TECHNICAL REPORT

TR-157
Component Objects for CWMP

Issue: 1
Issue Date: March 2009

Issue History

Issue Number	Issue Date	Issue Editor	Changes
1	March 2009	John Blackford, 2Wire, Inc. TimSpets, Westell	Original

Comments or questions about this Technical Report should be directed to:

Editors:	John Blackford	2Wire, Inc.	jblackford@2wire.com
	Tim Spets	Westell	tspets@westell.com
BroadbandHome™ WG Chairs	Greg Bathrick	PMC-Sierra	
	Heather Kirksey	Alcatel-Lucent	

Table of Contents

List of Tables

List of Figures

Summary

The architecture of TR-069 Amendment 2 [1] and TR-106 Amendment 1 [2] enables device management of CPE devices in the customer's home, including the home gateway, and devices behind it.

This Technical Report defines additional management objects for use in CWMP managed devices. The objects may exist at the top level of a hierarchy, or in some cases within an existing object. The objects are intended for use in all CWMP root objects (both Device and InternetGatewayDevice). The objects define varying functionality, diagnostics, etc., that are agnostic to the type of device.

The additional management objects defined in this Technical Report include the following:

Enhanced device diagnostic and monitoring capabilities
These enhanced features include the ability to monitor device memory and process status as well as reporting of temperature sensor status and alarms. Two diagnostic tests have also been added: Namespace Lookup and hardware-specific self-test.

Autonomous Transfer and Multi-cast Download Policy Configuration
This specification completes the additions to CWMP undertaken in collaboration with DVB to ensure TR-069's ability to meet the needs of IP video environments. In [1] capabilities for multi-cast download and autonomous transfers were added to the CWMP protocol; in this Technical Report, objects have been added for managing the policies around autonomous transfer reporting and configuring the multicast download availability.

Simple Firewall
Simple firewall management has been defined in this specification.

USB Hosts
This specification contains objects that enable the remote management of USB Hosts and policies for the behavior of attached USB devices.

UPnP and DLNA discovery
UPnP is a widely deployed home networking technology; DLNA digital home servers and digital home players use UPnP technology to provide content streaming and sharing across devices in the home. Objects defined in this specification enable the reporting of UPnP and DLNA devices and capabilities in the home network in order to give service providers increased visibility into the subscriber home.

Periodic Stats
The periodic stats object allows for the collection and reporting of performance monitoring data for TR-069 enabled devices.

In summary, these enhancements continue to ensure that CWMP is meeting the management needs of operators' next generation home networking services.

1 Purpose and Scope

1.1 Purpose

The purpose of this Technical Report is to provide Component Objects for CWMP.

A Component Object is defined as an object and their contained parameters intended for use in any applicable CWMP root data model (both Device and InternetGatewayDevice). The object(s) may reside at the top level or an appropriate sub-object level.

1.2 Scope

This Technical Report defines Component Objects for use in CWMP managed devices for all root data models. The current root data models are InternetGatewayDevice:1 defined in TR-098 Amendment 2 [3] and a Device:1 defined in TR-106 Amendment 2 [2].

Example of including an object at the Top level:

Device.*NSlookupDiagnostics.*
InternetGatewayDevice.*NSlookupDiagnostics.*

Example of including an object under a currently defined object:

Device.DeviceInfo.*MemoryStatus.*
InternetGatewayDevice.DeviceInfo.*MemoryStatus.*

The Parameter model in Section 4 is derived from an XML document that adheres to the CWMP DM schema as defined in Annex A of TR-106 Amendment 2 [2]. The XML can be found at http://www.broadband-forum.org/cwmp/tr-157-1-0-0.xml.

Sections containing "Theory of Operations" for Component Objects are located in the appendices.

2 References and Terminology

2.1 Conventions

In this Technical Report several words are used to signify the requirements of the specification. These words are often capitalized.

MUST	This word, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications must be understood and carefully weighted before choosing a different course.
MAY	This word, or the adjective "OPTIONAL", means that this item is one of an allowed set of alternatives. An implementation that does not include this option MUST be prepared to inter-operate with another implementation that does include the option.

2.2 References

The following references constitute provisions of this Technical Report. At the time of publication, the editions indicated were valid. All references are subject to revision; users of this Technical Report are therefore encouraged to investigate the possibility of applying the most recent edition of the references listed below. A list of currently valid Broadband Forum Technical Reports is published at www.broadband-forum.org.

[1] TR-069 Amendment 2, *CPE WAN Management Protocol,* Broadband Forum Technical Report, 2007

[2] TR-106 Amendment 2, *Data Model Template for TR-069-Enabled Devices,* Broadband Forum Technical Report, 2008

[3] TR-098 Amendment 2, *Internet Gateway Device Data Model for TR-069,* Broadband Forum Technical Report, 2008

[4] DLNA Networked Device Interoperability Guidelines, *DLNA Networked Device Interoperability Guidelines, Volume 2: Media Format Profiles.*, DLNA, October 2006, http://www.dlna.org/industry/certification/guidelines/

[5] TS 102 824, *Digital Video Broadcasting (DVB);Remote Management and Firmware Update System for DVB IP Services*, ETSI, July 2008, http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=27769

[6] ICSA Modular Firewall Certification Criteria, *Required Services Security Policy - Small/Medium Business (SMB) Category module - version 4.0*, ICSA Labs, http://www.icsalabs.com/icsa/docs/html/communities/firewalls/pdf/criteria/SMB.pdf

[7] ISO/IEC 13818-6:1998, *Information Technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC*, ISO, 1998, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25039

[8] Organizationally Unique Identifiers (OUIs), http://standards.ieee.org/faqs/OUI.html

[9] RFC 1323, *TCP Extensions for High Performance*, IETF, May 1992, http://www.ietf.org/rfc/rfc1323.txt

[10] RFC 2581, *TCP Congestion Control*, IETF, April 1999, http://www.ietf.org/rfc/rfc2581.txt

[11] RFC 2582, *The NewReno Modification to TCP's Fast Recovery Algorithm*, IETF, April 1999, http://www.ietf.org/rfc/rfc2582.txt

[12] RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*, IETF RFC, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[13] RFC 2818, *HTTP Over TLS*, IETF, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[14] RFC 2974, *Session Announcement Protocol*, IETF, October 2000, http://www.ietf.org/rfc/rfc2974.txt

[15] RFC 3066, *Tags for the Identification of Languages*, IETF RFC, http://www.ietf.org/rfc/rfc3066.txt

[16] RFC 3926, *FLUTE - File Delivery over Unidirectional Transport*, IETF, October 2004, http://www.ietf.org/rfc/rfc3926.txt

[17] RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, IETF RFC, http://www.ietf.org/rfc/rfc3986.txt

[18] RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, IETF RFC, July 2005, http://www.ietf.org/rfc/rfc4122.txt

[19] RFC 793, *Transmission Control Protocol*, IETF, September 1981, http://www.ietf.org/rfc/rfc793.txt

[20] UPnP Device Architecture, *UPnP Device Architecture 1.0*, UPnP Forum, April 2008, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf

[21] USB 1.0, *USB 1.0 Specification*, USB-IF, January 1996, http://www.usb.org/developers/docs/

[22] USB 2.0, *USB 2.0 Specification*, USB-IF, April 2000, http://www.usb.org/developers/docs/usb_20_122208.zip

[23] USB 3.0, *USB 3.0 Specification*, USB-IF, November 2008, http://www.usb.org/developers/docs/usb_30_spec.zip

2.3 Definitions

The following terminology is used throughout this Technical Report:

ACS	Auto-Configuration Server. This is a component in the broadband network responsible for auto-configuration of the CPE for advanced services.
CPE	Customer Premises Equipment; refers to any TR-069-compliant device and therefore covers both Internet Gateway devices and LAN-side end devices.
CWMP	CPE WAN Management Protocol. Defined in TR-069 Amendment 2 [1], CWMP is a communication protocol between an ACS and CPE that defines a mechanism for secure auto-configuration of a CPE and other CPE management functions in a common framework.

2.4 Abbreviations

This Technical Report defines the following abbreviations:

CPU	Central Processing Unit
DDD	Device Description Document
DLNA	Digital Living Network Alliance
DNS	Domain Name System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IGD	Internet Gateway Device
LAN	Local Area Network
NAT	Network Address Translation
QoS	Quality of Service
RAM	Random Access Memory
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TR	Technical Report
URL	Universal Resource Locator
USB	Universal Serial Bus
USB-IF	USB Implementer's Forum
USN	Unique Service Name
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
WAN	Wide Area Network
WG	Working Group
XML	Extensible Markup Language

3 Technical Report Impact

3.1 Energy Efficiency

TR-157 has no impact on energy efficiency.

3.2 IPv6

TR-157 has no impact on IPv6 support and compatibility.

3.3 Security

There are no relevant security issues relating to TR-157.

4 Data Model Definition

4.1 Parameter Definitions

To support the functionality defined in this specification, an extension to the Device data model and InternetGatewayDevice data model is specified in Table 1. For the Device data model, this extension is considered part of Device:1.3 (version 1.3 of the Device data model), which extends version 1.2 of the Device data model defined in TR-106 Amendment 2 [2]. For the InternetGatewayDevice data model, this extension is considered part of InternetGatewayDevice:1.5 (version 1.5 of the InternetGatewayDevice data model), which extends version 1.4 of the InternetGatewayDevice data model defined in TR-098 Amendment 2 [3].

Table 1 lists the Component Objects and their associated parameters. The notation used to indicate the data type of each parameter follows the notation defined in TR-106 Amendment 2 [2].

