TECHNICAL REPORT

DSL Forum
TR-106 Amendment 1

Data Model Template
for TR-069-Enabled Devices

November 2006

Produced by:
DSLHome-Technical Working Group

Editors:

Jeff Bernstein, 2Wire
Tim Spets, Westell
Christele Bouchat, Alcatel
John Blackford, 2Wire
Mike Digdon, SupportSoft
Heather Kirksey, Motive
William Lupton, 2Wire
Anton Okmianski, Cisco

Working Group Chair:

Greg Bathrick, PMC-Sierra
Heather Kirksey, Motive

Abstract:
This document specifies a generic data model applicable to all TR-069-enabled devices.

Notice:

The DSL Forum is a non-profit corporation organized to create guidelines for DSL network system development and deployment. This Technical Report has been approved by members of the Forum. This document is not binding on the DSL Forum, any of its members, or any developer or service provider. The document is subject to change, but only with approval of members of the Forum.

Version History

Version Number	Version Date	Version Editor	Changes
Issue 1	September 2005	Jeff Bernstein, 2Wire Christele Bouchat, Alcatel Tim Spets, Westell	Issue 1
Issue 1 Amendment 1	November 2006	Jeff Bernstein, 2Wire John Blackford, 2Wire Mike Digdon, SupportSoft Heather Kirksey, Motive William Lupton, 2Wire Anton Okmianski, Cisco	Clarification of original document

Contents

1 Introduction

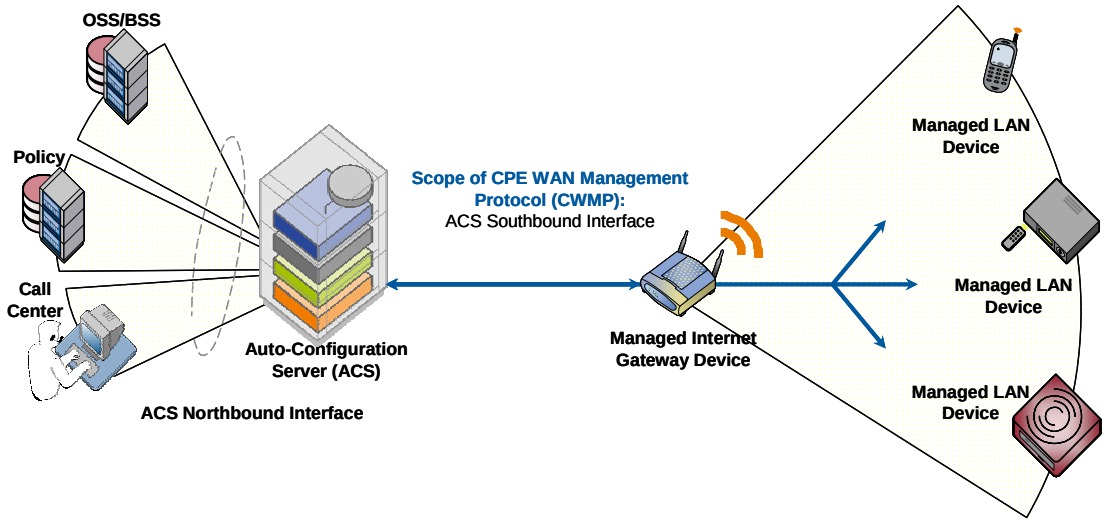
This document specifies a baseline object structure and set of TR-069-accessible parameters to be available on any TR-069-enabled device [2]. TR-069 defines the generic requirements of the management protocol methods which can be applied to any TR-069 CPE. It is intended to support a variety of different functionalities to manage a collection of CPE, including the following primary capabilities:

- Auto-configuration and dynamic service provisioning
- Software/firmware image management
- Status and performance monitoring
- Diagnostics

If TR-069 defines the generic methods for any device, other documents (such as this one) specify the managed objects, or data models, on which the generic methods act to configure, diagnose, and monitor the state of specific devices and services.

The following figure places TR-069 in the end-to-end management architecture:

Figure 1 – Positioning in the End-to-End Architecture



The ACS is a server that resides in the network and manages devices in the subscriber premises. It uses the methods, or RPCs, defined to TR-069 to get and set the state of the device, initiate diagnostic tests, download and upload files, and manage events. Some portions of this state are common across managed devices and some are relevant only to certain devices types or services.

For a particular type of device, it is expected that the baseline defined in this document would be augmented with additional objects and parameters specific to the device type. The data-model used in any TR-069-capable device must follow the guidelines described in this document. These guidelines include the following aspects:

- Structural requirements for the data hierarchy
- Requirements for versioning of data models
- Requirements for defining profiles
- A set of common data objects
- A baseline profile for any device supporting these common data objects

1.1 Terminology

The following terminology is used throughout the series of documents defining the CPE WAN Management Protocol.

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.
Common Object	An object defined in this specification that may be contained either directly within the "Device" Root Object or within a Service Object contained within the "Services" object.
CWMP	CPE WAN Management Protocol. Defined in [2], 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.
Data Model	A hierarchical set of Parameters that define the managed objects accessible via TR-069 for a particular device or service.
Device	Used here as a synonym for CPE.
Event	An indication that something of interest has happened that requires the CPE to notify the ACS.
Internet Gateway Device	A CPE device that is either a B-NT (broadband network termination) or a broadband router.
Object	A named collection of Parameters and/or other Objects.
Parameter	A name-value pair representing a manageable CPE parameter made accessible to an ACS for reading and/or writing.
RPC	Remote Procedure Call.
Profile	A named collection of requirements relating to a given object.
Root Object	The top-level object of a device's data model that contains all of the manageable objects. The name of the Root Object is either "Device" or "InternetGatewayDevice"—the former is used for all types of devices except an Internet Gateway Device.
Service Object	The top-most object associated with a specific service or application within which all objects and parameters associated with the service are contained.

1.2 Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

2 Architecture

2.1 Data Hierarchy

The data-model for a TR-069-capable device will follow a common set of structural requirements. The detailed structure depends on the nature of the device.

A device will always have a single Root Object, which will be called either "Device" or "InternetGateway-Device". The latter is exclusively to accommodate the existing TR-098 specification and is only to be used if the device is an Internet Gateway Device.

In most cases, the Root Object contains two types of sub-elements: the Common Objects defined in this specification (applicable only to the "Device" Root Object), and a single "Services" object that contains all Service Objects associated with specific services or applications.

To accommodate the existing TR-098 specification, if the device is an Internet Gateway Device, the Root Object will also contain the application-specific objects associated with an Internet Gateway Device. In this case, the InternetGatewayDevice object plays the role of both a Root Object and a Service Object.

A single device might include more than one Service Object. For example, a device that serves both as a VoIP endpoint and a game device, might include both VoIP-specific and game-specific Service Objects.

A single device might also include more than one instance of the same type of Service Object. An example of when this might be appropriate is a TR-069 capable device that proxies the management functions for one or more other devices that are not TR-069 capable. In this case, the ACS would communicate directly only with the TR-069 capable device, which would incorporate the data-models for all devices for which it is serving as a management proxy. For example, a video device serving as a management proxy for three VoIP phones would contain in its data model a video-specific Service Object plus three instances of a VoIP-specific Service Object. Note that whether a device is serving as a management proxy for another device or whether it has that functionality embedded in it is generally opaque to the ACS.

2.1.1 Data Hierarchy Requirements

The data model for a TR-069-capable device (other than an Internet Gateway Device) MUST adhere to the following structural requirements:

1) The data model MUST contain exactly one Root Object, called "Device".
2) The Root Object MUST contain a "DeviceSummary" parameter as specified in section 3.7.
3) The Root Object MAY contain any of the Common Objects defined in section 3.4.
4) The Root Object MUST contain exactly one "Services" object.
5) The "Services" object MUST contain all of the Service Objects supported by the device. Each Service Object contains all of the objects and parameters for a particular service or application.
6) The "Services" object MAY contain more than one Service Object, each corresponding to a distinct service or application type.
7) The "Services" object MAY contain more than one instance of a Service Object of the same type.
8) Each Service Object instance MUST be appended with an instance number (assigned by the CPE) to allow for the possibility of multiple instances of each. For example, if the device supports the Service Object ABCService, the first instance of this Service Object might be "ABCService.1".
9) For each supported type of Service Object, a corresponding parameter in the "Services" object MUST indicate the number of instances of that Service Object type. If a particular Service Object type is supported by the device but there are currently no instances present, this parameter MUST still be present with a value of zero. The name of this parameter MUST be the name of the Service Object concatenated with "NumberOfEntries". For example, for a device that contains instances of ABCService, there MUST be a corresponding parameter in the "Services" object called "ABCServiceNumberOfEntries".
10) Each Service Object MAY contain secondary copies of some of the Common Objects defined in this specification. The specific set of Common Objects that might be contained within a Service Object is specified in section 3.4.

