*Frameworx Specification*

Activation and Configuration API REST Specification

**TMF640**

**Release 15.5.1**

**May 2016**

|  |  |
| --- | --- |
| **Latest Update: Frameworx Release 15.5** | **TM Forum Approved** |
| **Version 1.0.2** | **IPR Mode: RAND** |

# NOTICE

Copyright © TM Forum 2016. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or associations to TM FORUM, except as needed for the purpose of developing any document or deliverable produced by a TM FORUM Collaboration Project Team (in which case the rules applicable to copyrights, as set forth in the [TM FORUM IPR Policy](http://www.tmforum.org/IPRPolicy/11525/home.html), must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by TM FORUM or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and TM FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

TM FORUM invites any TM FORUM Member or any other party that believes it has patent claims that would necessarily be infringed by implementations of this TM Forum Standards Final Deliverable, to notify the TM FORUM Team Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this deliverable.

The TM FORUM invites any party to contact the TM FORUM Team Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this TM FORUM Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this TM FORUM Standards Final Deliverable. TM FORUM may include such claims on its website, but disclaims any obligation to do so.

TM FORUM takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this TM FORUM Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on TM FORUM's procedures with respect to rights in any document or deliverable produced by a TM FORUM Collaboration Project Team can be found on the TM FORUM website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this TM FORUM Standards Final Deliverable, can be obtained from the TM FORUM Team Administrator. TM FORUM makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Direct inquiries to the TM Forum office:

4 Century Drive
Suite 100
Parsippany, NJ 07054, USA

Tel No.  +1 973 944 5100

Fax No.  +1 973 944 5110

TM Forum Web Page: [www.tmforum.org](http://www.tmforum.org/)

# Table of Contents

[NOTICE 2](#_Toc437593115)

[Table of Contents 4](#_Toc437593116)

[List of Tables 6](#_Toc437593117)

[Introduction 7](#_Toc437593118)

[RESOURCE MODEL 8](#_Toc437593119)

[Managed Entity and Task Resource Models 8](#_Toc437593120)

[Service 8](#_Toc437593121)

[Resource (Abstract) 10](#_Toc437593122)

[Monitor 11](#_Toc437593123)

[Notification Resource Models 12](#_Toc437593124)

[Servicecreationnotification 12](#_Toc437593125)

[ServiceValueChangeNotification 13](#_Toc437593126)

[ServicestateChangeNotification 13](#_Toc437593127)

[ServiceDeletionNotification 14](#_Toc437593128)

[RESOURCEValueChangeNotification 14](#_Toc437593129)

[RESOURCEstateChangeNotification 14](#_Toc437593130)

[RESOURCEDeletionNotification 15](#_Toc437593131)

[RESOURCECREATENotification 15](#_Toc437593132)

[Monitorcreationnotification 16](#_Toc437593133)

[MonitorValueChangeNotification 16](#_Toc437593134)

[MonitorstateChangeNotification 17](#_Toc437593135)

[MonitoreDeletionNotification 18](#_Toc437593136)

[API OPERATION TEMPLATES 19](#_Toc437593137)

[GET /API/service/{ID} 20](#_Toc437593138)

[POST /api/activation/service 21](#_Toc437593139)

[PATCH /apI/service/{ID} 25](#_Toc437593140)

[DELETE /api/activation/service/{ID} 26](#_Toc437593141)

[HEAD /api/activation/service/{ID} 27](#_Toc437593142)

[POST|PUT|PATCH|DELETE /api/…/monitor 28](#_Toc437593143)

[GET /api/monitor/{ID} 28](#_Toc437593144)

[HEAD /api/activation/MONITOR/{ID} 29](#_Toc437593145)

[API NOTIFICATION TEMPLATES 31](#_Toc437593146)

[REGISTER LISTENER POST /hub 31](#_Toc437593147)

[UNREGISTER LISTENER DELETE hub/{id} 32](#_Toc437593148)

[publish {EventTYPE} POST /listener 32](#_Toc437593149)

[open items 33](#_Toc437593150)

[Administrative Appendix 34](#_Toc437593151)

[VERSION HISTORY 34](#_Toc437593152)

[Release historY 34](#_Toc437593153)

[Acknowledgments 34](#_Toc437593154)

# List of Tables

N/A

# Introduction

The following document is intended to provide details of the REST API for Activation and Configuration.

