure that these networks have certain properties such as security or acceleration features, or to address particular network topologies. The present document assumes that externally-managed internal VLs are managed by the NFVO and created towards the VIM. NOTE 5: External and/or externally-managed internal VLs can be passed in VNF lifecycle management operation requests such as InstantiateVnf or ChangeVnfFlavor, and/or in the grant response. The NFVO may choose to override in the grant response external and/or externally-managed VL instances that have been passed previously in the associated VNF lifecycle management request, if the lifecycle management request has originated from the NFVO itself.			

9.5.3 Referenced structured data types

9.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but are not resource representations.

9.5.3.2 Type: ResourceDefinition

This type provides information of an existing or proposed resource used by the VNF. It shall comply with the provisions defined in table 9.5.3.2-1.

Table 9.5.3.2-1: Definition of the ResourceDefinition data type

Attribute name	Data type	Cardinality	Description
id	IdentifierLocal	1	Identifier of this "ResourceDefinition" structure, unique at least within the scope of the "GrantRequest" structure.
type	Enum (inlined)	1	Type of the resource definition referenced. Permitted values: - COMPUTE - VL - STORAGE - LINKPORT
vduId	IdentifierInVnfd	0..1	Reference to the related VDU in the VNFD applicable to this resource. Shall only be present if a VDU is applicable to this resource.
resourceTemplateId	IdentifierInVnfd	0..1	Reference to a resource template (VnfVirtualLinkDesc, VirtualComputeDesc, VnfExtCpd, VirtualStorageDesc) in the VNFD. Shall be present for the planned creation of new resources, including temporary resources, and for the modification of existing resources. Shall be absent otherwise.
resource	ResourceHandle	0..1	Resource information for an existing resource. Shall be present for resources that are planned to be deleted or modified. Shall be absent otherwise.

9.5.3.3 Type: GrantInfo

This type contains information about a Compute, storage or network resource whose addition/update/deletion was granted. It shall comply with the provisions defined in table 9.5.3.3-1.

Table 9.5.3.3-1: Definition of the GrantInfo data type

Attribute name	Data type	Cardinality	Description
resourceDefinitionId	IdentifierLocal	1	Identifier of the related "ResourceDefinition" structure from the related "GrantRequest" structure.
reservationId	Identifier	0..1	The reservation identifier applicable to the VNFC/VirtualLink/VirtualStorage. It shall be present for new resources when policy is GRANT_RESERVE_MULTI and an applicable reservation exists; shall not be present otherwise.
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to be used to manage this resource. Shall be present for new resources, and shall be absent for resources that have already been allocated. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the "vimConnectionInfo" attribute of the "VnfInstance" structure. This attribute shall only be supported when VNF-related Resource Management in direct mode is applicable.

Attribute name	Data type	Cardinality	Description
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of the virtualised resource. Shall be present for new resources, and shall be absent for resources that have already been allocated. This attribute shall only be supported when VNF-related Resource Management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
zoneId	IdentifierLocal	0..1	Reference to the identifier of the "ZoneInfo" structure in the "Grant" structure defining the resource zone into which this resource is to be placed. Shall be present for new resources if the zones concept is applicable to them (typically, Compute resources), and shall be absent for resources that have already been allocated.
resourceGroupId	IdentifierInVim	0..1	Identifier of the "infrastructure resource group", logical grouping of virtual resources assigned to a tenant within an Infrastructure Domain, to be provided when allocating the resource. If the VIM connection referenced by "vimConnectionId" applies to multiple infrastructure resource groups, this attribute shall be present for new resources. If the VIM connection referenced by "vimConnectionId" applies to a single infrastructure resource group, this attribute may be present for new resources. This attribute shall be absent for resources that have already been allocated.

9.5.3.4 Type: ZoneInfo

This type provides information regarding a resource zone. It shall comply with the provisions defined in table 9.5.3.4-1.

Table 9.5.3.4-1: Definition of the ZoneInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierLocal	1	The identifier of this ZoneInfo instance, for the purpose of referencing it from other structures in the "Grant" structure.
zoneId	Identifier	1	The identifier of the resource zone, as managed by the resource management layer (typically, the VIM).
vimConnectionId	Identifier	0..1	Identifier of the connection to the VIM that manages the resource zone. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the " vimConnectionInfo" attribute of the "VnfInstance" structure. This attribute shall only be supported and present when VNF-related Resource Management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management the resource zone. This attribute shall only be supported and present when VNF-related Resource Management in indirect mode is applicable. The identification scheme is outside the scope of the present document.

9.5.3.5 Type: ZoneGroupInfo

This type provides information regarding a resource zone group. A resource zone group is a group of one or more related resource zones which can be used in resource placement constraints. To fulfil such constraint, the NFVO may decide to place a resource into any zone that belongs to a particular group.

NOTE: A resource zone group can be used to support overflow from one resource zone into another, in case a particular deployment supports only non-elastic resource zones.

The ZoneGroupInfo type shall comply with the provisions defined in table 9.5.3.5-1.

Table 9.5.3.5-1: Definition of the ZoneGroupInfo data type

Attribute name	Data type	Cardinality	Description
zoneId	IdentifierLocal	1..N	References of identifiers of "ZoneInfo" structures, each of which provides information about a resource zone that belongs to this group.

9.5.3.6 Type: PlacementConstraint

This type provides information regarding a resource placement constraint. A set of such constraints may be sent by the VNFM to the NFVO to influence the resource placement decisions made by the NFVO as part of the granting process. A placement constraint defines a condition to the placement of new resources, considering other new resources as well as existing resources.

EXAMPLE: The following rules influence the placement of a set of resources such that they are placed in the same Network Function Virtualisation Infrastructure Point of Presence (NFVI-PoP) but in different resource zones:

```
{type="affinity"; scope="NFVI_POP"; {resource1,resource2}}
{type="anti-affinity"; scope="ZONE"; {resource1,resource2}}
```

The PlacementConstraint type shall comply with the provisions defined in table 9.5.3.6-1.

Table 9.5.3.6-1: Definition of the PlacementConstraint data type

Attribute name	Data type	Cardinality	Description
affinityOrAntiAffinity	Enum (inlined)	1	The type of the constraint. Permitted values: - AFFINITY - ANTI_AFFINITY
scope	Enum (inlined)	1	The scope of the placement constraint indicating the category of the "place" where the constraint applies. Permitted values: - NFVI_POP - ZONE - ZONE_GROUP - NFVI_NODE
resource	ConstraintResourceRef	2..N	References to resources in the constraint rule.

9.5.3.7 Type: VimConstraint

This type provides information regarding a VIM selection constraint. A set of such constraints may be sent by the VNFM to the NFVO to influence the VIM selection decisions made by the NFVO as part of the granting process.

The VimConstraint type shall comply with the provisions defined in table 9.5.3.7-1.

Table 9.5.3.7-1: Definition of the VimConstraint data type

Attribute name	Data type	Cardinality	Description
sameResourceGroup	Boolean	0..1	If present and set to true, this signals that the constraint applies not only to the same VIM connection, but also to the same infrastructure resource group.
resource	ConstraintResourceRef	2..N	References to resources in the constraint rule. The NFVO shall ensure that all resources in this list are managed through the same VIM connection. If "sameResourceGroup" is set to true, the NFVO shall further ensure that all resources in this list are part of the same infrastructure resource group in that VIM connection.

9.5.3.8 Type: ConstraintResourceRef

This type references a resource either by its VIM-level identifier for existing resources, or by the identifier of a "ResourceDefinition" structure in the "GrantRequest" structure for new resources.

The ConstraintResourceRef type shall comply with the provisions defined in table 9.5.3.8-1.

Table 9.5.3.8-1: Definition of the ConstraintResourceRef data type

Attribute name	Data type	Cardinality	Description
idType	Enum (inlined)	1	The type of the identifier. Permitted values: - RES_MGMT: Resource-management-level identifier; this identifier is managed by the VIM in the direct mode of VNF-related resource management, and is managed by the NFVO in the indirect mode) - GRANT: Reference to the identifier of a "ResourceDefinition" structure in the "GrantRequest" structure.
resourceId	IdentifierInVim	1	An actual resource-management-level identifier (idType=RES_MGMT), or an identifier that references a "ResourceDefinition" structure in the related "GrantRequest" structure (idType=GRANT).
vimConnectionId	Identifier	0..1	Identifier of the VIM connection for managing the resource. It shall only be present when idType = RES_MGMT. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the "vimConnectionInfo" attribute of the "VnfInstance" structure. This attribute shall only be supported when VNF-related resource management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifier of the resource provider. It shall only be present when idType = RES_MGMT. This attribute shall only be supported when VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.

9.5.3.9 Type: VimComputeResourceFlavour

If the VIM requires the use of virtual compute resource flavours during compute resource instantiation, it is assumed that such flavours are selected or created by the NFVO based on the information in the virtual compute descriptor defined in the VNFD.

This type defines the mapping between a virtual compute descriptor in the VNFD and the corresponding compute resource flavour managed by the NFVO in the VIM. It shall comply with the provisions defined in table 9.5.3.9-1.

Table 9.5.3.9-1: Definition of the VimComputeResourceFlavour data type

Attribute name	Data type	Cardinality	Description
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to access the flavour referenced in this structure. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the "vimConnectionInfo" attribute of the "VnfInstance" structure. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of the virtualised resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
vnfdVirtualComputeDescId	IdentifierInVnfd	1	Identifier which references the virtual compute descriptor in the VNFD that maps to this flavour.
vimFlavourId	IdentifierInVim	1	Identifier of the compute resource flavour in the resource management layer (i.e. VIM).

9.5.3.10 Type: VimSoftwareImage

This type contains a mapping between a software image definition the VNFD and the corresponding software image managed by the NFVO in the VIM which is needed during compute resource instantiation. It shall comply with the provisions defined in table 9.5.3.10-1.

Table 9.5.3.10-1: Definition of the VimSoftwareImage data type

Attribute name	Data type	Cardinality	Description
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to access the software image referenced in this structure. The applicable "VimConnectionInfo" structure, which is referenced by vimConnectionId, can be obtained from the "vimConnectionInfo" attribute of the "VnfInstance" structure. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of the virtualised resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
vnfdSoftwareImageId	IdentifierInVnfd	1	Identifier which references the software image descriptor in the VNFD.

Attribute name	Data type	Cardinality	Description
vimSoftwareImageId	IdentifierInVim	1	Identifier of the software image in the resource management layer (i.e. VIM).

9.5.4 Referenced simple data types and enumerations

9.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

9.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

9.5.4.3 Enumeration: GrantedLcmOperationType

The enumeration GrantedLcmOperationType defines the permitted values to represent VNF lifecycle operation types in grant requests. It shall comply with the provisions defined in table 9.5.4.3-1.

Table 9.5.4.3-1: Enumeration GrantedLcmOperationType

Enumeration value	Description
INSTANTIATE	Represents the "Instantiate VNF" LCM operation.
SCALE	Represents the "Scale VNF" LCM operation.
SCALE_TO_LEVEL	Represents the "Scale VNF to Level" LCM operation.
CHANGE_FLAVOUR	Represents the "Change VNF Flavour" LCM operation.
TERMINATE	Represents the "Terminate VNF" LCM operation.
HEAL	Represents the "Heal VNF" LCM operation.
OPERATE	Represents the "Operate VNF" LCM operation.
CHANGE_EXT_CONN	Represents the "Change external VNF connectivity" LCM operation.

10 VNF Package Management interface

10.1 Description

This interface allows the VNFM to obtain VNF package information from the NFVO, and to retrieve API version information.

The operations provided through this interface are:

- Query VNF Package, including obtaining the VNFD
- Fetch VNF Package
- Fetch VNF Package Artifacts
- Subscribe
- Query Subscription Info
- Terminate Subscription
- Notify

10.1a API version

For the VNF package management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

10.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnfpkgm" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 10.2-1 shows the overall resource URI structure defined for the VNF Package Management interface.

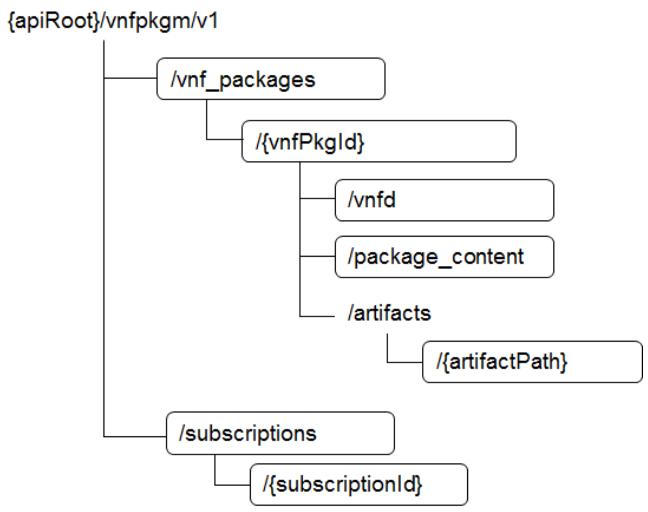

Figure 10.2-1: Resource URI structure of the VNF Package Management Interface

Table 10.2-1 lists the individual resources defined, and the applicable HTTP methods.

The NFVO shall support responding to requests for all HTTP methods on the resources in table 10.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 10.2-1: Resources and methods overview of the VNF Package Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
VNF packages	/vnf_packages	GET	M	Query VNF packages information
Individual VNF package	/vnf_packages/{vnfPkgId}	GET	M	Read information about an individual VNF package
VNFD of an individual VNF package	/vnf_packages/{vPkgId}/vnfd	GET	M	Read VNFD of an on-boarded VNF package
VNF package content	/vnf_packages/{vPkgId}/package_content	GET	M	Fetch an on-boarded VNF package
Individual VNF package artifact	/vnf_packages/{vPkgId}/artifacts/{artifactPath}	GET	M	Fetch individual VNF package artifact
Subscriptions	/subscriptions	POST	M	Subscribe to notifications related to on-boarding and/or changes of VNF package
		GET	M	Query multiple subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription resource
		DELETE	M	Terminate a subscription
Notification endpoint	(client-provided)	POST	See note	Notify about VNF package on-boarding or change. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the VNFM. If the VNFM supports invoking the POST method on the "Subscription" resource towards the NFVO, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

10.3 Sequence diagrams (informative)

10.3.1 Flow of querying/reading VNF package information

This clause describes a sequence for querying information about one or multiple VNF packages.

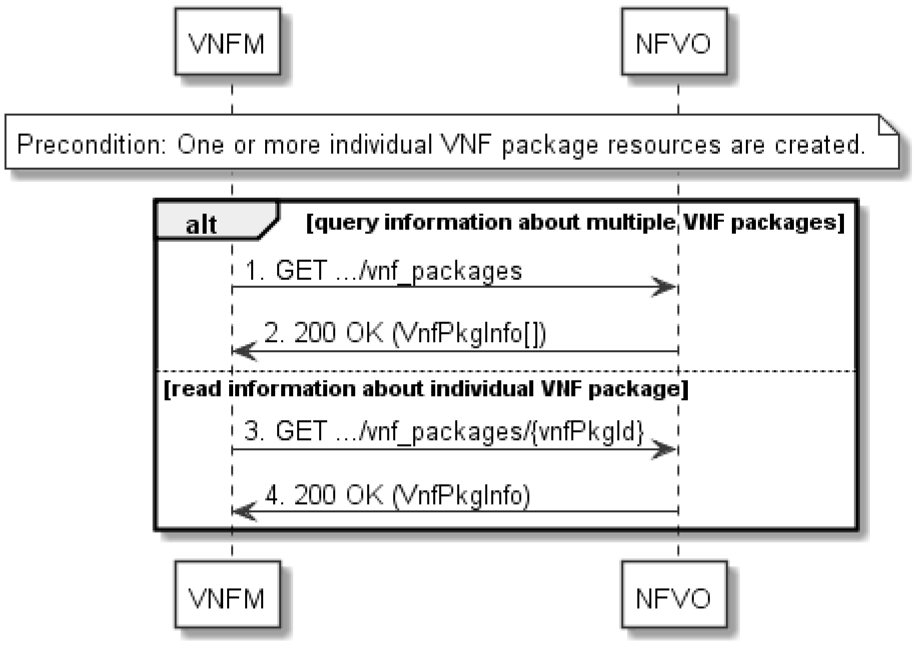
VNFM
NFVO
Precondition: One or more individual VNF package resources are created.
alt
[query information about multiple VNF packages]
1. GET .../vnf_packages
2. 200 OK (VnfPkgInfo[])
[read information about individual VNF package]
3. GET .../vnf_packages/{vnfPkgId}
4. 200 OK (VnfPkgInfo)
VNFM
NFVO

Figure 10.3.1-1: Flow of querying/reading VNF package information

Precondition: One or more individual VNF package resources are created.

VNF package information query, as illustrated in figure 10.3.1-1, consists of the following steps:

1) If the VNFM intends to query information about multiple VNF packages, it sends a GET request to the "VNF packages" resource.
2) The NFVO returns a "200 OK" response, and includes in the payload body zero or more data structures of type "VnfPkgInfo".
3) If the VNFM intends to read information about a particular VNF package, the VNFM sends a GET request to the "Individual VNF package" resource, addressed by the appropriate VNF package identifier in its resource URI.
4) The NFVO returns a "200 OK" response, and includes in the payload body a data structure of type "VnfPkgInfo".

Postcondition: Upon successful completion, the VNFM gets the information of the VNF packages or the VNF package.

Error handling: In case of failure, appropriate error information is provided in the response.

10.3.2 Flow of reading the VNFD of an on-boarded VNF package

This clause describes the procedure for reading the VNFD of an on-boarded VNF package.