An Internet Gateway Device MUST adhere to the above requirements with the following exceptions:

1) The data model MUST contain exactly one Root Object, called "InternetGatewayDevice".

2) The Root Object MAY contain any of the objects specific to an Internet Gateway Device as defined in [3].

3) The "InternetGatewayDevice" Root Object MUST NOT directly contain any of the Common Objects defined in this specification. While [3] defines objects very similar to some of the Common Objects defined here, they are not identical and MUST NOT be considered the same as the Common Objects. (Service Objects within the "Services" object MAY contain Common Objects with the limitations specified in section 3.4.)

4) The "Services" object MAY be absent if the device supports no Service Objects other than InternetGatewayDevice.

5) The "DeviceSummary" parameter MAY be absent only in an Internet Gateway Device that supports the InternetGatewayDevice version 1.0 data-model, as defined in section 2.4.2 of [3], and no other Service Objects.[1]

Formally, the top level of the data hierarchy is defined as follows:

```
Element = Root
        | Root ".DeviceSummary"
        | Root ".Services." ServiceObject "." Instance
        | Root ".Services." ServiceObject "NumberOfEntries"
        | Root ".Services." ServiceObject "." Instance "." SecondaryCommonObject
        | DeviceRoot "." CommonObject
        | GatewayRoot "." GatewaySpecificObject   ; As defined in [3]

Root = DeviceRoot
     | GatewayRoot

DeviceRoot = "Device"

GatewayRoot = "InternetGatewayDevice"

CommonObject  = "DeviceInfo"
              | "Config"
              | "UserInterface"
              | "ManagementServer"
              | "GatewayInfo"
              | "Time"
              | "LAN"

SecondaryCommonObject  = "DeviceInfo"
                       | "Config"
                       | "UserInterface"
                       | "Time"
                       | "LAN"

Instance = NONZERODIGIT [DIGIT]*
```

2.1.2 Data Hierarchy Examples

Below are some examples of data hierarchies for various types of devices. (Objects are shown in bold text, parameters are shown in plain text.)

Simple device supporting the ABCService Service Object:

Device
DeviceSummary
DeviceInfo
ManagementServer
Services

[1] The implication of this requirement is that if an Internet Gateway Device supports one or more Service Objects (for example, the VoiceService object defined in TR-104), the Internet Gateway Device is required to support version 1.1 or greater of the InternetGatewayDevice root object as defined in TR-098.

ABCServiceNumberOfEntries = 1
ABCService.1
ABCServiceSpecificObjects

Device supporting both ABCService and XYZService Service Objects:

Device
DeviceSummary
DeviceInfo
ManagementServer
Time
UserInterface
LAN
Services
ABCServiceNumberOfEntries = 1
ABCService.1
ABCServiceSpecificObjects
XYZServiceNumberOfEntries = 1
XYZService.1
XYZServiceSpecificObjects

Internet Gateway Device that also supports the ABCService and XYZService Service Objects:

InternetGatewayDevice
DeviceSummary
DeviceInfo
ManagementServer
Time
UserInterface
Layer3Forwarding
LANDeviceNumberOfEntries = 1
LANDevice.1
WANDeviceNumberOfEntries = 1
WANDevice.1
Services
ABCServiceNumberOfEntries = 1
ABCService.1
ABCServiceSpecificObjects
XYZServiceNumberOfEntries = 1
XYZService.1
XYZServiceSpecificObjects

Device supporting the ABCService Service Object and proxying for two devices supporting the functionality of the XYZService Service Object:

Device
DeviceSummary
DeviceInfo
ManagementServer
Config
GatewayInfo
Time
UserInterface
LAN
Services
ABCServiceNumberOfEntries = 1
ABCService.1
ABCServiceSpecificObjects
XYZServiceNumberOfEntries = 2
XYZService.1
DeviceInfo
XYZServiceSpecificObjects

XYZService.2
DeviceInfo
XYZServiceSpecificObjects

Internet Gateway Device also serving as a management proxy for three devices supporting the functionality of the ABCService Service Object:

InternetGatewayDevice
DeviceSummary
DeviceInfo
ManagementServer
Time
UserInterface
Layer3Forwarding
LANDeviceNumberOfEntries = 1
LANDevice.1
WANDeviceNumberOfEntries = 1
WANDevice.1
Services
ABCServiceNumberOfEntries = 3
ABCService.1
DeviceInfo
ABCServiceSpecificObjects
ABCService.2
DeviceInfo
ABCServiceSpecificObjects
ABCService.3
DeviceInfo
ABCServiceSpecificObjects

2.2 Object Versioning

To allow the definition of a Service Object or Root Object to change over time, the definition of a Service Object or Root Object MUST have an explicitly specified version.

Version numbering of Service Objects and Root Objects is defined to use a major/minor version numbering convention. The object version is defined as a pair of integers, where one integer represents the major version, and the second integer represents the minor version. The version MUST be written with the two integers separated by a dot (Major.Minor).

The first version of a given object SHOULD be defined as version "1.0".

For each subsequent version of the object, if the later version is compatible with the previous version, then the major version SHOULD remain unchanged, and the minor version SHOULD be incremented by one. For example, the next compatible version after "2.17" would be "2.18". The requirements for a version to be considered compatible with an earlier version are described in section 2.2.1.

For each subsequent version of the object, if the later version is not compatible with the previous version, then the major version MUST increment by one, and the minor version MAY reset back to zero. For example, the next incompatible version after "2.17" might be "3.0".

2.2.1 Requirements for Compatible Versions

For one version of an object to be considered compatible with another version, the later version MUST be a strict superset of the earlier version. Using major/minor versioning, this requirement applies only between minor versions that share the same major version.

More specifically, this requires the following of the later version with respect to all earlier versions to which it is to be compatible:

- The later version MAY add objects and parameters not previously in any earlier version, but MUST NOT remove objects or parameters already defined in earlier versions.

- The later version MUST NOT modify the definition of any parameter or object already defined in an earlier version (unless the original definition was clearly in error and has to be modified as an erratum or clarified through a corrigendum process).
- The later version MUST NOT require any of the objects or parameters that have been added since the earliest compatible version to be explicitly operated upon by the ACS to ensure proper operation of the device (except those functions specifically associated with functionality added in later versions). That is, the later version will accommodate an ACS that knows nothing of elements added in later versions.

The goal of the above definition of compatibility is intended to ensure bi-directional compatibility between an ACS and CPE. Specifically that:

- If an ACS supports only an earlier version of an object as compared to the version supported by the CPE, the ACS can successfully manage that object in the CPE as if it were the earlier version.
- If a CPE supports only an earlier version of an object as compared to the version supported by an ACS, the ACS can successfully manage that object in the CPE as if it were the later version (without support for new components defined only in later versions).

2.2.2 Version Notation

For objects, the following notation is defined to identify specific versions:

Notation	Description	Example
ObjectName:Major.Minor	Refers to a specific version of the object.	Device:1.0
ObjectName:Major	Refers to any minor version of the object with the specified major version.	Device:1
ObjectName	Refers to any version of the object.	Device

Note that the version notation defined here is *only* to be used for purposes of documentation and in the content of the DeviceSummary parameter defined in section 3.7. The actual names of objects and parameters in the data model MUST NOT include version numbers.

2.3 Profiles

To limit the variability that an ACS needs to accommodate among various devices that it might manage, it is useful to define "profiles" that express specific sets of requirements, support for which can be explicitly indicated by a device.

A profile is a named collection of requirements associated with a given object. A device can indicate support for one or more profiles. A device supporting a profile means that the device supports all of the requirements defined by that profile. When a device supports all requirements defined by a profile, the device MUST indicate support for that profile. The use of profiles allows the ACS a shorthand means of discovering support for entire collections of capabilities in a device.

The following sections define the conventions to be used when defining profiles associated with TR-069 data models.

2.3.1 Scope of Profiles

A given profile is defined only in the context of a specific Service Object or Root Object with a specific major version. For each profile definition, the specific object name and major version to which the profile is to apply MUST be explicitly identified.

A profile's name MUST be unique among profiles defined for the same object and major version, but a name MAY be reused to define a different profile for a distinct combination of object name and major version. For example, if we define profile "A" associated with object "X:2" (major version 2 of object X), the same name "A" might be used to define a different profile for object "Y:1" or for object "X:3".

