ONEM2M TECHNICAL SPECIFICATION

Document Number	TS-0006-V1.0.1
Document Name:	Management enablement (BBF)
Date:	2015-January-30
Abstract:	Specifies the usage of the BBF TR-069 protocol and the corresponding message flows including normal cases as well as error cases to fulfil the oneM2M management requirements. • Protocol mapping between the oneM2M service layer and BBF TR-069 protocol. The Mca reference point, ms interface and la interface are possibly involved in this protocol mapping. • Mapping between the oneM2M management related resources and the TR-069 protocol RPCs and TR-181i2 data model. • Specification of new TR-181 data model elements to fulfil oneM2M specific management requirements that cannot be currently translated.

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Contents

1. Scope

The present document describes the protocol mappings between the management Resources for oneM2M and the BBF TR-181i2 Data Model [6].

2 References

2.1 Normative references

The following referenced documents are necessary, partially or totally, for the application of the present document. Their use in the context of this TS is specified by the normative statements that are referring back to this clause.

[1] oneM2M TS-0001: "Functional Architecture".

[2] oneM2M TS-0004: "Service Layer Core Protocol Specification".

[3] oneM2M TS-0011: "Definitions and Acronyms".

[4] BBF: "TR-069 CPE WAN Management Protocol" Issue: 1 Amendment 5, November 2013.

[5] BBF: "TR-106 Data Model Template for TR-069-Enabled Devices", Issue 1, Amendment 7, September 2013.

[6] BBF: "TR-181 Device Data Model for TR-069, Issue 2 Amendment 8", September 2014.

[7] BBF: "TR-131 ACS Northbound Interface Requirements, Issue:1", November 2009.

2.2 Informative references

[i.1] oneM2M Drafting Rules (http://member.onem2m.org/Static_pages/Others/Rules_Pages/oneM2M-Drafting-Rules-V1_0.doc)

3 Definitions, symbols, abbreviations and acronyms

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS-0011 [3] apply.

CPE Proxier A CPE that is capable of proxying the communication between an ACS and a Proxied Device as defined in TR-069 [4].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS-0011 [3] apply.

ACS	Auto-Configuration Server
ADN	Application Dedicated Node
AE	Application Entity
ASN	Application Service Node
CMDH	Communication Management and Delivery Handling
CPE	Customer Premise Equipment
CSE	Common Services Entity
CWMP	CPE WAN Management Protocol
DU	Deployment Unit

IN-CSE	CSE which resides in the Infrastructure Node
LAN	Local Area Network
MN	Middle Node
OUI	Organizationally Unique Identifier
PC	Product Class
RPC	Remote Procedure Call
SN	Serial Number
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universal Unique Identifier
XML	Extensible Markup Language

3.4 Acronyms

For the purposes of the present document, the acronyms given in TR-0004 [3] apply.

4 Conventions

The key words “Shall”, ”Shall not”, “May”, ”Need not”, “Should”, ”Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 Mapping of basic data types

TR-106 [5] specifies the object structure supported by TR-069 enabled devices and specifies the structural requirements for the data hierarchy. This clause includes the mapping attribute data types to TR-181 [6] parameters which follows the conventions of section 3 of TR-106 [5] and data types described in Table 4 of TR-106 [5].

Table 5-1: Data Type Mapping

oneM2M Data Types	Mapping to data types in TR-106	Conversion Notes
xs:boolean	boolean	
xs:string	string	Mapping is constrained to the size of the string
xs:unsignedInt	unsignedInt	
xs:unsignedLong	unsignedLong	
xs:integer	long	Mapping is constrained to the size of the long data type.
Xs:positiveInteger	unsignedLong	Mapping is constrained to a lower limit of 1 and the size of the unsignedLong data type.
Xs:nonNegativeInteger	unsignedLong	Mapping is constrained the size of the unsignedLong data type.
Comma separated Lists	Comma separated Lists	Data structure is represented by comma separated list as described in section 3.2.3 of TR-106 [5].

In some instances the conversion of the contents between data types will cause an error to occur (e.g., xs:integer to long). When an error occurs in the conversion of a data type, the STATUS_BAD_REQUEST response status code.

6 Mapping of identifiers

The TR-069 [4] specification defines three (3) types of devices, known as CPEs, that are capable of being managed from the perspective of the TR-069 agent:

- CPE that hosts the TR-069 agent: Section A.3.3.1 Inform of TR-069 [4] defines the required fields for a CPE to be identified. These fields include the OUI and Serial Number of the CPE assigned by the CPE manufacturer. Optionally the manufacturer may assign a Product Class to the CPE. The format of the identifier is as follows: OUI-[PC-]SN.

- Virtual Device: This type of device is addressed as a CPE. The Virtual Device has its own OUI-[PC-]SN as represented by the CPE Proxier. The CPE Proxier emulates a CWMP agent for each Virtual Device.
- Embedded Device: This type of device is addressed as one or more objects within the data model of the CPE that hosts the TR-069 agent.

6.1 Mapping of Device identifiers to the Node Resource

Node Resources are identified for each instance of an ADN, ASN and MN node and are identified using the M2M Node Identifier (M2M-Node-ID) defined in the oneM2M Functional Architecture [1].

CPE Device identifiers shall map to the nodeID attribute of the <node> resource. The CPE Device identifiers are obtained from the contents of the following attributes:

- Device.DeviceInfo.ManufacturerOUI
- Device.DeviceInfo.ProductClass
- Device.DeviceInfo.SerialNumber

Virtual Device identifiers shall map to the nodeID attribute of the <node> resource. The Virtual Device identifiers are obtained from the CPE Proxier using the contents of the attributes:

- Device.ManagementServer.VirtualDevice.{i}.ManufacturerOUI
- Device.ManagementServer.VirtualDevice.{i}.ProductClass
- Device.ManagementServer.VirtualDevice.{i}.SerialNumber

Embedded Device identifiers shall map to the nodeID attribute of the <node> resource. The Embedded Device identifiers are obtained using the containing CPE Device or Virtual Device identifiers along with the contents of the attributes of the:

- Device.ManagementServer.EmbeddedDevice.{i}.ControllerID
- Device.ManagementServer.EmbeddedDevice.{i}.ProxiedDeviceID

6.2 Identifier of an object instance

The TR-069 [4] specification permits objects to have multiple object instances where each object instance is contained within the objectPath attribute of the Resource within the context of the Resource's objectId as defined in clause 7.1.

In order to allow the AE or CSE that originated the request that manipulates a Resource to easily align the M2M Service Layer with the Resource's external technology identifier, the value of the object instance "{i}" should be a part of the identifier of the Resource in the M2M Service Layer where possible. For example if the [areaNetwork] resource has an object instance identifier of "Device.X_oneM2M_org_CSE.1.M2MareaNetworkDevice.[foo]" then the M2M Service Layer Resource should be identified using the object instance of the underlying technology (e.g., "/foo" for the Resource areaNetwork).

7 Mapping of resources

This clause contains all information on how to map management resources from TS-0004 [2] to managed objects and parameters as defined in the TR-181 [6] data model or the Remote Procedure Calls (RPCs) in TR-069 [4].

7.1 General mapping assumptions

TR-069 [4] specifies a protocol for communication between a CPE (Customer Premises Equipment) and an ACS (Auto-Configuration Server). Any TR-069 enabled device has to follow the data model as described in the TR-106 [5] and TR-181 [6] as well as RPCs described in TR-069 [4].

As TR-181 [6] is the model that the Resources are mapped, all Resources shall have the bjected of the TR-181[6] namespace (e.g., "urn:broadband-forum-org:tr-181-2-7-0").

7.1.1 Mapping of Device identifiers

The Device identifiers for CPEs are mapped to the Resource Typse [deviceInfo].

For CPE and Virtual Devices map their Device Identifiers (OUI-[PC-]SN) to the manufacturer, deviceType and deviceLabel attributes of the Resource [deviceInfo].

For Embedded Devices, the ControllerID and ProxiedDeviceID parameters of the Device.ManagementServer.EmbeddedDevice.{i} object instance are mapped to the deviceLabel attribute of the Resource [deviceInfo] as a comma separated list: "Device.ManagementServer.EmbeddedDevice.{i}.ControllerID, Device.ManagementServer.EmbeddedDevice.{i}.ProxiedDeviceID".