Table 1 Parameter Definitions of Component Objects

Name	Type	Write	Description	Object Default
UserNumberOfEntries	unsignedInt	-	Number of entries in the User table.	-
SmartCardReaderNumberOfEntries	unsignedInt	-	Number of entries in the SmartCardReader table.	-
.DeviceInfo.	object	-	This object contains general device information.	-
.DeviceInfo.MemoryStatus.	object	-	Status of the device's volatile physical memory.	-
Total	unsignedInt	-	The total physical RAM, in *kilobytes*, installed on the device.	-
Free	unsignedInt	-	The free physical RAM, in *kilobytes*, currently available on the device.	-
.DeviceInfo.ProcessStatus.	object	-	Status of the processes on the device.	-
CPUUsage	unsignedInt [:100]	-	The total amount of the CPU, in *percent*, rounded up to the nearest whole *percent*. In the case that multiple CPU are present, this value represents the average of all CPU.	-
ProcessNumberOfEntries	unsignedInt	-	Number of entries in the Process table. Since a Process can come and go very quickly, the CPE SHOULD place a locally specified limit on the frequency at which it will notify the ACS of value changes, as described in TR-069a2[1] Section 3.2.1.	-
.DeviceInfo.ProcessStatus.Process.{i}.	object	-	List of all processes running on the device. At most one entry in this table can exist with a given value for *PID*.	-
PID	unsignedInt	-	The Process Identifier.	-
Command	string(256)	-	The name of the command that has caused the process to exist.	-
Size	unsignedInt	-	The size in *kilobytes* of the memory occupied by the process.	-

Name	**Type**	**Write**	**Description**	**Object Default**
Priority	unsignedInt [:99]	-	The priority of the process where 0 is highest.	-
CPUTime	unsignedInt	-	The amount of time in *milliseconds* that the process has spent taking up CPU time since the process was started.	-
State	string	-	The current state that the process is in. Enumeration of: *Running* *Sleeping* *Stopped* *Idle* (OPTIONAL) *Uninterruptible* (OPTIONAL) *Zombie* (OPTIONAL)	-
.DeviceInfo.TemperatureStatus.	object	-	Status of the temperature of the device.	-
TemperatureSensorNumberOfEntries	unsignedInt	-	Number of entries in TemperatureSensor table.	-
.DeviceInfo.TemperatureStatus. TemperatureSensor.{i}.	object	-	This object represents information that the device has obtained via sampling an internal temperature sensor. At most one entry in this table can exist with a given value for *Name*.	-
Enable	boolean	W	Indicates whether or not the temperature sensor is enabled.	-
Status	string	-	The status of this temperature sensor. Enumeration of: *Disabled* (The sensor is not currently sampling the temperature.) *Enabled* (The sensor is currently sampling the temperature.) *Error* (The sensor error currently prevents sampling the temperature.)	-
Reset	boolean	W	When set to *true*, resets the temperature sensor. When read, this parameter returns *false*, regardless of the actual value.	-
ResetTime	dateTime	-	The time at which this temperature sensor was reset. Reset can be caused by: *Status* transition from *Disabled* to *Enabled* *Reset* set to *true*. An internal reset of the temperature sensor (including a reboot of the device). The Unknown Time value, as defined in TR-106a2[2], indicates that this temperature sensor has never been reset, which can only happen if it has never been enabled.	-
Name	string(256)	-	Name of this temperature sensor. This text MUST be sufficient to distinguish this temperature sensor from other temperature sensors.	-

Name	Type	Write	Description	Object Default
Value	int[-274:]	-	This temperature sensor's last good reading in *degrees celsius*. A value of -274 (which is below absolute zero) indicates a good reading has not been obtained since last reset.	-
LastUpdate	dateTime	-	The time at which this temperature sensor's last good reading was obtained. The Unknown Time value, as defined in TR-106a2[2], indicates a good reading has not been obtained since last reset.	-
MinValue	int[-274:]	-	This temperature sensor's lowest value reading in *degrees celsius* since last reset. A value of -274 (which is below absolute zero) indicates a good reading has not been obtained since last reset.	-
MinTime	dateTime	-	The time at which this temperature sensor's lowest value was read. The Unknown Time value, as defined in TR-106a2[2], indicates a good reading has not been obtained since last reset.	-
MaxValue	int[-274:]	-	This temperature sensor's highest value reading in *degrees celsius* since last reset. A value of -274 (which is below absolute zero) indicates a good reading has not been obtained since last reset.	-
MaxTime	dateTime	-	The time at which this temperature sensor's highest value was read. The Unknown Time value, as defined in TR-106a2[2], indicates a good reading has not been obtained since last reset.	-
LowAlarmValue	int[-274:]	W	This temperature sensor's low alarm value in *degrees celsius*. A value of -274 (which is below absolute zero) indicates a non configured value.	-
LowAlarmTime	dateTime	-	Initial time at which this temperature sensor's *LowAlarmValue* was encountered. This value is only set the first time the alarm is seen and not changed until the next reset. The Unknown Time value, as defined in TR-106a2[2], indicates that an alarm has not been encountered since the last reset.	-
HighAlarmValue	int[-274:]	W	This temperature sensor's high alarm value in *degrees celsius*. A value of -274 (which is below absolute zero) indicates a non configured value.	-
HighAlarmTime	dateTime	-	Initial time at which this temperature sensor's *HighAlarmValue* was encountered. This value is only set the first time the alarm is seen and not changed until the next reset. The Unknown Time value, as defined in TR-106a2[2], indicates that an alarm has not been encountered since the last reset.	-

Name	Type	Write	Description	Object Default
.DeviceInfo.NetworkProperties.	object	-	This object defines the parameters that describe how the device handles network traffic.	-
MaxTCPWindowSize	unsignedInt	-	The maximum number of *bytes* of outstanding data a sender can send on a particular connection prior to an acknowledgment RFC793[19]. Any scaling factor SHOULD be included in this parameter RFC1323[9].	-
TCPImplementation	string	-	Comma-separated list of strings. Indicates the TCP congestion control mechanism(s) implemented. Each list item is an enumeration of: *Tahoe* (Represents the base TCP implementation in RFC793[19] and elements of RFC2582[11]) *Reno* (Represents the base TCP implementation in RFC793[19] with the additional algorithms defined in RFC2581[10]) *New Reno* (Described as a modification to the Reno algorithms in RFC2582[11]) *Vegas* (An emerging TCP congestion control mechanism) Tahoe, Reno, and New Reno are defined in RFC2582[11]	-
.ManagementServer.	object	-	This object contains parameters relating to the CPE's association with an ACS.	-
.ManagementServer. AutonomousTransferCompletePolicy.	object	-	This object allows configuration of CPE policy for notification of AUTONOMOUS TRANSFER COMPLETE events, defined in TR-069a2[1]. The CPE policy determines the conditions under which the CPE notifies the ACS of the completion of file transfers that were not specifically requested by the ACS.	-
Enable	boolean	W	Enable/disable CPE notification of AUTONOMOUS TRANSFER COMPLETE events to the ACS.	-
TransferTypeFilter	string	W	Indicates the transfer types that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER COMPLETE events. Transfer types not indicated by this filter MUST NOT be included when the CPE notifies the ACS. Enumeration of: *Upload* *Download* *Both* (Upload and Download) Note that this includes any backup or restore operations that were not specifically requested by the ACS. A backup is regarded as an Upload and a restore is regarded as a Download.	-

Name	Type	Write	Description	Object Default
FileTypeFilter	string (1024)	W	Comma-separated list (maximum length 1024) of strings. Indicates the file types that MUST be included when the CPE notifies the ACS of AUTONOMOUS TRANSFER COMPLETE events. File types omitted from this list MUST NOT be included when the CPE notifies the ACS. *1 Firmware Upgrade Image* (Download Only) *2 Web Content* (Download Only) *3 Vendor Configuration File* (Download or Upload) *4 Vendor Log File* (Upload Only) *X [0-9A-F]{6} .** (For Vendor-Specific File Types, could be for either Download or Upload) Additionally, the following format is defined to allow the unique definition of vendor-specific file types: *"X <OUI> <Vendor-specific identifier>"* <OUI> is replaced by a 6 hexadecimal-digit OUI (organizationally unique identifier) as defined in [8], with all upper-case letters and any leading zeros included. The OUI used for a given vendor-specific file type MUST be one that is assigned to the organization that defined this file type (which is not necessarily the same as the vendor of the CPE or ACS). Note that an empty string indicates that all file types are excluded from this filter, effectively disabling CPE notification of AUTONOMOUS TRANSFER COMPLETE events to the ACS.	-
.UserInterface.	object	-	This object contains parameters relating to the user interface of the CPE.	-
.UserInterface.RemoteAccess.	object	-	This object contains parameters relating to remotely accessing the CPE's user interface. Remote access is defined as any entity not of a local subnet attempting to connect to the CPE. Remote access requires user authentication. To provide remote access authentication the CPE MUST support a *.User.{i}.* with at least one instance that has *.User.{i}.RemoteAccessCapable* set to *true*.	-
Enable	boolean	W	Enables/Disables remotely accessing the CPE's user interface.	-
Port	unsignedInt [:65535]	W	Destination TCP port required for remote access connection.	-