A given profile is defined in association with a minimum minor version of a given object. The minimum required version of an object is the minimum version that includes all of the required elements defined by the profile. For each profile definition, the specific minimum version MUST be explicitly identified.

2.3.2 Multiple Profile Support

For a given type of Service Object, multiple profiles MAY be defined. Profiles MAY be defined that have either independent or overlapping requirements.

To maximize interoperability, a device MUST indicate all profiles that it supports. That is, it MUST indicate all profiles whose definition is a subset of the support provided by that device. Doing so maximizes the likelihood that an ACS will be aware of the definition of the indicated profiles. For example, if profile "A" is a subset of profile "B", and a device supports both, by indicating support for both "A" and "B" an ACS that is unaware of profile "B" will at least recognize the device's support for profile "A".

2.3.3 Profile Versions

To allow the definition of a profile to change over time, the definition of every profile MUST have an associated version number.

Version numbering of profiles is defined to use a minor-only version numbering convention. That is, for a given profile name, each successive version MUST be compatible with all earlier versions. Any incompatible change to a profile MUST use a different profile name.

For one version of a profile to be considered compatible with another version, the later version MUST be a strict superset of the earlier version. This requires the following of the later version with respect to all earlier versions to which it is to be compatible:

- The later version MAY add requirements that were not in earlier versions of the profile, but MUST NOT remove requirements.
- The later version MAY remove one or more conditions that had previously been placed on a requirement. For example, if a previous profile required X only if condition A was true, then the later profile might require X unconditionally.

For profiles, the following notation is defined to identify specific versions:

Notation	Description	Example
ProfileName:Version	Refers to a specific version of the profile.	Baseline:1
ProfileName	Refers to any version of the profile.	Baseline

2.3.4 Baseline Profiles

For every Service Object (and Root Object) there SHOULD be at least one profile defined. In many cases it is desirable to define a Baseline profile that indicates the minimum requirements required for any device that supports that object. Where a Baseline profile is defined, it would normally be expected that all implementations of the corresponding object would indicate support for the Baseline profile in addition to any other profiles supported.

2.3.5 Types of Requirements in a Profile

Because a profile is defined within the context of a single object (and major version), all of the requirements associated with the profile MUST be specific to the data-model associated with that object.

Profile requirements can include any of the following types of requirements associated with an object's data model:

- A requirement for read support of a Parameter.
- A requirement for write support of a Parameter.

- A requirement for support of a sub-object contained within the overall object.
- A requirement for the ability to add or remove instances of a sub-object.
- A requirement to support active and/or passive notification for a Parameter.
- A requirement to support access control for a given Parameter.

For each of the requirement categories listed above, a profile can define the requirement unconditionally, or can place one or more conditions on the requirement. For example, a profile might require that a Parameter be supported for reading only if the device supports some other parameter or object (one that is not itself required by the profile). Such conditions will be directly related to the data model of the overall object associated with the profile.

Because a device has to be able to support multiple profiles, all profiles MUST be defined such they are non-contradictory. As a result, profiles MUST only define minimum requirements to be met, and MUST NOT specify negative requirements. That is, profiles will not include requirements that specify something that is not to be supported by the device, or requirements that exclude a range of values.

2.4 DEPRECATED and OBSOLETED Items

The key word "DEPRECATED" in the data-model definition for any TR-069-capable device is to be interpreted as follows: This term refers to an object, parameter or parameter value that is defined in the current version of the standard but is meaningless, inappropriate, or otherwise unnecessary. It is intended that such objects, parameters or parameter values will be removed from the next major version of the data-model. Requirements on how to interpret or implement deprecated objects, parameters or parameter values are given below. For more information on how to interpret or implement specific deprecated objects, parameters or parameter values, refer to the definition of the object or parameter.

The key word "OBSOLETED" in the data-model definition for any TR-069-capable device is to be interpreted as follows: This term refers to an object, parameter or parameter value that meets the requirements for being deprecated, and in addition is obsolete. Such objects, parameters or parameter values can be removed from a later minor version of a data-model, or from a later version of a profile, without this being regarded as breaking backwards compatibility rules. Requirements on how to interpret or implement obsoleted objects, parameters or parameter values are given below. For more information on how to interpret or implement specific obsoleted objects, parameters or parameter values, refer to the definition of the object or parameter.

2.4.1 Requirements for DEPRECATED Items

This section defines requirements that apply to all DEPRECATED objects, parameters and parameter values unless specifically overridden by the object or parameter definition.

Data-model requirements:

1) The definition of a DEPRECATED parameter, object or parameter value MUST include an explanation of why the item is deprecated.
2) The definition of a DEPRECATED parameter, object or parameter value MAY specify further requirements relating to the item; such requirements MAY override CPE or ACS requirements specified in this section.

CPE requirements:

1) A DEPRECATED parameter MUST have a value which is valid for its data type and fulfils any range (for numeric parameters), length (for string or base64 parameters) and enumerated value (for string parameters) requirements.
2) Detailed behavioral requirements for a DEPRECATED parameter, e.g. that its value is a unique key, MAY be ignored by the CPE.

3) The CPE MUST, if such operations are permitted by the data model definition, permit creation of DEPRECATED objects, modification of DEPRECATED parameters, and setting of DEPRECATED parameter values. However, it MAY choose not to apply such changes to its operational state.

4) Regardless of whether DEPRECATED changes are applied to the CPE operational state, a read of a DEPRECATED writable parameter SHOULD return the value that was last written, i.e. the CPE is expected to store the value even if it chooses not to apply it to its operational state.

5) When the ACS modifies the value of a DEPRECATED parameter, the CPE MAY choose not to check whether the new parameter value is valid for its data type and fulfils any range (for numeric parameters), length (for string or base64 parameters) and enumerated value (for string parameters) requirements.

6) The CPE MAY reject an attempt by the ACS to set any parameter to a DEPRECATED value.

ACS requirements:

1) The ACS SHOULD NOT create DEPRECATED objects, modify DEPRECATED parameters, or set DEPRECATED parameter values.

2) The ACS SHOULD ignore DEPRECATED objects, parameters and parameter values.

3) The ACS MUST NOT set a DEPRECATED parameter to a value that is invalid for its data type or fails to fulfil any range (for numeric parameters), length (for string or base64 parameters) or enumerated value (for string parameters) requirements.

4) The ACS MUST NOT set any parameter to a DEPRECATED value.

2.4.2 Requirements for OBSOLETED Items

This section defines requirements that apply to all OBSOLETED objects, parameters or parameter values unless specifically overridden by the object or parameter definition.

An OBSOLETED object, parameter or parameter must meet all the requirements of the previous section. In addition, the following data-model requirements apply.

1) An OBSOLETED object, parameter or parameter value MAY be removed from a later minor version of a data-model without this being regarded as breaking backwards compatibility rules.

2) An OBSOLETED object, parameter or parameter value MUST NOT be removed from the current version of a profile, but MAY be removed from a later version of a profile without this being regarded as breaking backwards compatibility rules.

3) A data-model definition MUST include a list of those OBSOLETED objects, parameters or parameter values that have been removed from the data-model or from its profiles. This is to prevent future namespace conflicts.

3 Object Definitions

3.1 General Notation

Parameter names use a hierarchical form similar to a directory tree. The name of a particular Parameter is represented by the concatenation of each successive node in the hierarchy separated with a "." (dot), starting at the trunk of the hierarchy and leading to the leaves. When specifying a partial path, indicating an intermediate node in the hierarchy, the trailing "." (dot) is always used as the last character.

Parameter names MUST be treated as case sensitive.

In some cases, where multiple instances of an object can occur, the placeholder node name "{i}" is shown. In actual use, this placeholder is to be replaced by an instance number, which MUST be a positive integer

(≥ 1). Because in some cases object instances can be deleted, instance numbers will in general not be contiguous.

3.2 Data Types

The parameters defined in this specification make use of a limited subset of the default SOAP data types [5]. The complete set of data types along with the notation used to represent these types is listed in Table 1.