Although all the examples are relative to Services the same API can be used to Activate and Configure Services or Resources.

# RESOURCE MODEL

## Managed Entity and Task Resource Models

## Service

Services (CFS or RFS) may be activated or configured by the Activation and Configuration API.

|  |
| --- |
| {              "id" : "id1234567890",     "href" : "http//..", "state" : "active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]}"serviceRelationship" : [{ "type" : "contains", "service" : { “id”: “43”, "href" : " <http://server:port/inventoryApi/service/44>"….May contain the fully embedded service or only an hyperlink } }], "supportingService”:[  {  “id”: “59”, "href":" <http://server:port/inventoryApi/service/59>"…. May contain the fully embedded service or only an hyperlink }], "supportingResource":[  {  “id”: “46779”, "her":<http://server:port/inventoryApi/logicalPort/46779>…. May contain the fully embedded resource or only an hyperlink }], "relatedParty": [ { "role": "Owner", "id": "1234", "href": "http ://serverLocation:port/partyManagement/partyRole/1234" }] } } |

Service attributes description (these are as per the Service model as used in the Service Inventory specification:

|  |  |
| --- | --- |
| Field | Description |
| id | Identifier of a service instance. Required to be unique. Used in URIs as the identifier of the service (for modify or delete use cases) |
| href | Reference to the owned Service (useful for delete or modify command) |
| category | The category of the service (e.g. CFS, RFS) |
| name | The name of the service  |
| description | A description of the service (what it provides) |
| serviceState | The lifecycle state of the service (as per state diagram below) |
| serviceSpecification | The service specification (default values, etc. are fetched from the catalogue) |
| serviceConfigSpec | Ordered service config (default values, etc. are fetched from the catalogue) |
| policy |  |
| serviceCharacteristic | A name/value pair list used to store instance specific values of attributesThe behavior is equivalent to a MAP data structure where only one entry for any given value of "name" can exist. |
| serviceRelationship | Linked Services to the one instantiate, it can be :* “reliesOn” if the Service needs another already owned Service to rely on (e.g. an option on an already owned mobile access Service)
* “targets” or “isTargeted” (depending on the way of expressing the link) for any other kind of links that may be useful
 |
| relatedParty | Party linked at the Service level (it may be a User for example) |
|  |  |



Figure 1 – Service State model

## Resource (Abstract)

Resources (Logical or Physical) may be configured by the Activation and Configuration API

Resource is an abstract resource. The intent is to describe the common set of attributes shared by all concrete resource (e.g. cross-connect, shelf). The abstract Resource collection is a container or collection for all the “concrete” resources.

The “type” attribute is meant to be implemented by all concrete resources and is a reserved attribute the value of is equal to the name of the resource. For example a shelf resource has the type value equal to "shelf".

|  |
| --- |
| {              "id" : "id1234567890",     "href" : "http//..", “type” : “resourcetype”, "state" : "active","commonName": "<<value>>","aliasNameList": [ { "name": "", "value": "" }],"description": "<<value>>","managementDomain": "<<value>>","adminState": "<<value>>","operationalState": "<<value>>","usageState": "<<value>>","version": "", "resourceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/resourceSpecification/conferenceBridgeEquipment"     }, "resourceCharacteristic":[      {      "name":"x",          "value":"1" }, {          "name":"y",          "value":"2" },      {          "name":"z",          "value":"3" }     ]},"resourceRelationship" : [{ "type" : "contains", "resource" : { “id”: “43”, "href" : “…”,….May contain the fully embedded service or only an hyperlink } }],  "relatedParty": [ { "role": "Owner", "id": "1234", "href": "http ://serverLocation:port/partyManagement/partyRole/1234" }] } |

## Monitor

The monitor resource is used to monitor the execution of async requests on specific resource.

