swagger /esc-etsi-api Explore

ESC ETSI API 4.0.0 OAS3

/esc-etsi-api

Documentation :

ETSI-MANO REST Northbound API

This new REST API is another programmatic interface to ESC that uses a REST architecture. The API accepts and returns HTTP or HTTPS messages that contain JavaScript Object Notation (JSON).

It is the payloads for these request/responses that a defined by the European Telecommunications Standards Institute (ETSI), specifically around Management and Orchestration (MANO). It contains its own data model, designed around the ETSI-MANO specification (ETSI GS NFV-SOL 003 V2.3.1), that abstracts away from the ESC core data model.

This initial implementation of the ETSI-MANO standards for NFV is to address the Or-Vnfm reference point, i.e. the interface between the Network Function Virtualisation Orchestrator (NFVO) and the Virtual Network Function Manager (VNFM).

The Or-Vnfm reference point to details the interactions to onboard ETSI-compliant VNF packages, manage resources, and VNF lifecycle management (LCM) operations.

During the lifespan of a VNF Instance, it moves between INSTANTIATED and NOT_INSTANTIATED states, whereas operations that perform LCM operations have a more complex state machine, as per the diagram below.

The ETSI-MANO specification considers provisioning of many components of a network service outside the remit of the VNFM, namely:

- Tenants
- Images
- Flavours
- External Networks/Virtual Link
- Externally Managed Internal Virtual Link
- Subnets

This means that LCM operations on an instance of a VNF submitted to the ETSI-MANO REST API expect these resources to be created out-of-band (OOB) as far as the VNFM is concerned. It is likely that these resources are created via the NVFO, either at the time of onboarding the VNF package or onboarding the tenant, and will be represented by VIM (Virtual Infrastructure Manager) identifiers in the request to ESC.

Managing Resources

Managing Resources via the ETSI-MANO API The ETSI-MANO API communicates with NFVO for lifecycle management. A configuration template, the Virtual Network Function Descriptor (VNFD) file describes the deployment parameters and operational behaviors of a VNF type. The VNFD is used in

the process of deploying a VNF and managing the lifecycle of a VNF instance. The flow of operations to deploy a VNF instance is:

1. Create VNF Identifier
2. Instantiate VNF The flow of operations to fully undeploy (and release resources used by a VNF instance) is:
3. Terminate VNF
4. Delete VNF Identifier

The other LCM operations are applicable once the VNF has been instantiated, except from Query which is applicable at any time since it does not modify the VNF.

LCM Operations

Here is an overview of the operations that can affect a VNF instance.

- **Create VNF Identifier**: Generate a new VNF Instance Id (a universally unique identifier) that is subsequently used as a handle to reference the instance upon which to execute further operations.
- **Instantiate VNF**: Deploy a new VNF instance in the VIM. The Instantiate request will contain instance-specific values and this, coupled with the VNFD and the Grant information will provide all the information required by the VIM to deploy the VNF. The VNFD is retrieved from the NFVO as part of this call flow which provides the resource requirements for the VNF to be instantiated. This data set is then further supplemented by requesting permission from the NFVO to continue with the request which returns Grant information that converts some of these resource requirements to actual resources that are reserved in the VIM.
- **Operate VNF**: Allow a VNF instance to be started or stopped. The resources are not released or changed, but the VNF instance in the VIM is toggled between these two states.
- **Query VNF**: Query one or more VNF instances known to ESC. This is a specific REST endpoint that can be filtered to find specific instances. In this initial release, the instances can be filtered by the VNF Instance Id.
- **Terminate VNF**: Undeploy the VNF instance in the VIM. The resources themselves remain reserved for the VNF instance, however the VNF itself is undeployed.
- **Delete VNF Identifier**: The resources are fully released in the VIM and in ESC and the associated VNF instance identifer is also released.

Server

`/vnflcm/v1`

vnf_instances

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

`GET` `/vnf_instances` Query multiple VNF instances

POST	/vnf_instances	Create a VNF Instance resource
GET	/vnf_instances/{vnfInstanceId}	Read an individual VNF resource
DELETE	/vnf_instances/{vnfInstanceId}	Delete a VNF instance resource
POST	/vnf_instances/{vnfInstanceId}/instantiate	Instantiate a VNF
POST	/vnf_instances/{vnfInstanceId}/operate	Operate a VNF Instance
POST	/vnf_instances/{vnfInstanceId}/terminate	Terminate a VNF Instance

vnf_lcm_op_occs

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.