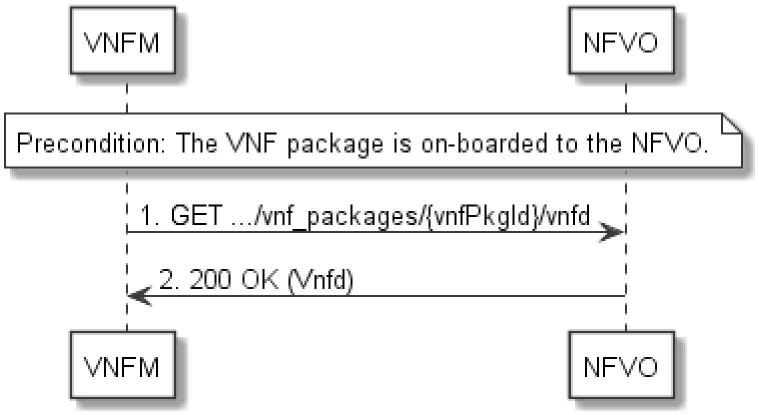
VNFM
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
1. GET .../vnf_packages/{vnfPkgId}/vnfd
2. 200 OK (Vnfd)
VNFM
NFVO

Figure 10.3.2-1: Flow of reading VNFD

Precondition: The VNF package is on-boarded to the NFVO.

The procedure consists of the following steps as illustrated in figure 10.3.2-1:

1) The VNFM sends a GET request to the "VNFD in an individual VNF package" resource.
2) The NFVO returns a "200 OK" response, and includes a copy of the VNFD from the VNF package in the payload body.

10.3.3 Flow of fetching an on-boarded VNF package

This clause describes a sequence for fetching the content of an on-boarded VNF package.

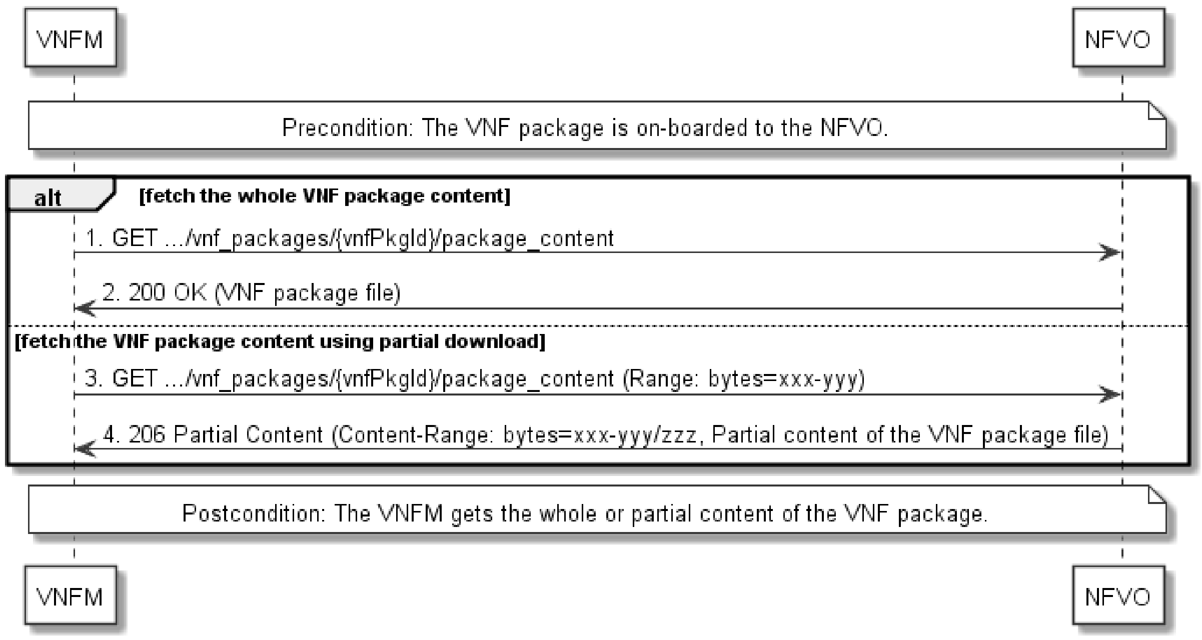
VNFM
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
alt [fetch the whole VNF package content]
1. GET .../vnf_packages/{vnfPkgId}/package_content
2. 200 OK (VNF package file)
[fetch the VNF package content using partial download]
3. GET .../vnf_packages/{vnfPkgId}/package_content (Range: bytes=xxx-yyy)
4. 206 Partial Content (Content-Range: bytes=xxx-yyy/zzz, Partial content of the VNF package file)
Postcondition: The VNFM gets the whole or partial content of the VNF package.
VNFM
NFVO

Figure 10.3.3-1: Flow of fetching an on-boarded VNF package

Precondition: The VNF package is on-boarded to the NFVO.

Fetching an on-boarded VNF package, as illustrated in figure 10.3.3-1, consists of the following steps:

1) If fetching the whole VNF package content, the VNFM sends a GET request to the " VNF package content" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the VNF package file in the payload body.

3) If fetching the VNF package content using partial download, the VNFM sends a GET request to the "VNF package content" resource, and includes a "Range" HTTP header indicating the partition of the VNF package content needs to be transferred.

4) The NFVO returns a "206 Partial Content" response with a payload body containing the partial content of the VNF package, and a "Content-Range" HTTP header indicating the byte range enclosed in the payload and the complete length of the VNF package content.

Postcondition: Upon successful completion, the VNFM gets the whole or partial content of the VNF package.

Error handling: In case of failure, appropriate error information is provided in the response.

10.3.4 Flow of fetching a VNF package artifact

This clause describes a sequence for fetching an individual artifact contained in an on-boarded VNF package.

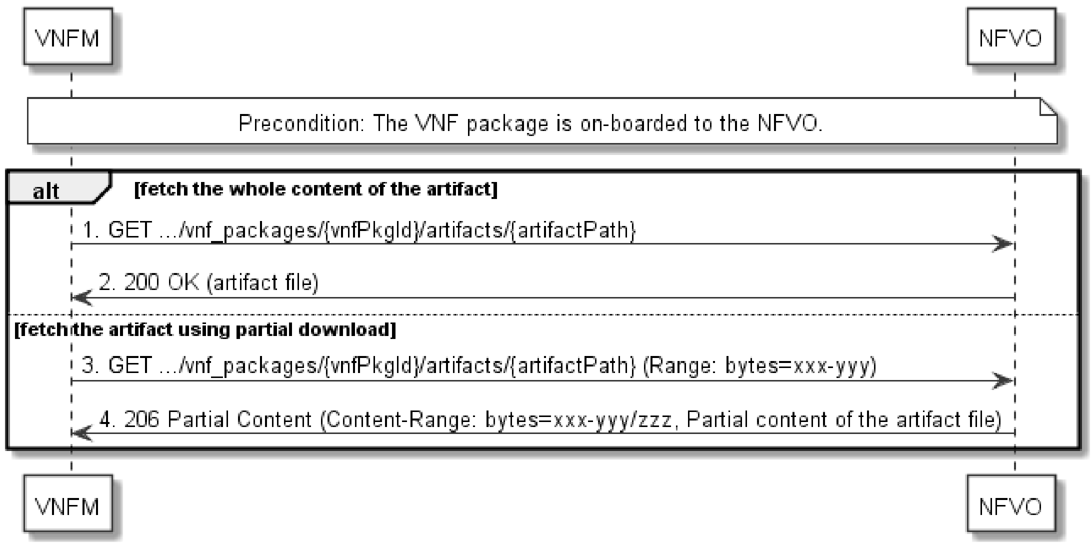
VNFM
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
alt
[fetch the whole content of the artifact]
1. GET .../vnf_packages/{vnfPkgId}/artifacts/{artifactPath}
2. 200 OK (artifact file)
[fetch the artifact using partial download]
3. GET .../vnf_packages/{vnfPkgId}/artifacts/{artifactPath} (Range: bytes=xxx-yyy)
4. 206 Partial Content (Content-Range: bytes=xxx-yyy/zzz, Partial content of the artifact file)
VNFM
NFVO

Figure 10.3.4-1: Flow of fetching a VNF package artifact

Precondition: The VNF package is on-boarded to the NFVO.

Fetching an individual artifact contained in an on-boarded VNF package, as illustrated in figure 10.3.4-1, consists of the following steps:

1) If fetching the whole content of the artifact, the VNFM sends a GET request to the "Individual VNF package artifact" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the applicable artifact file from the VNF package in the payload body.

3) If fetching the artifact using partial download, the VNFM sends a GET request to the "Individual VNF package artifact" resource, and includes a "Range" HTTP header indicating the partition of the artifact needs to be transferred.

4) The NFVO returns a "206 Partial Content" response with a payload body containing the partial content of the artifact file, and a "Content-Range" HTTP header indicating the byte range enclosed in the payload and the complete length of the artifact file.

Error handling: In case of failure, appropriate error information is provided in the response.

10.3.5 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF package management.

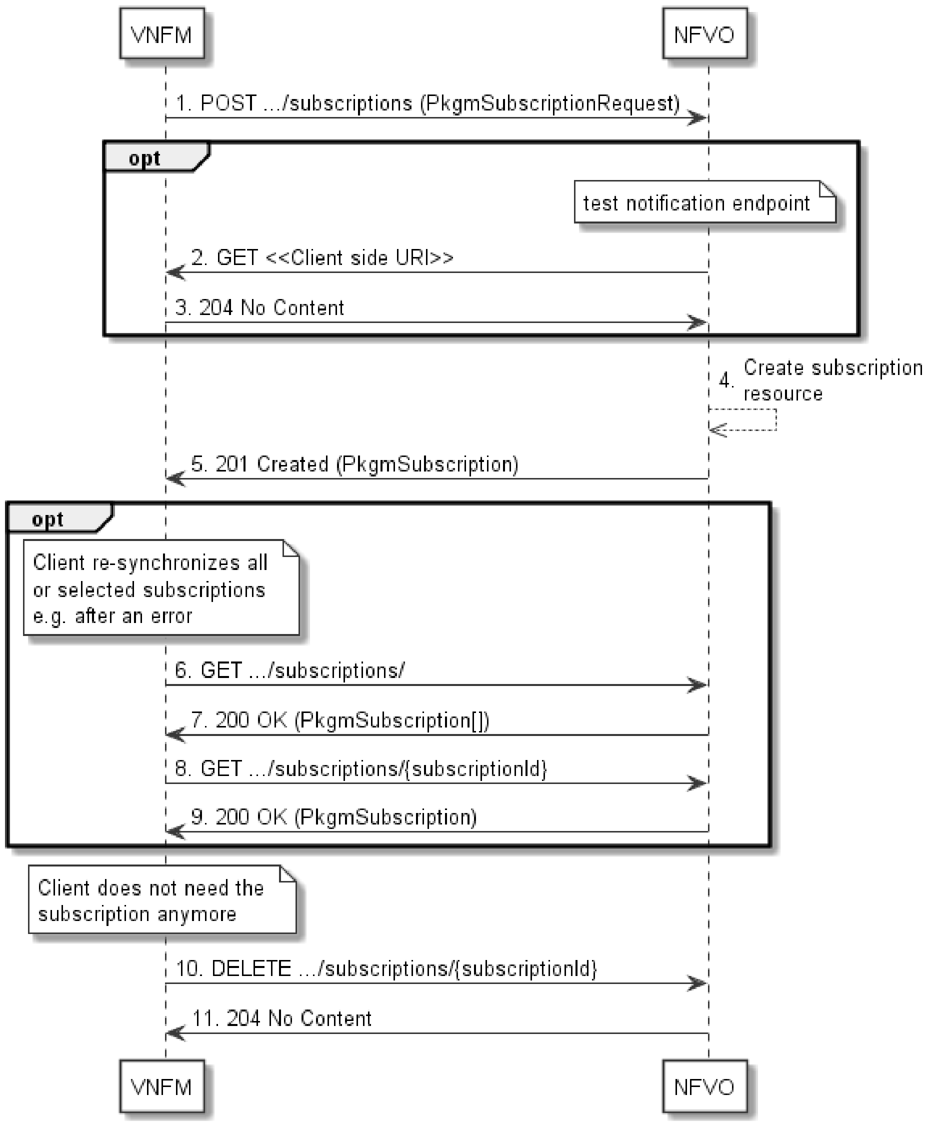
VNFM
NFVO
1. POST .../subscriptions (PkgmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (PkgmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (PkgmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (PkgmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
VNFM
NFVO

Figure 10.3.5-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 10.3.5-1:

1) The VNFM sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "PkgmSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the VNFM as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

3) In that case, the VNFM returns a "204 No Content" response to indicate success.

4) The NFVO creates a new subscription to notifications related to VNF package on-boarding or changes, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "PkgmSubscription" representing the subscription resource just created by the NFVO, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) If desired, e.g. to recover from an error situation, the VNFM may obtain information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the VNFM.

8) If desired, e.g. to recover from an error situation, the VNFM may obtain information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the NFVO returns a "200 OK" response that contains a representation of that individual subscription.

10) If the VNFM does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11) The NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

10.3.6 Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF package management.

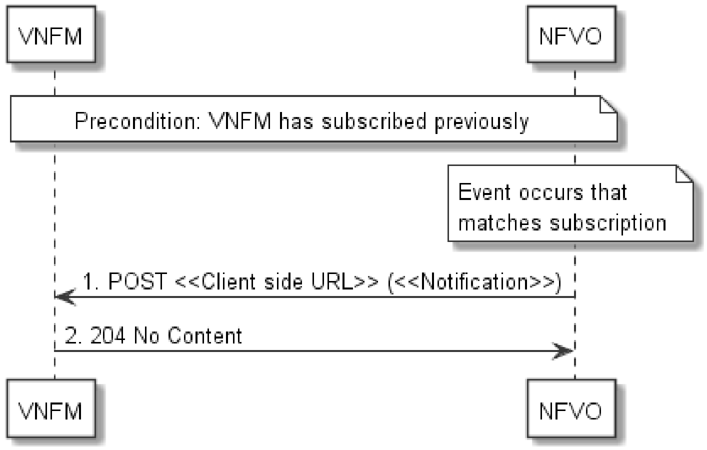
VNFM
NFVO
Precondition: VNFM has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URL>> (<<Notification>>)
2. 204 No Content
VNFM
NFVO

Figure 10.3.6-1: Flow of sending notifications

Precondition: The VNFM has subscribed previously for notifications related to VNF package management.

The procedure consists of the following steps as illustrated in figure 10.3.6-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the VNFM has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API (see clauses 10.5.2.5 and 10.5.2.6).

2) The VNFM acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the NFVO does not receive the "204 No Content" response from the VNFM, it can retry sending the notification.

10.4 Resources

10.4.1 Introduction

This clause defines all the resources and methods provided by the VNF package management interface.

10.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF package management interface.

10.4.2 Resource: VNF packages

10.4.2.1 Description

This resource represents VNF packages. The client can use this resource to query information of the VNF packages.

10.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages

This resource shall support the resource URI variables defined in table 10.4.2.2-1.

Table 10.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

10.4.2.3 Resource methods

10.4.2.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.2.3.2 GET

The GET method queries the information of the VNF packages matching the filter.

This method shall follow the provisions specified in the tables 10.4.2.3.2-1 and 10.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The VNFM may supply this parameter. All attribute names that appear in the VnfPkgInfo and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO should support this parameter.

Name	Cardinality	Description
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The NFVO shall support this parameter. The following attributes shall be excluded from the VnfPkgInfo structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - softwareImages - additionalArtifacts - userDefinedData.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 10.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfPkgInfo	0..N	200 OK	Information about zero or more VNF packages was queried successfully. The response body shall contain in an array the VNF package info representations that match the attribute filter, i.e. zero or more VNF package info representations as defined in clause 10.5.2.2. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.3 Resource: Individual VNF package

10.4.3.1 Description

This resource represents an individual VNF package. The client can use this resource to read information of the VNF package.

10.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}

This resource shall support the resource URI variables defined in table 10.4.3.2-1.

Table 10.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

10.4.3.3 Resource methods

10.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.3.3.2 GET

The GET method reads the information of an individual VNF package.

This method shall follow the provisions specified in the tables 10.4.3.3.2-1 and 10.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.3.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfPkgInfo	1	200 OK	Information of the VNF package was read successfully. The response body shall contain the VNF package info representation defined in clause 10.5.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.4 Resource: VNFD in an individual VNF package

10.4.4.1 Description

This resource represents the VNFD contained in an on-boarded VNF package. The client can use this resource to obtain the content of the VNFD.

10.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/vnfd

This resource shall support the resource URI variables defined in table 10.4.4.2-1.

Table 10.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

10.4.4.3 Resource methods

10.4.4.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.4.3.2 GET

The GET method reads the content of the VNFD within a VNF package.

The VNFD can be implemented as a single file or as a collection of multiple files. If the VNFD is implemented in the form of multiple files, a ZIP file embedding these files shall be returned. If the VNFD is implemented as a single file, either that file or a ZIP file embedding that file shall be returned.

The selection of the format is controlled by the "Accept" HTTP header passed in the GET request:

- If the "Accept" header contains only "text/plain" and the VNFD is implemented as a single file, the file shall be returned; otherwise, an error message shall be returned.
- If the "Accept" header contains only "application/zip", the single file or the multiple files that make up the VNFD shall be returned embedded in a ZIP file.
- If the "Accept" header contains both "text/plain" and "application/zip", it is up to the NFVO to choose the format to return for a single-file VNFD; for a multi-file VNFD, a ZIP file shall be returned.

The default format of the ZIP file shall be the one specified in ETSI GS NFV-SOL 004 [2] where only the YAML files representing the VNFD, and information needed to navigate the ZIP file and to identify the file that is the entry point for parsing the VNFD (such as TOSCA-meta or manifest files or naming conventions) are included.