7.1.2 Mapping of Embedded Devices

The TR-181 [6] specification does not provide a mechanism where Embedded Devices provide information related to the Device.DeviceInfo objects and sub-objects. Instead the TR-181 [6] provides this information in a manner that is reliant on the Embedded Device's underlying technology (e.g., ZigBee®, UpnP).

As such the mapping of the [memory] and [battery] Resources are implementation specific for each underlying technology and is outside the scope of this specification.

7.2 Resource [deviceInfo]

The Resource [deviceInfo] is a read-only Resource that shall map to the Device.DeviceInfo object of TR-181 [6] for CPE and Virtual Devices.

The information shall be retrieved using the GetParameterValues RPC of TR-069 [4].

Note: The SerialNumber, ModelNumber, ProductClass attributes for a Virtual device are the same values as the Device.ManagementServer.VirtualDevice.{i} object in the CPE Proxier.

Table 7.2-1: Resource [deviceInfo] for CPE and Virtual Devices

Attribute Name of [deviceInfo]	TR-181 Parameter
deviceLabel	Device.DeviceInfo.SerialNumber
manufacturer	Device.DeviceInfo.Manufacturer
model	Device.DeviceInfo.ModelNumber
deviceType	Device.DeviceInfo.ProductClass
fwVersion	Device.DeviceInfo.SoftwareVersion if the device supports only 1 software version. If the device support multiple software versions this shall map to Device.DeviceInfo.AdditionalSoftwareVersion
swVersion	Device.DeviceInfo.SoftwareVersion
hwVersion	Device.DeviceInfo.HardwareVersion

Table 7.2-2: Resource [deviceInfo] for Embedded Devices

Attribute Name of [deviceInfo]	TR-181 Parameter
deviceLabel	Comma separated list: "Device.ManagementServer.EmbeddedDevice.{i}.ControllerID, Device.ManagementServer.EmbeddedDevice.{i}.ProxiedDeviceID
manufacturer	No mapping available
model	No mapping available
deviceType	No mapping available
fwVersion	No mapping available
swVersion	No mapping available
hwVersion	No mapping available

7.3 Resource [memory]

The Resource [memory] is a read-only Resource that shall map to the Device.DeviceInfo.MemoryStatus object of TR-181 [6] for CPE and Virtual Devices.

The information shall be retrieved using the GetParameterValues RPC of TR-069 [4].

Attempts to modify the attributes of the memory Resource causes an error code "operation unsupported" to be returned.

Table 7.3-1: Resource [memory]

Attribute Name of [memory]	TR-181 Parameter
memAvailable	Device.DeviceInfo.MemoryStatus.Free
memTotal	Device.DeviceInfo.MemoryStatus.Total

7.4 Resource [battery]

The Resource [battery] is a read-only Resource that shall map to an instance of Device.DeviceInfo.X_oneM2M_org_BatteryStatus.Battery.{i} object for CPE and Virtual Devices.

The information shall be retrieved using the GetParameterValues RPC of TR-069 [4].

Table 7.4-1: Resource [battery]

Attribute Name of [battery]	TR-181 Parameter
batteryLevel	Device.DeviceInfo.X_oneM2M_org_BatteryStatus.Battery.{i}.Level
batteryStatus	Device.DeviceInfo.X_oneM2M_org_BatteryStatus.Battery.{i}.Status

7.5 Resource [areaNwkInfo]

The Resource [areaNwkInfo] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.M2MareaNetwork.{i} object.

As the Resource [areaNwkInfo] is a multi-instance Resource, the M2MareaNetwork object is a multi-object instance that can be created and deleted.

The M2MareaNetwork instance shall be created using the Add Object RPC of TR-069 [4].

The M2MareaNetwork instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of an M2MareaNetwork shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of an M2MareaNetwork shall be modified using the SetParameterValues RPC of TR-069 [4].

Table 7.5-1: Resource [areaNwkInfo]

Attribute Name of [areaNwkInfo]	X_oneM2M_org Parameter
areaNwkType	Device.X_oneM2M_org_CSE.{i}.M2MareaNetwork.{i}.Type
listOfDevices	Device.X_oneM2M_org_CSE.{i}.M2MareaNetwork.{i}.ListOfDevices

7.6 Resource [areaNwkDeviceInfo]

The Resource [areaNwkDeviceInfo] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.AreaNetworkDevice.{i} object.

As the Resource [areaNwkDeviceInfo] is a multi-instance Resource, the AreaNetworkDevice object is a multi-object instance that can be created and deleted.

Instances of the Resource [areaNwkDeviceInfo] are referenced in the listOfDevices attribute of the associated Resource [areaNwkInfo].

The M2MareaNetworkDevice instance shall be created using the Add Object RPC of TR-069 [4].

The M2MareaNetworkDevice instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of an M2MareaNetworkDevice shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of an M2MareaNetworkDevice shall be modified using the SetParameterValues RPC of TR-069 [4].

Table 7.6-1: Resource [areaNwkDeviceInfo]

Attribute Name of [areaNwkDeviceInfo]	X_oneM2M_org Parameter
devId	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.Host
devType	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.Type
areaNwkId	Reference to Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.M2MareaNetwork
sleepInterval	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.SleepInterval
sleepDuration	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.SleepDuration

Attribute Name of [areaNwkDeviceInfo]	X_oneM2M_org Parameter
status	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.Status
listOfNeighbors	Device.X_oneM2M_org_CSE.{i}.M2MareaNetworkDevice.{i}.Neighbors

7.7 Resource [eventLog]

The Resource [eventLog] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i} object.

The EventLog instance shall be created using the Add Object RPC of TR-069 [4].

The EventLog instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of an EventLog instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of an EventLog instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.7-1: Resource [eventLog]

Attribute Name of [eventLog]	TR-181 Parameter
logTypeId	Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i}.Type
logData	Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i}.Data
logActionStatus	Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i}.Status
logStart	Set to “True” , the Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i}.Enable parameter is set to “True”.
logStop	Set to “True” , the Device.DeviceInfo.X_oneM2M_org_Diagnostics.EventLog.{i}.Enable parameter is set to “False”.

7.8 Resource [deviceCapability]

The Resource [deviceCapability] represents a capability of device that can be administratively enabled or disabled. The lists of capabilities that are managed are defined in the enumeration of the capabilityName attribute. The TR-181 [6] data model defines a subset of capabilities listed in the deviceCapability enumeration. The supported device capabilities within TR-181 [6] include:

- LAN Interfaces: USB, Wi-Fi, HomePlug, MoCA, UPA
- Hardware Capabilities: SmartCardReader

The information shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The capabilities shall be enabled and disabled using the SetParameterValues RPC of TR-069 [4].

328 **Table 7.8-1: Resource [capabilityInstance]**

Attribute Name of [capabilityInstance]	TR-181 Parameter
capabilityName	This attribute is fixed based on the value of the capabilityName attribute.
Attached	Returns “True”
capabilityActionStatus	Status is defined as: • Success if the SetParameterValues RPC indicates that the operation was sucessfull. • Failure if the response to the SetParameterValues RPCs indicates that the operation failed. • In process if the SetParameterValues RPC is initiatied but the response to the SetParameterValues RPC has not been received.
currentState	USB: Device.USB.Interface.{i}.Enable Wi-Fi: Device.Wi-Fi.Radio.{i}.Enable HomePlug: Device.HomePlug.Interface.{i}.Enable MoCA: Device.MoCA.Interface.{i}.Enable UPA: Device.UPA.Interface.{i}.Enable SmartCardReader: Device.SmartCardReaders.SmartCardReader.{i}.Enable
enable	USB: Device.USB.Interface.{i}.Enable Wi-Fi: Device.Wi-Fi.Radio.{i}.Enable HomePlug: Device.HomePlug.Interface.{i}.Enable MoCA: Device.MoCA.Interface.{i}.Enable UPA: Device.UPA.Interface.{i}.Enable SmartCardReader: Device.SmartCardReaders.SmartCardReader.{i}.Enable
disable	Same parameter is used to disable a capability as the enable attribute.