|  |
| --- |
| { "id": "", "state": "MonitorState", “type”: “monitor” "request": { "method": "", "to": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "response": { "statusCode": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "href": "", "sourceHref": "http//.."} |

|  |  |
| --- | --- |
| Fields | Description |
| id | Identifier of an instance of the monitor. Required to be unique within the resource type. Used in URIs as the identifier for specific instances of a type. |
| type | Name of the resource type i.e monitor |
| request | Represent the request |
| response | Represent the response |
| state | The Monitor state of the resource. InProgess, InError, Completed |
| href | The self ref |
| sourceHref | The monitored resource href |

## Notification Resource Models

### Servicecreationnotification

Used to notify that a service has just been newly provisioned

|  |
| --- |
| { "event": { "service":  { "id": "42", // Following a whole representation of the Service with all its attributes  See Service Resource. } }, "eventType": "serviceCreationNotification"} |

### ServiceValueChangeNotification

Used to notify that a service has been re-provisioned (and some values changed)

|  |
| --- |
| { "event": { "service":  { "id": "42", // Following a whole representation of the Service with all its attributes  See Service Resource. } } "eventType": "serviceValueChangeNotification"} |

### ServicestateChangeNotification

Used to notify that a service state has just changed

|  |
| --- |
| { "event": { "service":  { "id": "42", "state": "Inactive" } }, "eventType": "serviceStateChangeNotification"} |

### ServiceDeletionNotification

Used to notify that a service has been deleted.

|  |
| --- |
| { "event": { "service":  { "id": "42" } }, "eventType": "serviceDeletionNotification"} |

### RESOURCEValueChangeNotification

Used to notify that a service has been re-provisioned (and some values changed)

|  |
| --- |
| { "event": { "resource":  { "id": "42", // Following a whole representation of the Resource with all its changed attributes   } } "eventType": "resourceValueChangeNotification"} |

### RESOURCEstateChangeNotification

Used to notify that a resource state has just changed

|  |
| --- |
| { "event": { "resource":  { "id": "42", "adminState": "newstate" } }, "eventType": "resourceStateChangeNotification"} |

### RESOURCEDeletionNotification

Used to notify that a resource has been deleted.

|  |
| --- |
| { "event": { "resource":  { "id": "42" } }, "eventType": "resourceDeletionNotification"} |

### RESOURCECREATENotification

Used to notify that a resource has been created

|  |
| --- |
| { "event": { "resource":  { "id": "42", // Following a whole representation of the Resource with all its changed attributes   } } "eventType": "resourceCreateNotification"} |

### Monitorcreationnotification

Used to notify that a resource was created in the inventory

|  |
| --- |
| { "event": { "monitor": { "id": "", "type": "monitor", "state": "MonitorState", "request": { "method": "", "to": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "response": { "body": "", "statusCode": "", "header": [ { "name": "", "value": "" } ] }, "href": "" } }, "eventType": "monitorCreationNotification"} |

### MonitorValueChangeNotification

Used to notify that a resource value was modified in the inventory

|  |
| --- |
| { "event": { "monitor": { "id": "", "type": "monitor", "state": "MonitorState", "request": { "method": "", "to": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "response": { "body": "", "statusCode": "", "header": [ { "name": "", "value": "" } ] }, "href": "" } }, "eventType": "monitorValueChangeNotification"} |

### MonitorstateChangeNotification

Used to notify

|  |
| --- |
| { "event": { "monitor": { "id": "", "type": "monitor", "state": "MonitorState", "request": { "method": "", "to": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "response": { "body": "", "statusCode": "", "header": [ { "name": "", "value": "" } ] }, "href": "" } }, "eventType": "monitorStateChangeNotification"} |

### MonitoreDeletionNotification

Used to notify

|  |
| --- |
| { "event": { "id": "http://server/api/activation/monitor/77" }, "eventType": "monitorDeletionNotification"} |

# API OPERATION TEMPLATES

For every single of operation on the entities use the following templates and provide sample REST requests and responses.