This method shall follow the provisions specified in the tables 10.4.4.3.2-1 and 10.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.4.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a		The request shall contain the appropriate entries in the "Accept" HTTP header as defined above.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, the content of the VNFD is returned. The payload body shall contain a copy of the file representing the VNFD or a ZIP file that contains the file or multiple files representing the VNFD, as specified above. The "Content-Type" HTTP header shall be set according to the format of the returned file, i.e. to "text/plain" for a YAML file or to "application/zip" for a ZIP file.
	ProblemDetails	0..1	406 Not Acceptable	If the "Accept" header does not contain at least one name of a content type for which the NFVO can provide a representation of the VNFD, the NFVO shall respond with this response code. The "ProblemDetails" structure may be included with the "detail" attribute providing more information about the error.

	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that "onboardingState" of the VNF package has a value different from "ONBOARDED". The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.5 Resource: VNF package content

10.4.5.1 Description

This resource represents a VNF package identified by the VNF package identifier allocated by the NFVO. The client can use this resource to fetch the content of the VNF package.

10.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content

This resource shall support the resource URI variables defined in table 10.4.5.2-1.

Table 10.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

10.4.5.3 Resource methods

10.4.5.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.5.3.2 GET

The GET method fetches the content of a VNF package identified by the VNF package identifier allocated by the NFVO.

This method shall follow the provisions specified in the tables 10.4.5.3.2-1 and 10.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.5.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			The request may contain a "Range" HTTP header to obtain single range of bytes from the VNF package file. This can be used to continue an aborted transmission. If the NFVO does not support range requests, it should return the whole file with a 200 OK response instead.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, a copy of the VNF package file is returned. The response body shall include a copy of the VNF package file. The "Content-Type HTTP" header shall be set according to the type of the file, i.e. to "application/zip" for a VNF Package as defined in ETSI GS NFV-SOL 004 [2].
	n/a	1	206 Partial Content	On success, if the NFVO supports range requests, a single consecutive byte range from the content of the VNF package file is returned. The response body shall contain the requested part of the VNF package file. The "Content-Range" HTTP header shall be provided according to IETF RFC 7233 [15]. The "Content-Type" HTTP header shall be set as defined above for the "200 OK" response.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that "onboardingState" of the VNF package has a value different from "ONBOARDED". The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	416 Range Not Satisfiable	The byte range passed in the "Range" header did not match any available byte range in the VNF package file (e.g. "access after end of file"). The response body may contain a ProblemDetails structure.

	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.6 Resource: Individual VNF package artifact

10.4.6.1 Description

This resource represents an individual artifact contained in a VNF package. The client can use this resource to fetch the content of the artifact.

10.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/artifacts/{artifactPath}

This resource shall support the resource URI variables defined in table 10.4.6.2-1.

Table 10.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note 1.
artifactPath	Sequence of one or more path segments representing the path of the artifact within the VNF package. EXAMPLE: foo/bar/run.sh See note 2.
NOTE 1: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification. NOTE 2: This identifier can be retrieved from the "artifactPath" attribute of the applicable "additionalArtifacts" entry in the body of the response to a GET request querying the "Individual VNF package" or the " VNF packages" resource.	

10.4.6.3 Resource methods

10.4.6.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.6.3.2 GET

The GET method fetches the content of an artifact within a VNF package.

This method shall follow the provisions specified in the tables 10.4.6.3.2-1 and 10.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.6.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a		The request may contain a "Range" HTTP header to obtain single range of bytes from an artifact file. This can be used to continue an aborted transmission. If the NFVO does not support range requests, it should return the whole file with a 200 OK response instead.	
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a	1	200 OK	On success, the content of the artifact is returned. The payload body shall contain a copy of the artifact file from the VNF package, as defined by ETSI GS NFV-SOL 004 [2]. The "Content-Type" HTTP header shall be set according to the content type of the artifact file. If the content type cannot be determined, the header shall be set to the value "application/octet-stream".
	n/a	1	206 Partial Content	A single consecutive byte range from the content of the artifact file, if the NFVO supports range requests. The response body shall contain the requested part of the artifact file from the VNF package, as defined by ETSI GS NFV-SOL 004 [2]. The "Content-Type" HTTP header shall be set according to the content type of the artifact file. If the content type cannot be determined, the header shall be set to the value "application/octet-stream". The "Content-Range" HTTP header shall be provided according to IETF RFC 7233 [15].
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that "onboardingState" of the VNF package has a value different from "ONBOARDED". The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	416 Range Not Satisfiable	The byte range passed in the "Range" header did not match any available byte range in the artifact file (e.g. "access after end of file"). The response body may contain a ProblemDetails structure.

	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.7 Resource: Subscriptions

10.4.7.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to the VNF package management, and to query its subscriptions.

10.4.7.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/subscriptions

This resource shall support the resource URI variables defined in table 10.4.7.2-1.

Table 10.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

10.4.7.3 Resource methods

10.4.7.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 10.4.7.3.1-1 and 10.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the VNFM, and might make sense only in very rare use cases. Consequently, the NFVO may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 10.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	PkgmSubscriptionRequest	1	Details of the subscription to be created.	
Response body	Data type	Cardinality	Response Codes	Description
	PkgmSubscription	1	201 Created	Representation of the created subscription resource. The HTTP response shall include a "Location" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the NFVO is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.7.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 10.4.7.3.2-1 and 10.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The VNFM may supply this parameter. All attribute names that appear in the PkgmSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 10.4.7.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PkgmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of VNF package management subscriptions as defined in clause 10.5.2.4. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.8 Resource: Individual subscription

10.4.8.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to the VNF package management.

10.4.8.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 10.4.8.2-1.

Table 10.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

10.4.8.3 Resource methods

10.4.8.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.8.3.2 GET

The GET method reads an individual subscription.

This method shall follow the provisions specified in the tables 10.4.8.3.2-1 and 10.4.8.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.8.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
	Data type	Cardinality	Response Codes	Description
Response body	PkgmSubscription	1	200 OK	Representation of the subscription resource.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.8.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 10.4.8.3.5-1 and 10.4.8.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.8.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.8.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.9 Resource: Notification endpoint

10.4.9.1 Description

This resource represents a notification endpoint.

The API producer can use this resource to send notifications related to VNF package management events to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

10.4.9.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 10.4.9.2-1.

Table 10.4.9.2-1: Resource URI variables for this resource

Name	Definition
n/a	

10.4.9.3 Resource methods

10.4.9.3.1 POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 10.4.9.3.1-1 and 10.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one of the alternatives defined in table 10.4.9.3.1-2.

Table 10.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	VnfPackageOnboardingNotification	1		A notification about on-boarding of a VNF package.
	VnfPackageChangeNotification	1		A notification about changes of status in a VNF package.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.9.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 10.4.9.3.2-1 and 10.4.9.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 10.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 10.4.9.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

10.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.4.9.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

10.5 Data model

10.5.1 Introduction

This clause defines the request and response data structures of the VNF package management interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

10.5.2 Resource and notification data types

10.5.2.1 Introduction

This clause defines data structures to be used in resource representations and notifications.

10.5.2.2 Type: VnfPkgInfo

This type represents the information of a VNF package. It shall comply with the provisions defined in table 10.5.2.2-1.

Table 10.5.2.2-1: Definition of the VnfPkgInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO.
vnfdId	Identifier	0..1	This identifier, which is managed by the VNF provider, identifies the VNF package and the VNFD in a globally unique way. It is copied from the VNFD of the on-boarded VNF package. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfProvider	String	0..1	Provider of the VNF package and the VNFD. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfProductName	String	0..1	Name to identify the VNF product. Invariant for the VNF product lifetime. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfSoftwareVersion	Version	0..1	Software version of the VNF. This is changed when there is any change to the software included in the VNF package. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfdVersion	Version	0..1	The version of the VNFD. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
checksum	Checksum	0..1	Checksum of the on-boarded VNF package. It shall be present after the VNF package content has been on-boarded and absent otherwise.
softwareImages	VnfPackageSoftwareImageInfo	0..N	Information about VNF package artifacts that are software images.

Attribute name	Data type	Cardinality	Description
			This attribute shall not be present before the VNF package content is on-boarded. Otherwise, this attribute shall be present unless it has been requested to be excluded per attribute selector.
additionalArtifacts	VnfPackageArtifactInfo	0..N	Information about VNF package artifacts contained in the VNF package that are not software images. This attribute shall not be present before the VNF package content is on-boarded. Otherwise, this attribute shall be present if the VNF package contains additional artifacts.
onboardingState	PackageOnboardingStateType	1	On-boarding state of the VNF package.
operationalState	PackageOperationalStateType	1	Operational state of the VNF package. See note 1.
usageState	PackageUsageStateType	1	Usage state of the VNF package. See note 2.
userDefinedData	KeyValuePairs	0..1	User defined data for the VNF package.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>vnfd	Link	0..1	Link to the VNFD resource. This link shall be present after the VNF package content is on-boarded.
>packageContent	Link	1	Link to the " VNF package content" resource.
NOTE 1: If the value of the onboardingState attribute is not equal to "ONBOARDED", the value of the operationalState attribute shall be equal to "DISABLED". NOTE 2: If the value of the onboardingState attribute is not equal to "ONBOARDED", the value of the usageState attribute shall be equal to "NOT_IN_USE".			

10.5.2.3 Type: PkgmSubscriptionRequest

This type represents a subscription request related to VNF package management notifications about VNF package on-boarding or changes. It shall comply with the provisions defined in table 10.5.2.3-1.

Table 10.5.2.3-1: Definition of the PkgmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	PkgmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

10.5.2.4 Type: PkgmSubscription

This type represents a subscription related to notifications about VNF package management. It shall comply with the provisions defined in table 10.5.2.4-1.

Table 10.5.2.4-1: Definition of the PkgmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	PkgmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.

10.5.2.5 Type: VnfPackageOnboardingNotification

This type represents a VNF package management notification, which informs the receiver that the onboarding process of a VNF package is complete and the package is ready for use. It shall comply with the provisions defined in table 10.5.2.5-1. The support of this notification is mandatory.

The notification shall be triggered by the NFVO when the "onboardingState" attribute of a new VNF package has changed to "ONBOARDED".

Table 10.5.2.5-1: Definition of the VnfPackageOnboardingNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfPackageOnboardingNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfPkgId	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO. Its value is the same as the value of the "id" attribute of the related "Individual VNF package" resource.
vnfdId	Identifier	1	This identifier, which is managed by the VNF provider, identifies the VNF package and the VNFD in a globally unique way. It is copied from the VNFD of the on-boarded VNF package.
_links	PkgmLinks	1	Links to resources related to this notification.

10.5.2.6 Type: VnfPackageChangeNotification

This type represents a VNF package management notification, which informs the receiver of a change of the status in an on-boarded VNF package. Only changes in the "operationalState" attribute of an on-boarded VNF package and the deletion VNF package will be reported. Changes in the "usageState" and "onboardingState" attributes are not reported. The notification shall comply with the provisions defined in table 10.5.2.6-1. The support of this notification is mandatory.

The notification shall be triggered by the NFVO when there is a change in the status of an onboarded VNF package, as follows:

- The "operationalState" attribute of a VNF package has changed, and the "onboardingState" attribute of the package has the value "ONBOARDED" (i.e. the package has been onboarded previously).
- The on-boarded VNF package has been deleted.

Table 10.5.2.6-1: Definition of the VnfPackageChangeNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfPackageChangeNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfPkgId	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO. Its value is the same as the value of the "id" attribute of the related "Individual VNF package" resource.
vnfdId	Identifier	1	Identifier of the VNFD contained in the VNF package, which also identifies the VNF package. This identifier is allocated by the VNF provider and copied from the VNFD.
changeType	PackageChangeType	1	The type of change of the VNF package.
operationalState	PackageOperationalStateType	0..1	New operational state of the VNF package. Only present when changeType is OP_STATE_CHANGE.
_links	PkgmLinks	1	Links to resources related to this notification.

10.5.3 Referenced structured data types

10.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but are not resource representations.

10.5.3.2 Type: VnfPackageSoftwareImageInfo

This type represents an artifact contained in a VNF package which represents a software image. It shall comply with provisions defined in table 10.5.3.2-1.

Table 10.5.3.2-1: Definition of the VnfPackageSoftwareImageInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of the software image.
name	String	1	Name of the software image.
provider	String	1	Provider of the software image.
version	Version	1	Version of the software image.
checksum	Checksum	1	Checksum of the software image file.
containerFormat	Enum (inlined)	1	Container format indicates whether the software image is in a file format that also contains metadata about the actual software. Permitted values: - AKI: a kernel image format - AMI: a machine image format - ARI: a ramdisk image format - BARE: the image does not have a container or metadata envelope - DOCKER: docker container format - OVA: OVF package in a tarfile - OVF: OVF container format See note 1.

Attribute name	Data type	Cardinality	Description
diskFormat	Enum (inlined)	1	Disk format of a software image is the format of the underlying disk image. Permitted values: • AKI: a kernel image format • AMI: a machine image format • ARI: a ramdisk image format • ISO: an archive format for the data contents of an optical disc, such as CD-ROM • QCOW2: a common disk image format, which can expand dynamically and supports copy on write • RAW: an unstructured disk image format • VDI: a common disk image format • VHD: a common disk image format • VHDX: enhanced version of VHD format • VMDK: a common disk image format See note 2.
createdAt	DateTime	1	Time when this software image was created.
minDisk	UnsignedInt	1	The minimal disk for this software image in bytes.
minRam	UnsignedInt	1	The minimal RAM for this software image in bytes.
size	UnsignedInt	1	Size of this software image in bytes.
userMetadata	KeyValuePairs	0..1	User-defined data.
imagePath	String	1	Path in the VNF package, which identifies the image artifact and also allows to access a copy of the image artifact.
NOTE 1: The list of permitted values was taken from "Container formats" in [i.7]. NOTE 2: The list of permitted values was adapted from "Disk formats" in [i.7].			

10.5.3.3 Type: VnfPackageArtifactInfo

This type represents an artifact other than a software image which is contained in a VNF package. It shall comply with provisions defined in table 10.5.3.3-1.

Table 10.5.3.3-1: Definition of the VnfPackageArtifactInfo data type

Attribute name	Data type	Cardinality	Description
artifactPath	String	1	Path in the VNF package, which identifies the artifact and also allows to access a copy of the artifact.
checksum	Checksum	1	Checksum of the artifact file.
metadata	KeyValuePairs	0..1	The metadata of the artifact that are available in the VNF package, such as Content type, size, creation date, etc.

10.5.3.4 Type: PkgmNotificationsFilter

This type represents a subscription filter related to notifications related to VNF package management. It shall comply with the provisions defined in table 10.5.3.4-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 10.5.3.4-1: Definition of the PkgmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: • VnfPackageOnboardingNotification • VnfPackageChangeNotification See note 1.
vnfProductsFromProviders	Structure (inlined)	0..N	If present, match VNF packages that contain VNF products from certain providers. See note 2.
>vnfProvider	String	1	Name of the VNF provider to match.
>vnfProducts	Structure (inlined)	0..N	If present, match VNF packages that contain VNF products with certain product names, from one particular provider.
>>vnfProductName	String	1	Name of the VNF product to match.
>>versions	Structure (inlined)	0..N	If present, match VNF packages that contain VNF products with certain versions and a certain product name, from one particular provider.
>>>vnfSoftwareVersion	Version	1	VNF software version to match.
>>>vnfdVersions	Version	0..N	If present, match VNF packages that contain VNF products with certain VNFD versions, a certain software version and a certain product name, from one particular provider.
vnfdId	Identifier	0..N	Match VNF packages with a VNFD identifier listed in the attribute. See note 2.
vnfPkgId	Identifier	0..N	Match VNF packages with a package identifier listed in the attribute. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise. See note 2.
operationalState	PackageOperationalStateType	0..N	Match particular operational state of the VNF package. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise.
usageState	PackageUsageStateType	0..N	Match particular usage state of the VNF package. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise.
NOTE 1: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems. NOTE 2: The attributes "vnfProductsFromProviders", "vnfdId" and "vnfPkgId" are alternatives to reference to particular VNF packages in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

10.5.3.5 Type: PkgmLinks

This type represents the links to resources that a VNF package management notification can contain. It shall comply with the provisions defined in table 10.5.3.5-1.

Table 10.5.3.5-1: Definition of the PkgmLinks data type

Attribute name	Data type	Cardinality	Description
vnfPackage	NotificationLink	1	Link to the resource representing the VNF package to which the notified change applies, i.e. the individual VNF package resource that represents the VNF package.
subscription	NotificationLink	1	Link to the related subscription.

10.5.3.6 Type: Checksum

This type represents the checksum of a VNF package or an artifact file. It shall comply with the provisions defined in table 10.5.3.6-1.

Table 10.5.3.6-1: Definition of the Checksum data type

Attribute name	Data type	Cardinality	Description
algorithm	String	1	Name of the algorithm used to generate the checksum, as defined in ETSI GS NFV-SOL 004 [2]. For example, SHA-256, SHA-512.
hash	String	1	The hexadecimal value of the checksum.

10.5.4 Referenced simple data types and enumerations

10.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

10.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

10.5.4.3 Enumeration: PackageOperationalStateType

The enumeration PackageOperationalStateType shall comply with the provisions defined in table 10.5.4.3-1.

Table 10.5.4.3-1: Enumeration PackageOperationalStateType

Enumeration value	Description
ENABLED	The VNF package is enabled, i.e. it can be used for instantiation of new VNF instances.
DISABLED	The VNF package is disabled, i.e. it cannot be used for further VNF instantiation requests (unless and until the VNF package is re-enabled).

10.5.4.4 Enumeration: PackageUsageStateType

The enumeration PackageUsageStateType shall comply with the provisions defined in table 10.5.4.4-1.

Table 10.5.4.4-1: Enumeration PackageUsageStateType

Enumeration value	Description
IN_USE	VNF instances instantiated from this VNF package exist.
NOT_IN_USE	No existing VNF instance is instantiated from this VNF package.