329

330 # 7.9 Resource [firmware]

331 The Resource [firmware] represents a firmware instance and is not considered a TR-069 managed entity within the
332 device until the firmware Resource’s update attribute has been written a value of “True”. When this occurs, the TR-069
333 Download RPC shall be invoked.

334

335 Note: In many instances, the server from which the firmware is downloaded requires authentication in the form of
336 Username and Password credentials. The CSE that executes firmware download shall maintain the mapping of the

username and password of the download server needed to download the firmware outside the lifecycle of the specific firmware.

Table 7.9-1: Resource [firmware]

Attribute Name of [firmware]	RPC Download Arguments
URL	URL
update	When set to the value of "True" executes the Download operations with a FileType "1 Firmware Upgrade Image" is performed.
	Username: Received from the CSE for the download server where the update is set to "True".
	Password: Received from the CSE for the download server where the update is set to "True".
	CommandKey: Automatically set by the CSE where the update is set to "True" in order to correlate the TransferComplete response.
	FileSize: 0 (not used)
	TargetFileName: <empty> (not used)
	DelaySeconds: 0 (immediate)
	SuccessURL: <empty> (not used)
	FailureURL: <empty> (not used)

7.10 Resource [software]

The Resource [software] is a multi-instance Resource where each instance of the Resource maps directly to an instance of Device.SoftwareModules.DeploymentUnit.{i} object for the deployment aspects (install, uninstall) of the Resource [software]. The install and uninstall operation of the Resource [software] is performed using a combination of the ChangeDUState and ChangeDUStateComplete RPCs.

Once a Resource [software] has been installed, the Resource shall be mapped to the associated Device.SoftwareModules.ExecutionUnit.{i} objects in order to activate and deactivate the associated execution unit.

The Resource [software] version and name shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The activate and deactivate operations of the Resource [software] shall be performed by manipulating the Device.SoftwareModules.ExecutionUnit.{i}.RequestedState parameter using the SetParameterValues RPC.

Note: The Resource [software] provides support for only 1 Execution Unit per Deployment Unit. If a Deployment Unit is discovered by the M2M Service Layer that contains multiple Execution Units for a Deployment Unit; only 1 Execution Unit is exposed. The selection of which Execution Unit is implementation specific.

Table 7.10-1: Resource [software]

Attribute Name of [software]	Description
version	Device.SoftwareModules.DeploymentUnit.{i}.Version

Attribute Name of [software]	Description
name	Device.SoftwareModules.DeploymentUnit.{i}.Name
URL	Device.SoftwareModules.DeploymentUnit.{i}.URL
install	Use the ChangeDUState:InstallOpStruct
installStatus	Status is defined as: • Success if the ChangeDUStateComplete RPC indicates that the operation was sucessfull. • Failure if the response to the ChangeDUState or ChangeDUStateComplete RPCs indicates that the operation failed. • In process if the ChangeDUState RPC is initiatied but the ChangeDUStateComplete RPC has not been received.
Activate	The action that activates software previously installed.
Deactivate	The action that deactivates software.
activeStatus	Status is defined as: • Success if the SetParameterValues RPC indicates that the operation was sucessfull. • Failure if the response to the SetParameterValues RPCs indicates that the operation failed. • In process if the SetParameterValues RPC is initiatied but the response to the SetParameterValues RPC has not been received.

356

357 **Table 7.10-2: RPC ChangeDUState:InstallOpStruct Arguments**

RPC ChangeDUState:InstallOpStruct Argument
URL: URL of the Server that M2M Node uses to download the DU.
Username: Username credential of Server that the CPE uses to download the DU – Supplied by the CSE.
Password: Password credential of Server that the CPE uses to download the DU – Supplied by the CSE.
UUID: Supplied by the CSE and used to correlate the DU for the uninstall operation.
ExecurtionEnvRef: <empty> not used

358

359 **Table 7.10-3: RPC ChangeDUState:UninstallOpStruct Arguments**

RPC ChangeDUState:Uninstall OpStruct Argument
UUID: UUID of the DU that was installed – Maintained by

RPC ChangeDUState:Uninstall OpStruct Argument
the CSE.
ExecutionEnvRef: <empty> not used

7.11 Resource [reboot]

The Resource [reboot] maps to either the Reboot RPC or FactoryReset RPC of TR-069 [4].

When the reboot attribute of the Resource [reboot] is set to "True", the CSE shall execute the Reboot RPC of TR-069[4].

When the factoryReset attribute of Resource [reboot] is set to "True", the CSE shall execute the FactoryReset RPC of TR-069[4].

Table 7.11-1: Resource [reboot]

Attribute Name of [reboot]	Description
reboot	Executes the Reboot RPC
factoryReset	FactoryReset RPC

Table 7.11-2: RPC Reboot Arguments

RPC Reboot Arguments
CommandKey: Automatically set by the CSE where the reboot is set to "True" in order to correlate the "M-Reboot" Event from the next Inform.

7.12 Resource [cmdhPolicy]

The Resource [cmdhPolicy] represents a set of rules defining which CMDH parameters will be used by default when a request issued by a local originator contains the **ec** (event category) parameter but not all other CMDH parameters, see clause D.12 of TS-0001 [1].

The Resource [cmdhPolicy] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i} object.

The Policy instance shall be created using the Add Object RPC of TR-069 [4].

The Policy instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a Policy instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a Policy instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12-1: Resource [cmdhPolicy]

Attribute Name of [cmdhPolicy]	X_oneM2M_org Parameter
name	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.Name

Attribute Name of [cmdhPolicy]	X_oneM2M_org Parameter
cmdhDefaults	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.DefaultRule
cmdhLimits	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.LimitRules
cmdhNetworkAccessRules	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.NetworkAccessECRules
cmdhBuffer	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.BufferRules

7.12.1 Resource [activeCmdhPolicy]

The Resource [activeCmdhPolicy] provides a link to the currently active set of CMDH policies, see clause D.12.1 of TS-0001 [1].

The Resource [activeCmdhPolicy] is mapped to the Enable parameter of the Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i} object.

The information of a Policy instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.1-1: Resource [activeCmdhPolicy]

Attribute Name of [activeCmdhPolicy]	X_oneM2M_org Parameter
cmdhPolicy	Device.X_oneM2M_org_CSE.{i}.CMDH.Policy.{i}.Enable At most one Policy instance shall be enabled at a time. As such the Policy instance that has the Enable parameter with a value of "True" is the active CMDH policy.

7.12.2 Resource [cmdhDefaults]

The Resource [cmdhDefaults] defines default CMDH policy values, see clause D.12.2 of TS-0001 [1].

The Resource [cmdhDefaults] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.Default.{i} object.

The Default instance shall be created using the Add Object RPC of TR-069 [4].

The Default instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a Default instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a Default instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.2-1: Resource [cmdhDefaults]

Attribute Name of [cmdhDefaults]	X_oneM2M_org Parameter
cmdhDefEcValue	Device.X_oneM2M_org_CSE.{i}.CMDH.Default.{i}.DefaultECRules
cmdhEcDefParamValues	Device.X_oneM2M_org_CSE.{i}.CMDH.Default.{i}.DefaultECParamRules

7.12.3 Resource [cmdhDefEcValues]

The Resource [cmdhDefEcValues] represents a value for the **ec** (event category) parameter of an incoming request, see clause D.12.3 of TS-0001 [1].

The Resource [cmdhDefEcValues] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i} object.

The DefaultECRule instance shall be created using the Add Object RPC of TR-069 [4].

The DefaultECRule instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a DefaultECRule instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a DefaultECRule instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.3-1: Resource [cmdhDefEcValues]

Attribute Name of [cmdhDefEcValues]	X_oneM2M_org Parameter
order	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.Order
defEcValue	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.EventCategory
requestOrigin	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.RequestOrigin
requestContext	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.RequestContext
requestContextNotification	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.RequestContextNotificationEnable
requestCharacteristics	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECRule.{i}.RequestCharacteristics

7.12.4 Resource [cmdhEcDefParamValues]

The Resource [cmdhEcDefParamValues] represents a specific set of default values for the CMDH related parameters **rqet** (request expiration timestamp), **rset** (result expiration timestamp), **oet** (operational execution time), **rp** (response persistence) and **da** (delivery aggregation) that are applicable for a given **ec** (event category) if these parameters are not specified in the request, see clause D.12.4 of TS-0001 [1].