Name	**Type**	**Write**	**Description**	**Object Default**
SupportedProtocols	string	-	Comma-separated list of strings. Indicates the protocols that are supported by the CPE for the purpose of remotely accessing the user interface. Each list item is an enumeration of: *HTTP* (As defined in RFC2616[12]) *HTTPS* (As defined in RFC2818[13])	-
Protocol	string	W	The value MUST be a member of the list reported by the *SupportedProtocols* parameter. This is the protocol currently being used for remote access.	-
.UserInterface.LocalDisplay.	object	-	This object describes how to remotely manage the initial positioning of a user interface on a device's local display.	-
Movable	boolean	W	Controls whether the user is allowed to change the GUI window position on the local CPE's display.	-
Resizable	boolean	W	Controls whether the user is allowed to resize the GUI window on the local CPE's display.	-
PosX	int	W	The horizontal position of the User Interface's top left corner within the local CPE's display measured from the top left corner, expressed in *pixels*.	-
PosY	int	W	The vertical position of the User Interface's top left corner within the local CPE's display measured from the top left corner, expressed in *pixels*.	-
Width	unsignedInt	W	The width of the user interface within the local CPE's display, expressed in *pixels*.	-
Height	unsignedInt	W	The height of the user interface within the local CPE's display, expressed in *pixels*.	-
DisplayWidth	unsignedInt	-	The width of the local CPE's display, expressed in *pixels*.	-
DisplayHeight	unsignedInt	-	The height of the local CPE's display, expressed in *pixels*.	-
.User.{i}.	object	W	This object contains parameters relating to the user characteristics. At most one enabled entry in this table can exist with a given value for *Username*.	-
Enable	boolean	W	Enables/disables this user object instance. If the User being configured is currently accessing the device then a disable MUST apply to the next user session and the current user session MUST NOT be abruptly terminated.	false
RemoteAccessCapable	boolean	W	Allows this user to remotely access the UserInterface via the mechanism defined in *.UserInterface.RemoteAccess.*	false
Username	string(64)	W	Name of the current user. MUST NOT be an empty string for an enabled entry.	<Empty>
Password	string(64)	W	The user's password. When read, this parameter returns an empty string, regardless of the actual value.	-

Name	Type	Write	Description	Object Default
Language	string(16)	W	String describing the default language for the local configuration interface, specified according to RFC3066[15]. If an empty string, *.UserInterface.CurrentLanguage* is used.	<Empty>
.UPnP.	object	-	This object contains all UPnP related objects and parameters including Device and Discovery related objects and parameters.	-
.UPnP.Device.	object	-	Configuration Object for UPnP Access.	-
Enable	boolean	W	Enables/Disables UPnP support.	-
UPnPMediaServer	boolean	W	Enables/Disables UPnP Media Server.	-
UPnPMediaRenderer	boolean	W	Enables/Disables UPnP Media Renderer.	-
UPnPWLANAccessPoint	boolean	W	Enables/Disables UPnP Wireless Access Point.	-
UPnPQoSDevice	boolean	W	Enables/Disables UPnP QoS Device.	-
UPnPQoSPolicyHolder	boolean	W	Enables/Disables UPnP QoS Policy Holder.	-
UPnPIGD	boolean	W	Enables/Disables UPnP IGD.	-
.UPnP.Device.Capabilities.	object	-	This object defines what UPnP capabilities this device has.	-
UPnPArchitecture	unsignedInt	-	Numeric value indicating the version of supported architecture for UPnP. A value of 0 indicates no support.	-
UPnPMediaServer	unsignedInt	-	Numeric value indicating the supported revision for UPnP Media Server. A value of 0 indicates no support.	-
UPnPMediaRenderer	unsignedInt	-	Numeric value indicating the supported revision for UPnP Media Renderer. A value of 0 indicates no support.	-
UPnPWLANAccessPoint	unsignedInt	-	Numeric value indicating the supported revision for UPnP Wireless Access Point. A value of 0 indicates no support.	-
UPnPBasicDevice	unsignedInt	-	Numeric value indicating the supported revision for UPnP Basic Device. A value of 0 indicates no support.	-
UPnPQoSDevice	unsignedInt	-	Numeric value indicating the supported revision for UPnP Qos Device. A value of 0 indicates no support.	-
UPnPQoSPolicyHolder	unsignedInt	-	Numeric value indicating the supported revision for UPnP Qos Policy Holder. A value of 0 indicates no support.	-
UPnPIGD	unsignedInt	-	Numeric value indicating the supported revision for UPnP IGD. A value of 0 indicates no support.	-
.UPnP.Discovery.	object	-	UPnP UPnP-DAv1[20] SSDP discovered root devices, embedded devices and embedded services. The CPE MAY, but need not, retain some or all of the information in this object across reboots.	-
RootDeviceNumberOfEntries	unsignedInt	-	Number of entries in RootDevice table.	-
DeviceNumberOfEntries	unsignedInt	-	Number of entries in Device table.	-
ServiceNumberOfEntries	unsignedInt	-	Number of entries in Service table.	-

Name	Type	Write	Description	Object Default
.UPnP.Discovery.RootDevice.{i}.	object	-	UPnP root device table. This table contains an entry for each UPnP root device that has been discovered via SSDP. At most one entry in this table can exist with a given value for *UUID*.	-
Status	string	-	The status of the UPnP root device. Enumeration of: *LeaseActive* (Device is active and UPnP lease has not expired.) *LeaseExpired* (Device is inactive because UPnP lease has expired.) *ByebyeReceived* (Device is inactive because byebye message was received.) The ability to list inactive UPnP root devices is OPTIONAL. The length of time an inactive device remains listed in this table is a local matter to the CPE.	-
UUID	string(36)	-	This UPnP root device's UUID (Universally Unique IDentifier) RFC4122[18], extracted from any of its USN (Unique Service Name) headers. This is a 36-byte string that uniquely identifies the device, the following is an example: *02c29d2a-dbfd-2d91-99c9-306d537e9856* *[0-9A-Fa-f]{8}-([0-9A-Fa-f]{4}-){3}[0-9A-Fa-f]{12}*	-
USN	string(256)	-	The value of the USN (Unique Service Name) header for this UPnP root device. Three discovery messages are sent for root devices, and this SHOULD be the value of the USN header of the following form: *uuid:device-UUID::urn:domain-name:device:deviceType:v* SSDP is an unreliable protocol and it is possible that no discovery message containing the USN header of the above form was ever received. If so, one of the other two forms MAY be used: *uuid:device-UUID::upnp:rootdevice* *uuid:device-UUID* (for root device UUID)	-
LeaseTime	unsignedInt	-	The UPnP root device lease time in *seconds*, extracted from the CACHE-CONTROL header.	-
Location	string(256)	-	The value of the LOCATION header for this UPnP root device, which is the URL of the root device's DDD (Device Description Document).	-

Name	**Type**	**Write**	**Description**	**Object Default**
Server	string(256)	-	The value of the SERVER header for this UPnP root device, which is a string of the following form: *OS/version UPnP/udaversion product/version* where **UPnP** is a literal string, **udaversion** is the version of the UPnP Device Architecture.	-
Host	string (1024)	-	Comma-separated list (maximum length 1024) of strings (maximum item length 256). Indicates the full path names of all Host table entries, whether active or inactive, that correspond to this UPnP root device. As such entries are added to or removed from the Host tables, the value of this parameter MUST be updated accordingly.	-
.UPnP.Discovery.Device.{i}.	object	-	UPnP embedded device table. This table contains an entry for each UPnP embedded device that has been discovered via SSDP. At most one entry in this table can exist with a given value for *UUID*.	-
Status	string	-	The status of the UPnP embedded device. Enumeration of: *LeaseActive* (Device is active and UPnP lease has not expired.) *LeaseExpired* (Device is inactive because UPnP lease has expired.) *ByebyeReceived* (Device is inactive because byebye message was received.) The ability to list inactive UPnP embedded devices is OPTIONAL. The length of time an inactive device remains listed in this table is a local matter to the CPE.	-
UUID	string(36)	-	This UPnP embedded device's UUID (Universally Unique IDentifier) RFC4122[18], extracted from any of its USN (Unique Service Name) headers. This is a 36-byte string that uniquely identifies the device, the following is an example: *02c29d2a-dbfd-2d91-99c9-306d537e9856* *[0-9A-Fa-f]{8}-([0-9A-Fa-f]{4}-){3}[0-9A-Fa-f]{12}*	-

Name	Type	Write	Description	Object Default
USN	string(256)	-	The value of the USN (Unique Service Name) header for this UPnP embedded device. Two discovery messages are sent for embedded devices, and this SHOULD be the value of the USN header of the following form: *uuid:device-UUID::urn:domain-name:device:deviceType:v* SSDP is an unreliable protocol and it is possible that no discovery message containing the USN header of the above form was ever received. If so, the other form MAY be used: *uuid:device-UUID*	-
LeaseTime	unsignedInt	-	The UPnP embedded device lease time in *seconds*, extracted from the CACHE-CONTROL header.	-
Location	string(256)	-	The value of the LOCATION header for this UPnP embedded device, which is the URL of the root device's DDD (Device Description Document).	-
Server	string(256)	-	The value of the SERVER header for this UPnP embedded device, which is a string of the following form: *OS/version UPnP/udaversion product/version* where **UPnP** is a literal string, **udaversion** is the version of the UPnP Device Architecture.	-
Host	string (1024)	-	Comma-separated list (maximum length 1024) of strings (maximum item length 256). Indicates the full path names of all Host table entries, whether active or inactive, that correspond to this UPnP embedded device. As such entries are added to or removed from the Host tables, the value of this parameter MUST be updated accordingly.	-
.UPnP.Discovery.Service.{i}.	object	-	UPnP embedded service table. This table contains an entry for each UPnP embedded service that has been discovered via SSDP. At most one entry in this table can exist with a given value for *USN*.	-