10.5.4.5 Enumeration: PackageChangeType

The enumeration PackageChangeType shall comply with the provisions defined in table 10.5.4.5-1.

Table 10.5.4.5-1: Enumeration PackageChangeType

Enumeration value	Description
OP_STATE_CHANGE	The "operationalState" attribute has been changed.
PKG_DELETE	The VNF package has been deleted.

10.5.4.6 Enumeration: PackageOnboardingStateType

The enumeration PackageOnboardingStateType shall comply with the provisions defined in table 10.5.4.6-1.

Table 10.5.4.6-1: Enumeration PackageOnboardingStateType

Enumeration value	Description
CREATED	The VNF package resource has been created.
UPLOADING	The associated VNF package content is being uploaded.
PROCESSING	The associated VNF package content is being processed, e.g. validation.
ONBOARDED	The associated VNF package content has been successfully on-boarded.

11 Virtualised Resources Quota Available Notification interface

11.1 Description

This interface allows the VNFM to subscribe to notifications on the availability of the virtualised resources quotas, and allows the NFVO to provide such notification to the subscriber. Further, this interface allows API version information retrieval.

Support for this interface is optional.

The operations provided through this interface are:

- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

11.1a API version

For the virtualised resource quota available notification interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 2 and the PATCH version field shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.3.1 of the present document, and the MINOR version 1 corresponds to the version of the API specified in version 2.4.1 of the present document.

11.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vrqan" shall be used to represent {apiName}. All resource URIs in the sub-clauses below are defined relative to the above base URI.

Figure 11.2-1 shows the overall resource URI structure defined for the Virtualised Resources Quota Available Notification interface.

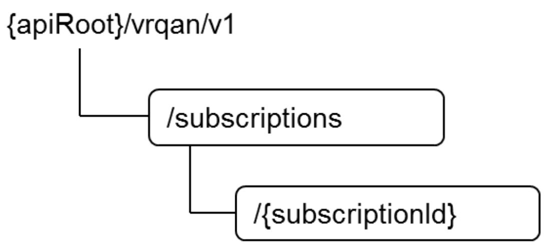

Figure 11.2-1: Resource URI structure of Virtualised Resources Quota Available Notification Interface

Table 11.2-1 lists the individual resources defined, and the applicable HTTP methods.

If the NFVO supports the Virtualised Resources Quota Available Notification interface, the NFVO shall support responding to requests for all HTTP methods on the resources in table 11.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 11.2-1: Resources and methods overview of the Virtualised Resources Quota Available Notification interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
Subscriptions	/subscriptions	POST	M	Subscribe to the notifications related to the availability of the virtualised resources quotas
		GET	M	Query subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read individual subscription
		DELETE	M	Terminate subscription
Notification endpoint	(client-provided)	POST	See note	Notify about the availability of the virtualised resources quota. See note
		GET	See note	Test the notification endpoint. See note
NOTE: If the NFVO supports the Virtualised Resources Quota Available Notification interface, the NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the VNFM. If the VNFM supports the Virtualised Resources Quota Available Notification interface, it shall support responding to the HTTP requests defined for the "Notification endpoint" resource.				

11.3 Sequence diagrams (informative)

11.3.1 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to the availability of the virtualised resources quotas.

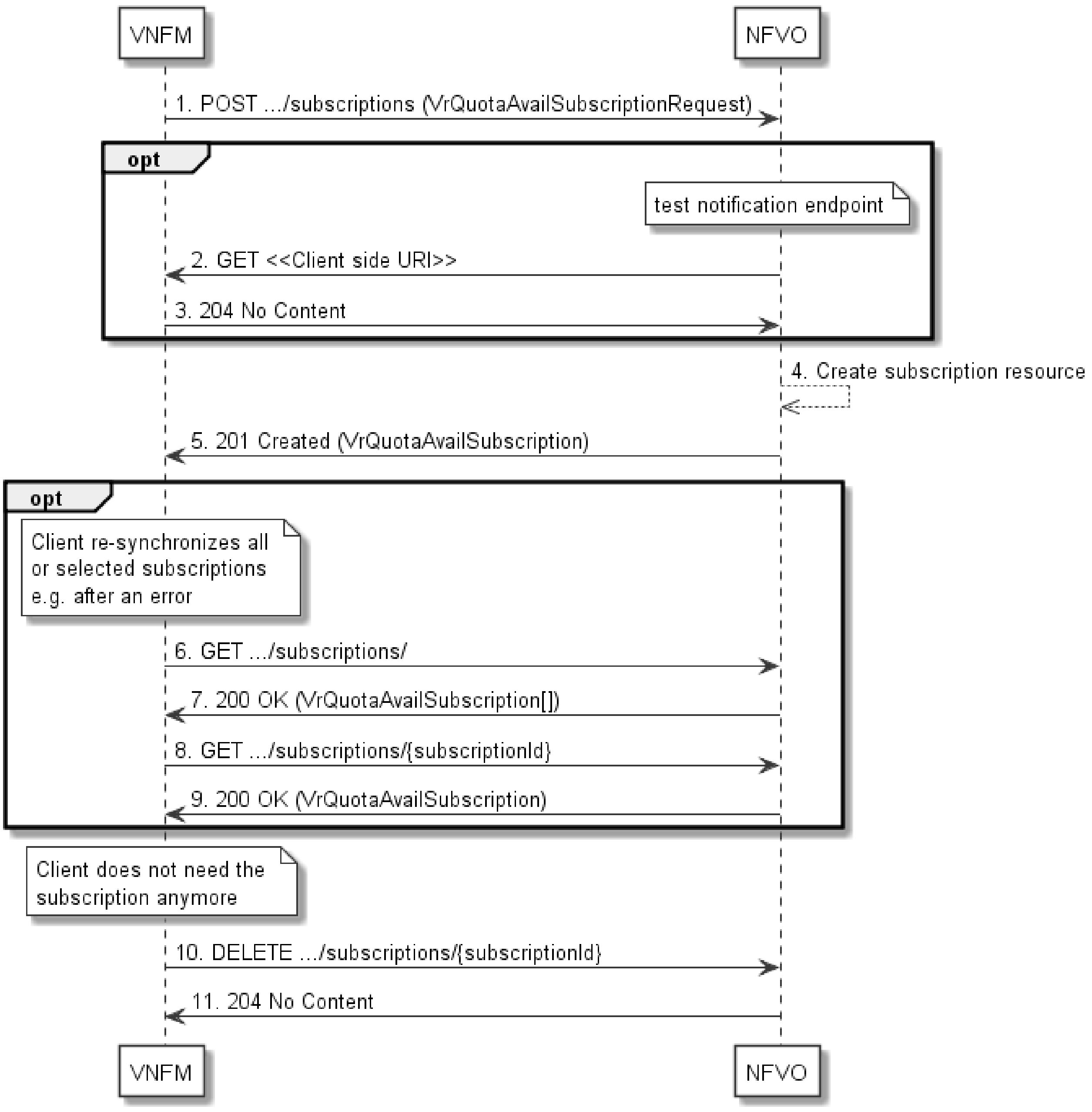
VNFM
NFVO
1. POST .../subscriptions (VrQuotaAvailSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (VrQuotaAvailSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (VrQuotaAvailSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (VrQuotaAvailSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
VNFM
NFVO

Figure 11.3.1-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 11.3.1-1:

1) The VNFM sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "VrQuotaAvailSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the NFVO will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the VNFM as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

3) In that case, the VNFM returns a "204 No Content" response to indicate success.

4) The NFVO creates a new subscription to notifications related to the availability of the virtualised resources quotas, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "VrQuotaAvailSubscription" representing the subscription resource just created by the NFVO, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) If desired, e.g. to recover from an error situation, the VNFM may obtain information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the VNFM.

8) If desired, the VNFM may obtain information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the NFVO returns a "200 OK" response that contains a representation of that individual subscription.

10) If the VNFM does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11) The NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

11.3.2 Flow of sending notifications

This clause describes the procedure of sending notifications related to the availability of virtualised resources quota.

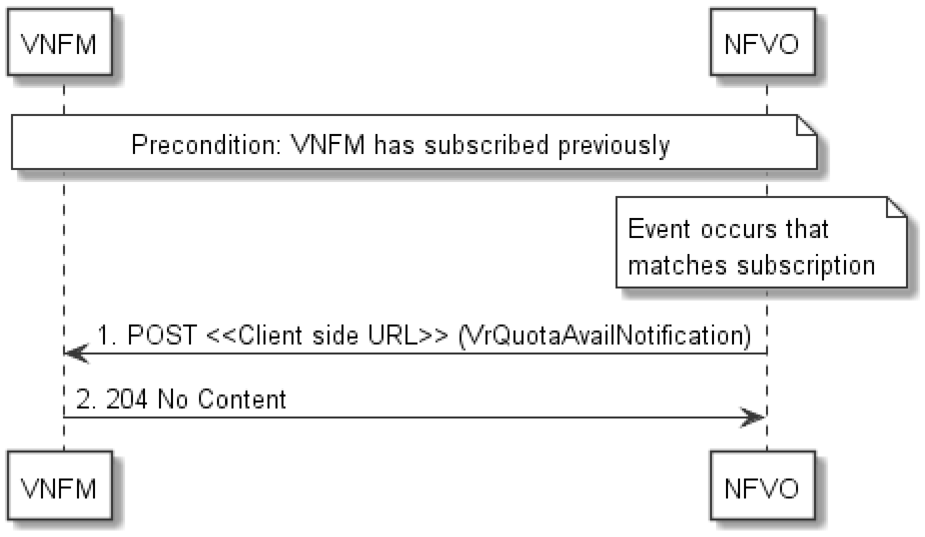
VNFM
NFVO
Precondition: VNFM has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URL>> (VrQuotaAvailNotification)
2. 204 No Content
VNFM
NFVO

Figure 11.3.2-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in figure 11.3.2-1:

Precondition: The VNFM has subscribed previously to notifications related to the availability of virtualised resources quotas.

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a VrQuotaAvailNotification that includes information about the event, and sends it in the body of a POST request to the URI which the VNFM has registered as part of the subscription request.

2) The VNFM acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the NFVO does not receive the "204 No Content" response from the VNFM, it can retry sending the notification.

11.4 Resources

11.4.1 Introduction

This clause defines all the resources and methods provided by the virtualised resources quota available notification interface.

11.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the virtualised resources quota available Notification interface.

11.4.2 Resource: Subscriptions

11.4.2.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to the availability of the virtualised resources quotas, and to query its subscriptions.

11.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vrqan/v1/subscriptions

This resource shall support the resource URI variables defined in table 11.4.2.2-1.

Table 11.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

11.4.2.3 Resource methods

11.4.2.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 11.4.2.3.1-1 and 11.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the VNFM, and might make sense only in very rare use cases. Consequently, the NFVO may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 11.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 11.4.2.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**		**Description**
	VrQuotaAvailSubscriptionRequest	1		Details of the subscription to be created.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VrQuotaAvailSubscription	1	201 Created	Representation of the created subscription resource. The HTTP response shall include a "Location" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the NFVO is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.2.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 11.4.2.3.2-1 and 11.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 11.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The VNFM may supply this parameter. All attribute names that appear in the VrQuotaAvailSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the VNFM if the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 11.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VrQuotaAvailSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of virtualised resource quota available subscriptions as defined in clause 11.5.2.3. If the VNFM supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Response too big. If the VNFM supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.3 Resource: Individual subscription

11.4.3.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to the availability of the virtualised resources quotas.

11.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vrqan/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 11.4.3.2-1.

Table 11.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

11.4.3.3 Resource methods

11.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.3.3.2 GET

The GET method reads an individual subscription.

This method shall follow the provisions specified in the tables 11.4.3.3.2-1 and 11.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 11.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 11.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VrQuotaAvailSubscription	1	200 OK	Representation of the subscription resource
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.3.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 11.4.3.3.5-1 and 11.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 11.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 11.4.3.3.5-2: Details of the DELETE request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The subscription resource was deleted successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.4 Resource: Notification endpoint

11.4.4.1 Description

This resource represents a notification endpoint.

The API producer can use this resource to send notifications related to virtualised resources quota availability to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

11.4.4.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 11.4.4.2-1.

Table 11.4.4.2-1: Resource URI variables for this resource

Name	Definition
n/a	

11.4.4.3 Resource methods

11.4.4.3.1 POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 11.4.4.3.1-1 and 11.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 11.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 11.4.4.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	VrQuotaAvailNotification	1	A notification related to the availability of the virtualised resources quota.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.4.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 11.4.4.3.2-1 and 11.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 11.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 11.4.4.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4 / 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

11.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

11.5 Data model

11.5.1 Introduction

This clause defines the request and response data structures of the Virtualised Resources Quota Available Notification interface. If a request or response contains attributes not defined in the present document, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

11.5.2 Resource and notification data types

11.5.2.1 Introduction

This clause defines data structures to be used in resource representations and notifications.

11.5.2.2 Type: VrQuotaAvailSubscriptionRequest

This type represents a subscription request related to notifications related to the availability of the virtualised resources quotas. It shall comply with the provisions defined in table 11.5.2.2-1.

Table 11.5.2.2-1: Definition of the VrQuotaAvailSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	VrQuotaAvailNotificationsFilter	0..1	Input filter for selecting notifications to subscribe to. This filter can contain information about specific attributes of the virtualised resources quota.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

11.5.2.3 Type: VrQuotaAvailSubscription

This type represents a subscription related to notifications related to the availability of the virtualised resources quotas. It shall comply with the provisions defined in table 11.5.2.3-1.

Table 11.5.2.3-1: Definition of the VrQuotaAvailSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	VrQuotaAvailNotificationsFilter	0..1	Input filter for selecting notifications to subscribe to. This filter can contain information about specific attributes of the virtualised resources quota.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.

11.5.2.4 Type: VrQuotaAvailNotification

This type represents a notification which indicates the availability of a quota applicable to the consumer. It shall comply with the provisions defined in table 11.5.2.4-1. Support of this notification is mandatory if the Virtualised Resources Quota Available Notification interface is supported.

The notification shall be triggered by the NFVO when a virtualised resource quota applicable to the consumer has been set.

Table 11.5.2.4-1: Definition of the VrQuotaAvailNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VrQuotaAvailNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
resourceGroupId	IdentifierInVim	1	Identifier of the "infrastructure resource group", logical grouping of virtual resources assigned to a tenant within an Infrastructure Domain.
vimConnectionInfo	VimConnectionInfo	0..1	Information about the VIM connection to manage the virtualised resources quota. This attribute shall only be supported and present when VNF-related Resource Management in direct mode is applicable.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of the virtualised resources quota. This attribute shall only be supported and present when VNF-related Resource Management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
_links	QuotaAvailLinks	1	Links to resources related to this notification.

11.5.3 Referenced structured data types

11.5.3.1 Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

11.5.3.2 Type: VrQuotaAvailNotificationsFilter

This type represents a subscription filter related to notifications about the availability of the virtualised resources quotas. It shall comply with the provisions defined in table 11.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 11.5.3.2-1: Definition of the VrQuotaAvailNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vimIds	Identifier	0..N	Match VIMs that were created the quota for a consumer of the virtualised resources. This attribute shall only be supported when VNF-related Resource Management in direct mode is applicable.
resourceProviderIds	Identifier	0..N	Match the entities responsible for the management of the virtualised resources that were allocated by the NFVO. This attribute shall only be supported when VNF-related Resource Management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceTypes	Enum (inlined)	0..N	Match particular resource types. Permitted values: - COMPUTE - STORAGE - NETWORK
resourceGroupIds	IdentifierInVim	0..N	Match the "infrastructure resource groups" that are logical groupings of the virtualised resources assigned to a tenant within an infrastructure Domain.

11.5.3.3 Type: QuotaAvailLinks

This type represents the links to resources that a notification of type "VrQuotaAvailNotification" can contain. It shall comply with the provisions defined in table 11.5.3.3-1.

Table 11.5.3.3-1: Definition of the QuotaAvailLinks data type

Attribute name	Data type	Cardinality	Description
subscription	NotificationLink	1	Link to the related subscription.

Annex A (informative): Mapping operations to protocol elements

A.1 Overview

This annex provides the mapping between operations as defined in ETSI GS NFV-IFA 007 [1] and the corresponding resources and HTTP methods defined in the present document.

A.2 VNF Package Management interface

Table A.2-1

ETSI GS NFV-IFA 007 [1] operation	HTTP method	Resource	Direction
Query VNF Package Info	GET	vnfpkgm/v1/vnf_packages	VNFM → NFVO
	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}	VNFM → NFVO
	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/vnfd	VNFM → NFVO
Fetch VNF Package	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content	VNFM → NFVO
Fetch VNF Package Artifacts	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/artifacts/{artifactPath}	VNFM → NFVO
Subscribe	POST	vnfpkgm/v1/subscriptions	VNFM → NFVO
Query Subscription Information	GET	vnfpkgm/v1/subscriptions	VNFM → NFVO
	GET	vnfpkgm/v1/subscriptions/{subscriptionId}	VNFM → NFVO
Terminate subscription	DELETE	vnfpkgm/v1/subscriptions/{subscriptionId}	VNFM → NFVO
Notify	POST	(client-provided)	NFVO → VNFM

A.3 VNF Lifecycle Operation Granting interface

Table A.3-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Grant Lifecycle Operation	POST	grant/v1/grants	VNFM → NFVO
	GET	grant/v1/grants/{grantId}	VNFM → NFVO

A.4 Virtualised Resources Management interfaces in indirect mode

This group of interfaces is outside the scope of the present document.