The Resource [cmdhEcDefParamValues] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i} object.

The DefaultECParamRule instance shall be created using the Add Object RPC of TR-069 [4].

The DefaultECParamRule instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a DefaultECParamRule instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a DefaultECParamRule instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.4-1: Resource [cmdhEcDefParamValues]

Attribute Name of [cmdhEcDefParamValues]	X_oneM2M_org Parameter
applicableEventCategory	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.EventCategories
defaultRequestExpTime	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.RequestExpTime
defaultResultExpTime	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.ResultExpTime

Attribute Name of [cmdhEcDefParamValues]	X_oneM2M_org Parameter
defaultOpExecTime	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.OperationExecTime
defaultRespPersistence	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.ResponsePersistence
defaultDelAggregation	Device.X_oneM2M_org_CSE.{i}.CMDH.DefaultECParamRule.{i}.DeliveryAggregation

7.12.5 Resource [cmdhLimits]

The Resource [cmdhLimits] represents limits for CMDH related parameter values, see clause D.12.5 of TS-0001 [1].

The Resource [cmdhLimits] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i} object.

The Limit instance shall be created using the Add Object RPC of TR-069 [4].

The Limit instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a Limit instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a Limit instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.5-1: Resource [cmdhLimits]

Attribute Name of [cmdhLimits]	X_oneM2M_org Parameter
order	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.Order
requestOrigin	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.RequestOrigin
requestContext	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.RequestContext
requestContextNotification	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.RequestContextNotificationEnable
requestCharacteristics	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.RequestCharacteristics
limitsEventCategory	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.EventCategories
limitsRequestExpTime	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.RequestExpTime
limitsResultExpTime	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.ResultExpTime
limitsOpExecTime	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.OperationExecTime
limitsRespPersistence	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.ResponsePersistence
limitsDelAggregation	Device.X_oneM2M_org_CSE.{i}.CMDH.Limit.{i}.DeliveryAggregation

7.12.6 Resource [cmdhNetworkAccessRules]

The Resource [cmdhNetworkAccessRules] defines the usage of underlying networks for forwarding information to other CSEs during processing of CMDH-related requests in a CSE, see clause D.12.6 of TS-0001 [1].

The Resource [cmdhNetworkAccessRules] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessECRule.{i} object.

The NetworkAccessECRule instance shall be created using the Add Object RPC of TR-069 [4].

The NetworkAccessECRule instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a NetworkAccessECRule instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a NetworkAccessECRule instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.6-1: Resource [cmdhNetworkAccessRules]

Attribute Name of [cmdhNetworkAccessRules]	X_oneM2M_org Parameter
applicableEventCategories	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessECRule.{i}.EventCategories
cmdhNwAccessRule	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessECRule.{i}.NetworkAccessRules

7.12.7 Resource [cmdhNwAccessRule]

The Resource [cmdhNwAccessRule] define limits in usage of specific underlying networks for forwarding information to other CSEs during processing of CMDH-related requests, see clause D.12.7 of TS-0001 [1].

The Resource [cmdhNwAccessRule] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessECRule.{i} object.

The NetworkAccessRule instance shall be created using the Add Object RPC of TR-069 [4].

The NetworkAccessRule instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a NetworkAccessRule instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a NetworkAccessRule instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.7-1: Resource [cmdhNwAccessRule]

Attribute Name of [cmdhNwAccessRule]	X_oneM2M_org Parameter
targetNetwork	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.TargetNetworks
minReqVolume	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.MinimumReqVolume
backOffParameters	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.BackoffTime
	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.BackoffTimeIncrement
	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.MaximumBackoffTime
otherConditions	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.OtherConditions
allowedSchedule	Device.X_oneM2M_org_CSE.{i}.CMDH.NetworkAccessRule.{i}.AllowedSchedule

7.12.8 Resource [cmdhBuffer]

The Resource [cmdhBuffer] represents limits in usage of buffers for temporarily storing information that needs to be forwarded to other CSEs during processing of CMDH-related requests in a CSE, see clause D.12.8 of TS-0001 [1].

The Resource [cmdhBuffer] is a multi-instance Resource where each instance of the Resource shall map to an instance of Device.X_oneM2M_org_CSE.{i}.CMDH.Buffer.{i} object.

The Buffer instance shall be created using the Add Object RPC of TR-069 [4].

The Buffer instance shall be deleted using the Delete Object RPC of TR-069 [4].

The information of a Buffer instance shall be retrieved using the GetParameterValues RPC of TR-069 [4].

The information of a Buffer instance shall be updated using the SetParameterValues RPC of TR-069 [4].

Table 7.12.8-1: Resource [cmdhBuffer]

Attribute Name of [cmdhBuffer]	X_oneM2M_org Parameter
applicableEventCategory	Device.X_oneM2M_org_CSE.{i}.CMDH.Buffer.{i}.EventCategories
maxBufferSize	Device.X_oneM2M_org_CSE.{i}.CMDH.Buffer.{i}.MaximumBufferSize
storagePriority	Device.X_oneM2M_org_CSE.{i}.CMDH.Buffer.{i}.StoragePriority

7.13 Resource Type <mgmtCmd>

Each mgmtCmd Resource shall map to BBF TR-069 RPC commands based on the value of cmdType. Accordingly, execReqArgs shall contain arguments related to the corresponding BBF TR-069 RPCs. The details about corresponding procedure mapping are described in section 8.2.

Table 7.13-1: Resource Type <mgmtCmd>

Attribute cmdType of mgmtCmd	Attribute execReqArgs of mgmtCmd
cmdType = RESET	Shall include all arguments related to BBF FactoryReset RPC
cmdType = REBOOT	Shall include all arguments related to BBF Reboot RPC
cmdType = UPLOAD	Shall include all arguments related to BBF Reboot RPC
cmdType = DOWNLOAD	Shall contain all arguments related to BBF Reboot RPC
cmdType = SOFTWAREINSTALL	Shall contain all arguments related to BBF ChangeDUState RPC which shall contain "InstallOpStruct" structure.
cmdType = SOFTWAREUNINSTALL	Shall contain all arguments related to BBF ChangeDUState RPC which shall contain "UninstallOpStruct" structure.

7.14 Resource Type <execInstance>

The <execInstance> resource from TS-0004 [2] shall map to BBF CancelTransfer RPC commands when it is disabled/cancelled using a Update operation or deleted using a Delete operation. The details are described in section 8.2.

8 Mapping of procedures for management

This clause contains all information on how to map management resource primitives from TS-0004 [2] to the Remote Procedure Calls (RPCs) in TR-069 [4].

8.1 Resource Type <mgmtObj> primitive mappings

This clause contains all information on how to map Resource Type <mgmtObj> primitives from TS-0004 [2] to the Remote Procedure Calls (RPCs) in TR-069 [4].

8.1.1 Alias-Based Addressing Mechanism

In order to utilize the Alias-Based Addressing Mechanism, the mechanism has to be supported by the ACS and CPE in order to map the M2M Service Layer identifier for the Resource instance to the CPE object instance. If the Alias-Based Addressing Mechanism feature is not supported by either the ACS or CPE, the CSE has to retain the mapping of the these M2M Resource instance identifiers.

8.1.2 Create primitive mapping

The Create Request and Response primitives shall map to the AddObject RPC. The AddObject RPC is defined in TR-069 [4] as a synchronous RPC and returns a successful response or one of the following fault codes in Table 8.1.2-1.

Table 8.1.2-1: AddObject Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST
9005	Invalid Parameter name (associated with Set/GetParameterValues, GetParameterNames, Set/GetParameterAttributes, AddObject, and DeleteObject)	STATUS_NOT_IMPLEMENTED

8.1.2.1 M2M Service Layer Resource Instance Identifier mapping

When the Resource is a multi-instance Resource, the AddObject RPC should utilize the Alias-Based Addressing Mechanism as defined in Section 3.6.1 of TR-069 [4] in order to use the Resource instance value of the URI.