Name	Type	Write	Description	Object Default
Status	string	-	The status of the UPnP embedded service. Enumeration of: *LeaseActive* (Service is active and UPnP lease has not expired.) *LeaseExpired* (Service is inactive because UPnP lease has expired.) *ByebyeReceived* (Service is inactive because byebye message was received.) The ability to list inactive UPnP embedded services is OPTIONAL. The length of time an inactive service remains listed in this table is a local matter to the CPE.	-
USN	string(256)	-	The value of the USN (Unique Service Name) header for this UPnP embedded service. This is of the following form: *uuid:device-UUID::urn:domain-name:service:serviceType:v*	-
LeaseTime	unsignedInt	-	The UPnP embedded service lease time in *seconds*, extracted from the CACHE-CONTROL header.	-
Location	string(256)	-	The value of the LOCATION header for this UPnP embedded service, which is the URL of the root device's DDD (Device Description Document).	-
Server	string(256)	-	The value of the SERVER header for this UPnP embedded service, which is a string of the following form: *OS/version UPnP/udaversion product/version* where **UPnP** is a literal string, **udaversion** is the version of the UPnP Device Architecture.	-
Host	string (1024)	-	Comma-separated list (maximum length 1024) of strings (maximum item length 256). Indicates the full path names of all Host table entries, whether active or inactive, that correspond to this UPnP embedded service. As such entries are added to or removed from the Host tables, the value of this parameter MUST be updated accordingly.	-
.DLNA.	object	-	This object contains all DLNA related objects and parameters.	-
.DLNA.Capabilities.	object	-	DLNA capabilities.	-
HNDDeviceClass	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the supported DLNA Home Network Device Classes DLNA-NDIG[4] Table 4-1.	-
DeviceCapability	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the supported DLNA Device Capabilities DLNA-NDIG[4] Table 4-2.	-

Name	Type	Write	Description	Object Default
HIDDeviceClass	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the supported DLNA Home Infrastructure Device Classes DLNA-NDIG[4] Table 4-4.	-
ImageClassProfileID	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the DLNA Image Class Profile IDs supported by this device, from Tables 5-2 and 5-3 of DLNA-NDIG[4].	-
AudioClassProfileID	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the DLNA Audio Class Profile IDs supported by this device, from Tables 5-4 through 5-10 of DLNA-NDIG[4].	-
AVClassProfileID	string (256)	-	Comma-separated list (maximum length 256) of strings. Indicates the DLNA AV Class Profile IDs supported by this device, from Tables 5-11 through 5-15 of DLNA-NDIG[4].	-
MediaCollectionProfileID	string (256)	-	Comma-separated list (maximum length 256) of strings. Indicates the DLNA Media Collection Profile IDs supported by this device DLNA-NDIG[4] Table 5-16.	-
PrinterClassProfileID	string(256)	-	Comma-separated list (maximum length 256) of strings. Indicates the DLNA Printer Class Profile IDs supported by this device DLNA-NDIG[4] Table 5-17.	-
.SmartCardReader.{i}.	object	-	This object describes the characteristics of the smart card reader. At most one entry in this table can exist with a given value for *Name*.	-
Enable	boolean	W	Enables or disables this smart card reader.	-
Status	string	-	Indicates the status of this smart card reader. Enumeration of: *Disabled* *Enabled* (Indicates the smart card reader is enabled and functioning properly.) *Error* (Indicates the smart card reader is enabled and not functioning properly.)	-
Name	string(256)	-	Human-readable name associated with this smart card reader.	-
Reset	boolean	W	When set to *true*, resets the SmartCard Reader and the associated SmartCard. When read, this parameter returns *false*, regardless of the actual value.	-

Name	Type	Write	Description	Object Default
ResetTime	dateTime	-	The time at which this SmartCard Reader was reset. Reset can be caused by: *Status* transition from Disabled to Enabled *Reset* set to *true*. An internal reset of the SmartCard Reader (including a reboot of the device). Unknown Time value indicates that this SmartCard Reader has never been reset, which can only happen if it has never been enabled.	-
DecryptionFailedCounter	unsignedInt	-	Counter incremented once each time decryption cannot be carried out. This counter relates to the smart card reader, not to the smart card itself, i.e. it is reset when the *Reset* parameter is used and not when a Smart Card is inserted or removed.	-
DecryptionFailedNoKeyCounter	unsignedInt	-	Counter incremented once each time the key is not available to decrypt it. This is a subset of the more general *DecryptionFailedCounter* within the same object and thus will always be less than that parameter. This counter relates to the smart card reader, not to the smart card itself, i.e. it is reset when the *Reset* parameter is used and not when a Smart Card is inserted or removed.	-
.SmartCardReader.{i}.SmartCard.	object	-	Status of currently associated smart card.	-
Status	string	-	Status of the Smart Card. Enumeration of: *None* (Indicates that no Smart Card is inserted.) *Running* (Indicates a Smart Card is present and working normally.) *Error* (Indicates the Smart Card is present and in an error condition.)	-
Type	string	-	Smart Card Type. Enumeration of: *CA* *DRM* *UICC* Vendors can extend the enumerated values with vendor specific extensions, in which case the rules outlined in TR-106a2[2] Section 3.3 MUST be adhered to.	-

Name	**Type**	**Write**	**Description**	**Object Default**
Application	string	-	Comma-separated list of strings. Indicates Smart Card Application(s). *Application* is only relevant when *Type* has a value of UICC, otherwise it is an empty string. Each list item is an enumeration of: *SIM* *USIM* *ISIM* Vendors can extend the enumerated values with vendor specific extensions, in which case the rules outlined in TR-106a2[2] Section 3.3 MUST be adhered to.	-
SerialNumber	string(256)	-	The Smart Card Serial Number or an empty string if the Smart Card serial Number is not available, e.g. in the case of IPTV due to restrictions of the Service Delivery Platform.	-
ATR	string (1024)	-	The Smart Card answer to a reset action. Issued by the Smart Card upon reset.	-
.SelfTestDiagnostics.	object	-	This diagnostics test is vendor-specific and MAY include testing hardware, software, and/or firmware.	-

Name	Type	Write	Description	Object Default
DiagnosticsState	string	W	Indicates availability of diagnostic data. Enumeration of: *None* *Requested* *Complete* *Error_Internal* *Error_Other* If the ACS sets the value of this parameter to *Requested*, the CPE MUST initiate the corresponding diagnostic test. When writing, the only allowed value is *Requested*. To ensure the use of the proper test parameters (the writable parameters in this object), the test parameters MUST be set either prior to or at the same time as (in the same SetParameterValues) setting the DiagnosticsState to Requested. When requested, the CPE SHOULD wait until after completion of the communication session with the ACS before starting the diagnostic. When the test is completed, the value of this parameter MUST be either *Complete* (if the test completed successfully), or one of the Error values listed above. If the value of this parameter is anything other than *Complete*, the values of the results parameters for this test are indeterminate. When the diagnostic initiated by the ACS is completed (successfully or not), the CPE MUST establish a new connection to the ACS to allow the ACS to view the results, indicating the Event code "8 DIAGNOSTICS COMPLETE" in the Inform message. After the diagnostic is complete, the value of all result parameters (all read-only parameters in this object) MUST be retained by the CPE until either this diagnostic is run again, or the CPE reboots. After a reboot, if the CPE has not retained the result parameters from the most recent test, it MUST set the value of this parameter to *None*. Modifying any of the writable parameters in this object except for this one MUST result in the value of this parameter being set to *None*. While the test is in progress, modifying any of the writable parameters in this object except for this one MUST result in the test being terminated and the value of this parameter being set to *None*. While the test is in progress, setting this parameter to *Requested* (and possibly modifying other writable parameters in this object) MUST result in the test being terminated and then restarted using the current values of the test parameters.	-

Name	Type	Write	Description	Object Default
Results	string (1024)	-	Results of self-test (vendor specific).	-
.NSLookupDiagnostics.	object	-	This object defines access to an IP-layer NS Lookup test for the specified IP interface. When initiated, the NS Lookup test will contact *DNSServer* and look up *HostName* *NumberOfRepetitions* times. There will be a *Result* instance for each time the device performs a DNS lookup, which is determined by the value of *NumberOfRepetitions*. Any previous *Result* instances are removed when a new test is initiated.	-

Name	Type	Write	Description	Object Default
DiagnosticsState	string	W	Indicates availability of diagnostic data. Enumeration of: *None* *Requested* *Complete* *Error_DNSServerNotResolved* (Unable to resolve DNSServer Name) *Error_Internal* *Error_Other* If the ACS sets the value of this parameter to *Requested*, the CPE MUST initiate the corresponding diagnostic test. When writing, the only allowed value is *Requested*. To ensure the use of the proper test parameters (the writable parameters in this object), the test parameters MUST be set either prior to or at the same time as (in the same SetParameterValues) setting the DiagnosticsState to *Requested*. When requested, the CPE SHOULD wait until after completion of the communication session with the ACS before starting the diagnostic. When the test is completed, the value of this parameter MUST be either *Complete* (if the test completed successfully), or one of the Error values listed above. If the value of this parameter is anything other than *Complete*, the values of the results parameters for this test are indeterminate. When the diagnostic initiated by the ACS is completed (successfully or not), the CPE MUST establish a new connection to the ACS to allow the ACS to view the results, indicating the Event code "8 DIAGNOSTICS COMPLETE" in the Inform message. After the diagnostic is complete, the value of all result parameters (all read-only parameters in this object) MUST be retained by the CPE until either this diagnostic is run again, or the CPE reboots. After a reboot, if the CPE has not retained the result parameters from the most recent test, it MUST set the value of this parameter to *None*. Modifying any of the writable parameters in this object except for this one MUST result in the value of this parameter being set to *None*. While the test is in progress, modifying any of the writable parameters in this object except for this one MUST result in the test being terminated and the value of this parameter being set to *None*. While the test is in progress, setting this parameter to *Requested* (and possibly modifying other writable parameters in this object) MUST result in the test being terminated and then restarted using the current values of the test parameters.	-