Table 1 – Data types

Type	Description
object	A container for parameters and/or other objects. The full path name of a parameter is given by the parameter name appended to the full path name of the object it is contained within.
string	For strings listed in this specification, a maximum allowed length can be listed using the form string(N), where N is the maximum string length in characters. For all strings a maximum length is either explicitly indicated or implied by the size of the elements composing the string. For strings in which the content is an enumeration, the longest enumerated value determines the maximum length. If a string does not have an explicitly indicated maximum length or is not an enumeration, the default maximum is 16 characters.
int	Integer in the range –2147483648 to +2147483647, inclusive. For some int types listed, a value range is given using the form int[Min:Max], where the Min and Max values are inclusive. If either Min or Max are missing, this indicates no limit.
unsignedInt	Unsigned integer in the range 0 to 4294967295, inclusive. For some unsignedInt types listed, a value range is given using the form unsignedInt[Min:Max], where the Min and Max values are inclusive. If either Min or Max are missing, this indicates no limit.
boolean	Boolean, where the allowed values are “0”, “1”, “true”, and “false”. The values “1” and “true” are considered interchangeable, where both equivalently represent the logical value *true*. Similarly, the values “0” and “false” are considered interchangeable, where both equivalently represent the logical value *false*.
dateTime	The subset of the ISO 8601 date-time format defined by the SOAP dateTime type. All times MUST be expressed in UTC (Universal Coordinated Time) unless explicitly stated otherwise in the definition of a parameter of this type. If absolute time is not available to the CPE, it SHOULD instead indicate the relative time since boot, where the boot time is assumed to be the beginning of the first day of January of year 1, or 0001-01-01T00:00:00. For example, 2 days, 3 hours, 4 minutes and 5 seconds since boot would be expressed as 0001-01-03T03:04:05. Relative time since boot MUST be expressed using an untimezoned representation. Any untimezoned value with a year value less than 1000 MUST be interpreted as a relative time since boot. If the time is unknown or not applicable, the following value representing “Unknown Time” MUST be used: 0001-01-01T00:00:00Z. Any dateTime value other than one expressing relative time since boot (as described above) MUST use timezoned representation (that is, it MUST include a timezone suffix).
base64	Base64 encoded binary. A maximum allowed length can be listed using the form base64(N), where N is the maximum length in characters after Base64 encoding.

All IP addresses are represented as strings either using IPv4 dotted-decimal notation or using any of the IPv6 text representations defined in [7]. Unspecified or inapplicable IP addresses and subnet masks MUST be represented as empty strings unless otherwise specified by the parameter definition.

All MAC addresses are represented as strings of 12 hexadecimal digits (digits 0-9, letters A-F or a-f) displayed as six pairs of digits separated by colons. Unspecified or inapplicable MAC addresses MUST be represented as empty strings unless otherwise specified by the parameter definition.

For unsignedInt parameters that are used for statistics, e.g. for byte counters, the actual value of the statistic might be greater than the maximum value that can be represented as an unsignedInt. Such values SHOULD wrap around through zero.

For strings that are defined to contain comma-separated lists, the format is defined as follows. Between every pair of successive items in a comma-separated list there MUST be a separator. The separator MUST include exactly one comma character, and MAY also include one or more space characters before or after the comma. The entire separator, including any space characters, MUST NOT be considered part of the list items it separates. The last item in a comma-separated list MUST NOT be followed with a separator. Individual items in a comma-separated list MUST NOT include a space or comma character within them. If an item definition requires the use of spaces or commas, that definition MUST specify the use of an escape mechanism that prevents the use of these characters.

For string parameters whose value is defined to contain the full hierarchical name of an object, the representation of the object name MUST NOT include a trailing "dot." An example of a parameter of this kind in the InternetGatewayDevice data model is InternetGatewayDevice.Layer3Forwarding.Default-ConnectionService. For this parameter, the following is an example of a properly formed value:

InternetGatewayDevice.WANDevice.1.WANConnectionDevice.2.WANPPPConnection.1

3.3 Vendor-Specific Parameters

A vendor MAY extend the standardized parameter list with vendor-specific parameters and objects. Vendor-specific parameters and objects MAY be defined either in a separate naming hierarchy or within the standardized naming hierarchy.

The name of a vendor-specific parameter or object not contained within another vendor-specific object MUST have the form:

X_<VENDOR>_VendorSpecificName

In this definition <VENDOR> is a unique vendor identifier, which MAY be either an OUI or a domain name. The OUI or domain name used for a given vendor-specific parameter MUST be one that is assigned to the organization that defined this parameter (which is not necessarily the same as the vendor of the CPE or ACS). An OUI is an organizationally unique identifier as defined in [4], which MUST formatted as a six-hexadecimal-digit string using all upper-case letters and including any leading zeros. A domain name MUST be upper case with each dot (".") replaced with a hyphen or underscore.

The VendorSpecificName MUST be a valid string as defined in 3.2, and MUST NOT contain a "." (period) or a space character.

Note – the use of the string "X_" to indicate a vendor-specific parameter implies that no standardized parameter can begin with "X_".

The name of a vendor-specific parameter or object that is contained within another vendor-specific object which itself begins with the prefix described above need not itself include the prefix.

The full path name of a vendor-specific parameter or object MUST NOT exceed 256 characters in length.

Below are some example vendor-specific parameter and object names:

Device.UserInterface.X_012345_AdBanner

Device.X_EXAMPLE-COM_MyConfig.Status

When appropriate, a vendor MAY also extend the set of values of an enumeration. If this is done, the vendor-specified values MUST be in the form "X_<VENDOR>_VendorSpecificValue". The total length of such a string MUST NOT exceed 31 characters.

3.4 Common Object Definitions

Table 2 provides a summary of the common data objects that are defined in this specification.

Table 2 – Summary of Common Data Objects

Object Name	Allowed Location in Hierarchy	Description
DeviceInfo	Root and Service Objects	General information about the device, including its identity and version information.
ManagementServer	Root	Parameters associated with the communication between the CPE and an ACS.
GatewayInfo	Root	Information to identify an Internet Gateway Device through which the CPE is connected.
Time	Root and Service Objects	Parameters associated with an NTP or SNTP time client on the CPE.
Config	Root and Service Objects	Contains general configuration state.
UserInterface	Root and Service Objects	Parameters related to the user interface of the CPE.
LAN	Root and Service Objects	Parameters related to IP-based LAN connectivity of the CPE.

Table 3 lists the Common Objects and their associated parameters defined for "Device", version 1.1. This definition is a superset of previously defined version, 1.0.

For a given implementation of this data model, the CPE MUST indicate support for the highest version number of any object or parameter that it supports. For example, even if the CPE supports only a single parameter that was introduced in version 1.1, then it will indicate support for version 1.1. The version number associated with each object and parameter is shown in the Version column of Table 3.

Table 3 – Common Object definitions for Device:1

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
DeviceSummary	string(1024)	-	See section 3.7.	-	1.0
.DeviceInfo.	object	-	This object contains general device information.	-	1.0
Manufacturer	string(64)	-	The manufacturer of the CPE (human readable string).	-	1.0

[2] The name of a Parameter is formed from the concatenation of the base path (see section 2.1), the object name shown in the yellow header, and the individual Parameter name.

[3] "W" indicates the parameter MAY be writable (if "W" is not present, the parameter is defined as read-only). For an object, "W" indicates object instances can be Added or Deleted.

[4] The default value of the parameter on creation of an object instance via TR-069. If the default value is an empty string, this is represented by the symbol <Empty>. A hyphen indicates that no default value is specified. For a parameter in which no default value is specified, on creation of an a parent object instance, the CPE MUST set the parameter to a value that is valid according to the definition of that parameter.

[5] The Version column indicates the minimum data-model version required to support the associated Parameter or Object.