8.1.3 Delete primitive mapping

8.1.3.1 Delete primitive mapping for deletion of Object Instances

The Delete Request and Response primitives that results in the deletion of a Resource shall map to the DeleteObject RPC. The DeleteObject RPC is defined in TR-069 [4] as a synchronous RPC and returns a successful response or one of the following fault codes in Table 8.1.3.1-1.

Table 8.1.3.1-1: DeleteObject Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9005	Invalid Parameter name (associated with Set/GetParameterValues, GetParameterNames, Set/GetParameterAttributes, AddObject, and DeleteObject)	STATUS_NOT_IMPLEMENTED

8.1.3.2 Delete primitive mapping for software un-install operation

The Delete Request and Response primitives that results in a software un-install operation (e.g., Resource [software]) shall use the ChangeDUState mechanism defined in TR-069 [4]. The ChangeDUState mechanism is an asynchronous command that consists of the synchronous ChangeDUState RPC for the un-installation request and the asynchronous ChangeDUStateComplete RPC. The ChangeDUState RPC returns a successful response or one of the following fault codes in Table 8.1.3.2-1. A successful response means that the CPE has accepted the ChangeDUState RPC.

Table 8.1.3.2-1: ChangeDUState Fault Code Mapping

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST

Once the CPE has attempted to change the state of the deployment unit, the CPE reports the result of the state change operation using the ChangeDUStateComplete RPC. The ChangeDUStateComplete RPC indicates a successful operation or one of the following fault codes in Table 8.1.3.2-2.

Table 8.1.3.2-2: ChangeDUStateComplete Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9022	Invalid UUID Format (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_BAD_REQUEST
9023	Unknown Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_BAD_REQUEST
9024	Disabled Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_BAD_REQUEST
9025	Deployment Unit to Execution Environment Mismatch (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_BAD_REQUEST
9026	Duplicate Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_BAD_REQUEST
9027	System Resources Exceeded (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_BAD_REQUEST
9028	Unknown Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update and Uninstall)	STATUS_BAD_REQUEST
9029	Invalid Deployment Unit State (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update and Uninstall)	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9030	Invalid Deployment Unit Update – Downgrade not permitted (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST
9031	Invalid Deployment Unit Update – Version not specified (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST
9032	Invalid Deployment Unit Update – Version already exists (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST

8.1.4 Update primitive mapping

8.1.4.1 Update primitive mapping for Parameter modifications

The Update Request and Response primitives that modifies the value of Resource attributes shall map to the SetParameterValues RPC. The SetParametersValue RPC is defined in TR-069 [4] as a synchronous RPC and returns a successful response or one of the following fault codes in Table 8.1.4.1-1.

Table 8.1.4.1-1: SetParameterValues Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST
9005	Invalid Parameter name (associated with Set/GetParameterValues, GetParameterNames, Set/GetParameterAttributes, AddObject, and DeleteObject)	STATUS_NOT-IMPLEMENTED
9006	Invalid Parameter type (associated with SetParameterValues)	STATUS_BAD_REQUEST
9007	Invalid Parameter value (associated with SetParameterValues)	STATUS_BAD_REQUEST
9008	Attempt to set a non-writable Parameter (associated with SetParameterValues)	STATUS_BAD_REQUEST

8.1.4.2 Update primitive mapping for upload file transfer operations

The Update Request and Response primitives that results in an upload file transfer operation (e.g., logStop attribute of the Resource [eventLog]) shall use the Upload mechanism defined in TR-069 [4]. The Upload mechanism is an asynchronous command that consists of the synchronous Upload RPC for the Upload and the asynchronous TransferComplete RPC. The Upload RPC returns a successful response or one of the following fault codes in Table 8.1.4.2-1. A successful response means that the CPE has accepted the Upload RPC.

Table 8.1.4.2-1: Upload Fault Code Mapping

Fault code	Description	Response Status Code

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST
9011	Upload failure (associated with Upload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST

Once the CPE has attempted to upload the file, the CPE reports the result of the Upload operation using the TransferComplete RPC. The TransferComplete RPC indicates a successful operation or one of the following fault codes in Table 8.1.4.2-2.

Table 8.1.4.2-2: TransferComplete Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9011	Upload failure (associated with Upload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9014	File transfer failure: unable to join multicast group (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9019	File transfer failure: file authentication failure (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9020	File transfer failure: unable to complete download within specified time windows (associated with TransferComplete method).	STATUS_BAD_REQUEST

8.1.4.3 Update primitive mapping for download file transfer operations

The Update Request and Response primitives that results in a download file transfer operation (e.g., update attribute of Resource [firmware]) shall use the Download mechanism defined in TR-069 [4]. The Download mechanism is an asynchronous command that consists of the synchronous Download RPC for the Download and the asynchronous TransferComplete RPC. The Download RPC returns a successful response or one of the following fault codes in Table 8.1.4.3-1. A successful response means that the CPE has accepted the Download RPC.

Table 8.1.4.3-1: Download Fault Code Mapping

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST

Once the CPE has attempted to download the file, the CPE reports the result of the download operation using the TransferComplete RPC. The TransferComplete RPC indicates a successful operation or one of the following fault codes in Table 8.1.4.3-2.

557 **Table 8.1.4.3-2: TransferComplete Fault Code Mapping**

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9011	Upload failure (associated with Upload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9014	File transfer failure: unable to join multicast group (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9019	File transfer failure: file authentication failure (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST
9020	File transfer failure: unable to complete download within specified time windows (associated with TransferComplete method).	STATUS_BAD_REQUEST

558

559 ### 8.1.4.4 Update primitive mapping for reboot operation

560 The Update Request and Response primitives that results in a reboot operation (e.g., reboot attribute of Resource
561 [reboot]) shall use the Reboot RPC defined in TR-069 [4]. The Reboot RPC is asynchronous command. The Reboot
562 RPC returns a successful response or one of the following fault codes in Table 8.1.4.4-1.

563 **Table 8.1.4.4-1: Reboot Fault Code Mapping**

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST

564

565 8.1.4.5 Update primitive mapping for factory reset operation

566 The Update Request and Response primitives that results in a factory reset operation (e.g., factoryReset attribute of
567 Resource [reboot]) shall use the FactoryReset RPC defined in TR-069 [4]. The FactoryReset RPC is an asynchronous
568 command. The FactoryReset RPC returns a successful response or one of the following fault codes in Table 8.1.4.5-1.

569 **Table 8.1.4.5-1: FactoryReset Fault Code Mapping**

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST

570

571 8.1.4.6 Update primitive mapping for software install operation

572 The Update Request and Response primitives that results in a software installation operation (e.g., install attribute of
573 Resource [software]) shall use the ChangeDUState mechanism defined in TR-069 [4]. The ChangeDUState mechanism
574 is an asynchronous command that consists of the synchronous ChangeDUState RPC for the download and the
575 asynchronous ChangeDUStateComplete RPC. The ChangeDUState RPC returns a successful response or one of the
576 following fault codes in Table 8.1.4.6-1. A successful response means that the CPE has accepted the ChangeDUState
577 RPC.

578 **Table 8.1.4.6-1: ChangeDUState Fault Code Mapping**

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST

579

580 Once the CPE has attempted to change the state of the deployment unit, the CPE reports the result of the state change
581 operation using the ChangeDUStateComplete RPC. The ChangeDUStateComplete RPC indicates a successful operation
582 or one of the following fault codes in Table 8.1.4.6-2.

583 **Table 8.1.4.6-2: ChangeDUStateComplete Fault Code Mapping**

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_BAD_REQUEST
9022	Invalid UUID Format (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_BAD_REQUEST
9023	Unknown Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_BAD_REQUEST
9024	Disabled Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_BAD_REQUEST
9025	Deployment Unit to Execution Environment Mismatch (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_BAD_REQUEST
9026	Duplicate Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_BAD_REQUEST
9027	System Resources Exceeded (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_BAD_REQUEST
9028	Unknown Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update and Uninstall)	STATUS_BAD_REQUEST
9029	Invalid Deployment Unit State (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update and Uninstall)	STATUS_BAD_REQUEST

Fault code	Description	Response Status Code
9030	Invalid Deployment Unit Update – Downgrade not permitted (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST
9031	Invalid Deployment Unit Update – Version not specified (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST
9032	Invalid Deployment Unit Update – Version already exists (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_BAD_REQUEST

8.1.5 Retrieve primitive mapping

The Retrieve Request and Response primitives shall map to the GetParameterValues RPC. The GetParametersValue RPC is defined in TR-069 [4] as a synchronous RPC and returns a successful response or one of the following fault codes in Table 8.1.5-1.