A.5 Virtualised Resources Quota Available Notification interface

Table A.5-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Subscribe	POST	vrqan/v1/subscriptions	VNFM → NFVO
Query Subscription Information	GET	vrqan/v1/subscriptions	NFVO → VNFM
	GET	vrqan/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Terminate subscription	DELETE	vrqan/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Notify	POST	(client-provided)	NFVO → VNFM

A.6 VNF Lifecycle Management interface

Table A.6-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Create VNF Identifier	POST	vnflcm/v1/vnf_instances	NFVO → VNFM
Instantiate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/instantiate	NFVO → VNFM
Scale VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/scale	NFVO → VNFM
Scale VNF to Level	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/scale_to_level	NFVO → VNFM
Change VNF Flavour	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/change_flavour	NFVO → VNFM
Terminate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/terminate	NFVO → VNFM
Delete VNF Identifier	DELETE	vnflcm/v1/vnf_instances/{vnfInstanceId}	NFVO → VNFM
Query VNF	GET	vnflcm/v1/vnf_instances/{vnfInstanceId}	NFVO → VNFM
	GET	vnflcm/v1/vnf_instances	NFVO → VNFM
Heal VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/heal	NFVO → VNFM
Operate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/operate	NFVO → VNFM
Change External VNF Connectivity	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/change_ext_conn	NFVO → VNFM
Modify VNF Information	PATCH	vnflcm/v1/vnf_instances/{vnfInstanceId}	NFVO → VNFM
Get Operation Status	GET	vnflcm/v1/vnf_lcm_op_occs	NFVO → VNFM
	GET	vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}	NFVO → VNFM
Subscribe	POST	vnflcm/v1/subscriptions	NFVO → VNFM
Query Subscription Information	GET	vnflcm/v1/subscriptions	NFVO → VNFM
	GET	vnflcm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Terminate Subscription	DELETE	vnflcm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Notify	POST	(client-provided)	VNFM → NFVO

A.7 VNF Performance Management interface

Table A.7-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Create PM Job	POST	vnfpm/v1/pm_jobs	NFVO → VNFM
Delete PM Job	DELETE	vnfpm/v1/pm_jobs/{pmJobId}	NFVO → VNFM
Query PM Job	GET	vnfpm/v1/pm_jobs	NFVO → VNFM
	GET	vnfpm/v1/pm_jobs/{pmJobId}	NFVO → VNFM
Create Threshold	POST	vnfpm/v1/thresholds	NFVO → VNFM
Delete Threshold	DELETE	vnfpm/v1/thresholds/{thresholdId}	NFVO → VNFM
Query Threshold	GET	vnfpm/v1/thresholds	NFVO → VNFM
	GET	vnfpm/v1/thresholds/{thresholdId}	NFVO → VNFM
Subscribe	POST	vnfpm/v1/subscriptions	NFVO → VNFM
Query Subscription Information	GET	vnfpm/v1/subscriptions	NFVO → VNFM
	GET	vnfpm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Terminate Subscription	DELETE	vnfpm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Notify	POST	(client-provided)	VNFM → NFVO

A.8 VNF Fault Management interface

Table A.8-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Get Alarm List	GET	vnffm/v1/alarms	NFVO → VNFM
Acknowledge Alarm	PATCH	vnffm/v1/alarms/{alarmId}	NFVO → VNFM
Subscribe	POST	vnffm/v1/subscriptions	NFVO → VNFM
Query Subscription Information	GET	vnffm/v1/subscriptions	NFVO → VNFM
	GET	vnffm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Terminate Subscription	DELETE	vnffm/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Notify	POST	(client-provided)	VNFM → NFVO

A.9 VNF Indicator interface

Table A.9-1

ETSI GS NFV-IFA 007 operation	HTTP method	Resource	Direction
Get Indicator Value	GET	vnfind/v1/indicators	NFVO → VNFM
	GET	vnfind/v1/indicators/{vnfInstanceId}	NFVO → VNFM
	GET	vnfind/v1/indicators/{vnfInstanceId}/{indicatorId}	NFVO → VNFM
Subscribe	POST	vnfind/v1/subscriptions	NFVO → VNFM
Query Subscription Information	GET	vnfind/v1/subscriptions	NFVO → VNFM
	GET	vnfind/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Terminate Subscription	DELETE	vnfind/v1/subscriptions/{subscriptionId}	NFVO → VNFM
Notify	POST	(client-provided)	VNFM → NFVO

Annex B (informative): Explanations

B.1 Introduction

This annex provides explanations of certain concepts introduced in the present document.

In clause B.2, the underlying concepts of scaling a VNF instance are explained.

In clause B.3, examples of VNF connectivity patterns, and change of VNF external connectivity, are provided.

B.2 Scaling of a VNF instance

A VNF instance can be scaled in the following ways:

- scale out: adding additional VNFC instances to the VNF to increase capacity
- scale in: removing VNFC instances from the VNF, in order to release unused capacity

This mechanism is called "horizontal scaling".

NOTE: Besides that, there is also "vertical scaling" which is not supported in the present document, and which includes scale up (adding further resources to existing VNFC instances) and scale down (removing resources from existing VNFC instances).

Potentially, different *aspects* of a VNF can be scaled independently. For example, a VNF could be designed to provide static capacity such as database nodes and dynamic capacity such as query processing nodes. Such a VNF might be scaled w.r.t. two separate aspects: the 'static capacity' aspect can be scaled by adding VNFCs from VNF Deployment Units (VDUs) defining database nodes, and the 'dynamic capacity' aspect can be scaled by adding VNFCs from VDUs defining query processing nodes. In complex VNF designs, scaling a VNF often requires adding/removing a number of related VNFC instances of several different types, possibly based on multiple VDUs. For example, in a high availability configuration, it might be required to add in each scaling step a pair of VNFC instances, one in active and one in standby configuration. The scaling aspects valid for a particular VNF are declared in the VNFD.

Each scaling aspect can only be scaled in discrete steps, the so-called "*scaling steps*". Each scaling step corresponds to adding or removing an *increment* (set of VNFCs based on one or more VDUs, and the related virtualised storage / virtualised network resources) to or from the VNF instance, and (re)configuring the virtualised resources. Per increment, the VNFM will figure out the necessary set of VNFCs and the related set of resources based on VNF-specific rules, for instance using the lifecycle management script associated to the Scale VNF or Scale VNF to Level event.

When scaling a VNF for a particular aspect, the number of scaling steps to apply to that aspect can be provided as a parameter. A scaling step is the smallest unit by which a particular aspect of a VNF can be scaled, and is mapped by the VNFM to the addition (or removal) of a certain set of resources. For each scaling aspect, the minimum scale level is assumed as zero, and the maximum scale level is defined in the VNFD. The maximum scale level corresponds to the maximum number of scaling steps that can be performed for this aspect, starting from the minimum scale level (i.e. zero). The maximum scale level represents the maximum configuration of that aspect of the VNF in a given deployment flavour. The minimum scale level represents the minimum configuration of that aspect of the VNF in a given deployment flavour. It usually corresponds to some deployed resources, but it is also possible to define in the VNFD that certain VDUs may not always have a corresponding VNFC instance, i.e. for certain aspects the minimum configuration may indeed be empty.

At each point in time between the completed VNF instantiation and the VNF termination, the current "size" of a particular scaling aspect of the VNF can be expressed by the current scale level w.r.t. that aspect. When the VNF is instantiated, the current scale level is initialized with values that are defined as part of the instantiation level in the VNFD for the associated aspect. Figure B.2-1 illustrates the concepts described above.

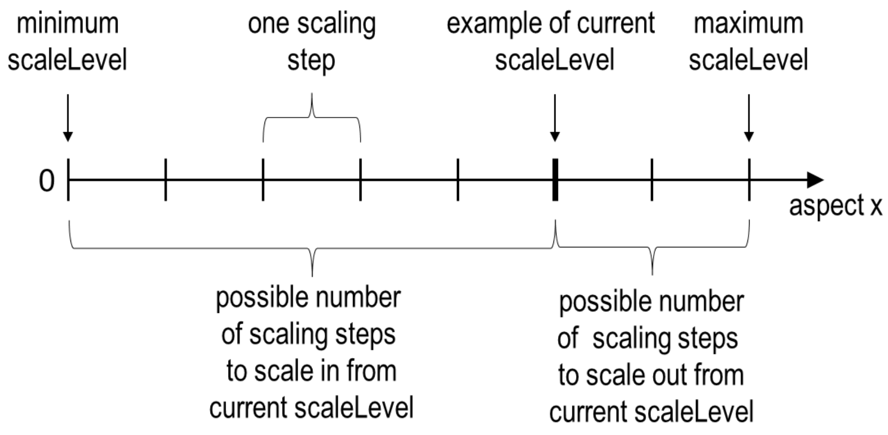
minimum scaleLevel
one scaling step
example of current scaleLevel
maximum scaleLevel
0
aspect x
possible number of scaling steps to scale in from current scaleLevel
possible number of scaling steps to scale out from current scaleLevel

Figure B.2-1: Illustrating the concepts of scale level and scaling steps for a particular scaling aspect

As indicated above, a VNF can have one or more scaling aspects. Each individual aspect has a current scale level. All pairs of (aspect, scaleLevel) together are called the *scale status* of the VNF instance and can be obtained from the "scaleStatus" attribute of the VnfInstance structure which is returned when reading the "Individual VNF instance" resource or when querying the "VNF instances" resource. Example 1 illustrates a possible scale status.

EXAMPLE 1:

```
"scaleStatus": [
    {"aspectId": "processing", "scaleLevel": "2"},
    {"aspectId": "database", "scaleLevel": "3"}
]
```

When requesting scaling of a VNF instance, there are two methods: Scale VNF (see clause 5.4.5) and Scale VNF to Level (see clause 5.4.6). When using "Scale VNF", the scaling request defines how many increments (scaling steps) are requested to be added to or removed from the current "size" (scale level) *for a single aspect*. Depending on the VNF capabilities, single-step scaling or multiple-step scaling can be supported in a single scale request. When using "Scale VNF to Level", the scale request defines a target size of the VNF instance by defining the requested target size *for all aspects at once*, independent from the current scale status (current size) of the VNF instance. The target size can be expressed by referencing pre-defined sizes (called *instantiation levels*) declared in the VNFD, or by explicitly providing the target scale level for each scaling aspect, as illustrated in example 2.

EXAMPLE 2:

```
"scaleInfo": [
    {"aspectId": "processing", "scaleLevel": "4"},
    {"aspectId": "database", "scaleLevel": "2"}
]
```

These combinations allow four sub-modes of scaling:

- Scale VNF with a single step.
- Scale VNF with multiple steps.
- Scale VNF to Level based on pre-defined sizes (instantiation levels) only.
- Scale VNF to Level with arbitrary sizes.

B.3 Examples of VNF connectivity patterns

B.3.1 Overview

Clause B.3.3 illustrates examples of possible connectivity patterns for a VNF. The purpose is to illustrate the relationship among the different information elements specified in clause 8.5 that are used to describe the connectivity of and within a VNF instance.

NOTE: The information related to connectivity as shown in clause B.3.2 is to be understood in the context of the present document, i.e. availability of certain information on the Or-Vnfm reference point follows the conditions that are detailed in the respective attribute descriptions and notes in the present document.

This clause also illustrates the use of the "Change external VNF connectivity" task resource to re-connect external CPs of a VNF instance to a different external VL.

B.3.2 Example of a VNF instance with two different types of external connections points

The present example shows a regular connectivity pattern of a VNF where the two external CPs of the VNF use different connectivity patterns. Figure B.3.2-1 illustrates the example, from which it is highlighted the following:

- An external CP of the VNF instance (see VnfExtCp #1) that maps to an internal CP, i.e. a CP of a specific VNFC.
- An external CP of the VNF instance (see VnfExtCp #2) that refers to a link port of an internal VL of the VNF (see VnfLinkPort #2.2).
- An internal VL of the VNF instance (see VnfVirtualLink #1) that is only used for connectivity of VNFCs within the VNF.
- An internal VL of the VNF instance (see VnfVirtualLink #2) that is used as provider of a link port for connectivity of external CPs of the VNF.
- Link ports of internal VLs of the VNF instance (see VnfLinkPort #1.1 to #1.3 and VnfLinkPort #2.1) that are optionally exposed on Or-Vnfm reference point.
- Internal CPs, i.e. CPs of specific VNFCs (see grey VNFC CPs) that are optionally exposed on the Or-Vnfm reference point.

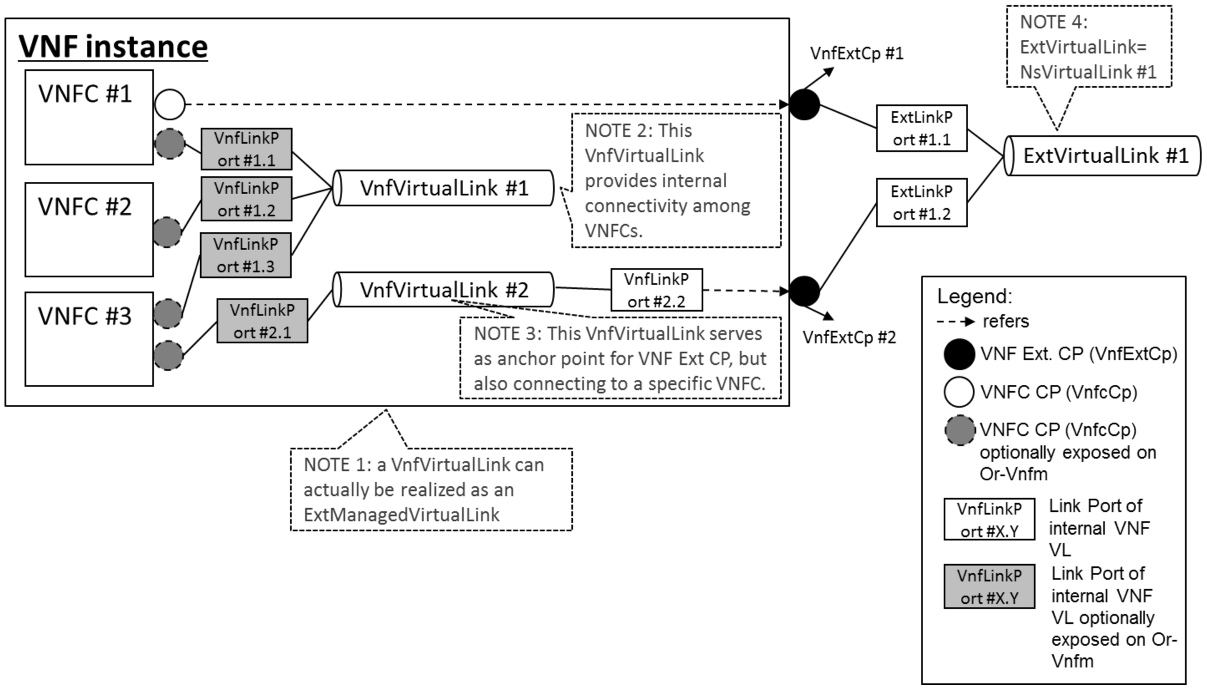
VNF instance
VNFC #1
VNFC #2
VNFC #3
VnfLinkPort #1.1
VnfLinkPort #1.2
VnfLinkPort #1.3
VnfLinkPort #2.1
VnfLinkPort #2.2
VnfVirtualLink #1
VnfVirtualLink #2
NOTE 2: This VnfVirtualLink provides internal connectivity among VNFCs.
NOTE 3: This VnfVirtualLink serves as anchor point for VNF Ext CP, but also connecting to a specific VNFC.
NOTE 1: a VnfVirtualLink can actually be realized as an ExtManagedVirtualLink
VnfExtCp #1
VnfExtCp #2
ExtLinkPort #1.1
ExtLinkPort #1.2
ExtVirtualLink #1
NOTE 4: ExtVirtualLink= NsVirtualLink #1
Legend:
refers
VNF Ext. CP (VnfExtCp)
VNFC CP (VnfcCp)
VNFC CP (VnfcCp) optionally exposed on Or-Vnfm
VnfLinkPort #X.Y
Link Port of internal VNF VL
Link Port of internal VNF VL optionally exposed on Or-Vnfm

Figure B.3.2-1: Example of a VNF instance with two different types of external connections points

B.3.3 Example of changing VNF connectivity

This example illustrates changing the external connectivity of a VNF instance using the "Change external VNF connectivity" task resource (clause 5.4.11). The scenario depicted disconnects from a "source" external VL all those external CP instances that were created based on a particular CPD, and connects them to a "target" external VL.

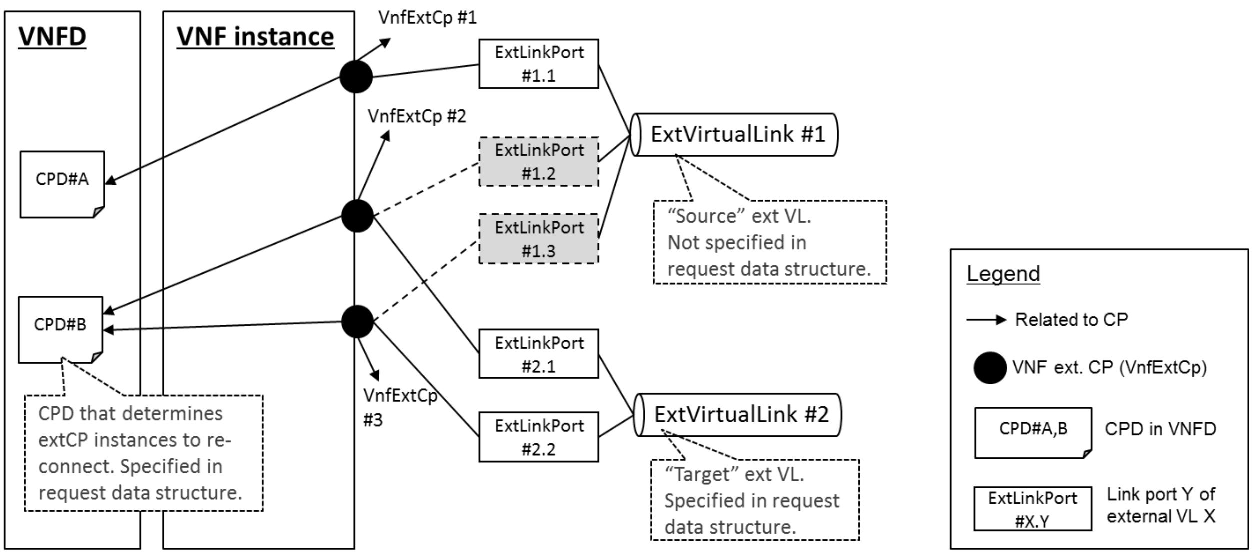

Figure B.3.3-1: Illustration of disconnecting external CPs from one external VL and connecting them to another external VL

Annex C (normative): VimConnectionInfo registry

C.1 Purpose

This annex defines the basic structure of the entries of a registry for VimConnectionInfo parameters, the structure of the identifiers, and the registration template. The registry contains a reference to the present document to indicate where the structure of the VimConnectionInfo data type is defined.