GET	/vnf_lcm_op_occs	Query multiple VNF lifecycle management operation occurrences
GET	/vnf_lcm_op_occs/{vnfLcmOpOccId}	Read an individual VNF lifecycle management operation occurrence
POST	/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail	Mark a VNF lifecycle management operation occurrence as failed
POST	/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback	Rollback a VNF lifecycle management operation occurrence
POST	/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel	Cancel a VNF lifecycle management operation occurrence

Models

Link {

description:	*This type represents a link to a resource.*
href*	string($uri) URI of the referenced resource.

}

KeyValuePairs {

description: *This type represents a list of key-value pairs. The order of the pairs in the list is not significant.*

}

VnfInstanceSubscriptionFilter {

description: *This type represents subscription filter criteria to match VNF instances.*

vnfdIds [...]

vnfProductsFromProviders [...]

vnfInstanceIds [...]

vnfInstanceNames [...]

}

VimConnectionInfo {

description: *This type represents parameters needed to connect to a VIM for managing the resources of a VNF instance.*

id* string($uuid)
The identifier of the VIM Connection. This identifier is managed by the NFVO.

vimId string($uuid)
The identifier of the VIM instance. This identifier is managed by the NFVO.

vimType* string
Discriminator for the different types of the VIM information.

interfaceInfo KeyValuePairs {...}

accessInfo KeyValuePairs {...}

extra KeyValuePairs {...}

}

ResourceHandle {

description: *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.*

vimConnectionId string($uuid)
Identifier of the VIM connection to manage the resource.

resourceProviderId string($uuid)
Identifier of the entity responsible for the management of the resource.

resourceId* string($uuid)
Identifier of the resource in the scope of the VIM or the resource provider.

vimLevelResourceType string
Type of the resource in the scope of the VIM or the resource provider.

}

FixedNetworkAddressData {

description: *This type represents a network address that is requested to be assigned.*

macAddress MacAddress string
ipAddress IpAddress string
subnetId string
Identifier of the subnet in the VIM. This attribute may be present if the "ipAddress" attribute is present, and shall be absent otherwise.

}

DynamicNetworkAddressData {

description: *This type represents a network address that is requested to be assigned.*

macAddress MacAddress string
numIpAddresses* integer($int32)
Number of IP addresses to assign dynamically. Shall be greater than zero.

subnetId string
Subnet defined by the identifier of the subnet resource in the VIM. In case this attribute is present, an IP addresses from that subnet will be assigned; otherwise, IP addresses not bound to a subnet will be assigned.

subnetIpRanges [...]

}

VnfExtCpData {

description: *This type represents an external CP.*

cpdId* string($uuid)
The identifier of the CPD in the VNFD.

fixedAddresses [...]
dynamicAddresses [...]

}

ExtVirtualLinkData {

description: *This type represents an external VL.*

id* string($uuid)
The identifier of the external VL instance.

vimConnectionId string($uuid)
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 string($uuid)
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.

resourceId* string($uuid)
The identifier of the resource in the scope of the VIM or the resource provider.

extCps [...]

}

ExtManagedVirtualLinkData {

description: *This type represents an externally-managed internal VL.*

id* string($uuid)
The identifier of the externally-managed internal VL instance.

virtualLinkDescId* string($uuid)
The identifier of the VLD in the VNFD for this VL.

vimConnectionId string($uuid)
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 string($uuid)
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.

resourceId* string($uuid)
The identifier of the resource in the scope of the VIM or the resource provider.

}

LcmOperationType string

The enumeration LcmOperationType represents those lifecycle operations that trigger a VNF lifecycle management operation occurrence notification.

Enum:

Array [9]

VnfInstance {

description: *This type represents a VNF instance.*

id* string($uuid)
Identifier of the VNF instance.

vnfInstanceName string
Name of the VNF instance.

vnfInstanceDescription string
Human-readable description of the VNF instance.

vnfdId* string($uuid)
Identifier of the VNFD on which the VNF instance is based.

vnfProvider* string
Provider of the VNF and the VNFD. The value is copied from the VNFD.

vnfProductName* string
Name to identify the VNF Product. The value is copied from the VNFD.

vnfSoftwareVersion* string
Software version of the VNF. The value is copied from the VNFD.

vnfdVersion* string

Identifies the version of the VNFD. The value is copied from the VNFD.

onboardedVnfPkgInfoId* string($uuid)
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.

vnfConfigurableProperties KeyValuePairs {...}

vimConnectionInfo [...]

instantiationState* string
The instantiation state of the VNF.