Table 8.1.5-1: GetParameterValues Fault Code Mapping

Fault code	Description	Response Status Code
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot used to indicate Parameters in error)	STATUS_BAD_REQUEST
9005	Invalid Parameter name (associated with Set/GetParameterValues, GetParameterNames, Set/GetParameterAttributes, AddObject, and DeleteObject)	STATUS_BAD_REQUEST

8.1.6 Notify primitive mapping

The NotifyRequest and Response primitives permit notifications to AE or CSEs that have subscribed to a Resource.

While TR-069 [4] has the capability to notify the subscribed ACS when an object's parameter has been modified, TR-069 [4] does not have the capability for an ACS to be notified if any parameter within the object has been modified unless the ACS individually subscribes to all the parameters of the object.

As such the procedure for mapping the Notify Request and Response primitives for TR-069 [4] is not possible unless the CSE subscribes to receive notification to all the parameters of an Object that are mapped to the Resource's attributes.

Note: In many implementations, subscribing to all the parameters of an Object that are mapped to the Resource can cause performance issues in the CPE as well as the CSE. As such using the attribute based subscription capabilities of TR-069 [4] for subscription of Resources should be avoided when possible.

8.1.6.1 Procedure for subscribed Resource attributes.

When a <subscription> Resource for a <mgmtObj> Resource is Created, Deleted or Updated the CSE shall map to the SetParameterAttributes RPC in the following manner:

- TR-069 [4] provides the capability to subscribe to changes of a specific attribute through the use of the SetParameterAttributes RPC using the "Active" value for the Notification parameter.

- TR-069 [4] provides the capability to un-subscribe to changes of a specific attribute through the use of the SetParameterAttributes RPC using the "None" value for the Notification parameter.

The SetParametersAttributes RPC is defined in TR-069 [4] as a synchronous RPC and returns a successful response or one of the following fault codes in Table 8.1.6.1-1.

Table 8.1.6.1-1: SetParameterAttributes Fault Code Mapping

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_BAD_REQUEST
9001	Request denied (no reason specified)	STATUS_BAD_REQUEST
9002	Internal error	STATUS_BAD_REQUEST
9003	Invalid arguments	STATUS_BAD_REQUEST
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_BAD_REQUEST
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_BAD_REQUEST

8.1.6.2 Notification primitive mapping

Notify Request and Response primitives shall map to the TR-069 notification mechanism. CPEs produce notifications for subscribed attributes using the TR-069 Inform method, the Inform method has an argument Event that has as one of the EventCodes with the value "4 VALUE CHANGE" indicating that a subscribed parameter's value has changed. The parameter(s) that have changed are included ParameterList argument of the Inform method.

The ParameterList argument is list of name-value pairs; the name is parameter name and shall be mapped to the objectPath attribute of the Resource while the value is the most recent value of the parameter.

Note: TR-069 CPEs do not report value changes of parameters that were modified by the ACS.

8.2 <mgmtCmd> and <execInstance> resource primitive mappings

8.2.1 Update (Execute) primitive for the <mgmtCmd> resource

When the Update Request primitive for <mgmtCmd> resource addresses the execEnable attribute of the <mgmtCmd> resource, it effectively triggers an Execute <mgmtCmd> procedure..

The Hosting CSE performs command conversion of its <execInstance> sub-resources. The mapping between the <execInstance> attributes and the TR-069 [4] RPC procedures triggered is based on the value of the cmdType attribute of the <mgmtCmd> resource defined in Table 8.2.1-1. The CPE acceptance of the corresponding RPC procedures is indicated by returning a successful Response primitive to the initial Update Request.

The Fault Codes which may be returned by the CPE to the Hosting CSE are mapped onto execStatus codes and stored in the corresponding <execInstance> attributes, and are detailed in the following sub-sections

Table 8.2.1-1 Mapping of Execute <mgmtCmd> primitives to BBF TR-069 RPC

cmdType value	BBF TR-069 RPCs
"DOWNLOAD"	Download RPC (see section 8.2.1.1) and TransferComplete RPC (section 8.2.1.3)
"UPLOAD"	Upload RPC (section 8.2.1.2) and TransferComplete RPC (section 8.2.1.3)
"SOFTWAREINSTALL"	ChangeDUState RPC (section 8.2.1.4) and ChangeDUStateComplete RPC (section 8.2.1.5)
"SOFTWAREUNINSTALL"	ChangeDUState RPC (section 8.2.1.4) and ChangeDUStateComplete RPC (section 8.2.1.5)
"REBOOT"	Reboot RPC (section 8.2.1.6)
"RESET"	Factory reset RPC (section 8.2.1.7)

8.2.1.1 Execute File Download

The download file transfer operation may use the Download mechanism defined in TR-069 [4]. The Download mechanism is an asynchronous command which returns a successful response or one of the following fault codes mapped onto execStatus values as detailed in Table 8.2.1.1-1. A successful response to the Update primitive triggering the Execute procedure means that the CPE has accepted the Download RPC.

Table 8.2.1.1-1: Download Fault Code Mapping

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST_DENIED
9002	Internal error	STATUS_INTERNAL_ERROR
9003	Invalid arguments	STATUS_INVALID_ARGUMENTS

Fault code	Description	execStatus Code
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_RESOURCES_EXCEEDED
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_FILE_TRANSFER_FAILED
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods, not associated with Scheduled Download method).	STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_UNSUPPORTED_PROTOCOL

8.2.1.2 Execute File Upload Operations

The upload file transfer operation shall use the Upload mechanism defined in TR-069 [4]. The Upload mechanism is an asynchronous command that consists of the synchronous Upload RPC for the Upload and the asynchronous TransferComplete RPC. The Upload RPC returns a successful response or one of the following fault codes mapped onto execStatus values as detailed in Table 8.2.1.2-1. A successful response to the Update primitive triggering the execute procedure means that the CPE has accepted the Upload RPC in Table 8.2.1.2-1.

Table 8.2.1.2-1: Upload Fault Code Mapping

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9002	Internal error	STATUS_INTERNAL_ERROR
9003	Invalid arguments	STATUS_INVALID_ARGUMENTS
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_RESOURCES_EXCEEDED
9011	Upload failure (associated with Upload, TransferComplete or AutonomousTransferComplete methods).	STATUS_UPLOAD_FAILED

Fault code	Description	execStatus Code
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_UNSUPPORTED_PROTOCOL

650

651 8.2.1.3 Report Results using TransferComplete RPC

652 After a File Download or Upload has been attempted, the result of the operation is reported using the TransferComplete
653 RPC. The TransferComplete RPC indicates a successful operation or one of the following fault codes mapped onto
654 execStatus values in Table 8.2.1.3-2.

655

656 **Table 8.2.1.3-2: TransferComplete Fault Code Mapping**

Fault code	Description	execStatus Code
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9002	Internal error	STATUS_INTERNAL_ERROR
9010	File transfer failure (associated with Download, ScheduleDownload, TransferComplete or AutonomousTransferComplete methods).	STATUS_FILE_TRANSFER_FAILED
9011	Upload failure (associated with Upload, TransferComplete or AutonomousTransferComplete methods).	STATUS_UPLOAD_FAILED
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE
9014	File transfer failure: unable to join multicast group (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_FILE_TRANSFER_FAILED_MULTICAST_GROUP_UNABLE_JOIN
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_SERVER_CONTACT_FAILED
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_FILE_ACCESS_FAILED

Fault code	Description	execStatus Code
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_DOWNLOAD_INCOMPLETE
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_FILE_CORRUPTED
9019	File transfer failure: file authentication failure (associated with Download, TransferComplete or AutonomousTransferComplete methods).	STATUS_FILE_TRANSFER_FILE_AUTHENTICATION_FAILURE
9020	File transfer failure: unable to complete download within specified time windows (associated with TransferComplete method).	STATUS_FILE_TRANSFER_WINDOW_EXCEEDED

8.2.1.4 Execute Software Operations with ChangeDUState RPC

The software installation and uninstall operations shall use the ChangeDUState mechanism defined in TR-069 [4]. The ChangeDUState mechanism is an asynchronous command that consists of the synchronous ChangeDUState RPC and returns a successful response or one of the fault codes mapped onto execStatus values as detailed in Table 8.2.1.4.-1. A successful response to the Update primitive triggering the Execute procedure means that the CPE has accepted the ChangeDUState RPC.