C.2 Registry content

The primary elements of the registry are:

Registered identifier

- vimType: Identifier of a set of VimConnectionInfo parameters (mandatory)

NOTE 1: The registration authority is required to ensure global uniqueness of registered identifiers.

Registered interface information for a particular vimType

- interfaceInfo: Interface information as a list of key names with data type, permissible values and description (mandatory)
- accessInfo: Access information as a list of key names with data type, permissible values and description (mandatory)
- extra: Additional specific information as a list of key names with data type, permissible values and description (optional)

Registrant information

- Registrant Name: Name of the company or organization registering the vimType (mandatory)
- Previous Registrant Name(s): Name or names of the company or organization to whom the registered identifier has belonged previously, e.g. due to buyout, merger, acquisition (optional)

NOTE 2: It is assumed that the registration authority will manage further information related to the identity of the registrant (e.g. contact information).

Additional information

- Solution Name: Name of the VIM for which the VimConnectionInfo parameter set is being registered (e.g. "OpenStack Release xyz with Keystone") (mandatory)
- Description: General description of the VIM for which the VimConnectionInfo parameter set is being registered (e.g. "ETSI-registered VIM Connection Info to enable the use of Openstack rel xyz with SOL003. Interface is using keystone as the gateway. Valid for releases starting from xyz"). (mandatory)
- Specification URI: Publically reachable URI of the specification that defines further details of the particular VIM Connection Info registered. Needs to be long-lived. (recommended)
- Registration Date: Date of the registration (mandatory)

C.3 Structure of the vimType identifier

A registered vimType identifier shall comply with the following syntax:

- It shall have the following structure:
  `<registrant> "." <vimName> ["." <version>].`
- <registrant> and <vimName> shall be strings that contain only uppercase letters, digits, "_" and "-".
- <registrant> shall be a concise string that represents the registrant (e.g., the company or organization name) chosen at registration time.
- The <registrant> value of "PRIVATE" is reserved for private use by implementations and shall not be used in registered identifiers.
- <vimName> shall be a string that represents the VIM Type.
- <version> shall be a string with the following structure:
  `"V_" <x> [ "_" <y>].`
- Providing a <version> is optional. In the <version> string, "x" and "y" represent a sequence of digits that denote the major (x) and minor (y) part of the version number of the VIM. Providing <y> is optional.
- If there are no changes of the interface in subsequent versions of the VIM, i.e. the registration of the previous version can still be used with the new versions, a registration of these subsequent versions is not required.

C.4 Initial registration

C.4.1 Instructions for data structure definition

Data structure definitions shall be submitted using JSON schema according to [i.8], separately for the three objects "interfaceInfo" (mandatory), "accessInfo" (mandatory) and "extra" (optional). The "description" keyword in the JSON schema shall be used to appropriately document each attribute in the data structure. The JSON schema shall include information about allowed values where applicable, formulated as JSON schema constraints (such as "minimumInclusive") or documented in text form using the "description" keyword.

In the registration template, items that are mandatory to be provided are marked with [M]; optional items are marked with [O].

C.4.2 Template

1 Solution information

Solution Name [M]	<Name of the VIM for which the VimConnectionInfo parameter set is being registered.> *EXAMPLE: "OpenStack Release xyz with Keystone"*
Description [M]	General description of the VIM for which the VimConnectionInfo parameter set is being registered *EXAMPLE: "ETSI-registered VIM Connection Info to enable the use of Openstack rel xyz with SOL003. Interface is using keystone as the gateway. Valid for releases starting from xyz".*
Specification URI [O]	<Publically reachable, long-lived URI of a specification that defines further details of the particular VIM Connection Info registered. Optional, but recommended.>

2 Registration information

Registrant name [M]	<Name of the legal entity requesting registration>
Registrant address [M]	<Address of the legal entity requesting registration>
Registrant contact [M]	<Name and email address of the contact person or the function of the legal entity requesting registration>
Registration date [M]	<The date when the registration request was sent>

3 Requested vimType identifier

Registrant [M]		VIM Name [M]		Version [O]
	.		.	

4 JSON schema definition of "interfaceInfo"

Purpose: Provides information about the interface or interfaces to the VIM, such as the URI of an interface endpoint to communicate with the VIM.

interfaceInfo [M]

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "interfaceInfo",
    "type": "object",
    "properties": {
      <include list of properties here >
    },
    "required": [<define properties that are required>]
}
```

5 JSON schema definition of "accessInfo"

Purpose: Provides authentication credentials for accessing the VIM, and other access-related information such as tenants or infrastructure resource groups.

accessInfo [M]

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "accessInfo",
    "type": "object",
    "properties": {
      <include list of properties here >
    },
    "required": [<define properties that are required>]
}
```

6 JSON schema definition of "extra"

Purpose: Provides optional additional VIM type specific information.

extra [O]

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "extra",
    "type": "object",
    "properties": {
      <include list of properties here >
    },
    "required": [<define properties that are required>]
}
```

C.5 Registration update

Only limited parts of the registration information are allowed to be updated:

- Registrant name (in case of mergers etc.). In this case, the registrant information will be updated and the previous registrant information will be preserved in a special "previous registrants" section.
- Registrant contact data (in case of change of contact person).
- Specification URI (in case of update of URI).

C.6 Initial registry content

C.6.1 Registration for ETSINFV.OPENSTACK_KEYSTONE.V_2

1 Solution information

Solution Name [M]	OpenStack with Keystone V2
Description [M]	ETSI-registered VIM Connection Info defining the interface and access parameters to use an OpenStack-based VIM with Keystone V2, to be signalled via the APIs specified in ETSI GS NFV-SOL 003. Keystone is used for access control to the VIM interfaces.
Specification URI [O]	https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/003/

2 Registration information

Registrant name [M]	ETSI ISG NFV

3 Requested vimType identifier

Registrant [M]		VIM Name [M]		Version [O]
ETSINFV	.	OPENSTACK_KEYSTONE	.	V_2

4 JSON schema definition of "interfaceInfo"

Purpose: Provides information about the interface or interfaces to the VIM, such as the URI of an interface endpoint to communicate with the VIM.

interfaceInfo [M]

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "title": "interfaceInfo",
   "additionalProperties": false,
   "required": [
      "endpoint"
   ],
   "type": "object",
   "properties": {
      "endpoint": {
         "type": "string",
         "format": "url",
         "description": "The url representing the interface endpoint."
```

```
       },
       "trustedCertificates": {
          "items": {
              "type": "string",
              "format": "byte"
          },
          "type": "array",
          "description": "A collection of base64 encoded certificates to be
trusted in relation to the endpoint."
       },
       "skipCertificateHostnameCheck": {
          "default": false,
          "type": "boolean",
          "description": "Certificate hostname check for the endpoint can be
skipped by setting this field to true."
       },
       "skipCertificateVerification": {
          "default": false,
          "type": "boolean",
          "description": "Certificate verification for the endpoint can be
skipped by setting this field to true."
       }
   }
}
```

5 JSON schema definition of "accessInfo"

Purpose: Provides authentication credentials for accessing the VIM, and other access-related information such as tenants or infrastructure resource groups.

accessInfo [M]

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "title": "accessInfo",
   "additionalProperties": false,
   "required": [
       "username",
       "password",
       "region",
       "tenant"
   ],
   "type": "object",
   "properties": {
       "username": {
          "type": "string",
          "description": "The username to use for access."
       },
       "region": {
          "type": "string",
          "description": "The OpenStack region to use for the VIM connection."
       },
       "password": {
          "writeOnly": true,
          "type": "string",
          "format": "password",
          "description": "The password to use for access. Required for input,
not returned on output."
       },
       "tenant": {
          "type": "string",
          "description": "The OpenStack tenant to use for the VIM connection."
```

```
        }
    }
}
```

6 JSON schema definition of "extra"

Purpose: Provides optional additional VIM type specific information.

extra [O]
not specified

C.6.2 Registration for ETSINFV.OPENSTACK_KEYSTONE.V_3

1 Solution information

Solution Name [M]	OpenStack with Keystone V3
Description [M]	ETSI-registered VIM Connection Info defining the interface and access parameters to use an OpenStack-based VIM with Keystone V2, to be signaled via the APIs specified in ETSI GS NFV-SOL 003. Keystone is used for access control to the VIM interfaces.
Specification URI [O]	https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/003/

2 Registration information

Registrant name [M]	ETSI ISG NFV

3 Requested vimType identifier

Registrant [M]		VIM Name [M]		Version [O]
ETSINFV	.	OPENSTACK_KEYSTONE	.	V_3

4 JSON schema definition of "interfaceInfo"

Purpose: Provides information about the interface or interfaces to the VIM, such as the URI of an interface endpoint to communicate with the VIM.

interfaceInfo [M]

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "interfaceInfo",
    "additionalProperties": false,
    "required": [
        "endpoint"
    ],
    "type": "object",
    "properties": {
        "endpoint": {
            "type": "string",
            "format": "url",
            "description": "The url representing the interface endpoint."
        },
        "trustedCertificates": {
            "items": {
                "type": "string",
                "format": "byte"
            },
          "type": "array",
          "description": "A collection of base64 encoded certificates to be
trusted in relation to the endpoint."
       },
       "skipCertificateHostnameCheck": {
          "default": false,
          "type": "boolean",
          "description": "Certificate hostname check for the endpoint can be
skipped by setting this field to true."
       },
       "skipCertificateVerification": {
          "default": false,
          "type": "boolean",
          "description": "Certificate verification for the endpoint can be
skipped by setting this field to true."
       }
    }
}
```

5 JSON schema definition of "accessInfo"

Purpose: Provides authentication credentials for accessing the VIM, and other access-related information such as tenants or infrastructure resource groups.

accessInfo [M]

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "accessInfo",
    "additionalProperties": false,
    "required": [
       "username",
       "password",
       "region",
       "project",
        "projectDomain",
        "userDomain"
    ],
    "type": "object",
    "properties": {
       "username": {
          "type": "string",
          "description": "The username to use for access."
       },
       "userDomain": {
          "type": "string",
          "description": "The OpenStack user domain to use for the VIM
connection."
        },
       "region": {
          "type": "string",
          "description": "The OpenStack region to use for the VIM connection."
       },
       "password": {
          "writeOnly": true,
          "type": "string",
          "format": "password",
          "description": "The password to use for access. Required for input,
not returned on output."
       },
       "project": {
          "type": "string",
          "description": "The OpenStack project to use for the VIM
connection."
       },
       "projectDomain": {
          "type": "string",
          "description": "The OpenStack project domain to use for the VIM
connection"
       }
    }
}
```

6 JSON schema definition of "extra"

Purpose: Provides optional additional VIM type specific information.

extra [O]
not specified

Annex D (informative): Complementary material for API utilization

To complement the definitions of each method, resource, and data type defined in the main body of the present document, the ETSI NFV ISG is providing supplementary description files, compliant to the OpenAPI Specification [i.9], for the Or-Vnfm reference point. These supplementary description files, containing the OpenAPI specification for each API defined in the present document, are located at https://forge.etsi.org/rep/nfv/NFV-SOL003.

In case of discrepancies between the supplementary files and the related data structure definitions in the main body of the present document, the data structure definitions take precedence.

The OpenAPI representations referenced above:

1) use the MAJOR.MINOR.PATCH version fields to signal the version of the API as defined in the present document, and

2) use the "impl" version parameter to represent changes to the OpenAPI representation without changing the present document (see clause 4.6.1.2).

The full version identifier of an API appears in the corresponding OpenAPI file, in the "version" subfield of the "info" field, where numerical fields are separated by a dot, as illustrated below.

EXAMPLE