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
ManufacturerOUI	string(6)	-	Organizationally unique identifier of the device manufacturer. Represented as a six hexadecimal-digit value using all upper-case letters and including any leading zeros. The value MUST be a valid OUI as defined in [4].	-	1.0
ModelName	string(64)	-	Model name of the CPE (human readable string).	-	1.0
Description	string(256)	-	A full description of the CPE device (human readable string).	-	1.0
ProductClass	string(64)	-	Identifier of the class of product for which the serial number applies. That is, for a given manufacturer, this parameter is used to identify the product or class of product over which the SerialNumber parameter is unique.	-	1.0
SerialNumber	string(64)	-	Serial number of the CPE.	-	1.0
HardwareVersion	string(64)	-	A string identifying the particular CPE model and version.	-	1.0
SoftwareVersion	string(64)	-	A string identifying the software version currently installed in the CPE. To allow version comparisons, this element SHOULD be in the form of dot-delimited integers, where each successive integer represents a more minor category of variation. For example, 3.0.21where the components mean: Major.Minor.Build.	-	1.0
EnabledOptions	string(1024)	-	Comma-separated list of the OptionName of each Option that is currently enabled in the CPE. The OptionName of each is identical to the OptionName element of the OptionStruct described in [2]. Only those options are listed whose State indicates the option is enabled.	-	1.0
AdditionalHardwareVersion	string(64)	-	A comma separated list of any additional versions. Represents any additional hardware version information the vendor might wish to supply.	-	1.0
AdditionalSoftwareVersion	string(64)	-	A comma separated list of any additional versions. Represents any additional software version information the vendor might wish to supply.	-	1.0
ProvisioningCode	string(64)	W	Identifier of the primary service provider and other provisioning information, which MAY be used by the ACS to determine service provider-specific customization and provisioning parameters.	-	1.0
DeviceStatus	string	-	Current operational status of the device. Enumeration of: “Up” ”Initializing” “Error” “Disabled”	-	1.0
UpTime	unsignedInt	-	Time in seconds since the CPE was last restarted.	-	1.0
FirstUseDate	dateTime	-	Date and time in UTC that the CPE first both successfully established an IP-layer network connection and acquired an absolute time reference using NTP or equivalent over that network connection. The CPE MAY reset this date after a factory reset. If NTP or equivalent is not available, this parameter, if present, SHOULD be set to the Unknown Time value.	-	1.0
DeviceLog	string(32K)	-	Vendor-specific log(s).	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
.ManagementServer.	object	-	This object contains parameters relating to the CPE's association with an ACS.	-	1.0
URL	string(256)	W	URL, as defined in [8], for the CPE to connect to the ACS using the CPE WAN Management Protocol. This parameter MUST be in the form of a valid HTTP or HTTPS URL. The "host" portion of this URL is used by the CPE for validating the ACS certificate when using SSL or TLS. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset.	-	1.0
Username	string(256)	W	Username used to authenticate the CPE when making a connection to the ACS using the CPE WAN Management Protocol. This username is used only for HTTP-based authentication of the CPE. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset.	-	1.0
Password	string(256)	W	Password used to authenticate the CPE when making a connection to the ACS using the CPE WAN Management Protocol. This password is used only for HTTP-based authentication of the CPE. When read, this parameter returns an empty string, regardless of the actual value. Note that on a factory reset of the CPE, the value of this parameter might be reset to its factory value. If an ACS modifies the value of this parameter, it SHOULD be prepared to accommodate the situation that the original value is restored as the result of a factory reset.	-	1.0
PeriodicInformEnable	boolean	W	Whether or not the CPE MUST periodically send CPE information to the ACS using the Inform method call.	-	1.0
PeriodicInformInterval	unsignedInt [1:]	W	The duration in seconds of the interval for which the CPE MUST attempt to connect with the ACS and call the Inform method if PeriodicInformEnable is true.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
PeriodicInformTime	dateTime	W	An absolute time reference in UTC to determine when the CPE will initiate the periodic Inform method calls. Each Inform call MUST occur at this reference time plus or minus an integer multiple of the PeriodicInformInterval. PeriodicInformTime is used only to set the "phase" of the periodic Informs. The actual value of PeriodicInformTime can be arbitrarily far into the past or future. For example, if PeriodicInformInterval is 86400 (a day) and if PeriodicInformTime is set to UTC midnight on some day (in the past, present, or future) then periodic Informs will occur every day at UTC midnight. These MUST begin on the very next midnight, even if PeriodicInformTime refers to a day in the future. The Unknown Time value defined in section 3.2 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 PeriodicInformInterval. If absolute time is not available to the CPE, its periodic Inform behavior MUST be the same as if the PeriodicInformTime parameter was set to the Unknown Time value.	-	1.0
ParameterKey	string(32)	-	ParameterKey provides the ACS a reliable and extensible means to track changes made by the ACS. The value of ParameterKey MUST be equal to the value of the ParameterKey argument from the most recent successful SetParameterValues, AddObject, or DeleteObject method call from the ACS. The CPE MUST set ParameterKey to the value specified in the corresponding method arguments if and only if the method completes successfully and no fault response is generated. If a method call does not complete successfully (implying that the changes requested in the method did not take effect), the value of ParameterKey MUST NOT be modified. The CPE MUST only modify the value of ParameterKey as a result of SetParameterValues, AddObject, DeleteObject, or due to a factory reset. On factory reset, the value of ParameterKey MUST be set to empty.	-	1.0
ConnectionRequestURL	string(256)	-	HTTP URL, as defined in [8], for an ACS to make a Connection Request notification to the CPE. In the form: http://host:port/path The "host" portion of the URL MAY be the IP address for the management interface of the CPE in lieu of a host name.	-	1.0
ConnectionRequestUsername	string(256)	W	Username used to authenticate an ACS making a Connection Request to the CPE.	-	1.0
ConnectionRequestPassword	string(256)	W	Password used to authenticate an ACS making a Connection Request to the CPE. When read, this parameter returns an empty string, regardless of the actual value.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
UpgradesManaged	boolean	W	Indicates whether or not the ACS will manage upgrades for the CPE. If true (1), the CPE SHOULD NOT use other means other than the ACS to seek out available upgrades. If false (0), the CPE MAY use other means for this purpose.	-	1.0
KickURL	string(256)	-	Present only for a CPE that supports the Kicked RPC method. LAN-accessible URL, as defined in [8], from which the CPE can be "kicked" to initiate the Kicked RPC method call. MUST be an absolute URL including a host name or IP address as would be used on the LAN side of the CPE.	-	1.0
DownloadProgressURL	string(256)	-	Present only for a CPE that provides a LAN-side web page to show progress during a file download. LAN-accessible URL, as defined in [8], to which a web-server associated with the ACS MAY redirect a user's browser on initiation of a file download to observer the status of the download.	-	1.0
UDPConnectionRequestAddress	string(256)	-	Address and port to which an ACS MAY send a UDP Connection Request to the CPE (see Annex G of [2]). This parameter is represented in the form of an Authority element as defined in [8]. The value MUST be in one of the following two forms: host:port host When STUNEnable is true, the "host" and "port" portions of this parameter MUST represent the public address and port corresponding to the NAT binding through which the ACS can send UDP Connection Request messages (once this information is learned by the CPE through the use of STUN). When STUNEnable is false, the "host" and "port" portions of the URL MUST represent the local IP address and port on which the CPE is listening for UDP Connection Request messages. The second form of this parameter MAY be used only if the port value is equal to "80".	-	1.1
UDPConnectionRequestAddressNotification-Limit	unsignedInt	W	The minimum time, in seconds, between Active Notifications resulting from changes to the UDP-ConnectionRequestAddress (if Active Notification is enabled).	-	1.1
STUNEnable	boolean	W	Enables or disables the use of STUN by the CPE. This applies only to the use of STUN in association with the ACS to allow UDP Connection Requests.	-	1.1
STUNServerAddress	string(256)	W	Host name or IP address of the STUN server for the CPE to send Binding Requests if STUN is enabled via STUNEnable. If empty and STUNEnable is true, the CPE MUST use the address of the ACS extracted from the host portion of the ACS URL.	-	1.1
STUNServerPort	unsignedInt [0:65535]	W	Port number of the STUN server for the CPE to send Binding Requests if STUN is enabled via STUNEnable. By default, this SHOULD be the equal to the default STUN port, 3478.	-	1.1