Enum:

Array [2]

instantiatedVnfInfo {...}

metadata KeyValuePairs {...}

extensions KeyValuePairs {...}

_links* {...}

}

CreateVnfRequest {

description: *This type represents request parameters for the "Create VNF identifier" operation.*

vnfdId* string($uuid)
Identifier that identifies the VNFD which defines the VNF instance to be created.

vnfInstanceName string
Human-readable name of the VNF instance to be created.

vnfInstanceDescription string
Human-readable description of the VNF instance to be created.

}

InstantiateVnfRequest {

description: *This type represents request parameters for the "Instantiate VNF" operation.*

flavourId* string($uuid)
Identifier of the VNF deployment flavour to be instantiated.

instantiationLevelId string($uuid)
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 [...]

extManagedVirtualLinks [...]

vimConnectionInfo [...]

localizationLanguage string
Localization language of the VNF to be instantiated.

additionalParams KeyValuePairs {...}

}

ScaleVnfRequest {

description: *This type represents request parameters for the "Scale VNF" operation.*

type* `string`
Indicates the type of the scale operation requested.

Enum:

Array [2]

aspectId* `string($uuid)`
Identifier of the scaling aspect.

numberOfSteps `integer($int32)`
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 {...}

}

ScaleVnfToLevelRequest {

description: *This type represents request parameters for the "Scale VNF to Level" operation.*

instantiationLevelId `string($uuid)`
Identifier of the target instantiation level of the current deployment flavour to which the VNF is requested to be scaled.

scaleInfo [...]

additionalParams KeyValuePairs {...}

}

ChangeVnfFlavourRequest {

description: *This type represents request parameters for the "Change VNF flavour" operation.*

newFlavourId* `string($uuid)`
Identifier of the VNF deployment flavour to be instantiated.

instantiationLevelId `string($uuid)`
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 [...]

extManagedVirtualLinks [...]

vimConnectionInfo [...]

additionalParams KeyValuePairs {...}

}

TerminateVnfRequest {

description: *This type represents request parameters for the "Terminate VNF" operation.*

terminationType* `string`
Indicates whether forceful or graceful termination is requested.

Enum:

Array [2]

gracefulTerminationTimeout integer($int32)
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.

additionalParams KeyValuePairs {...}

}

HealVnfRequest {

description: *This type represents request parameters for the "Heal VNF" operation.*

cause string
Indicates the reason why a healing procedure is required.

additionalParams KeyValuePairs {...}

}

OperateVnfRequest {

description: *This type represents request parameters for the "Operate VNF" operation.*

changeStateTo* VnfOperationalStateType string
Enum:

Array [2]

stopType StopType string
Enum:

Array [2]

gracefulStopTimeout integer($int32)
The time interval (in seconds) to wait for the VNF to be taken out of service during graceful stop, before stopping the VNF. Ignored if changeStateTo=STARTED.

additionalParams KeyValuePairs {...}

}

ChangeExtVnfConnectivityRequest {

description: *This type represents request parameters for the "Change external VNF connectivity" operation to modify the external connectivity of a VNF instance.*

extVirtualLinks
[...]

vimConnectionInfo
[...]

additionalParams KeyValuePairs {...}

}

VnfInfoModifications {

description: *This type represents attribute modifications for an "Individual VNF instance" resource, i.e. modifications to a resource representation based on the "VnfInstance" data type.*

vnfInstanceName string
New value of the "vnfInstanceName" attribute in "VnfInstance", or "null" to remove the attribute.

vnfInstanceDescription string
New value of the "vnfInstanceDescription" attribute in "VnfInstance", or "null" to remove the attribute.

onboardedVnfPkgInfoId string($uuid)
New value of the "onboardedVnfPkgInfoId" attribute in "VnfInstance". The value "null" is not permitted.

vnfConfigurableProperties KeyValuePairs {...}

metadata KeyValuePairs {...}

extensions KeyValuePairs {...}

vimConnectionInfo [...]

}

VnfLcmOpOcc {

description: *This type represents a VNF lifecycle management operation occurrence.*

id* string($uuid)
Identifier of this VNF lifecycle management operation occurrence.

operationState* LcmOperationStateType string
Enum:

Array [7]

stateEnteredTime* string($date-time)
Date-time when the current state was entered.

startTime* string($date-time)
Date-time of the start of the operation.

vnfInstanceId* string($uuid)
Identifier of the VNF instance to which the operation applies.

grantId string($uuid)
Identifier of the grant related to this VNF LCM operation occurrence, if such grant exists.

operation* LcmOperationType string
The enumeration LcmOperationType represents those lifecycle operations that trigger a VNF lifecycle management operation occurrence notification.

Enum:

Array [9]

isAutomaticInvocation* boolean
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* {...}

isCancelPending* boolean
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 string

Enum:

Array [2]

error ProblemDetails {...}

resourceChanges {...}

changedInfo VnfInfoModifications {...}

changedExtConnectivity [...]