Remember that the following Uniform Contract rules must be used :

|  |  |  |
| --- | --- | --- |
| Operation on Entities | Uniform API Operation | Description |
| Query Entities | GET service | GET must be used to retrieve a representation of a service. |
| Create Entity | POST service | POST must be used to create a new service |
| Partial Update of an Entity | PATCH service | PATCH must be used to partially update a service |
| Remove an Entity | DELETE service | DELETE must be used to remove a service |
| Execute a Task  | POST service | POST must be used to execute Task Resources |
| Other Request Methods | POST service | Creating a graph. |

Filtering and attribute selection rules are described in the TMF REST Design Guidelines.

Notifications are also described in a subsequent section.

## GET /API/service/{ID}

This operation is used to retrieve service items.

The same API can be used for retrieving Resources (if the management scope of the controller is relative to Resources). i.e GET /api/Resource/{ID}

Note that collections can be retrieved via GET /API/service with no {ID}

Description :

* This operation is used to retrieve service information using the ID
* Attribute selection is enabled

Behavior :

* Status code 200 - if the request was successful
* Status code 404 Not found - the supplied ID does not match a known Service

|  |
| --- |
| **REQUEST** |
| GET /api/activation/service/id1234567890Accept: application/json |
| **RESPONSE** |
| 200Content-Type: application/json{              "id" : "id1234567890", "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]}  } |

## POST /api/activation/service

POST operations are used to create a service.

The same API can be used for creating Resources (if the management scope of the controller is relative to Resources). i.e POST /api/Resource/{ID}

Example 1 : The response of the operation cannot be sent back synchronously, a “monitor” resource hyperlink is given in the Response.

See TM Forum REST Design Guidelines for more information about asynchronous pattern and monitor resources.

|  |
| --- |
| **REQUEST** |
| POST /api/activation/serviceAccept: application/json {          "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]} } |
| **RESPONSE** |
| 202 AcceptedContent-Type: Application/JSONLocation: [http://server/api/activation/service/14](http://server/api/provisioning/service/14){// same as in request}Link: <[**http://server/api/activation/monitor/1514**>;rel=related;title=monitor](http://server/api/provisioning/monitor/1514%3E;rel=related;title=monitor),<[http://server/api/activation/service/14>;rel=self](http://server/api/provisioning/service/14%3E;rel=self),<[http://server/api/activation/service/14>;rel=canonical](http://server/api/provisioning/service/14%3E;rel=canonical) |

Example 2 : The response of the operation cannot be sent back synchronously, a “monitor” resource hyperlink is given in the Response. This is done under the control of the client.

A client may request the server to modify its asynchronous behavior with the following “Expect” headers:

* “Expect: 200-ok/201-created/204-no-content” **disables all asynchronous functionality**. The server may return a “417 Expectation Failed” if it is not willing to wait for an operation to complete.
* “Expect: 202-accepted” **explicitly request an asynchronous response**. The server may return a “417 Expectation Failed” if it is not willing to perform the request asynchronously.

If no expectation is provided, client must be prepared to accept a 202 Accepted status for any request other than GET.

See TM Forum REST Design Guidelines V3 for more information about asynchronous pattern and monitor resources.

|  |
| --- |
| **REQUEST** |
| POST /api/activation/serviceAccept: application/json**Expect: 202-accepted** {          "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]} } |
| **RESPONSE** |
| **202 Accepted**Content-Type: Application/JSONLocation: [http://server/api/activation/service/14](http://server/api/provisioning/service/14){// same as in request}Link: <[http://server/api/activation/monitor/1514>;rel=related;title=monitor](http://server/api/provisioning/monitor/1514%3E;rel=related;title=monitor),<[http://server/api/activation/service/14>;rel=self](http://server/api/provisioning/service/14%3E;rel=self),<[http://server/api/activation/service/14>;rel=canonical](http://server/api/provisioning/service/14%3E;rel=canonical) |

Example 3: the activation is realized immediately and a synchronous response is given to the caller.

|  |
| --- |
| **REQUEST** |
| POST /api/activation/serviceAccept: application/json {          "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]} } |
| **RESPONSE** |
| 200 OKContent-Type: Application/JSONLocation: [http://server/api/activation/service/14](http://server/api/provisioning/service/14){          "id" : "14", "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]} }Link: <[http://server/api/activation/service/14>;rel=self](http://server/api/provisioning/service/14%3E;rel=self),<[http://server/api/activation/service/14>;rel=canonical](http://server/api/provisioning/service/14%3E;rel=canonical) |

Example see TM Forum REST Design Guidelines.