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
STUNUsername	string(256)	W	If non-empty, the value of the STUN USERNAME attribute to be used in Binding Requests (only if message integrity has been requested by the STUN server). If empty, the CPE MUST NOT send STUN Binding Requests with message integrity.	-	1.1
STUNPassword	string(256)	W	The value of the STUN Password to be used in computing the MESSAGE-INTEGRITY attribute to be used in Binding Requests (only if message integrity has been requested by the STUN server). When read, this parameter returns an empty string, regardless of the actual value.	-	1.1
STUNMaximumKeepAlivePeriod	int[-1:]	W	If STUN Is enabled, the maximum period, in seconds, that STUN Binding Requests MUST be sent by the CPE for the purpose of maintaining the binding in the Gateway. This applies specifically to Binding Requests sent from the UDP Connection Request address and port. A value of -1 indicates that no maximum period is specified.	-	1.1
STUNMinimumKeepAlivePeriod	unsignedInt	W	If STUN Is enabled, the minimum period, in seconds, that STUN Binding Requests can be sent by the CPE for the purpose of maintaining the binding in the Gateway. This limit applies only to Binding Requests sent from the UDP Connection Request address and port, and only those that do not contain the BINDING-CHANGE attribute. This limit does not apply to retransmissions following the procedures defined in [9].	-	1.1
NATDetected	boolean	-	When STUN is enabled, this parameter indicates whether or not the CPE has detected address and/or port mapping in use. A true value indicates that the received MAPPED-ADDRESS in the most recent Binding Response differs from the CPE's source address and port. When STUNEnable is false, this value MUST be false.	-	1.1
.GatewayInfo.	object	-	This object contains information associated with a connected Internet Gateway Device.	-	1.0
ManufacturerOUI	string(6)	-	Organizationally unique identifier of the associated Internet Gateway Device. An empty string indicates that there is no associated Internet Gateway Device that has been detected.	-	1.0
ProductClass	string(64)	-	Identifier of the product class of the associated Internet Gateway Device. An empty string indicates either that there is no associated Internet Gateway Device that has been detected, or the Internet Gateway Device does not support the use of the product-class parameter.	-	1.0
SerialNumber	string(64)	-	Serial number of the associated Internet Gateway Device. An empty string indicates that there is no associated Internet Gateway Device that has been detected.	-	1.0
.Config.	object	-	This object contains general configuration parameters.	-	1.0
PersistentData	string(256)	W	Arbitrary user data that MUST persist across CPE reboots.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
ConfigFile	string(32K)	W	A dump of the currently running configuration on the CPE. This parameter enables the ability to backup and restore the last known good state of the CPE. It returns a vendor-specific document that defines the state of the CPE. The document MUST be capable of restoring the CPE's state when written back to the CPE using SetParameterValues.	-	1.0
.Time.	object	-	This object contains parameters relating an NTP or SNTP time client in the CPE.	-	1.0
NTPServer1	string(64)	W	First NTP timeserver. Either a host name or IP address.	-	1.0
NTPServer2	string(64)	W	Second NTP timeserver. Either a host name or IP address.	-	1.0
NTPServer3	string(64)	W	Third NTP timeserver. Either a host name or IP address.	-	1.0
NTPServer4	string(64)	W	Fourth NTP timeserver. Either a host name or IP address.	-	1.0
NTPServer5	string(64)	W	Fifth NTP timeserver. Either a host name or IP address.	-	1.0
CurrentLocalTime	dateTime	-	The current date and time in the CPE's local time zone.	-	1.0
LocalTimeZone	string(256)	W	The local time zone definition, encoded according to IEEE 1003.1 (POSIX). The following is an example value: "EST+5 EDT,M4.1.0/2,M10.5.0/2"	-	1.0
.UserInterface.	object	-	This object contains parameters relating to the user interface of the CPE.	-	1.0
PasswordRequired	boolean	W	Present only if the CPE provides a password-protected LAN-side user interface. Indicates whether or not the local user interface MUST require a password to be chosen by the user. If false, the choice of whether or not a password is used is left to the user.	-	1.0
PasswordUserSelectable	boolean	W	Present only if the CPE provides a password-protected LAN-side user interface and supports LAN-side Auto-Configuration. Indicates whether or not a password to protect the local user interface of the CPE MAY be selected by the user directly, or MUST be equal to the password used by the LAN-side Auto-Configuration protocol.	-	1.0
UpgradeAvailable	boolean	W	Indicates that a CPE upgrade is available, allowing the CPE to display this information to the user.	-	1.0
WarrantyDate	dateTime	W	Indicates the date and time in UTC that the warranty associated with the CPE is to expire.	-	1.0
ISPName	string(64)	W	The name of the customer's ISP.	-	1.0
ISPHelpDesk	string(32)	W	The help desk phone number of the ISP.	-	1.0
ISPHomePage	string(256)	W	The URL of the ISP's home page.	-	1.0
ISPHelpPage	string(256)	W	The URL of the ISP's on-line support page.	-	1.0
ISPLogo	base64 (5460)	W	Base64 encoded GIF or JPEG image. The binary image is constrained to 4095 bytes or less.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
ISPLogoSize	unsignedInt [0:4095]	W	Un-encoded binary image size in bytes. If ISPLogoSize input value is 0 then the ISPLogo is cleared. ISPLogoSize can also be used as a check to verify correct transfer and conversion of Base64 string to image size.	-	1.0
ISPMailServer	string(256)	W	The URL of the ISP's mail server.	-	1.0
ISPNewsServer	string(256)	W	The URL of the ISP's news server.	-	1.0
TextColor	string(6)	W	The color of text on the GUI screens in RGB hexidecimal notation (e.g., FF0088).	-	1.0
BackgroundColor	string(6)	W	The color of the GUI screen backgrounds in RGB hexidecimal notation (e.g., FF0088).	-	1.0
ButtonColor	string(6)	W	The color of buttons on the GUI screens in RGB hexidecimal notation (e.g., FF0088).	-	1.0
ButtonTextColor	string(6)	W	The color of text on buttons on the GUI screens in RGB hexidecimal notation (e.g., FF0088).	-	1.0
AutoUpdateServer	string(256)	W	The server the CPE can check to see if an update is available for direct download to it. This MUST NOT be used by the CPE if the Device.-ManagementServer.UpgradesManaged parameter is true (1).	-	1.0
UserUpdateServer	string(256)	W	The server where a user can check via a web browser if an update is available for download to a PC. This MUST NOT be used by the CPE if the Device.ManagementServer.UpgradesManaged parameter is true (1).	-	1.0
AvailableLanguages	string(256)	-	Comma-separated list of user-interface languages that are available, where each language is specified according to RFC 3066 [6].	-	1.0
CurrentLanguage	string(16)	W	Current user-interface language, specified according to RFC 3066 [6].	-	1.0
.LAN.	object	-	This object contains parameters relating to IP-based LAN connectivity of a device. This object relates only to IP-layer LAN capabilities. Lower-layer aspects of LAN connectivity are not considered part of the common data model defined in this specification. For a device that contains multiple IP interfaces, the scope of this object is limited to the default IP interface. Data that might be associated with other interfaces is not considered part of the common data model defined in this specification.	-	1.0
AddressingType	string	W	The method used to assign an address to this interface. Enumeration of: "DHCP" "Static" The ability to modify this parameter is OPTIONAL.	-	1.0
IPAddress	string	W	The current IP address assigned to this interface. The ability to modify this parameter is OPTIONAL, and this parameter cannot be modified if the AddressingType is "DHCP".	-	1.0
SubnetMask	string	W	The current subnet mask. The ability to modify this parameter is OPTIONAL, and this parameter cannot be modified if the AddressingType is "DHCP".	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
DefaultGateway	string	W	The IP address of the current default gateway for this interface. The ability to modify this parameter is OPTIONAL, and this parameter cannot be modified if the AddressingType is “DHCP”.	-	1.0
DNSServers	string(256)	W	Comma-separated list of IP address of the DNS servers for this interface. The ability to modify this parameter is OPTIONAL, and this parameter cannot be modified if the AddressingType is “DHCP”. If this parameter is modifiable, the device MAY ignore any DNS servers beyond the first two in the list.	-	1.0
MACAddress	string	W	The physical address of this interface. Writable only if MACAddressOverride is present and equal to true.	-	1.0
MACAddressOverride	boolean	W	Whether the value of MACAddress parameter can be overridden. When true, MACAddress is writable. When false, MACAddress is not writable, and the default MAC address assigned by the device SHOULD be restored.	-	1.0
DHCPOptionNumberOfEntries	unsignedInt	-	Number of entries in the DHCP option table.	-	1.0
.LAN.DHCPOption.{i}.	object	W	This object is for configuration of DHCP options. Each instance of this object represents a DHCP option to be included by the DHCP client in client requests. The DHCP client MAY include any other options not specified in this table.	-	1.0
Request	boolean	W	Whether this entry represents a request to the DHCP server, or a value to be sent by the DHCP client. When true, this entry represents a request. In this case, the DHCP client MUST include the specified Tag in the Parameter Request List, as defined in RFC 2132. The Value parameter is ignored in this case. When false, this entry represents a value to be sent by the DHCP client. In this case, the DHCP client MUST include a DHCP option formed from the Tag and Value parameters (with the Length derived from the length of the Value parameter).	-	1.0
Tag	unsignedInt [1:254]	W	Tag of the DHCP option as defined in RFC 2132.	-	1.0
Value	base64	W	Base64 encoded octet string to be used as the Value of the DHCP option if Request is false.	<Empty>	1.0
.LAN.Stats.	object	-	This object contains statistics for the default IP interface.	-	1.0
ConnectionUpTime	unsignedInt	-	The time in seconds that this IP interface has been connected. If the IP interface is using DHCP, this is the time that the DHCP client has been only in the Bound or Renewing states and the lower-layer interface has continuously maintained a link. If the IP interface is using static addressing, this is the time that the lower-layer interface has continuously maintained a link.	-	1.0
TotalBytesSent	unsignedInt	-	Total number of IP payload bytes sent over this interface since the device was last restarted as specified in DeviceInfo.UpTime.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
TotalBytesReceived	unsignedInt	-	Total number of IP payload bytes received over this interface since the device was last restarted as specified in DeviceInfo.UpTime.	-	1.0
TotalPacketsSent	unsignedInt	-	Total number of IP packets sent over this interface since the device was last restarted as specified in DeviceInfo.UpTime.	-	1.0
TotalPacketsReceived	unsignedInt	-	Total number of IP packets received over this interface since the device was last restarted as specified in DeviceInfo.UpTime.	-	1.0
CurrentDayInterval	unsignedInt	-	Number of seconds since the beginning of the period used for collection of CurrentDay statistics. The device MAY align the beginning of each CurrentDay interval with days in the UTC time zone, but is not required to do so.	-	1.0
CurrentDayBytesSent	unsignedInt	-	Total number of IP payload bytes sent over this interface since the beginning of the current-day interval as specified by CurrentDayInterval.	-	1.0
CurrentDayBytesReceived	unsignedInt	-	Total number of IP payload bytes received over this interface since the beginning of the current-day interval as specified by CurrentDayInterval.	-	1.0
CurrentDayPacketsSent	unsignedInt	-	Total number of IP packets sent over this interface since the beginning of the current-day interval as specified by CurrentDayInterval.	-	1.0
CurrentDayPacketsReceived	unsignedInt	-	Total number of IP packets received over this interface since the beginning of the current-day interval as specified by CurrentDayInterval.	-	1.0
QuarterHourInterval	unsignedInt	-	Number of seconds since the beginning of the period used for collection of QuarterHour statistics. The device MAY align the beginning of each QuarterHour interval with real-time quarter-hour intervals, but is not required to do so.	-	1.0
QuarterHourBytesSent	unsignedInt	-	Total number of IP payload bytes sent over this interface since the beginning of the quarter-hour interval as specified by QuarterHourInterval.	-	1.0
QuarterHourBytesReceived	unsignedInt	-	Total number of IP payload bytes received over this interface since the beginning of the quarter-hour interval as specified by QuarterHourInterval.	-	1.0
QuarterHourPacketsSent	unsignedInt	-	Total number of IP packets sent over this interface since the beginning of the quarter-hour interval as specified by QuarterHourInterval.	-	1.0
QuarterHourPacketsReceived	unsignedInt	-	Total number of IP packets received over this interface since the beginning of the quarter-hour interval as specified by QuarterHourInterval.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
.LAN.IPPingDiagnostics.	object	-	This object defines access to an IP-layer ping test for the default IP interface.	-	1.0
DiagnosticsState	string	W	Indicates availability of diagnostic data. One of: "None" "Requested" "Complete" "Error_CannotResolveHostName" "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.	-	1.0
Host	string(256)	W	Host name or address of the host to ping.	-	1.0
NumberOfRepetitions	unsignedInt [1:]	W	Number of repetitions of the ping test to perform before reporting the results.	-	1.0
Timeout	unsignedInt [1:]	W	Timeout in milliseconds for the ping test.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
DataBlockSize	unsignedInt [1:65535]	W	Size of the data block in bytes to be sent for each ping.	-	1.0
DSCP	unsignedInt [0:63]	W	DiffServ codepoint to be used for the test packets. By default the CPE SHOULD set this value to zero.	-	1.0
SuccessCount	unsignedInt	-	Result parameter indicating the number of successful pings (those in which a successful response was received prior to the timeout) in the most recent ping test.	-	1.0
FailureCount	unsignedInt	-	Result parameter indicating the number of failed pings in the most recent ping test.	-	1.0
AverageResponseTime	unsignedInt	-	Result parameter indicating the average response time in milliseconds over all repetitions with successful responses of the most recent ping test. If there were no successful responses, this value MUST be zero.	-	1.0
MinimumResponseTime	unsignedInt	-	Result parameter indicating the minimum response time in milliseconds over all repetitions with successful responses of the most recent ping test. If there were no successful responses, this value MUST be zero.	-	1.0
MaximumResponseTime	unsignedInt	-	Result parameter indicating the maximum response time in milliseconds over all repetitions with successful responses of the most recent ping test. If there were no successful responses, this value MUST be zero.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
.LAN.TraceRouteDiagnostics.	object	-	This object is defines access to an IP-layer trace-route test for the default IP interface.	-	1.0
DiagnosticsState	string	W	Indicates availability of diagnostic data. One of: "None" "Requested" "Complete" "Error_CannotResolveHostName" "Error_MaxHopCountExceeded" "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.	-	1.0
Host	string(256)	W	Host name or address of the host to find a route to.	-	1.0
Timeout	unsignedInt [1:]	W	Timeout in milliseconds for the trace route test.	-	1.0
DataBlockSize	unsignedInt [1:65535]	W	Size of the data block in bytes to be sent for each trace route.	-	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
MaxHopCount	unsignedInt [1:64]	W	The maximum number of hop used in outgoing probe packets (max TTL). The default is 30 hops.	-	1.0
DSCP	unsignedInt [0:63]	W	DiffServ codepoint to be used for the test packets. By default the CPE SHOULD set this value to zero.	-	1.0
ResponseTime	unsignedInt	-	Result parameter indicating the response time in milliseconds the most recent trace route test. If a route could not be determined, this value MUST be zero.	-	1.0
NumberOfRouteHops	unsignedInt	-	Result parameter indicating the number of hops within the discovered route. If a route could not be determined, this value MUST be zero.	-	1.0
.LAN.TraceRouteDiagnostics.RouteHops.{i}.	object	-	Result parameter indicating the components of the discovered route. If a route could not be determined, there will be no instances of this object.	-	1.0
HopHost	string(256)	-	Result parameter indicating the Host Name or IP Address of a hop along the discovered route.	-	1.0