Name	Type	Write	Description	Object Default
Interface	string(256)	W	The value MUST be the full path name of a table row. This parameter specifies the IP-layer interface over which the test is to be performed (i.e. the source IP address to use when performing the test). If an empty string is specified, the CPE MUST use its routing policy (Forwarding table entries), if necessary, to determine the appropriate interface.	-
HostName	string(256)	W	Specifies the Host Name that NS Lookup is to look for. The current domain name MUST be used unless the name is a fully qualified name.	-
DNSServer	string(256)	W	Specifies the DNS Server name or IP address that NS Lookup is to use for the lookup. The name of this server will be resolved using the default DNS server unless an IP address is provided. If an empty string is specified, the device's default DNS server will be used.	-
Timeout	unsignedInt	W	Timeout in *milliseconds* that indicates that a request has failed.	-
NumberOfRepetitions	unsignedInt	W	The number of times the device SHOULD repeat the execution of the NSLookup using the same input parameters. If the diagnostics test fails the CPE MAY terminate the test without completing the full number of repetitions. Each repetition will use a Result instance to hold the NSLookup result data.	-
SuccessCount	unsignedInt	-	Number of successfully executed repetitions.	-
ResultNumberOfEntries	unsignedInt	-	Total number of Result entries from the most recent invocation of the test.	-
.NSLookupDiagnostics.Result.{i}.	object	-	Results from the most recent invocation of the test, one instance per repetition.	-
Status	string	-	Result Parameter to represent whether the NS Lookup was successful or not. Errors for individual Result instances do not get bubbled up to *.NSLookupDiagnostics.DiagnosticsState*. A failure on a specific attempt does not mean that the overall test failed, but a failure on all attempts means that *.NSLookupDiagnostics.DiagnosticsState* SHOULD be *Error_Other*. Enumeration of: *Success* *Error_DNSServerNotAvailable* *Error_HostNameNotResolved* *Error_Timeout* *Error_Other*	-

Name	Type	Write	Description	Object Default
AnswerType	string	-	Result parameter to represent whether the answer is Authoritative or not. Enumeration of: *None* (Indicates that the NS Lookup failed to find the host.) *Authoritative* *NonAuthoritative*	-
HostNameReturned	string(256)	-	Result parameter to represent the fully qualified name for the Host Name in the calling parameter (e.g. HostName.DomainName); if no response was provided, then this parameter is an empty string.	-
IPAddresses	string	-	Comma-separated list (maximum 10 items) of IPAddresses. Indicates the IP Address results returned by the NS Lookup; if no response was provided, then this parameter is an empty string.	-
DNSServerIP	string	-	Result parameter to represent the actual DNS Server IP address that the NS Lookup used.	-
ResponseTime	unsignedInt	-	Response time (for the first response packet) in *milliseconds*, or 0 if no response was received.	-
.Firewall.	object	-	Firewall configuration object.	-
Config	string	W	How this firewall is configured. Enumeration of: *High* (The firewall implements the "Traffic Denied Inbound" and "Minimally Permit Common Services Outbound" components of the ICSA residential certification's Required Services Security Policy ICSA-Firewall[6].) *Low* (All Outbound traffic and pinhole-defined Inbound traffic is allowed.) *Off* (All Inbound and Outbound traffic is allowed, and the CPE is only protected by NAT settings.) Vendors can extend the enumerated values with vendor specific extensions, in which case the rules outlined in TR-106a2[2] Section 3.3 MUST be adhered to.	-
Version	string(16)	-	A string identifying the firewall settings version currently used in the CPE, or an empty string if the firewall settings are not associated with a version.	-
LastChange	dateTime	-	The time at which the firewall settings most recently changed.	-
.USBHosts.	object	-	This object models the CPE's USB Host controllers. See Appendix I for Theory of Operation.	-
HostNumberOfEntries	unsignedInt	-	Number of entries in the Host table.	-
.USBHosts.Host.{i}.	object	-	Table of CPE USB Host controllers. At most one entry in this table can exist with a given value for *Name*.	-
Enable	boolean	W	Enables or disables the USB Host controller.	-
Name	string(64)	-	User-readable host controller name.	-

Name	Type	Write	Description	Object Default
Type	string	-	Type of USB Host Enumeration of: *OHCI* (Open Host Controller Interface) *EHCI* (Enhanced Host Controller Interface) *UHCI* (Universal Host Controller Interface) *xHCI* (Extensible Host Controller Interface)	-
Reset	boolean	W	When set to *true*, reset the Host Controller and apply the reset signaling (see USB2.0[22] Chapter 7.1.7.5) to all of the Host Controller Hub downstream ports. The value is not saved in the device's state and setting it to *false* has no effect. When read, this parameter returns *false*, regardless of the actual value.	-
PowerManagementEnable	boolean	W	When set to *true*, *PowerManagementEnable* enables the Host Controller to invoke Power Management policy, i.e. controlled Suspend (see USB2.0[22], Chapters 4.3.2, 7.1.7.6, and 11.9). When set to *false PowerManagementEnable* immediately disables the Host controller Power Management policy.	-
USBVersion	string(4)	-	USB specification version with which the controller complies. Example: "1.1"	-
DeviceNumberOfEntries	unsignedInt	-	Number of entries in the Device table.	-
.USBHosts.Host.{i}.Device.{i}.	object	-	Table of connected USB devices. At most one entry in this table can exist with a given value for *DeviceNumber*.	-
DeviceNumber	unsignedInt	-	Device number on USB bus.	-
USBVersion	string(4)	-	USB specification version with which the device complies. Example: "1.1"	-
DeviceClass	hexBinary (1)	-	Class Code as assigned by USB-IF. When 0x00, each device specifies its own class code. When 0xFF, the class code is vendor specified.	-
DeviceSubClass	hexBinary (1)	-	Subclass code (assigned by USB-IF).	-
DeviceVersion	unsignedInt [:65535]	-	Device release number.	-
DeviceProtocol	hexBinary (1)	-	Protocol code (assigned by USB-IF).	-
ProductID	unsignedInt [:65535]	-	Product ID (assigned by manufacturer).	-
VendorID	unsignedInt [:65535]	-	Vendor ID (assigned by USB-IF).	-
Manufacturer	string(64)	-	Device Manufacturer string descriptor.	-
ProductClass	string(64)	-	Device Product Class string descriptor.	-
SerialNumber	string(64)	-	Device SerialNumber string descriptor.	-
Port	unsignedInt [:255]	-	Hub port on parent device. 0 when no parent.	-

Name	Type	Write	Description	Object Default
Rate	string	-	Speed of the USB device. Enumeration of: *Low* (1.5 Mbits/sec (187.5 KB/sec) defined in USB1.0[21]) *Full* (12 Mbits/sec (1.5 MB/sec) defined in USB1.0[21]) *High* (480 Mbits/sec (60 MB/sec) defined in USB2.0[22]) *Super* (5.0 Gbits/sec (625 MB/sec) defined in USB3.0[23]) Internal signaling between the connected USB device and the USB Host Controller provide the information needed to determine the negotiated rate.	-
Parent	string	-	The value MUST be the full path name of a row in the *.USBHosts.Host.{i}.Device.{i}* table. If the referenced object is deleted, the parameter value MUST be set to an empty string. This is a reference to the parent USB device (e.g. hub device). Example: *.USBHosts.Host.2.Device.3* This is an empty string for a device connected to the Host controller (root hub).	-
MaxChildren	unsignedInt	-	Number of ports. Only applies for hub device, equal to 0 for other devices.	-
IsSuspended	boolean	-	When *true* the associated Device is in a suspended (i.e. low-power) state (see USB2.0[22] Chapter 11.9). When *false* the associated Device is in any of the other states specified by the USB 2.0 Device State Machine (see USB2.0[22] Chapter 9.1.1).	-
IsSelfPowered	boolean	-	When *true* the associated device is at least partly powered by a local source (see USB2.0[22] Chapter 9.4.5). When *false* the associated device draws all the current it needs from the USB bus.	-
ConfigurationNumberOfEntries	unsignedInt	-	Number of entries in the Configuration table.	-
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.	object	-	Table of device configurations. At most one entry in this table can exist with a given value for *ConfigurationNumber*.	-
ConfigurationNumber	unsignedInt	-	The identifier for each Device Configuration.	-
InterfaceNumberOfEntries	unsignedInt	-	Number of entries in the Interface table.	-
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.Interface.{i}.	object	-	Table of device interface descriptors. At most one entry in this table can exist with a given value for *InterfaceNumber*.	-
InterfaceNumber	unsignedInt [:255]	-	Number of this interface (from USB interface descriptor).	-
InterfaceClass	hexBinary (1)	-	Class Code as assigned by USB-IF. When 0x00, each interface specifies its own class code. When 0xFF, the class code is vendor specified.	-