_links* {...}

}

CancelMode {

description: *This type represents a parameter to select the mode of cancelling an ongoing VNF LCM operation occurrence.*

cancelMode* CancelModeType string
Enum:

Array [2]

}

LccnSubscriptionRequest {

description: *This type represents a subscription request related to notifications about VNF lifecycle changes.*

filter LifecycleChangeNotificationsFilter {...}

callbackUri* string($uri)
The URI of the endpoint to send the notification to.

authentication SubscriptionAuthentication {...}

}

SubscriptionAuthentication {

description: *A data structure that defines the authorization requirements.*

authType* string
Defines the type of Authentication / Authorization to use when sending a notification.

Enum:

Array [2]

paramsBasic {...}

paramsOauth2ClientCredentials {...}

}

LccnSubscription {

description: *This type represents a subscription related to notifications about VNF lifecycle changes.*

id* string($uuid)
Identifier of this subscription resource.

filter LifecycleChangeNotificationsFilter {...}

callbackUri* string($uri)
The URI of the endpoint to send the notification to.

_links* {...}

}

VnfLcmOperationOccurrenceNotification {

description: *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.*

id* string($uuid)
Identifier of this notification

notificationType* string
Discriminator for the different notification types.

subscriptionId string($uuid)
Identifier of the subscription that this notification relates to.

timeStamp* string($date-time)
Date-time of the generation of the notification.

notificationStatus* string
Indicates whether this notification reports about the start of a lifecycle operation or the result of a lifecycle operation.

Enum:

Array [2]

operationState* LcmOperationStateType string
Enum:

Array [7]

vnfInstanceId* string($uuid)
The identifier of the VNF instance affected

operation* LcmOperationType string
The enumeration LcmOperationType represents those lifecycle operations that trigger a VNF lifecycle management operation occurrence notification.

Enum:

Array [9]

isAutomaticInvocation* string($boolean)
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).

vnfLcmOpOccId* string($uuid)
The identifier of the VNF lifecycle management operation occurrence associated to the notification.

affectedVnfcs [...]

affectedVirtualLinks [...]

affectedVirtualStorages [...]

changedInfo VnfInfoModifications {...}

changedExtConnectivity [...]

error [...]

_links* LccnLinks {...}

}

VnfIdentifierCreationNotification {

description:	*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*
id*	string($uuid) Identifier of this notification
notificationType*	string Discriminator for the different notification types.
subscriptionId	string($uuid) Identifier of the subscription that this notification relates to.
timeStamp*	string($date-time) Date-time of the generation of the notification.
vnfInstanceId*	string($uuid) The created VNF instance identifier
_links*	LccnLinks {...}

}

VnfIdentifierDeletionNotification {

description:	*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.*
id*	string($uuid) Identifier of this notification
notificationType*	string Discriminator for the different notification types.
subscriptionId	string($uuid) Identifier of the subscription that this notification relates to.
timeStamp*	string($date-time) Date-time of the generation of the notification.
vnfInstanceId*	string($uuid) The deleted VNF instance identifier
_links*	LccnLinks {...}

}

ExtVirtualLinkInfo {

description:	*This type represents information about an external VL.*
id*	string($uuid) Identifier of the external VL and the related external VL information instance
resourceHandle*	ResourceHandle {...}
linkPorts	[...]

}

ExtManagedVirtualLinkInfo {

description: *This type provides information about an externally-managed virtual link.*

id* string($uuid)
Identifier of the externally-managed internal VL and the related externally-managed VL information instance.

vnfVirtualLinkDescId* string($uuid)
Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.

networkResource* ResourceHandle {...}

vnfLinkPorts [...]

}

ScaleInfo {

description: *This type represents the scale level of a VNF instance related to a scaling aspect.*

aspectId* string($uuid)
Identifier of the scaling aspect

scaleLevel* integer($int32)
Indicates the scale level. The minimum value shall be 0 and the maximum value shall be <= maxScaleLevel as described in the VNFD.