3.5 Inform Requirements

For CPE supporting the *Device* Root Object, the CPE MUST include in the ParameterList argument of the Inform message all of the parameters listed in Table 4 that are present in the data model implementation (any that are not present in the implementation need not be included in the Inform).

Table 4 – Forced Inform parameters

Parameter
Device.DeviceSummary
Device.DeviceInfo.HardwareVersion
Device.DeviceInfo.SoftwareVersion
Device.ManagementServer.ConnectionRequestURL
Device.ManagementServer.ParameterKey
Device.LAN.IPAddress

Note – the Forced Inform requirements do not apply to secondary instances of any of the above parameters that might be contained within Service Objects.

3.6 Notification Requirements

CPE MUST support Active Notification (see [2]) for all parameters defined in the Common Object definitions for the *Device* Root Object (section 3.4) with the exception of those parameters listed in Table 5. For only those parameters listed Table 5, the CPE MAY reject a request by an ACS to enable Active Notification via the SetParameterAttributes RPC by responding with fault code 9009 as defined in [2] (Notification request rejected).

CPE MUST support Passive Notification (see [2]) for all parameters defined in the Common Object definitions for the *Device* Root Object, with no exceptions.

Table 5 – Parameters for which Active Notification MAY be denied by the CPE

Parameter[6]
.DeviceInfo.
ModelName
Description
UpTime
FirstUseDate
DeviceLog
.ManagementServer.
ParameterKey
.Time.
CurrentLocalTime
.LAN.Stats.
ConnectionUpTime
TotalBytesSent
TotalBytesReceived
TotalPacketsSent
TotalPacketsReceived
CurrentDayInterval
CurrentDayBytesSent
CurrentDayBytesReceived
CurrentDayPacketsSent
CurrentDayPacketsReceived
QuarterHourInterval
QuarterHourBytesSent
QuarterHourBytesReceived
QuarterHourPacketsSent
QuarterHourPacketsReceived
.LAN.IPPingDiagnostics.
DiagnosticsState
SuccessCount
FailureCount
AverageResponseTime
MinimumResponseTime
MaximumResponseTime

[6] The name of a Parameter referenced in this table is the concatenation of the base path (see section 2.1), the object name shown in the yellow header, and the individual Parameter name.

Parameter[6]
.LAN.TraceRouteDiagnostics.
DiagnosticsState
ResponseTime
NumberOfRouteHops
.LAN.TraceRouteDiagnostics.RouteHops.{i}.
HopHost

3.7 DeviceSummary Definition

The DeviceSummary parameter is defined to provide an explicit summary of the top-level data model of the device, including version and profile information. This parameter MAY be used by an ACS to discover the nature of the device and the ACS's compatibility with specific objects supported by the device.