Table 8.2.1.4-1: ChangeDUState Fault Code Mapping

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9002	Internal error	STATUS_INTERNAL_ERROR
9004	Resources exceeded (when used in association with SetParameterValues, this cannot be used to indicate Parameters in error)	STATUS_RESOURCES_EXCEEDED

8.2.1.5 Report Results with ChangeDUStateComplete RPC

After software installation and uninstall operations using a ChangeDUState mechanism as defined in TR-069 [4], the result of the state change operation is retrieved using the ChangeDUStateComplete RPC. The ChangeDUStateComplete RPC indicates a successful operation or one of the fault codes mapped onto execStatus values as detailed in Table 8.2.1.5.-1.

Table 8.2.1.5-1: ChangeDUStateComplete Fault Code Mapping

Fault code	Description	execStatus Code
9001	Request denied (no reason specified)	STATUS_REQUEST_DENIED

Fault code	Description	execStatus Code
9003	Invalid arguments	STATUS_INVALID_ARGUMENTS
9012	File transfer server authentication failure (associated with Upload, Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_SERVER_AUTHENTICATION_FAILURE
9013	Unsupported protocol for file transfer (associated with Upload, Download, ScheduleDownload, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_UNSUPPORTED_PROTOCOL
9015	File transfer failure: unable to contact file server (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_SERVER_CONTACT_FAILED
9016	File transfer failure: unable to access file (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_FILE_ACCESS_FAILED
9017	File transfer failure: unable to complete download (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_DOWNLOAD_INCOMPLETE
9018	File transfer failure: file corrupted or otherwise unusable (associated with Download, TransferComplete, AutonomousTransferComplete, DUStateChangeComplete, or AutonomousDUStateChangeComplete methods).	STATUS_FILE_TRANSFER_FAILED_FILE_CORRUPTED
9022	Invalid UUID Format (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_INVALID_UUID_FORMAT
9023	Unknown Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_UNKNOWN_EXECUTION_ENVIRONMENT
9024	Disabled Execution Environment (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update, and Uninstall)	STATUS_DISABLED_EXECUTION_ENVIRONMENT
9025	Deployment Unit to Execution Environment Mismatch (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_EXECUTION_ENVIRONMENT_MISMATCH
9026	Duplicate Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install only)	STATUS_DUPLICATE_DEPLOYMENT_UNIT
9027	System Resources Exceeded (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install and Update)	STATUS_SYSTEM_RESOURCES_EXCEEDED
9028	Unknown Deployment Unit (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update and Uninstall)	STATUS_UNKNOWN_DEPLOYMENT_UNIT

Fault code	Description	execStatus Code
9029	Invalid Deployment Unit State (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Install, Update and Uninstall)	STATUS_INVALID_DEPLOYMENT_UNIT_STATE
9030	Invalid Deployment Unit Update – Downgrade not permitted (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_DOWNGRADE_DISALLOWED
9031	Invalid Deployment Unit Update – Version not specified (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_UPGRADE_DISALLOWED
9032	Invalid Deployment Unit Update – Version already exists (associated with DUStateChangeComplete or AutonomousDUStateChangeComplete methods: Update only)	STATUS_INVALID_DEPLOYMENT_UNIT_UPDATE_VERSION_EXISTS

8.2.1.6 Execute Reboot operation

The reboot operation shall use the Reboot RPC defined in TR-069 [4]. The Reboot RPC is a synchronous command. A successful response to the Update primitive triggering the Execute procedure means that the CPE has accepted the Reboot RPC. The Reboot RPC returns a successful response or one of the fault codes mapped onto execStatus values as detailed in Table 8.2.1.6-1.

Table 8.2.1.6-1: Reboot Fault Code Mapping

Fault code	Description	execStatus Code
9001	Request denied (no reason specified)	STATUS_REQUEST_DENIED
9002	Internal error	STATUS_INTERNAL_ERROR
9003	Invalid arguments	STATUS_INVALID_ARGUMENTS

8.2.1.7 Execute Factory Reset operation

The factory reset operation shall use the FactoryReset RPC defined in TR-069 [4]. The FactoryReset RPC is a synchronous command. A successful response to the Update primitive triggering the Execute procedure means that the CPE has accepted the FactoryReset RPC. The FactoryReset RPC returns a successful response or one of the fault codes mapped onto execStatus values as detailed in Table 8.2.1.7-1.

Table 8.2.1.7-1: FactoryReset Fault Code Mapping

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST_DENIED
9002	Internal error	STATUS_INTERNAL_ERROR

Fault code	Description	execStatus Code
9003	Invalid arguments	STATUS_INVALID_ARGUMENTS

686

687 8.2.2 Delete <mgmtCmd> resource primitive mapping

688 The Delete Request primitive for the <mgmtCmd> resource may initiate TR-069 [4] RPC commands for the
689 corresponding <execInstance> sub-resources as follows:

690 • If there are no <execInstance> sub-resources with RUNNING execStatus, a successful response to the Delete
691 primitive is returned and the <mgmtCmd> resource is deleted without triggering any TR-069 [4] RPCs.

692 • If there are <execInstance> sub-resources with RUNNING execStatus that resulted in cancellable TR-069 [4]
693 RPCs (e.g. File Upload and File Download RPCs), a TR-069[4] CancelTransfer RPC shall be initiated for
694 each cancellable operation. Upon completion of all the cancellation operations, if any fault codes are returned
695 by the CPE, an unsuccessful Response to the Delete primitive with status code "Delete mgmtCmd-
696 execInstance cancellation error" is returned, and the <mgmtCmd> resource is not deleted. The execStatus
697 attribute of each specific <execInstance> is set to CANCELLED for successful RPCs or is determined from
698 the RPC fault codes as detailed in Table 8.2.2-1. If all cancellation operations are successful on the managed
699 entity, a successful Response to the Delete primitive is returned and the <mgmtCmd> resource is deleted.

700 • If there is at least one <execInstance> sub-resource with RUNNING execStatus that resulted in non-
701 cancellable TR-069 [4] RPCs (e.g. RPCs other than File Upload and File Download RPCs), the execStatus
702 attribute of the specific <execInstance> is changed to STATUS_NON_CANCELLABLE. An unsuccessful
703 Response to the Delete primitive with status code "Delete mgmtCmd- execInstance cancellation error" is
704 returned and the <mgmtCmd> resource is not deleted.

705 **Table 8.2.2-1: CancelTransfer Fault Code Mapping for Delete <mgmtCmd>**

Fault code	Description	Response Status Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9021	Cancelation of file transfer not permitted in current transfer state	STATUS_CANCELLATION_DENIED

706

707 8.2.3 Update (Cancel) <execInstance> primitive mapping

708 When the Update Request primitive for an <execInstance> sub-resource addresses the execDisable attribute of the
709 <execInstance > sub-resource, it effectively triggers a Cancel <execInstance> resource procedure.

710 The hosting CSE determines weather the <execInstance> resource has a RUNNING execStatus and weather the
711 resulting TR-069 [4] RPCs are cancellable. Currently, only the TR-069 File Upload and File Download RPCs are
712 cancellable using the TR-069 [4] CancelTransfer RPC.

713 • If the addressed <execInstance> sub-resource has an execStatus other than RUNNING, an un-successful
714 Response to the Update primitive is returned with status code "Cancel execInstance – already complete".