```
swagger: "2.0"
info:
  version: "1.0.0-impl:etsi.org:ETSI_NFV_OpenAPI:1"
  title: SOL003 VNF LCM
  license:
    name: "ETSI Forge copyright notice"
    url: https://forge.etsi.org/etsi-forge-copyright-notice.txt
basePath: "/vnflcm/v1"
```

END EXAMPLE

Annex E (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Uwe Rauschenbach, Nokia

Other contributors:

Anatoly Andrianov, Nokia

Ernest Bayha, Ericsson

Bruno Chatras, Orange

Wooyong Choi, SK Telecom

Haibin Chu, Ericsson

Szabolcs Deak, Nokia

Hunor Demeter, Nokia

Aijuan Feng, Huawei

Dmytro Gassanov, NetCracker

Lars-Erik Helander, Procera Networks

Junyi Jiang, Huawei

Shelby Kiewel, iconectiv

TaeYeon Kim, ETRI

Yuya Kuno, DOCOMO Communications Lab.

Thinh Nguyenphu, Nokia

Kazuaki Obana, DOCOMO Communications Lab.

Anne-Marie Praden, Gemalto

René Robert, Orange

Myung-Ki Shin, ETRI

Amanda Xiang, Huawei

Xu Yang, Huawei

Jong-Hwa Yi, ETRI

Annex F (informative): Change History

Date	Version	Information about changes
May 2016	0.0.1	Initial version based on - NFVSOL(16)000008r1 SOL003 ToC Proposal - NFVSOL(16)000009_SOL003_basic_skeleton
September 2016	0.1.0	Contributions incorporated - NFVSOL(16)000092_SOL003_structure_update - NFVSOL(16)000085r2_SOL003_InstantiateVnf_flow - NFVSOL(16)000076r2_SOL002___SOL003_-_4_2_Specification_Methodology - NFVSOL(16)000081r3_SOL003_-_Interface_design_for_CreateVNFId Editorials - Removed the old clause 5 and 6 structure as this is not in line anymore with the template - Added reference to RFC7159 as instructed in the Editor's note in document 76r2, removed the Editor's note, and changed the style of the reference to comply with ETSI Drafting Rules. - Added applicable abbreviations - Added changes of GS title page requested by Technical Steering Committee (NFVTSC(16)000052) - Added editorials requested by Ericsson
October 2016	0.1.5	Contributions incorporated that were agreed at SOL#11 call - NFVSOL(16)000105r1_SOL003_align_GS_with_template_changes_in_104 - NFVSOL(16)000096_SOL003_4_2_RFC3986 - NFVSOL(16)000101_SOL003__Removal_of_Editor_s_Note_in_Section_5_2_1_Creation - NFVSOL(16)000107r1_SOL003_Scope
November 2016	0.2.0	Contributions incorporated that were agreed at SOL#12 meeting in Bundang - NFVSOL(16)000097r3_SOL003_4_2_Normative_status_of_https - NFVSOL(16)000100r3_SOL003_Add_JSON_schema_Annex_skeleton - NFVSOL(16)000102r3_SOL003_VNF_Lifecycle_Management_interface___Terminate_VNF_fl - NFVSOL(16)000108r1_SOL003_Interface_overview_annex - NFVSOL(16)000109_SOL003_operations_mapping_annex - NFVSOL(16)000110r3_SOL002_SOL003_Delete_VNF_Identifier_Interface - NFVSOL(16)000113r3_SOL003_ScaleVnf_flow - NFVSOL(16)000115r2_SOL003_QueryVnf_flow - NFVSOL(16)000116r3_SOL003_GetOperationStatus_flow - NFVSOL(16)000118r2_SOL002_SOL003_instantiate_VNF_resource_description - NFVSOL(16)000119r3_SOL002_SOL003_terminate_VNF_resource_description - NFVSOL(16)000120r4_SOL003_LCM-Operate_Interface - NFVSOL(16)000122r1_SOL003_data_type_of_CreateVnfRequest - NFVSOL(16)000128_SOL003_small_change - NFVSOL(16)000129r2_SOL003_Instantiate_flow_changes - NFVSOL(16)000132r1_SOL003_LCCN_notification_interface_resources_and_methods - NFVSOL(16)000134r1_SOL003_Resource_of_VNF_lifecycle_operation_occurrences - NFVSOL(16)000137_SOL003_Apply_convention_change_from_document_136

Date	Version	Information about changes
November 2016	0.3.0	Contributions incorporated that were agreed at SOL#13 and SOL#14 - NFVSOL(16)000145r2_Conventions_simplifying_the_table - NFVSOL(16)000131r2_SOL003_Granting_interface_resources_and_methods - NFVSOL(16)000140R1_SOL003_Create_Delete_VNF_Identifier_-_make_consistent_with_REST - NFVSOL(16)000156R1 Change Flavour Inerface
December 2016	0.4.0	Contributions incorporated that were agreed in SOL#15 call and at SOL#16Shenzhen F2F - NFVSOL(16)000144r3_SOL003_refactoring_the_LCM_flows (aligned style of headlines of flows afterwards as an editorial action, and moved the Delete VNF Instance clause to a more appropriate place) - NFVSOL(16)000159R1_SOL003_LCCN_interface_flows - NFVSOL(16)000160r2_SOL003_LCCN_interface_subscriptions - NFVSOL(16)000161r1_SOL003_LCCN_interface_notification_resources - NFVSOL(16)000167_SOL003_Editors_note_on_granting_OperateVnf - NFVSOL(16)000172r1_SOL003_Attribute_filters (added clause "4.3.1 Introduction" as an editorial action on top of the text introduced by this contribution in order to improve the reading flow). - NFVSOL(16)000174r2_SOL003_LCM_error_handling - NFVSOL(16)000179r3_SOL003_Heal_VNF_Interface - NFVSOL(16)000180r2_SOL003_Scale_VNF_task - NFVSOL(16)000181r2_SOL003_Scale_VNF_to_Level_task - NFVSOL(16)000191r3_SOL003_VNF_Pkg_resources - NFVSOL(16)000193r2_SOL003_QueryVnfPackage_flow - NFVSOL(16)000196r1_SOL003_Query_VNF_resource_description - NFVSOL(16)000197r2_SOL003_General_Aspects__HTTP_headers - NFVSOL(16)000200r1_SOL003_align_resource_description_clauses - NFVSOL(16)000201r2_SOL003_Modify_VNF_Information_Interface - NFVSOL(16)000202r3_SOL003_VNF_Fault_Management_interface - NFVSOL(16)000208_SOL003_nicer_tables (aligned the table in the FM clause as an editorial action) Editorials: - Aligned the format of the references in clause 2.1 - Fixed a copy&paste error in the Granting interface (VNF lifecycle management interface → VNF lifecycle operation granting interface) - Added "Release 2" to title
January 2017	0.5.0	Contributions incorporated that were agreed in NFVSOL#17 call: - NFVSOL(17)000001r1_SOL003_merging_LCM_and_LCCN_interfaces - NFVSOL(16)000173r3_SOL003_Attribute_selectors - NFVSOL(16)000194r4_SOL003_FetchVnfPackage_flow Editorials: - Added "Draft" to page header
January 2017	0.6.0	Contributions incorporated that were agreed at NFVSOL#18 F2F in Munich: - NFVSOL(16)000139r1_SOL003_error_codes_to_signal_application_errors - NFVSOL(16)000195r4_SOL003_FetchVnfPackageArtifacts_flow - NFVSOL(16)000203r2_SOL003_LCCN_interface_notification_data_types (Rapporteur's addition when implementing this contribution: Removed the inline definitions of DateTime as this is now defined as a global type.) - NFVSOL(17)000005r3_SOL003_VnfLcOpOcc_data_structure - NFVSOL(17)000006r1_SOL003_LCM_error_handling_resources - NFVSOL(17)000007r2_SOL003_Indicator_interface_structure - NFVSOL(17)000008_SOL003_Replace_URL_by_URI - NFVSOL(17)000022r1_SOL003_ModifyVnfInformatin_flow - NFVSOL(17)000023r1_SOL003_VR_Quota_Available_resources - NFVSOL(17)000024r1_SOL003_VR_Quota_Available_notification_flow - NFVSOL(17)000026_SOL003_VNF_Pkgm_structure_update - NFVSOL(17)000027r1_SOL003_VnfConfig_implementing_IFA_verdict_part_1_-_PATCH - NFVSOL(17)000028r2_SOL003_VnfConfig_implementing_IFA_verdict_part_2_-_Change_ex - NFVSOL(17)000032r4_SOL003_Granting_resources - NFVSOL(17)000033r2_SOL003_Granting_data_types - NFVSOL(17)000035r1_SOL003_LCM_flow_update - NFVSOL(17)000046_SOL003__Change_to_the_Resource_Representation_of_an_Individu

Date	Version	Information about changes
		Editorials: - Fixed some typos - s/DF/deployment flavour/ - removed the mentioning of boldfaced text in the text referring to figure 5.5.2.1-1 - s/encodeddata/encoded data/ - s/and includes in the entity body binary-encoded data/and includes binary-encoded data in the entity body/ - Fixed leftovers of "information element" in Granting interface (copy&paste error from IFA007)
February 2017	0.7.0	Contributions incorporated that were agreed at NFVSOL#19 call and NFV#20 in Bilbao: - NFVSOL(17)000038r2_SOL003_Add_resource_and_enumeration_for_alarm - NFVSOL(17)000047r1_SOL003_PATCH_entity_body - NFVSOL(17)000079r1_SOL003_General_Data_types - NFVSOL(17)000048r2_SOL003_Indirect_RM - NFVSOL(17)000049_SOL003_Error_handling_Granting_interface - NFVSOL(17)000056r1_SOL003_PM_interface_structure (as agreed in Bilbao, clauses related to subscription resources have been moved to the end of the resources clauses to align with other APIs) - NFVSOL(17)000059_SOL003_Adding_criterion_to_LccnSubscriptionFilter - NFVSOL(17)000064r1_SOL002_SOL003_and_SOL005_Labelling_of_API_Na mes - NFVSOL(17)000066r2_SOL003_VimInfo_fixes_plus_InterfaceInfo_and_Access Info_defin - NFVSOL(17)000074_SOL003_VNF_package_management_notification_flows - NFVSOL(17)000080r1_SOL003_VNF_package_management_resources - NFVSOL(17)000083_SOL003_Error_handling_LCM_interface (with the following modification by the rapporteur: replaced multiple occurrences of the copy-paste error "corresponding to the instantiation operation" by "corresponding to the operation") - NFVSOL(17)000085r1_SOL003_Flows_for_Indicator_interface - NFVSOL(17)000086r1_SOL003_Resources_of_Indicator_interface - NFVSOL(17)000087r1_SOL003_Filters_and_Selectors_for_the_LCM_interface - NFVSOL(17)000090r2_SOL003_Links_for_LCM_and_Granting_interfaces - NFVSOL(17)000091_SOL003_LifecycleChangeNotification_terminology (with the following modifications by the rapporteur: (1) use lowercase in "lifecycle management operation occurrence" if it appears in flowing text. (2) change "represented by a VNF Lifecycle Operation Occurrence" → " represented by a VNF Lifecycle Operation Occurrence resource") - NFVSOL(17)000092r2_SOL003_VNF_Fault_Management_interface_data_mod el (with the following modifications by the rapporteur: as the contribution proposes two alternatives for the request body, added the sentence "Each notification request body shall include exactly one of the alternatives defined in table 7.3.6.3.1-2" that we use in other occurrences of this pattern. - NFVSOL(17)000096_SOL003_VNF_fault_management_subscription_and_notif ication_fl - NFVSOL(17)000097r2_SOL003_VNF_package_management_data_models - NFVSOL(17)000098r2_SOL003_notification_and_filter_design_for_vr_quota_a vail_not - NFVSOL(17)000099r1_SOL003_error_codes_design_for_vr_quota_avail_notifi cation_in - NFVSOL(17)000100r1_SOL003_Data_structures_of_the_Indicator_interface - Aligned Annex A with Conventions change in NFVSOL(17)000106_Conventions_Document_NFVSOL_17_000050__Swagge r_Representatio - NFVSOL(17)000111_SOL003_Conventions_move_Resource_structure_up_in_ the_TOC Editorials: - Sorted the references. Removed the two informative references to JSON schema as they are not referenced any longer. - Various typo and numbering corrections - In FM interface, moved subscriptions resources to the end of the resources clauses to align with the structure of the other APIs - Applied those changes in NFVSOL(17)000084r1_Template_changes_for_error_handling that are editorial (note that technical changes regarding error hand ling etc. require a contribution)

Date	Version	Information about changes
		- In a number of NOTEs, replaced "softwareVersion" by "vnfSoftwareVersion". In the table, the name of the attribute to which the note refers was changed to "vnfSoftwarVersion" but that was forgotten in the NOTE. - Moved the datatype definition for VnfLcOpOcc in front of the subscription data types in order to align with the sequence of the resources
March 2017	0.8.0	Stable Draft, with a small number of noted gaps as defined in contribution NFVSOL(17)000009r9. Contributions incorporated that were agreed at NFVSOL#21 call and NFV#22 in Piscataway: - NFVSOL(17)000078r3_SOL002_SOL003_VimId_fixes (Additionally, rapporteur deleted one related editor's note in table 9.5.3.3-1 that was forgotten to be deleted by contribution 78r3) - NFVSOL(17)000118r1_SOL003_VNF_performance_management_pmJobs_flows - NFVSOL(17)000119r3_SOL003_VNF_performance_management_thresholds_flows - NFVSOL(17)000120_SOL003_VNF_performance_management_subscription_and_notification - NFVSOL(17)000124r1_SOL003_Authorization - NFVSOL(17)000125r1_SOL002_SOL003_MonitoringParameters_data_structure - NFVSOL(17)000126r1_SOL002_SOL003_NetworkAddress_data_structure - NFVSOL(17)000127r2_SOL003_Fixes_related_to_IFA_decisions_in_Bilbao_and_beyond (Additional change was performed by the rapporteur to Annex B.2 to align with the removal of "Fetch VNF Package" which corresponds to the spirit of this contribution but was forgotten) - NFVSOL(17)000128r2_SOL002_SOL003_Addressing_Rapporteur_s_and_Editor_s_notes_bat - NFVSOL(17)000139_SOL003_apply_conventions_to_vr_quota_avail_notification_flow - NFVSOL(17)000141_SOL002_SOL003_Rename_ind_to_vnfind - NFVSOL(17)000147r1_SOL002_SOL003_Definitions__symbols_and_abbreviations - NFVSOL(17)000148r1_SOL003_Clause_4_1_Overview - NFVSOL(17)000149_SOL002_SOL003_LCM_ed_note_error_handling_bugfix - NFVSOL(17)000150r1_SOL002_SOL003_Adding_description_of_rollback_retry_cancel_fa - NFVSOL(17)000151r1_SOL003_SOL002_clause_5_6_1_Basic_concepts__for_LCM_errors_ - NFVSOL(17)000152_SOL003_Move_LcmOpType_to_global_types_list - NFVSOL(17)000153r2_SOL003_PM_ThresholdCriteria - NFVSOL(17)000154r2_SOL003_PM_interface_data_model - NFVSOL(17)000155r1_SOL003_SOL002_resolve_Auto-X_editor_s_note - NFVSOL(17)000157_SOL003_SOL002_5_2_1_Fixing_Vnf_Instance_Creation_flow - NFVSOL(17)000169_SOL003_Adding_VNFD_resource_in_VNF_package_management_interf - NFVSOL(17)000170r1_SOL003_Read_VNFD_flow - NFVSOL(17)000171r1_SOL003_Add_selectors_for_VNF_package_management_interface - NFVSOL(17)000172_SOL003_Add_filters_for_VNF_package_management_interface - NFVSOL(17)000173_SOL003_Add_filters_for_vr_quota_available_notification_inter - NFVSOL(17)000174r1_SOL003_modify_data_type_of_the_artifacts - NFVSOL(17)000175r1_SOL003_define_checksum_data_type_for_the_VNF_package_managem - NFVSOL(17)000176r1_SOL003_Add_alarm_resource_to_FM_and_fix_links - NFVSOL(17)000177r2_SOL003_VNF_fault_management_related_data_types - NFVSOL(17)000178_SOL003__Correction_of_vnfdId_Data_Type_in_VnfInstance__Claus - NFVSOL(17)000180r2_SOL003_VNF_Fault_Management_Get_Alarm_List_sequence_flow - NFVSOL(17)000181r3_ SOL003_Update_FM_Resources - NFVSOL(17)000182_SOL002_SOL003_Indicators clean up

Date	Version	Information about changes
		- NFVSOL(17)000187r1_SOL002_SOL003_Conventions_global_fix_for_normative_statement - NFVSOL(17)000188_SOL002_SOL003_Notification_id - NFVSOL(17)000189r1_SOL002_SOL003_VnfInstanceSubscriptionFilter_general_data_typ - NFVSOL(17)000190_SOL002_SOL003_VnfLcOpOcc_fixes_for_ModifyVnfInfo - NFVSOL(17)000191r1_SOL002_SOL003_state_change_timestamp_and_affected_resources (but "attributeChanges was not added to the exclude-default for VnfLcOpOcc in the last change, as this attribute has been removed from the contribution in r1, but removal from that particular change was missed) - NFVSOL(17)000192_SOL002_SOL003_Remove_editor_s_note_in_5_4_3_3_4 - NFVSOL(17)000193r1_SOL003_Resumed___partial_download_of_artifacts_and_VNF_packa (and applied the change we agreed in another document for the 200 OK response for fetching a VNF package artifact also to the 206 response) - NFVSOL(17)000199_SOL002_SOL003_Renaming_attribute_selectors - NFVSOL(17)000200_SOL002_SOL003_Attribute_filter_equality - NFVSOL(17)000201_SOL003_FM_align_fault_type_and_event_type_with_IFA_contribut - NFVSOL(17)000202r1_SOL003_Annex_B_Mapping - NFVSOL(17)000203_SOL003__VNF_FM_Interface_Terminology_Clean-Ups - NFVSOL(17)000204_SOL003_error_handling__filters_and_selectors_for_VNF_perform Editorials: - A number of small typos fixed - Removed ":" after "Location" when the "Location" HTTP header is mentioned. - Reworded two awkwardly-formulated sentences about the NFVO trying to recover from an error situation, in the subscription management flows of FM and PM i/fs) - Aligned the intro text of the clauses "Referenced simple types and enumerations", and replaced empty "Referenced simple types and enumerations" clauses by references to clause 4.4. - Captions of tables that define an enumeration have been aligned with the template - Changed name of the "Remarks" column to "Description" globally - Removed template leftovers from the front matter.
April 2017	V 0.8.1	Editorial changes to prepare for clean-up by editHelp!
April 2017	V 0.8.2	Clean-up done by ***editHelp!***