The DeviceSummary is defined as a list that includes the Root Object followed by all Service Object instances (or support for a Service Object type if no instances are currently present). For each of these objects, the DeviceSummary specifies the version of the object, the associated instance number used to identify the specific object instance, and a list of the supported profiles for that object.

The syntax of the DeviceSummary parameter is defined formally as follows:

```
DeviceSummary = RootObject [", " ServiceObject]*

RootObject = ObjectName ":" ObjectVersion "[](" ProfileList ")"

ServiceObject = ObjectName ":" ObjectVersion "[" [Instance] "](" ProfileList ")"

ObjectVersion = MajorVersion "." MinorVersion

ProfileList = [Profile [ ", " Profile]*]

Profile = ProfileName ":" ProfileVersion

MajorVersion = Integer

MinorVersion = Integer

ProfileVersion = Integer

Integer = DIGIT*

Instance = [ "+" ] NONZERODIGIT [DIGIT]*
```

For each object instance, the ObjectVersion element MUST indicate the major and minor versions of the object supported by the device.

The ObjectVersion for all objects defined prior to this specification for which explicit major and minor version numbers have not been defined is 1.0. Future updates to these objects will specify distinct version numbers.

The version for the "Device" object as defined in this specification is "1.0".

Instance is the instance number of the particular object instance. If the device supports an object type, but no instances are currently present, a single entry for this object MUST be listed in the DeviceSummary, and the instance number MUST be empty ("[]"). In this case, the device need not list support for specific profiles since the profile list might be dependent on the specific instance when it is instantiated.

If the instance number for an object might change (for example, if the instances represent physically separate devices, being managed by proxy, that can be connected or disconnected), the instance number MUST be prefixed with a "+" character. Lack of a "+" character indicates that the instance number is expected to remain unchanged.

For each object (Root Object and Service Objects), a device MUST list all profiles that it supports in the ProfileList element. That is, it MUST list all profiles for which the device's actual level of support is a superset. Each entry in the ProfileList MUST include the ProfileName and the ProfileVersion. The ProfileVersion is a single integer representing the minor version of the profile.

Vendor-specific objects and profiles MAY be included in this list, and if so MUST begin with X_<VENDOR>_, where <VENDOR> MUST be as defined in section 3.3.

3.7.1 DeviceSummary Examples

Below are some examples of the DeviceSummary parameter. (The first examples correspond directly to the examples given in section 2.1.2.)

Simple device supporting the ABCService Service Object:

"Device:1.0[](Baseline:1), ABCService:1.0[1](Baseline:1)"

Device supporting both ABCService and XYZService Service Objects:

"Device:1.0[](Baseline:1), ABCService:1.0[1](Baseline:1), XYZService:1.0[1](Baseline:1)"

Internet Gateway Device that also supports the ABCService and XYZService Service Objects:

"InternetGatewayDevice:1.0[](Baseline:1), ABCService:1.0[1](Baseline:1), XYZService:1.0[1](Baseline:1)"

Device supporting the ABCService Service Object and proxying for two devices supporting the functionality of the XYZService Service Object:

"Device:1.0[](Baseline:1), ABCService:2.17[1](Baseline:1), XYZService:1.2[1](Baseline:2), XYZService:1.2[2](Baseline:2, AnotherProfile:3)"

Internet Gateway Device also serving as a management proxy for three devices supporting the functionality of the ABCService Service Object:

"InternetGatewayDevice:1.0[](Baseline:1), ABCService:1.0[1](Baseline:1), ABCService:1.0[2](Baseline:1), ABCService:1.0[3](Baseline:1, AnotherProfile:1)"

Version 1.0 Internet Gateway Device with no additional service objects supported:

"InternetGatewayDevice:1.0[](Baseline:1)"

Device supporting the ability to proxy for devices supporting the functionality of the ABCService Service Object, but with no current instances of that object:

"Device:1.0[](Baseline:1), ABCService:2.17[]()"

Device supporting the ABCService Service Object with the baseline and a vendor-specific profile:

"Device:1.0[](Baseline:1), ABCService:2.17[1](Baseline:1, X_EXAMPLE-COM_MyProfile:2)"

Device supporting the ABCService Service Object, but with no profiles:

"Device:1.0[](Baseline:1), ABCService:2.17[1]()"

4 Profile Definitions

4.1 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	The object is REQUIRED to be present.
C	Creation and deletion of instances of the object via AddObject and DeleteObject is REQUIRED.
A	Creation of instances of the object via AddObject is REQUIRED, but deletion is not required.
D	Deletion of instances of the object via DeleteObject is REQUIRED, but creation is not required.

4.2 Baseline Profile

Table 6 defines the Baseline:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 6 – Baseline:1 Profile definition for Device:1

Name	Requirement
Device.	P
DeviceSummary	R
Device.DeviceInfo.	P
Manufacturer	R
ManufacturerOUI	R
ModelName	R
Description	R
SerialNumber	R
HardwareVersion	R
SoftwareVersion	R
DeviceStatus	R
UpTime	R
Device.ManagementServer.	P
URL	W
Username	W
Password	W
PeriodicInformEnable	W
PeriodicInformInterval	W
PeriodicInformTime	W
ParameterKey	R
ConnectionRequestURL	R
ConnectionRequestUsername	W
ConnectionRequestPassword	W
UpgradesManaged	W

4.3 GatewayInfo Profile

Table 7 defines the GatewayInfo:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 7 – GatewayInfo:1 Profile definition for Device:1

Name	Requirement
Device.GatewayInfo.	P
ManufacturerOUI	R
ProductClass	R
SerialNumber	R

4.4 Time Profile

Table 8 defines the Time:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 8 – Time:1 Profile definition for Device:1

Name	Requirement
Device.Time.	P
NTPServer1	W
NTPServer2	W
CurrentLocalTime	R
LocalTimeZone	W

4.5 LAN Profile

Table 9 defines the LAN:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 9 – LAN:1 Profile definition for Device:1

Name	Requirement
Device.LAN.	P
AddressingType	R
IPAddress	R
SubnetMask	R
DefaultGateway	R
DNSServers	R
MACAddress	R
Device.LAN.Stats.	P
ConnectionUpTime	R
TotalBytesSent	R
TotalBytesReceived	R
TotalPacketsSent	R
TotalPacketsReceived	R

4.6 IPPing Profile

Table 10 defines the IPPing:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 10 – IPPing:1 Profile definition for Device:1

Name	Requirement
Device.LAN.IPPingDiagnostics.	P
DiagnosticsState	W
Host	W
NumberOfRepetitions	W
Timeout	W
DataBlockSize	W
DSCP	W
SuccessCount	R
FailureCount	R
AverageResponseTime	R
MinimumResponseTime	R
MaximumResponseTime	R

4.7 TraceRoute Profile

Table 11 defines the TraceRoute:1 profile for the Device:1 object. The minimum required version for this profile is Device:1.0.

Table 11 – TraceRoute:1 Profile definition for Device:1

Name	Requirement
Device.LAN.TraceRouteDiagnostics.	P
DiagnosticsState	W
Host	W
Timeout	W
DataBlockSize	W
MaxHopCount	W
DSCP	W
ResponseTime	R
NumberOfRouteHops	R
Device.LAN.TraceRouteDiagnostics.RouteHops.{i}.	P
HopHost	R

4.8 UDPConnReq Profile

The UDPConnReq:1 profile for a Device implies support for all of the CPE requirements defined in Annex G of [2], including support for the data model parameters as shown in Table 12. The minimum required version for this profile is Device:1.1.

Table 12 – UDPConnReq :1 Profile definition for Device:1

Name	Requirement
Device.ManagementServer.	-
UDPConnectionRequestAddress	R
UDPConnectionRequestAddressNotificationLimit	W

Name	Requirement
STUNEnable	W
STUNServerAddress	W
STUNServerPort	W
STUNUsername	W
STUNPassword	W
STUNMaximumKeepAlivePeriod	W
STUNMinimumKeepAlivePeriod	W
NATDetected	R

Normative References

The following documents are referenced by this specification.

[1] RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt

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

[3] TR-098 Amendment 1, *Internet Gateway Device Data Model for TR-069*, DSL Forum Technical Report

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

[5] *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508

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

[7] RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, http://www.ietf.org/rfc/rfc3513.txt

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

[9] RFC 3489, *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*, http://www.ietf.org/rfc/rfc3489.txt