}

VnfcResourceInfo {

description: *This type represents the information on virtualised compute and storage resources used by a VNFC in a VNF instance*

id* string($uuid)
Identifier of this VnfcResourceInfo instance

vduId* string($uuid)
Reference to the applicable VDU in the VNFD.

computeResource ResourceHandle {...}

storageResourceIds [...]

reservationId string($uuid)
The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.

vnfcCpInfo {...}

metadata KeyValuePairs {...}

}

VnfVirtualLinkResourceInfo {

description: *This type represents the information that allows addressing a virtualised resource that is used by an internal VL instance in a VNF instance.*

id* string($uuid)
Identifier of this VnfVirtualLinkResourceInfo instance.

virtualLinkDescId* string($uuid)
Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.

networkResource* ResourceHandle {...}
reservationId string($uuid)
The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.

vnfLinkPorts [...]
metadata KeyValuePairs {...}

}

VirtualStorageResourceInfo {

description: *This type represents the information that allows addressing a virtualised resource that is used by a VNF instance*

id* string($uuid)
Identifier of this VirtualStorageResourceInfo instance.

virtualStorageDescId* string($uuid)
Identifier of the VirtualStorageDesc in the VNFD.

storageResource ResourceHandle {...}
reservationId string($uuid)
The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.

metadata KeyValuePairs {...}

}

VnfLinkPort {

description: *This type represents a link port of an internal VL of a VNF*

id* string($uuid)
Identifier of this link port as provided by the entity that has created the link port.

resourceHandle* ResourceHandle {...}
cpInstanceId string($uuid)
Identifier of the external CP of the VNF to be connected to this link port.

}

ExtLinkPort {

description: *This type represents a link port of an external VL, i.e. a port providing connectivity for the VNF to an NS VL.*

id* string($uuid)
Identifier of this link port as provided by the entity that has created the link port.

resourceHandle* ResourceHandle {...}
cpInstanceId string($uuid)
Identifier of the external CP of the VNF to be connected to this link port.

}

NetworkAddressInfo {

description: *This type represents information about a network address that has been assigned*

macAddress* MacAddress string

ipAddress IpAddress string

subnetIpRanges [...]

}

MonitoringParameter {

description: *This type represents a monitoring parameter that is tracked by the VNFM*

id* string($uuid)
Identifier of the monitoring parameter defined in the VNFD.

name string
Human readable name of the monitoring parameter, as defined in the VNFD.

value* {...}

timeStamp* string($date-time)
Represents the point in time when the measurement has been performed, as known to the VNFM.

}

LifecycleChangeNotificationsFilter {

description: *This type represents a subscription filter related to notifications about VNF lifecycle changes*

vnfInstanceSubscriptionFilter VnfInstanceSubscriptionFilter {...}

notificationTypes [...]

operationTypes [...]

operationStates [...]

}

AffectedVnfc {

description: *This type provides information about added, deleted, modified and temporary VNFCs.*

id* string($uuid)
Identifier of the Vnfc instance, identifying the applicable "vnfcResourceInfo" entry in the "VnfInstance" data type

vduId* string($uuid)
Identifier of the related VDU in the VNFD.

changeType* string
Signals the type of change

Enum:

Array [4]

computeResource* ResourceHandle {...}

addedStorageResourceIds [...]

removedStorageResourceIds [...]

}

AffectedVirtualLink {

description:	*This type provides information about added, deleted, modified and temporary VLs*
id*	string($uuid) Identifier of the virtual link instance, identifying the applicable "vnfVirtualLinkResourceInfo" entry in the "VnfInstance" data type
virtualLinkDescId*	string($uuid) Identifier of the related VLD in the VNFD.
changeType*	string Signals the type of change. Enum: Array [6]
networkResource*	ResourceHandle {...}

}

AffectedVirtualStorage {

description:	*This type provides information about added, deleted, modified and temporary virtual storage resources*
id*	string($uuid) Identifier of the storage instance, identifying the applicable "virtualStorageResourceInfo" entry in the "VnfInstance" data type
virtualLinkDescId*	string($uuid) Identifier of the related VirtualStorage descriptor in the VNFD.
changeType*	string Signals the type of change. Enum: Array [4]
storageResource*	ResourceHandle {...}

}

LccnLinks {

description:	*This type represents the links to resources that a notification can contain*
vnfInstance*	Link {...}
subscription*	Link {...}
vnfLcmOpOcc	Link {...}

}

VnfOperationalStateType string

Enum:

Array [2]

StopType string
Enum:

Array [2]

LcmOperationStateType string
Enum:

Array [7]

CancelModeType string
Enum:

Array [2]

MacAddress string

IpAddress string

ProblemDetails {

description:	*A JSON representation of a "ProblemDetails" data structure according to IETF RFC 7807 that provides additional details of the error*
type	string($uri) A URI reference according to IETF RFC 3986 [5] that identifies the problem type.
title	string A short, human-readable summary of the problem type.
status*	integer($int32) The HTTP status code for this occurrence of the problem
detail*	string A human-readable explanation specific to this occurrence of the problem.
instance	string($uri) A URI reference that identifies the specific occurrence of the problem.
additionalAttributes	[...]

}