May 2017	V 0.9.0	Contributions incorporated that were agreed at NFVSOL#23 and NFVSOL#25 calls and in email approval round #1: - NFVSOL(17)000209r1_SOL002_SOL003_SOL005_all_fields__and_defaults_for_selectors - NFVSOL(17)000225r2_SOL003_clause_6_Fetching_of_Performance_reports - NFVSOL(17)000226r1_SOL003_global_Harmonize_Query_vs_Read - NFVSOL(17)000237r1_SOL003_6_5_2_4_EN_add__id_to_ThresholdCrossedNotification - NFVSOL(17)000238_SOL003_6_5_2_5_EN_add__id_to_PerformanceInformationAvailable - NFVSOL(17)000242_SOL003_10_4_5_3_2_and_10_4_6_3_2_EN_Non-support_of_range_req - NFVSOL(17)000245r2_SOL002_SOL003_5_x_harmonization_of_VnfLcOpOcc_related_identi - NFVSOL(17)000247_SOL003_B_7_small_bugfix - NFVSOL(17)000258r1_SOL003_5_6_Better_name_for_unresolvable_fail - NFVSOL(17)000257_SOL003_5_3_vnfInstanceId_in_LCM_flow_diagrams (applies to ModifyVnfInfo) - NFVSOL(17)000260r4_SOL003_global_editorial_and_minor_technical_changes - NFVSOL(17)000294_SOL003_4_5_2_3_Clarification_of_two_access_token_usage (Rapporteur: and added "#2" in the text in a few additional places to align with the figure) - NFVSOL(17)000295_SOL003_4_5_2_3_flow_and_step_consistency - NFVSOL(17)000297_SOL003_5_3_12_Missing_error_handling_description - NFVSOL(17)000304_SOL003_removing_status_from_SoftwareImageInformation - NFVSOL(17)000305_SOL003_4_1_Overview_Consistency_Fix

Date	Version	Information about changes
		- NFVSOL(17)000309_SOL003_4_5_2_Authorization_flow_description_inconsistency - NFVSOL(17)000315_SOL003_4_5_2_fixing_HTTP_method_in_OAuth_2_0_flows - NFVSOL(17)000317_SOL003_5_5_2_13_operationType_in_VnfLcOpOcc (Rapporteur: In table 5.5.2.13-1, it was missed to delete "Type" from "operationType"; all other occurrences use "operation" not "operationType") - NFVSOL(17)000321_SOL003__9_5_2_3_Change_VimInfo_data_type_to_VimConnectionInf - NFVSOL(17)000323_SOL003_Clause_7_5_4_3_Change_the_description_of_PerceivedSev - NFVSOL(17)000329_SOL003_global_ignore_unknown_attributes - NFVSOL(17)000336r1_SOL003_many_Fixing_some_conditions (Rapporteur: and aligned the new text with 245r2, i.e. "VNF LCM operation occurrence" instead of "operation") Contributions incorporated that were agreed at NFVSOL#26 and NFVSOL#27 calls, in email approval rounds #2 and #3, and in SOL#28 F2F: - NFVSOL(17)000217r4_SOL003_VNF_FM_Acknowledge_Alarm_operation - NFVSOL(17)000218r1 - NFVSOL(17)000227r1_SOL003_4_3_2_2_EN_Attribute_filters_point_in_time - NFVSOL(17)000228r1_SOL003_4_4_1_3_EN_for_KeyValuePair___Object - NFVSOL(17)000229r1_SOL003_10_5_3_2_VnfPackageSoftwareImageInfo_fixes - NFVSOL(17)000230r3_SOL003_global_Apply_conventions_for_identifiers - NFVSOL(17)000231r1_SOL003_5_5_3_5_ENs_VimConnectionInfo_registry - NFVSOL(17)000232_SOL003_5_5_3_7_rapp_note__information_element - NFVSOL(17)000233r2_SOL003_5_5_3_7_to_5_5_3_9_resource_metadata - NFVSOL(17)000234r2_SOL003_5_6_2_3_EN_regarding_forceful_vs_graceful_cancel - NFVSOL(17)000235_SOL003_6_3_3_rapporteur_s_notes_regarding_PM_flow - NFVSOL(17)000236_SOL003_6_3_6_rapporteur_s_note_regarding_PM_flow_bugs - NFVSOL(17)000240_SOL003_9_5_3_3_EN_in_table_for_GrantInfo - NFVSOL(17)000241_SOL003_9_5_2_2_and_9_5_2_3_EN_and_rapporteurs_note - NFVSOL(17)000244_SOL003_5_x_Problem_with_Error_handling_ModifyVnfInfo - NFVSOL(17)000246r1_SOL003_5_4_16_Making_the_Fail_operation_synchronous - NFVSOL(17)000256r1_SOL003_4_4_1_3_VnfInstanceSubscriptionFilter_fixes - NFVSOL(17)000261r1_SOL003_10_3_3_and_10_4_5_3_2_ZIP_as_content_type_for_VNF_Pac - NFVSOL(17)000262_SOL003_9_4_3_Deletion_of_grants - NFVSOL(17)000280r2_SOL003_-_Clause_4_4_2_-_5_5_3_-_MAC_and_IP_address_represent (Rapporteur: renamed IPAddress to IpAddress to follow conventions) - NFVSOL(17)000282_SOL003_-_Removing_normative_dependencies_on_SOL001 - NFVSOL(17)000291_SOL003_Clause_4_4_2_simple_data_type_editor_note - NFVSOL(17)000293r1_SOL003_4_2_Consistency_of_URI_and_OAuth - NFVSOL(17)000298r2_SOL003_5_4_3_2_Improvement_of_resource_definition_descriptio - NFVSOL(17)000299r1_SOL003_5_4_13_2_Improvement_of_resource_definition_descripti - NFVSOL(17)000300_SOL003_4_4_2_Clear_meaning_of_IdentifierLocal_type - NFVSOL(17)000301r1_SOL003_5_4_16_Add_the_supplement_to_Finally_Failed - NFVSOL(17)000302r1_SOL003_4_3_5_5_Consistency_between_4_3_5_4_and_4_3_5_5 - NFVSOL(17)000303r3_SOL003_-_Editorial_changes - NFVSOL(17)000306r1_SOL003_4_3_2_2_Filter_Spec_Fix - NFVSOL(17)000307r2_SOL003_4_3_4_3_Adding_retry-after_header_field - NFVSOL(17)000308r1_SOL003_4_3_5_Bug_fix_and_modify_title - NFVSOL(17)000310r1_SOL003_4_5_3_Small_bug_fixes (rapporteur also accordingly changed other places that mention that " Link to the subscription that triggered the notification" to "Link to the related subscription")

Date	Version	Information about changes
		- NFVSOL(17)000311r1_SOL003_5_3_9_Retry_flow_error_handling_description_modificat - NFVSOL(17)000313_SOL003_5_4_12_3_2_Small_bug_fix - NFVSOL(17)000314_SOL003_5_5_2_2_modify_the_description_for_VnfInstance - NFVSOL(17)000316_SOL003_5_5_2_8_Graceful_Termination_clarification - NFVSOL(17)000318r2_SOL003_5_5_2_16_LCM_Notifications_race_condition - NFVSOL(17)000322r1_SOL003__Various_Clauses__Consistency_in_the_naming_of_the_ad - NFVSOL(17)000326_SOL003_6_5_3_2_PmFilter - NFVSOL(17)000328r4_SOL003_5_5_Making_the_relationship_between_vnflcOpOcc_and_no - NFVSOL(17)000330_SOL003_9_5_2_3_and_9_5_3_3_Grant_fixes - NFVSOL(17)000332r2_SOL003_5_5_3_17_Problem_with_storage_resources_in_AffectedVn - NFVSOL(17)000333r3_SOL003_5_5_3_15_wrong_use_of_Object_in_MonitoringParameter - NFVSOL(17)000334r1_SOL003_global_Subscribe_Notify_error_handling - NFVSOL(17)000335r1_SOL003_4_3_3_2_1_Fixes_to_attribute_selector - NFVSOL(17)000338_SOL003_5_5_3_5_Protecting_credentials_in_VimConnectionInfo - NFVSOL(17)000339_SOL003_5_4_2_3_2_Adding_vimConnections_to_exclude_default - NFVSOL(17)000341_SOL003_7_5_2_5_Refactoring_links_in_AlarmNotification - NFVSOL(17)000342_SOL002_SOL003_4_4_2_and_10_5_3_2_EN_on_Version_and_unsignedI - NFVSOL(17)000343_SOL002_SOL003_global_Renaming_of_attribute_filters_to_attrib. - NFVSOL(17)000347r4_SOL003_Annex__X___VIM_Registry - NFVSOL(17)000349r2_SOL003_SOL002_5_x_Error_code_404_if_task_resource_not_suppor - NFVSOL(17)000352r1_SOL003_7_5_3_4_Update_data_type_FaultyResourceInfo - NFVSOL(17)000354r3_SOL003_SOL002_autoscale_autohel_description - NFVSOL(17)000355_SOL003_SOL002_Replace_entity_body_by_payload_body - NFVSOL(17)000363_SOL002_SOL003_global_consistency_of_enum_type_names - NFVSOL(17)000368_SOL002_SOL003_4_3_3_1_adding_informative_to_overview_and_exa - NFVSOL(17)000374_SOL003_SOL002_Notification_Authorization_future_proofing - NFVSOL(17)000375r1_SOL003_SOL002_Explanation_of_OperateVnf - NFVSOL(17)000384_SOL003_SOL002_Refactoring_VNF_Instance_link_in_AlarmNotifica (This is a strict subset of 341) - NFVSOL(17)000389r2_SOL003_10_4_4_Fetch_the_VNFD - NFVSOL(17)000393_SOL002_SOL003_Move_VimConnectionInfo_to_the_correct_clause - NFVSOL(17)000394_SOL002_SOL003_address_comments_from_Procera_Networks - NFVSOL(17)000396r2_SOL002_SOL003_6_4_7_4_8_4_9_4_10_4_11_4_Addition_of_the_note - NFVSOL(17)000397_SOL002_SOL003_5_3_3_Clarification_of_notification_flow_in_th - NFVSOL(17)000399_SOL003_SOL002_changedExtVLs_in_LcmOpOccNotif Editorials: - Put "Notify" always as last row in the tables in all clauses of Annex B - Enforced naming convention of using plural for attributes of cardinality 0..N
June 2017	V 0.10.0	Contributions incorporated that were approved at NFVSOL#29 call: - NFVSOL(17)000337r1_SOL003_5_5_2_x_additionalParameters_missing_from_TerminateVn - NFVSOL(17)000387r2_SOL003_SOL002_EN_Scaling_explanation - NFVSOL(17)000403r3_SOL003 ExtCP ExtVL fixes related to IFA discussion - NFVSOL(17)000405r2_SOL003_SOL002_one_change_pulled_out_from_349r1 - NFVSOL(17)000408r3_SOL003_rapporteur_s_cleanup_of_V_0_9_0

Date	Version	Information about changes
		- NFVSOL(17)000414r3_SOL003_issue__resourceGroupId_in_Grant_interface - NFVSOL(17)000416_SOL003__Remove_Swagger_Annex_A - NFVSOL(17)000417_SOL003_SOL002_copying_over_a_note_from_IFA007 Editorials: - Fixing awkward "(s)" plural where applicable: (1) use plural where appropriate e.g. for arrays resource(s) → resources, (2) use singular where appropriate, e.g. if there's only a single entry such as notification about quota(s) → quota, (3) use alternatives where there are two distinct possibilities, e.g. VIM interface or VIM interfaces - Removed editor's notes not addressed by contributions, apart from the one in the registry
July 2017	V 0.10.1	- Editorial changes - Fixing the implementation of document NFVSOL(17)000199_SOL002_SOL003_Renaming_attribute_selectors
November 2017	V 2.3.2	Contributions incorporated that were approved at NFVSOL#40 call, NFVSOL#41 F2F and subsequent email approval: - NFVSOL(17)000565_SOL002_SOL003_Fixing_actors_in_authorization_flows - NFVSOL(17)000592r4_SOL003_miscellaneous_bugfixes - NFVSOL(17)000593r1_SOL002_SOL003_miscellaneous_small_bugfixes - NFVSOL(17)000594r1_SOL002_SOL003_Adding_405_response - NFVSOL(17)000596r1_SOL003_Adding_statement_for_mandatory_and_optional_HTTP_meth - NFVSOL(17)000600_SOL003_IFA027_reference - NFVSOL(17)000602_SOL003_Get_Alarm_List_query_fix - NFVSOL(17)000629_SOL003_VR_Quota_Avail_Notification_Trigger_Condition Editorials: - Added draft disclaimer box

Date	Version	Information about changes
November 2017	V 2.3.3	Contributions incorporated that were approved at NFVSOL#42 - NFVSOL#44 calls: - NFVSOL(17)000634_SOL003_VNFC_CP_changes_in_AffectedVnfc - NFVSOL(17)000667r2_SOL002_SOL003_Add_description_to_VNF_fault_management_interf - NFVSOL(17)000668_SOL002_SOL003_complement_the_description_of_CancelModeType - NFVSOL(17)000670r2_SOL002_SOL003_Fixing_statement_for_mandatory_and_conditional - NFVSOL(17)000691r1_SOL002_SOL003__Use_of_verbal_forms_for_the_expression_of_pr
December 2017	V 2.3.4	Contributions incorporated that were approved at NFVSOL#45 F2F, the subsequent email approval and the NFVSOL#46 and NFVSOL#47 calls: - NFVSOL(17)000388r6_SOL002_ed2_4_1_and_SOL003_ed_2_4_1__Authorization_of_API_Req - NFVSOL(17)000635r1_SOL002_SOL003_implicit_changes_in_VnfInfoModifications - NFVSOL(17)000646_SOL002_SOL003_Add_resource_metadata_to_AffectedVnfc_VirtualL - NFVSOL(17)000654r1_SOL003_Align_PkgmNotificationsFilter_with_VnfInstanceSubscri - NFVSOL(17)000666r1_SOL003_Refactoring_VNF_package_management_interface_to_align - NFVSOL(17)000669r3_SOL002_SOL003_Add_note_to_clarify_how_timeout_of_the_cancel_ - NFVSOL(17)000671r2_SOL002_SOL003_ExtCpData_changes_from_IFA1114r1_and_IFA1037r4 (in clause 5.5.3.5, rapporteur has changed in the Description column "vnfLinkPortInfo" to "vnfLinkPorts" to reflect the correct attribute name ("VnfLinkPortInfo" is the type, "vnfLinkPorts" is the attribute) - NFVSOL(17)000674r4_SOL002_SOL003_Authorization_method_negotiation - NFVSOL(17)000690r1_SOL003_5_5_2_10_Add_note_for_the_presence_condition_of_attri - NFVSOL(17)000695r1_SOL002_SOL003_Fixing_normative_status_of_notification_endpoi - NFVSOL(17)000698_SOL002_SOL003_allow_Fail_operation_in_FAILED_TEMP - NFVSOL(17)000706r1_SOL003_Solve_the_inconsistency_of_password_transmission - NFVSOL(17)000715r2_SOL002_SOL003_-_Double_Subscriptions_for_Notifications - NFVSOL(17)000718r2_SOL002_SOL003_sequence_of_responses_and_notifications - NFVSOL(17)000723r1_SOL003_fixing_VNF_connectivity_figure - NFVSOL(17)000730r1_SOL003_Fix_description_of_unsupported_method_for_notificatio - NFVSOL(17)000733_SOL003_align_normative_statements_in_resource_tables - NFVSOL(17)000734r1_SOL003_align_normative_statements_in_trigger_conditions - NFVSOL(17)000739_SOL003_VimConnectionInfo_fix_the_word_needed - NFVSOL(17)000749_SOL003_Add_error_code_for_fetching_package_content_or_artifa - NFVSOL(17)000754_SOL002_SOL003_Remove_redundant_description_of_vnfConfigurabl - NFVSOL(17)000757_SOL003_Add_clarification_of_ExManagedVirtualLink_and_ExtVirt - NFVSOL(17)000768_SOL003_Annex_A_2_Operation_Name_and_Resource_URI_Alignment - NFVSOL(17)000775_SOL003_VnfPkgm_undoing_the_timestamp_change_from_CR_666r1 - NFVSOL(17)000780r1_SOL003_Fixing_leftovers_of_onboardedVnfPkg_Info_Id - NFVSOL(17)000789r2_Additional_modifications_for_SOL002_and_SOL003 Some editorial fixes were applied.
February 2018	2.3.5	Incorporated change request: - NFVSOL(18)000022_SOL003_Temporarily_remove_IFA027_reference
February 2018	2.4.1	Publication

Date	Version	Information about changes
March 2018	2.4.2	Contributions incorporated that were approved at NFVSOL#55 F2F: - NFVSOL(18)000037r1_SOL003_API_Authorization_clarification - NFVSOL(18)000058r2_SOL003ed251_Making_Authz_negotiation_more_flexible - NFVSOL(18)000060r2_SOL003ed251_disambiguating_artifactPath - NFVSOL(18)000091r2_SOL003ed251_evolving_the_registry_annex_part_1 - NFVSOL(18)000081r1_SOL003ed251_empty_collections_clarification_adressing_Plugte
March 2018	2.4.3	Contributions incorporated that were approved at NFVSOL#57 call: - NFVSOL(18)000095r1_SOL003ed251_evolving_the_registry_annex_part_2
May 2018	2.4.4	Contributions incorporated that were approved from NFVSOL#58 call to NFVSOL#64 F2F: - NFVSOL(18)000127r1_SOL003ed251_fixing_tracker_issues_0007747_and_00077478 - NFVSOL(18)000059r1_SOL003_Updating_JSON_RFC_reference - NFVSOL(18)000131r2_SOL003ed251__Fix_cardinality_of_the_operationParams_attribut - NFVSOL(18)000153r6_SOL003_Version_Management (Rapporteur has slightly changed the statement "and API version retrieval" added to 5.2 etc. to read "and API version information retrieval", as we are not retrieving API versions, but API version information.) - NFVSOL(18)000181r2_SOL003ed251__Change_the_cardinality_of_the_subscriptionId_at - NFVSOL(18)000210r1_SOL003_Attribute_selectors - NFVSOL(18)000214r2_SOL003_Adding_status_codes - NFVSOL(18)000215_SOL003_Aligning_operation_type_enums_with_IFA007 - NFVSOL(18)000216r1_SOL003_MAC_address_optional_in_IpOverEthernetAddressInfo - NFVSOL(18)000217r1_SOL003_Small_fixes - NFVSOL(18)000218_SOL003_SOL005_VnfPkgm_small_fix - NFVSOL(18)000219_SOL003_fix_for_the_enhanced_patch_rules - NFVSOL(18)000220r1_SOL003_enhanced_patch_rules_-_deletion_of_array_entries - NFVSOL(18)000224r2_SOL003_Fixing_the_sequence_of_400_response_code_definitions - NFVSOL(18)000226r1_SOL003_different_names_for_virtual_link_descriptor_ids - NFVSOL(18)000227_SOL003_Granting_policies_from_IFA007 - NFVSOL(18)000234_SOL003_Fixing_cardinality_of_ConstraintResourceRef Editorials - TRUE -> true consistently - Removed smart quotes
June 2018	2.4.5	Contributions incorporated that were approved at NFVSOL#66 F2F: - NFVSOL(18)000053r2_SOL003ed251_Bring_back_IFA027_reference - NFVSOL(18)000209r2_SOL003_Attribute_filters - NFVSOL(18)000212r1_SOL003_Normative_attribute_filters_support (in addition to implementing the tracked changes in the CR, the rapporteur also applied one missed change in 6.4.2.3.2 that is part of the pattern: adding "by the VNFM in the filter expression" to the last sentence of the description of the "filter" parameter). - NFVSOL(18)000213r2_SOL003_Support_for_links_in_notifications - NFVSOL(18)000221_SOL003_metadata_for_CP_IEs - NFVSOL(18)000250_SOL003_small_fix_replace_queried_by_read - NFVSOL(18)000257_SOL003ed251__Remove_the_current_values_of_the_MonitoringPara
July 2018	2.4.6	Contributions incorporated that were approved in EA following NFVSOL#66 F2F: - NFVSOL(18)000309_SOL003_Define_Number_and_String_data_types - NFVSOL(18)000241r2_SOL003_Changes_from_IFA_CRs_412r2_and_411r1 Editorials: - Replaced "present specification" by "present document" - Fixed missing comma in table 4.3.2.2-1

Date	Version	Information about changes
July 2018	2.4.7	Contributions incorporated: - NFVSOL(18)000316r2_SOL003_Define_Minor_version_number - NFVSOL(18)000317_SOL003_Retry_as_reaction_to_error_responses_during_notificat - NFVSOL(18)000337_SOL003_attribute_selector_attribute_filter_small_fixes - NFVSOL(18)000339_SOL003_mirror_of_332_-_Add_annex_with_a_reference_to_OpenAPI - NFVSOL(18)000345_SOL003_registry_link_update - NFVSOL(18)000434r1_SOL003_remaining_ENs_resolution
August 2018	2.4.8	Draft as input for approval process towards publication as 2.5.1. Contributions incorporated: - NFVSOL(18)000418r2_SOL003_closing_pagination_gap - NFVSOL(18)000441_SOL003_Define_Patch_version_number - NFVSOL(18)000437r2_SOL003_VIM_registration - NFVSOL(18)000237r5_SOL003_-_API_Version_Identification o Rapporteur has applied editorial changes on top of this CR, including deletion of the trailing phrase "of the previous version (see clause 4.6.2.2)" in the second paragraph of clause 4.6.2.1 which did not fit to the sentence and is considered an editing leftover. - NFVSOL(18)000343r4_SOL003_version_signalling - NFVSOL(18)000471r1_SOL003ed251__Add_note_to_MAJOR_version_field Editorials: - Use "annex" (lowercase) consistently.

History

Document history		
V2.3.1	July 2017	Publication
V2.4.1	February 2018	Publication
V2.5.1	September 2018	Publication