## PATCH /apI/service/{ID}

PATCH operations are used to update services.

The response of the operation can be sent back synchronously or not, in this case a “monitor” resource hyperlink is given in the Response.

The same API can be used for updating Resources (if the management scope of the controller is relative to Resources). i.e paTCH /api/Resource/{ID}

See TM Forum REST Design Guidelines TMF 630-631 for more information about asynchronous pattern and monitor resources.

|  |
| --- |
| **REQUEST** |
| PATCH /api/activation/service/14Accept: application/json{ "state": "Active"} |
| **RESPONSE** |
| 202 AcceptedContent-Type: Application/JSON{              "id" : "14", "state" : "Active", "serviceSpecification":{      "id":"conferenceBridgeEquipment",      "href":"http: //serverlocation:port/catalogManagement/serviceSpecification /conferenceBridgeEquipment"     }, "serviceCharacteristic":[      {      "name":"numberOfVc500Units",          "value":"1" }, {          "name":"numberOfVc100Units",          "value":"2" },      {          "name":"routerType",          "value":"CiscoASR1000" }, {          "name":"powerSupply",          "value":"UK"      }     ]} } |

## DELETE /api/activation/service/{ID}

DELETE operations are used to delete services.

The response of the operation can be sent back synchronously or not, in this case a “monitor” resource hyperlink is given in the Response.

The same API can be used for deleting Resources (if the management scope of the controller is relative to Resources) i.e delete /api/Resource/{ID}

The response of the operation can be sent back synchronously or not, in this case a “monitor” resource hyperlink is given in the Response.

See TM Forum REST Design Guidelines TMF 630-631 for more information about asynchronous pattern and monitor resources.

Example 1 : Monitor response for the DELETE directive, the deletion will be performed

|  |
| --- |
| **REQUEST** |
| DELETE /api/activation/service/47Accept: application/json |
| **RESPONSE** |
| 202 AcceptedContent-Type: Application/JSONLink: <[http://server/api/activation/monitor/1545>;rel=related;title=monitor](http://server/api/provisioning/monitor/1545%3E;rel=related;title=monitor),<[http://server/api/activation/service/47>;rel=self](http://server/api/provisioning/service/47%3E;rel=self),<[http://server/api/activation/service/47>;rel=canonical](http://server/api/provisioning/service/47%3E;rel=canonical) |

Example 2 : Error performing the DELETE operation

|  |
| --- |
| **REQUEST** |
| DELETE /api/activation/service/47Accept: application/json |
| **RESPONSE** |
| 204 No Content |

## HEAD /api/activation/service/{ID}

This Uniform Contract operation is used to retrieve the header information that would be retrieved by the GET operation without retrieving the actual data.

The Header information provides information about the possible action and next state transitions.

The same API can be used for getting Resources (if the management scope of the controller is relative to Resources). i.e HEAD/api/Resource/{ID}

Behavior :

* Returns HTTP/1.1 status code 200 if the request was successful.

|  |
| --- |
| **REQUEST** |
| HEAD /api/activation/service/14Accept: application/json |
| **RESPONSE** |
| LINK: <[http://server/api/activation/service/14>;rel=self](http://server/api/provisioning/service/14%3E;rel=self),<[http://server/api/activation/service/14>;rel=canonical](http://server/api/provisioning/service/14%3E;rel=canonical) |

## POST|PUT|PATCH|DELETE /api/…/monitor

PUT|POST|PATCH|DELETE operations are not supported against /api/…/monitor.

|  |
| --- |
| **REQUEST** |
| PUT|POST|PATCH|DELETE /api/…/monitorAccept: application/json |
| **RESPONSE** |
| 405 Method Not Allowed |

## GET /api/monitor/{ID}

This Uniform Contract operation is used to get a Monitor. Monitors can be accessed via the Monitor collection or a singleton under their source for example api/activation/service/47/monitor.