Name	Type	Write	Description	Object Default
InterfaceSubClass	hexBinary (1)	-	Subclass code (assigned by USB-IF).	-
InterfaceProtocol	hexBinary (1)	-	Protocol code (assigned by USB-IF).	-
.PeriodicStatistics.	object	-	This object configures collection of periodic statistics for the device. Periodic statistics are measured over a sample interval (which can be aligned with absolute time) and are made available to the ACS as a comma-separated list of the most recent <n> samples. This object provides a single set of global settings that affect the entire device unless overridden locally.	-
MinSampleInterval	unsignedInt	-	Minimum sample interval in *seconds* that the CPE is able to support. A value of 0 indicates no specific minimum sample interval.	-
MaxReportSamples	unsignedInt	-	Maximum number of samples of each statistic that the CPE is able to store and report. A value of 0 indicates no specific maximum number of samples.	-
SampleSetNumberOfEntries	unsignedInt	-	Number of entries in SampleSet table.	-
.PeriodicStatistics.SampleSet.{i}.	object	W	Periodic statistics sample set table. Each sample set has its own sample interval etc. At most one enabled entry in this table can exist with a given value for *Name*.	-
Enable	boolean	W	Enables or disables collection of periodic statistics for this sample set. When collection of periodic statistics is enabled, any stored samples are discarded, and the first sample interval begins immediately.	false
Status	string	-	Indicates availability of Sample statistics. Enumeration of: *Disabled* (Collection is disabled.) *Enabled* (Collection is enabled.) *Trigger* (Collection is enabled and the ACS SHOULD now fetch the collected data.) The *Trigger* value is only used for triggering the ACS to fetch the collected data and can only be used when *FetchSamples* is in the range [1:*ReportSamples*]. The transition from *Enabled* to *Trigger* to *Enabled* MUST be instantaneous and so will result in only a single value change for notification purposes.	"Disabled"
Name	string(128)	W	The name of this sample set, which uniquely distinguishes each sample set.	<Empty>

Name	Type	Write	Description	Object Default
SampleInterval	unsignedInt [1:]	W	The sample interval in *seconds*. Each statistic is measured over this sample interval. The CPE MAY reject a request to set *SampleInterval* to less than *.PeriodicStatistics.MinSampleInterval*. Sample intervals MUST begin every *SampleInterval seconds*, with no delay between samples. If *SampleInterval* is changed while collection of periodic statistics is enabled, any stored samples are discarded, and the first sample interval begins immediately. For example, if *ReportSamples* is 24 and *SampleInterval* is 3600 (an hour), the CPE can store up to a day's worth of samples for each statistic.	3600
ReportSamples	unsignedInt [1:]	W	The number of samples that the CPE will store and report for each statistic. The CPE MUST permit *ReportSamples* to be set to at least *.PeriodicStatistics.MaxReportSamples*. If *ReportSamples* is changed while collection of periodic statistics is enabled, the CPE will truncate or extend its statistics buffers as appropriate, but statistics collection MUST NOT otherwise be affected. For example, if *ReportSamples* is 24 and *SampleInterval* is 3600 (an hour), the CPE can store up to a day's worth of samples for each statistic.	24

Name	Type	Write	Description	Object Default
TimeReference	dateTime	W	An absolute time reference in UTC to determine when sample intervals will complete. Each sample interval MUST complete at this reference time plus or minus an integer multiple of *SampleInterval*. *TimeReference* is used only to set the "phase" of the sample and fetch intervals. The actual value of *TimeReference* can be arbitrarily far into the past or future. This time reference also determines when the *Status Enabled* to *Trigger* to *Enabled* transitions that are controlled by *FetchSamples* will occur. If collection of periodic statistics is enabled and *FetchSamples* is in the range [1:*ReportSamples*] then each such *Status* transition MUST occur at this reference time plus or minus an integer multiple of *FetchSamples* * *SampleInterval* (the fetch interval). If *TimeReference* is changed while collection of periodic statistics is enabled, any stored samples are discarded, and the first sample interval begins immediately. The Unknown Time value defined in TR-106a2[2] indicates that no particular time reference is specified. That is, the CPE MAY locally choose the time reference, and is required only to adhere to the specified sample and fetch intervals. If absolute time is not available to the CPE, its sample and fetch interval behavior MUST be the same as if the *TimeReference* parameter was set to the Unknown Time value. For example, if *SampleInterval* is 3600 (an hour) and if *TimeReference* is set to UTC midnight on some day (in the past, present, or future) then sample intervals will complete on each UTC hour (00:00, 01:00, 02:00 etc). If, in addition, *FetchSamples* is 24, then the fetch interval is 86400 (a day) and *Status Enabled* to *Trigger* to *Enabled* transitions will occur every day at UTC midnight. Note that, if *TimeReference* is set to a time other than the Unknown Time, the first sample interval (which has to begin immediately) will almost certainly be shorter than *SampleInterval*). This is why *TimeReference* is defined in terms of when sample intervals complete rather than start.	0001-01-01T00:00:00Z

Name	**Type**	**Write**	**Description**	**Object Default**
FetchSamples	unsignedInt	W	The number of sample intervals to be collected before transitioning *Status* from *Enabled* to *Trigger* to *Enabled*. If this SampleSet is enabled and *FetchSamples* is in the range [1:*ReportSamples*] then *Status* MUST transition from *Enabled* to *Trigger* to *Enabled* on completion of every *FetchSamples* sample intervals. Otherwise, the transition MUST NOT occur. For example, if *ReportSamples* is 25 and *FetchSamples* is 24, then the CPE will store 25 values for each monitored parameter and the above *Status* transition will occur as the CPE stores each 24th of 25 sample intervals, which means that the ACS could delay for up to two sample intervals before reading the stored values and would still not miss any samples (see also *ForceSample*). To disable this trigger mechanism and still collect sampled statistics, *FetchSamples* can be set to either 0 or a value greater than *ReportSamples*.	0
ForceSample	boolean	W	When set to *true*, forces statistics for the current sample to be calculated and updated in the data model. Setting it to *false* has no effect. When read, this parameter returns *false*, regardless of the actual value. If this is the first time that *ForceSample* has been set to *true* during the current sample interval, this MUST cause a new value to be added to each of the periodic statistics comma-separated list parameters, and the *ReportEndTime* and all *SampleSeconds* parameters MUST be updated accordingly. If this is not the first time that *ForceSample* has been set to *true* during the current sample interval, then the new values that were added as described in the previous paragraph, and the *ReportEndTime* and all *SampleSeconds* parameters, MUST be updated accordingly. Note that *ForceSample* just provides a "sneak preview" of the current sample. It does not create a new sample and it does not interfere with the sample interval schedule. At the end of each sample interval, if *ForceSample* was set to *true* during the sample interval then the new values that were added as described above, and the *ReportEndTime* and all *SampleSeconds* parameters, will be updated accordingly. In other words, the partial sample data that was created when *ForceSample* was set to *true* will be updated one last time at the end of the sample interval.	false

Name	**Type**	**Write**	**Description**	**Object Default**
ReportStartTime	dateTime	-	The absolute time at which the sample interval for the first stored sample (for each statistic) started.	<Empty>
ReportEndTime	dateTime	-	The absolute time at which the sample interval for the last stored sample (for each statistic) ended. If *ForceSample* has been used to force statistics for the current sample to be calculated and updated in the data model, then *ReportEndTime* MUST be updated to reflect the actual time over which stored data was collected.	<Empty>
SampleSeconds	string	-	Comma-separated list of unsigned integers. Each entry indicates the number of *seconds* during which data was collected during the sample interval. Individual *SampleSeconds* values can be less than *SampleInterval*, for several reasons, including: *TimeReference* has been set to a time other than the Unknown Time and the current sample interval started part of the way through a scheduled sample interval. *ForceSample* has been used to force statistics for the current sample to be calculated and updated in the data model.	<Empty>
ParameterNumberOfEntries	unsignedInt	-	Number of entries in Parameter table.	0
.PeriodicStatistics.SampleSet.{i}.Parameter.{i}.	object	W	Periodic statistics parameter table for this sample set. This table contains entries for parameters whose values are to be sampled. Note that the comma-separated lists in this object (SampleSeconds, SuspectData and Values) only ever change (a) when first enabled, (b) when ForceSample is set to true (a "sneak preview" of the current sample), or (c) at the end of the sample interval. At most one enabled entry in this table can exist with a given value for *Reference*.	-
Enable	boolean	W	Enables or disables this object instance.	false
Reference	string(256)	W	The value MUST be the full path name of a parameter. This is the parameter being monitored by the Periodic Statistics mechanism.	<Empty>
SampleMode	string	W	Controls how this parameter's value is sampled. Enumeration of: *Current* (Sampled value is current value) *Change* (Sampled value is change in value since start of sample interval) Parameters of non-numeric types can only support *Current*. The value of the *SampleMode* MUST be ignored for such parameters.	"Current"