715 • If the addressed <execInstance> sub-resources has RUNNING execStatus and resulted in cancellable TR-069
716 [4] RPCs (e.g. File Upload and File Download RPCs), a BBF TR-069 [4] CancelTransfer RPC shall be
717 initiated. For a successful CancelTransfer RPC the execStatus attribute of the specific <execInstance> is set to
718 CANCELLED and a successful Response is sent to the Update primitive. For an unsuccessful CancelTransfer
719 RPC the execStatus attribute is determined from the RPC fault codes as detailed in Table 8.2.3-1 and an

unsuccessful Response is sent to the Update primitive with status code "Cancel execInstance – cancellation error".

- If the addressed <execInstance> sub-resources has RUNNING execStatus and resulted non-cancellable TR-069 [4] RPCs (e.g. RPCs other than File Upload and File Download RPCs), the execStatus attribute of the specific <execInstance> is changed to STATUS_NON_CANCELLABLE. An unsuccessful Response is sent to the Update primitive with status code "Cancel execInstance – not cancellable"

.

Table 8.2.3-1: CancelTransfer Fault Code Mapping for Update (Cancel) <execInstance>

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9021	Cancelation of file transfer not permitted in current transfer state	STATUS_REQUEST_UNSUPPORTED

8.2.4 Delete <execInstance> primitive mapping

The Delete Request primitive for an <execInstance> sub-resource may initiate TR-069 [4] RPC commands for the corresponding <execInstance> sub-resources as follows:

- If the addressed <execInstance> sub-resource has an execStatus other than RUNNING, an successful Response to the Delete primitive is returned and the <execInstance> sub-resource is deleted without triggering any TR-069 [4] RPCs.

- If the addressed <execInstance> sub-resource has RUNNING execStatus and resulted in cancellable TR-069 [4] RPCs (e.g. File Upload and File Download RPCs), a BBF TR-069 [4] CancelTransfer RPC shall be initiated. For a successful CancelTransfer RPC a successful response is sent to the Delete primitive and the <execInstance> sub-resource is deleted. For an unsuccessful CancelTransfer RPC the execStatus attribute is determined from the RPC fault codes as detailed in Table 8.2.4-1 and an unsuccessful Response is sent to the Delete primitive with status code "Delete execInstance – cancellation failed".

- If the addressed <execInstance> sub-resource has RUNNING execStatus and resulted non-cancellable TR-069 [4] RPCs (e.g. RPCs other than File Upload and File Download RPCs), the execStatus attribute is set to STATUS_NOT_CANCELLABLE and an unsuccessful Response is sent to the Update primitive with status code "Delete execInstance – not cancellable"

Table 8.2.4-1: CancelTransfer Fault Code Mapping for Delete <execInstance>

Fault code	Description	execStatus Code
9000	Method not supported	STATUS_REQUEST_UNSUPPORTED
9001	Request denied (no reason specified)	STATUS_REQUEST DENIED
9021	Cancelation of file transfer not permitted in current transfer state	STATUS_CANCELLATION_DENIED

9 Server Interactions

This clause specifies how the IN-CSE interacts with an ACS in order to manage the Resources described in this specification. The IN-CSE interaction with an ACS includes:

- Establishment of the communication session between the IN-CSE and ACS
- Processing of requests and notifications between the IN-CSE and the ACS
- Discovery

Note: The Broadband Forum has not defined a protocol specification for the Northbound Interface of an ACS. As such, this document only describes the expectations of this interface in the form of requirements on the ACS.

9.1 Communication Session Establishment

9.1.1 IN-CSE to ACS Communication Session Establishment

When the IN-CSE detects that it has to delegate an interaction with a device resource to an ACS, the IN-CSE establishes a communication session with the ACS. The establishment of a communication session between the IN-CSE and ACS provides security dimensions for Access control, Authentication, Non-repudiation, Data confidentiality, Communication security, Data integrity and Privacy adhering to the following TR-131 [7] Architectural requirement A7.

The IN-CSE may establish multiple sessions with an ACS based on the security model utilized between the IN-CSE and the ACS.

9.1.2 ACS to IN-CSE Communication Session Establishment

When the ACS detects a change to resources it manages that the IN-CSE has expressed interest, the ACS requests the IN-CSE to establish a session if a session doesn't exist for the resource being managed. The establishment of a communication session between the IN-CSE and ACS provides security dimensions for Access control, Authentication, Non-repudiation, Data confidentiality, Communication security, Data integrity and Privacy adhering to the following TR-131 [7] Architectural requirement A7.

The ACS may establish multiple sessions with an IN-CSE based on the security model utilized between the IN-CSE and the ACS.

While a session between the ACS and IN-CSE is not established, the AS retains any notifications or changes in the resources based on an Event retention policy (i.e., time, number of events).

When an ACS to IN-CSE interaction is required and a session does not exist, the ACS requests to initiate a session based on a Session Initiation Policy (i.e., Periodic contact establishment (schedule), upon event detection with timeframe window).

9.2.3 ACS and IN-CSE Communication Session Requirements

When establishing a session from the ACS to the IN-CSE:

- If a session doesn't exist between the IN-CSE and ACS, the ACS shall retain any notifications or changes in the resources based on an Event retention policy (i.e., time, number of events).
- When an ACS to IN-CSE interaction is required and a session does not exist, the ACS shall be capable to initiate a session based on a Session Initiation Policy (i.e., Periodic contact establishment (schedule), upon event detection with timeframe window)

9.2 Processing of Requests and Responses

9.2.1 Request and Notification Formatting

Requests and Notifications mechanisms between the IN-CSE and the DM Server format the XML schema of the CPE methods defined in TR-069 [4] as an ACS would format the CPE methods that it would pass to the CPE. The IN-CSE would then also process the CPE methods as defined in TR-069 [4]. Likewise the ACS would send notifications in the format of the XML schema of the CPE for sending events using the Inform RPC.

9.2.2 ACS Request Processing Requirements

When receiving requests from the IN-CSE the ACS shall be capable of defining mechanisms to support triggering of immediate operations to device. If the device is not available the ACS returns an appropriate error code.

The ACS shall provide capability for the IN-CSE to indicate request policies to include: Retry policy, Request Time out.

9.2.3 ACS Notification Processing Requirements

When sending notifications to the IN-CSE:

- The ACS shall be capable of providing a mechanism for the IN-CSE to subscribe to events.
- The ACS shall be capable of providing a list of events for which the IN-CSE can subscribe.
- The ACS shall be capable of providing a mechanism for the IN-CSE to unsubscribe from events.
- The ACS shall be capable of providing an event delivery mechanism.
- The ACS shall be capable of providing the capability for the IN-CSE to request event filters including: Event Code; Specific parameters changing value; Device; Any combination of the previous criteria.
- The IN-CSE shall be capable of subscribing to be notified of changes to resources it manages.
- The ACS shall be capable of notifying the IN-CSE of changes to resources to which the client has subscribed.

9.3 Discovery and Synchronization of Resources

For devices under management, the IN-CSE may discover resources of interest (metadata and values) within a device using the ACS.

For resources of interest, the IN-CSE may also express an interest to be notified of a resource if a resource is changed (added, deleted, updated).

The IN-CSE shall be capable to discover and subscribe to changes of resources in order to synchronize the IN-CSE with resources of interest of the ACS.

9.4 Access Management

Once a request has performed an Access Decision by the IN-CSE to allow the request, the IN-CSE shall select the appropriate ACS along with elements the ACS would need to implement access management within the ACS. These would include the Identity of the subject (oneM2M Originator) of the request which is needed in scenarios where the original issuer of the request is needed to be known – this could be done by correlating principals (e.g., Roles, Accounts) used by the IN-CSE and ACS.

9.4.1 Access Management Requirements

- The ACS shall be capable of providing a mechanism for the IN-CSE to discover the Access Management elements used to authorize and authenticate access to resources controlled by the ACS.
- The IN-CSE shall be capable of correlating Access Management elements provided by the ACS to Access Management elements used by the IN-CSE.
- The IN-CSE shall be capable of providing secured storage of Access Management elements within the IN-CSE.

10 New Management Technology Specific Resources

TR-181 [6] provides a list of management objects that have been standardized by the Broadband Forum and where possible, clause 7 provides a mapping of the Resources to standardized management objects. This clause provides the oneM2M vendor specific extensions to the TR-181 [6] data model as specified in the ts-0006-1-1.xml.

History

Publication history		
V1.0.1	30 Jan 2015	Release 1 - Publication