Behavior :

* What status and exception codes are returned.
* Returns HTTP/1.1 status code 200 if the request was successful.
* Any other special return and/or exception codes.

|  |
| --- |
| **REQUEST** |
| GET /api/activation/monitor/14Content-type: application/json |
| **RESPONSE** |
| 200Content-Type: application/json "id": "", "state": "MonitorState", “type”: “monitor” "request": { "method": "", "to": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "response": { "statusCode": "", "body": "", "header": [ { "name": "", "value": "" } ] }, "href": "", "sourceHref": "http//.."} |

Example see TM Forum REST Design Guidelines.

## HEAD /api/activation/MONITOR/{ID}

This Uniform Contract operation is used to retrieve the header information that would be retrieved by the GET operation without retrieving the actual data. Useful to know if an operation is still in progress on a resource for example.

Behavior :

* Returns HTTP/1.1 status code 200 if the request was successful.

|  |
| --- |
| **REQUEST** |
| HEAD /api/activation/monitor/14Accept: application/json |
| **RESPONSE** |
| E-tag: "adlkjaljadfljal" |

# API NOTIFICATION TEMPLATES

For every single of operation on the entities use the following templates and provide sample REST notification POST calls.

It is assumed that the Pub/Sub uses the Register and UnRegister mechanisms described in the REST Guidelines reproduced below.

##  REGISTER LISTENER POST /hub

Description :

Sets the communication endpoint address the service instance must use to deliver information about its health state, execution state, failures and metrics. Subsequent POST calls will be rejected by the service if it does not support multiple listeners. In this case DELETE /api/hub/{id} must be called before an endpoint can be created again.

Behavior :

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 409 if request is not successful.

|  |
| --- |
| **REQUEST** |
| POST /api/hubAccept: application/json{ "callback": "http://in.listener.com"} |
| **RESPONSE** |
| 201Content-Type: application/jsonLocation: /api/hub/42{ "id": "42", "callback": "http://in.listener.com", "query": null} |

## UNREGISTER LISTENER DELETE hub/{id}

Description :

Clears the communication endpoint address that was set by creating the Hub.

Behavior :

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 404 if the resource is not found.

|  |
| --- |
| **REQUEST** |
| DELETE /api/hub/{id}Accept: application/json |
| **RESPONSE** |
| 204 |

## publish {EventTYPE} POST /listener

Description :

Provide the Event description

Behavior :

Returns HTTP/1.1 status code 201 if the service is able to set the configuration.

|  |
| --- |
| **REQUEST** |
| POST /client/listenerAccept: application/json{  "event": { EVENT BODY  }, "eventType": "eventType"} |
| **RESPONSE** |
| 201Content-Type: application/json |

Example see TM Forum REST Design Guidelines.

## open items

|  |  |  |
| --- | --- | --- |
| **Item Number** | **Description** | **Resoultion** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

# Administrative Appendix

## VERSION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Date**  | **Modified by** | **Description**  |
| Version 1.0.0 | 12 Nov 2015 | Pierre GauthierTM Forum | Submitted for Fx15.5 |
| Version 1.0.1 | 12 Nov 2015 | Alicja KaweckiTM Forum | Updated cover, header; minor formatting/style corrections prior to publishing |
| Version 1.0.2 | 2 May 2016 | Alicja Kawecki,TM Forum | Updated cover, footer, Notice to reflect TM Forum Approved status |

## Release historY

|  |  |  |  |
| --- | --- | --- | --- |
| **Release Number** | **Date**  | **Release led by:** | **Description**  |
| 2.0 | 12 Nov 2015 | Pierre Gauthier pgauthier@tmforum.orgAndrew Forth aforth@amdocs.comAugust-Wilhelm JagauAugust-wilhelm.jagau@ericsson.com  | Final release of specification with additional comments on Resource usage |

## Acknowledgments

This document was prepared by members of the TM Forum API Program team.