Name	Type	Write	Description	Object Default
CalculationMode	string	W	Controls how this parameter's statistic is calculated from the sampled value(s). Enumeration of: *Latest* (Statistic is sampled value at end of sample interval) *Minimum* (Statistic is minimum sampled value during sample interval) *Maximum* (Statistic is maximum sampled value during sample interval) *Average* (Statistic is average (mean) sampled value during sample interval) Parameters of non-numeric types can only support *Latest*. The value of the *CalculationMode* MUST be ignored for such parameters. *SampleMode* MUST be applied before *CalculationMode*, i.e. the inputs to the calculation will have already accounted for *SampleMode*.	"Latest"
LowThreshold	int	W	The low threshold value that controls the calculation of *Failures*. A value equal to *HighThreshold* disables the threshold/failure mechanism. Parameters of non-numeric types cannot support the threshold/failure mechanism. The value of this parameter MUST be ignored for such parameters.	0
HighThreshold	int	W	The high threshold value that controls the calculation of *Failures*. A value equal to *LowThreshold* disables the threshold/failure mechanism. Parameters of non-numeric types cannot support the threshold/failure mechanism. The value of this parameter MUST be ignored for such parameters.	0
SampleSeconds	string	-	Comma-separated list of unsigned integers. Each entry indicates the number of *seconds* during which data was collected for this parameter during the sample interval. Individual *SampleSeconds* values can be less than *.PeriodicStatistics.SampleSet.{i}.SampleInterval*, for several reasons, including: Any of the reasons for which *.PeriodicStatistics.SampleSet.{i}.SampleSeconds* values might be less than *.PeriodicStatistics.SampleSet.{i}.SampleInterval*. The parameter doesn't exist, or was created or deleted during a sample interval.	<Empty>

Name	Type	Write	Description	Object Default
SuspectData	string	-	Comma-separated list of unsigned integers (range 0 to 1). Each entry is 0 if the sampled value is believed to be valid, or 1 if an event that might affect the validity of the sampled value occurred during the sample interval. For example, if the parameter value were to be reset during the sample interval then it would be appropriate to set *SuspectData* to 1.	<Empty>
Values	string	-	Comma-separated list of strings. Each entry indicates the value of the referenced parameter, as determined by *SampleMode*, during the sample interval. The statistics values in this comma-separated lists MUST be in time order, with the oldest one first and the most recent one last. If the *SampleMode* parameter is not present, or is inappropriate for the referenced parameter, the statistics values MUST be collected in Current mode.	<Empty>
Failures	unsignedInt	-	Counts the number of times (since this object instance was last enabled) that a newly-calculated sample value (accounting for *SampleMode*) transitioned from the “in range” state to the “out of range” state, or between the “out of range (low)” and “out of range (high)” states. The states are defined as follows: "in range” : current value is greater than *LowThreshold* and less than *HighThreshold*. "out of range” : current value is less than or equal to *LowThreshold*, or greater than or equal to *HighThreshold*. "out of range (low)” : current value is less than or equal to *LowThreshold*. "out of range (high)” : current value is greater than or equal to *HighThreshold*. Note that, if *LowThreshold* and *HighThreshold* are both the same, the threshold/failure mechanism is disabled, so the value of this parameter will not increment. This parameter can be incremented at any time during a sample interval, and might be incremented more than once during a single sample interval. For this reason, the CPE SHOULD place a locally specified limit on the frequency at which it will notify the ACS of such changes, as described in TR-069a2[1] Section 3.2.1. Parameters of non-numeric types cannot support the threshold/failure mechanism. The value of this parameter MUST be ignored for such parameters.	0

Name	Type	Write	Description	Object Default
.DownloadAvailability.	object	-	This object contains multicast announcement and query parameters used for the purpose of downloading files.	-
.DownloadAvailability. Announcement.	object	-	This object contains multicast announcement parameters used to download files.	-
Enable	boolean	W	Enable/disable CPE ability to receive and use multicast announcements from a server for the purpose of downloading files.	-
Status	string	-	The status of the announcement service. Enumeration of: *Disabled* *Enabled* *Error* (MAY be used by the CPE to indicate a locally defined error condition., OPTIONAL)	-
GroupNumberOfEntries	unsignedInt	-	Number of entries in the Group table.	-
.DownloadAvailability. Announcement.Group.{i}.	object	W	Multicast groups to which the CPE SHOULD listen for announcements. At most one enabled entry in this table can exist with a given value for *URL*.	-
Enable	boolean	W	Enable/disable listening to this multicast group.	false
Status	string	-	The status of this group table entry. Enumeration of: *Disabled* *Enabled* *Error* (MAY be used by the CPE to indicate a locally defined error condition, e.g. unable to parse received announcements., OPTIONAL)	"Disabled"
URL	string(256)	W	URL RFC3986[17] encoding the group address, source and port on which to listen, and other protocol information, e.g. expected announcement format. Depending on the application, the messages identified by this URL MAY directly contain the data to be downloaded, or alternatively MAY contain information informing the CPE how to obtain the data to be downloaded via a separate mechanism, which itself could involve a unicast or a multicast download protocol. Refer to DVB-TS.102.824[5] for an example of a URL format that identifies a SAP RFC2974[14] stream that indicates how to use either FLUTE RFC3926[16] or DSM-CC ISO-13818-6:1998[7] to perform the download.	<Empty>
.DownloadAvailability.Query.	object	-	This object contains multicast query parameters used to download files.	-
Enable	boolean	W	Enable/disable CPE ability to autonomously query a server for the purpose of downloading files.	-

Name	Type	Write	Description	Object Default
Status	string	-	The status of the query service. Enumeration of: *Disabled* *Enabled* *Error* (MAY be used by the CPE to indicate a locally defined error condition, e.g. unable to contact query response server., OPTIONAL)	-
URL	string(256)	W	URL RFC3986[17] of the query response server. Depending on the application, the protocol described by this URL MAY be a SOAP interface, or MAY be any other RPC mechanism. Refer to DVB-TS.102.824[5] for an example of a URL format that identifies a SOAP interface running over HTTP or HTTPS.	-

4.2 Inform Requirements

This specification does not change any of the Inform Requirements for the Device Root Object as they are defined in Section 3.5/TR-106 Amendment 2 [2] or the InternetGatewayDevice Root Object as they are defined in Section 2.4.1/TR-098 Amendment 2 [3].

4.3 Notification Requirements

CPE MUST support Active Notifications (see TR-069 Amendment 2 [1]) for all parameters defined in Table 1 with the exception of those parameters listed in Table 2. For only those parameters listed in Table 2, the CPE MAY reject a request by an ACS to enable an Active Notification via the SetParameterAttributes RPC by responding with fault code 9009 as defined in TR-069 Amendment 2 [1] (Notification request rejected).

CPE MUST support Passive Notifications (see TR-069 Amendment 2 [1]) for all parameters defined in Table 1, with no exceptions.

Table 2 Parameters for which Active Notification MAY be denied by the CPE

Name
.DeviceInfo.MemoryStatus.
Free
.DeviceInfo.ProcessStatus.
CPUUsage
.DeviceInfo.ProcessStatus.Process.{i}.
Size
CPUTime
State
.DeviceInfo.TemperatureStatus.TemperatureSensor.{i}.
Value
LastUpdate
.UserInterface.LocalDisplay.
PosX
PosY
Width
Height
.SmartCardReader.{i}.
DecryptionFailedCounter
DecryptionFailedNoKeyCounter
.SelfTestDiagnostics.
DiagnosticsState
Results
.NSLookupDiagnostics.
DiagnosticsState
SuccessCount
ResultNumberOfEntries
.NSLookupDiagnostics.Result.{i}.
Status
AnswerType
HostNameReturned
IPAddresses
DNSServerIP
ResponseTime
.PeriodicStatistics.SampleSet.{i}.
SampleSeconds

Name
.PeriodicStatistics.SampleSet.{i}.Parameter.{i}.
SampleSeconds
SuspectData
Values

5 Profile Definition

5.1 Notation

The following abbreviations are used to specify profile requirements:

Abbreviation	Description
R	Read support is REQUIRED
W	Both Read and Write support is REQUIRED. This MUST NOT be specified for a parameter that is defined as read-only.
P	This object is REQUIRED to be present.
C	Creation and Deletion of instances of the object via AddObject and DeleteObject is REQUIRED.
A	Creation of instance of the object via AddObject is REQUIRED, but Deletion is not required.
D	Deletion of instance of the object via DeleteObject is REQUIRED, but Creation is not required.

5.2 Memory Status Profile

Table 3 defines the MemoryStatus:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 3 MemoryStatus:1 Profile definition

Name	Requirement
.DeviceInfo.MemoryStatus.	P
Total	R
Free	R

5.3 Process Status Profile

Table 4 defines the ProcessStatus:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 4 ProcessStatus:1 Profile definition

Name	Requirement
.DeviceInfo.ProcessStatus.	P
CPUUsage	R
ProcessNumberOfEntries	R
.DeviceInfo.ProcessStatus.Process.{i}.	P
PID	R
Command	R
Size	R
Priority	R
CPUTime	R
State	R

5.4 Temperature Status Profile

Table 5 defines the TempStatus:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 5 TempStatus:1 Profile definition

Name	Requirement
.TemperatureStatus.	P
TemperatureSensorNumberOfEntries	R
.TemperatureStatus.TemperatureSensor.{i}.	P
Enable	W
Status	R
ResetTime	R
Name	R
Value	R
LastUpdate	R
MinValue	R
MinTime	R
MaxValue	R
MaxTime	R

5.5 Temperature Status Advanced Profile

The TempStatusAdv:1 profile is defined as the union of the TempStatus:1 profile and the additional requirements defined in Table 6. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 6 TempStatusAdv:1 Profile definition

Name	Requirement
.TemperatureStatus.TemperatureSensor.{i}.	P
Reset	W
LowAlarmValue	W
LowAlarmTime	R
HighAlarmValue	W
HighAlarmTime	R

5.6 Autonomous Transfer Complete Policy Profile

Table 7 defines the AutonXferComplPolicy:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 7 AutonXferComplPolicy:1 Profile definition

Name	Requirement
.ManagementServer.AutonomousTransferCompletePolicy.	P
Enable	W
TransferTypeFilter	W
FileTypeFilter	W

5.7 User Profile

Table 8 defines the User:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 8 User:1 Profile definition

Name	Requirement
UserNumberOfEntries	R
.User.{i}.	C
Enable	W
Username	W
Password	W

5.8 UPnP Device Profile

Table 9 defines the UPnPDev:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 9 UPnPDev:1 Profile definition

Name	Requirement
.UPnP.	P
.UPnP.Device.	P
Enable	W
UPnPMediaServer	W
UPnPMediaRenderer	W
UPnPWLANAccessPoint	W
UPnPQoSDevice	W
UPnPQoSPolicyHolder	W
UPnPIGD	W
.UPnP.Device.Capabilities.	P
UPnPArchitecture	R
UPnPMediaServer	R
UPnPMediaRenderer	R
UPnPWLANAccessPoint	R
UPnPBasicDevice	R
UPnPQoSDevice	R
UPnPQoSPolicyHolder	R
UPnPIGD	R

5.9 UPnP Discovery Basic Profile

Table 10 defines the UPnPDiscBasic:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 10 UPnPDiscBasic:1 Profile definition

Name	Requirement
.UPnP.	P
.UPnP.Discovery.	P
RootDeviceNumberOfEntries	R
.UPnP.Discovery.RootDevice.{i}.	P
Status	R
UUID	R
USN	R
LeaseTime	R
Location	R
Server	R

5.10 UPnP Discovery Advanced Profile

The UPnPDiscAdv:1 profile is defined as the union of the UPnPDiscBasic:1 profile and the additional requirements defined in Table 11. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 11 UPnPDiscAdv:1 Profile definition

Name	Requirement
.UPnP.Discovery.	P
DeviceNumberOfEntries	R
ServiceNumberOfEntries	R
.UPnP.Discovery.Device.{i}.	P
Status	R
UUID	R
USN	R
LeaseTime	R
Location	R
Server	R
.UPnP.Discovery.Service.{i}.	P
Status	R
USN	R
LeaseTime	R
Location	R
Server	R

5.11 Self Test Diagnostics Profile

Table 12 defines the SelfTestDiag:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 12 SelfTestDiag:1 Profile definition

Name	Requirement
.SelfTestDiagnostics.	P
DiagnosticsState	W
Results	R

5.12 Namespace Lookup Diagnostics Profile

Table 13 defines the NSLookupDiag:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 13 NSLookupDiag:1 Profile definition

Name	**Requirement**
.NSLookupDiagnostics.	P
DiagnosticsState	W
Interface	W
HostName	W
DNSServer	W
Timeout	W
NumberOfRepititions	W
SuccessCount	R
ResultNumberOfEntries	R
.NSLookupDiagnostics.Result.{i}.	P
Status	R
AnswerType	R
HostNameReturned	R
IPAddressess	R
DNSServerIP	R
ResponseTime	R

5.13 Simple Firewall Profile

Table 14 defines the SimpleFirewall:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 14 SimpleFirewall:1 Profile definition

Name	**Requirement**
.Firewall.	P
Config	W
Version	R
LastChange	R

5.14 USB Hosts Basic Profile

Table 15 defines the USBHostsBasic:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 15 USBHostsBasic:1 Profile definition

Name	Requirement
.USBHosts.	P
HostNumberOfEntries	R
.USBHosts.Host.{i}.	P
Enable	W
Name	R
Type	R
USBVersion	R
DeviceNumberOfEntries	R
.USBHosts.Host.{i}.Device.{i}.	P
DeviceNumber	R
USBVersion	R
DeviceClass	R
DeviceSubClass	R
DeviceVersion	R
DeviceProtocol	R
ProductID	R
VendorID	R
Manufacturer	R
ProductClass	R
SerialNumber	R
Port	R
Rate	R
Parent	R
MaxChildren	R
ConfigurationNumberOfEntries	R
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.	P
ConfigurationNumber	R
InterfaceNumberOfEntries	R
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.Interface.{i}.	P
InterfaceNumber	R
InterfaceClass	R
InterfaceSubClass	R
InterfaceProtocol	R

5.15 USB Hosts Advanced Profile

The USBHostsAdv:1 profile is defined as the union of the USBHostsBasic:1 profile and the additional requirements defined in Table 16. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 16 USBHostsAdv:1 Profile definition

Name	Requirement
.USBHosts.Host.{i}.	P
Reset	W
PowerManagementEnable	W
.USBHosts.Host.{i}.Device.{i}.	P
IsSuspended	R
IsSelfPowered	R
ConfigurationNumberOfEntries	R
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.	P
ConfigurationNumber	R
InterfaceNumberOfEntries	R
.USBHosts.Host.{i}.Device.{i}.Configuration.{i}.Interface.{i}.	P
InterfaceNumber	R
InterfaceClass	R
InterfaceSubClass	R
InterfaceProtocol	R

5.16 Periodic Statistics Base Profile

Table 17 defines the PeriodicStatsBase:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 17 PeriodicStatsBase:1 Profile definition

Name	Requirement
.PeriodicStatistics.	P
MinSampleInterval	R
MaxReportSamples	R
SampleSetNumberOfEntries	R
.PeriodicStatistics.SampleSet.{i}.	C
Name	W
SampleInterval	W
ReportSamples	W
ReportStartTime	R
ReportEndTime	R
SampleSeconds	R
ParameterNumberOfEntries	R
.PeriodicStatistics.SampleSet.{i}.Parameter.{i}.	C
Reference	W
SampleSeconds	R
SuspectData	R
Values	R

5.17 Periodic Statistics Advanced Profile

The PeriodicStatsAdv:1 profile is defined as the union of the PeriodicStatsBase:1 profile and the additional requirements defined in Table 18. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 18 PeriodicStatsAdv:1 Profile definition

Name	**Requirement**
.PeriodicStatistics.SampleSet.{i}.	C
Enable	W
Status	R
TimeReference	W
FetchSamples	W
ForceSamples	W
.PeriodicStatistics.SampleSet.{i}.Parameter.{i}.	C
Enable	W
SampleMode	W
CalculationMode	W
LowThreshold	W
HighThreshold	W
Failures	R

5.18 Download Availability Announcement Profile

Table 19 defines the DownloadAnnounce:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 19 DownloadAnnounce:1 Profile definition

Name	Requirement
.DownloadAvailability.	P
.DownloadAvailability.Announcement.	P
Enable	W
Status	R
GroupNumberOfEntries	R
.DownloadAvailability.Announcement.Group.{i}.	C
Enable	W
Status	R
URL	W

5.19 Download Availability Query Profile

Table 20 defines the DownloadQuery:1 profile for the Device:1 root object and the InternetGatewayDevice:1 root object. The minimum REQUIRED version for this profile is Device:1.3 and InternetGatewayDevice:1.5.

Table 20 DownloadQuery:1 Profile definition

Name	Requirement
.DownloadAvailability.	P
.DownloadAvailability.Query.	P
Enable	W
Status	R
URL	W

Appendix I. USB Host Theory of Operation

An increasing number of devices are equipped with a USB Host controller and USB host interface(s) / connector(s) (series A receptacle).

There are a number of use cases for adding a USB Host and connected devices to a CWMP data model. One example is retrieving the exact product identity of the connected device in the event of service issues such as printer or file sharing problems. Another example is notifying the user that a newly-connected device is not supported, e.g. due to a missing driver. Or the detection of the connection of a particular USB device could mean additional services for this device could be offered to the subscriber.

The data model contains the number of devices connected to each host controller. For each device, the main properties of the USB device descriptors as well as interface descriptors are represented. The latter is to support devices that only indicate class/subclass (and therefore device type) at the interface level.

Example USB topology of connected devices:

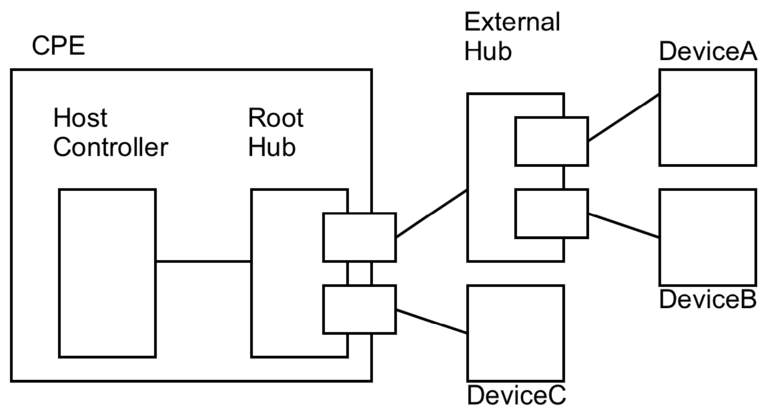

Figure 1 - Example USB Host Connections

All USB devices attach to a USB Host through a port on a USB entity called known as a hub. Hubs have status bits that are used to report the attachment or removal of a USB device on one of its ports. The USB Host queries the hub to retrieve these status bits. In the case of an attachment, the USB Host enables the port and addresses the USB device through the device's control pipe at the default address. Figure 1 depicts both a Root Hub and an External Hub that provide this service.

The USB Host assigns a unique USB address to the device and then determines if the newly attached USB device is a hub or function. The USB Host establishes its end of the control pipe for the USB using the assigned USB address and endpoint number zero. This is reflected in the data model by adding a new USBHosts.Host.{i}.Device.{i}. instance.

If the attached USB device is a hub and USB devices are attached to its ports, then the above procedure is followed for each of the attached USB devices.

If the attached USB device is a function, then attachment notifications will be handled by the USB Host software that is appropriate for the function.

End of Broadband Forum Technical Report TR-157