TECHNICAL REPORT

TR-106

Data Model Template for TR-069-Enabled Devices

Issue: 1 Amendment 2
Issue Date: November 2008

Notice

The Broadband Forum is a non-profit corporation organized to create guidelines for broadband network system development and deployment. This Broadband Forum Technical Report has been approved by members of the Forum. This Broadband Forum Technical Report is not binding on the Broadband Forum, any of its members, or any developer or service provider. This Broadband Forum Technical Report is subject to change, but only with approval of members of the Forum. This Technical Report is copyrighted by the Broadband Forum, and all rights are reserved. Portions of this Technical Report may be copyrighted by Broadband Forum members.

This Broadband Forum Technical Report is provided AS IS, WITH ALL FAULTS. ANY PERSON HOLDING A COPYRIGHT IN THIS BROADBAND FORUM TECHNICAL REPORT, OR ANY PORTION THEREOF, DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY:

(A) OF ACCURACY, COMPLETENESS, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE;
(B) THAT THE CONTENTS OF THIS BROADBAND FORUM TECHNICAL REPORT ARE SUITABLE FOR ANY PURPOSE, EVEN IF THAT PURPOSE IS KNOWN TO THE COPYRIGHT HOLDER;
(C) THAT THE IMPLEMENTATION OF THE CONTENTS OF THE DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

By using this Broadband Forum Technical Report, users acknowledge that implementation may require licenses to patents. The Broadband Forum encourages but does not require its members to identify such patents. For a list of declarations made by Broadband Forum member companies, please see http://www.broadband-forum.org. No assurance is given that licenses to patents necessary to implement this Technical Report will be available for license at all or on reasonable and non-discriminatory terms.

ANY PERSON HOLDING A COPYRIGHT IN THIS BROADBAND FORUM TECHNICAL REPORT, OR ANY PORTION THEREOF, DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW (A) ANY LIABILITY (INCLUDING DIRECT, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES UNDER ANY LEGAL THEORY) ARISING FROM OR RELATED TO THE USE OF OR RELIANCE UPON THIS TECHNICAL REPORT; AND (B) ANY OBLIGATION TO UPDATE OR CORRECT THIS TECHNICAL REPORT.

Broadband Forum Technical Reports may be copied, downloaded, stored on a server or otherwise re-distributed in their entirety only, and may not be modified without the advance written permission of the Broadband Forum.

The text of this notice must be included in all copies.

Issue History

Issue Number	Issue Date	Issue Editor	Changes
Issue 1	September 2005	Jeff Bernstein, 2Wire Christele Bouchat, Alcatel Tim Spets, Westell	Original
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
Issue 1 Amendment 2	November 2008	William Lupton, 2Wire Håkan Westin, Tilgin	Addition of data model definition XML Schema and normative XML common object and component definitions

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

Editors	William Lupton	2Wire	wlupton@2wire.com
	Håkan Westin	Tilgin	hakan.westin@tilgin.com
BroadbandHome™ Working Group Chairs	Greg Bathrick	PMC-Sierra	Greg_Bathrick@pmc-sierra.com
	Heather Kirksey	Motive	hkirksey@motive.com

Table of Contents

1 Introduction 8
1.1 Terminology 9
1.2 Document Conventions 10
2 Architecture 10
2.1 Data Hierarchy 10
2.1.1 Data Hierarchy Requirements 10
2.1.2 Data Hierarchy Examples 12
2.2 Object Versioning 14
2.2.1 Requirements for Compatible Versions 14
2.2.2 Version Notation 15
2.3 Profiles 15
2.3.1 Scope of Profiles 15
2.3.2 Multiple Profile Support 15
2.3.3 Profile Versions 16
2.3.4 Baseline Profiles 16
2.3.5 Types of Requirements in a Profile 16
2.4 DEPRECATED and OBSOLETED Items 17
2.4.1 Requirements for DEPRECATED Items 17
2.4.2 Requirements for OBSOLETED Items 18
3 Object Definitions 18
3.1 General Notation 18
3.2 Data Types 19
3.3 Vendor-Specific Parameters 21
3.4 Common Object Definitions 21
3.5 Inform Requirements 42
3.6 Notification Requirements 42
3.7 DeviceSummary Definition 44
3.7.1 DeviceSummary Examples 45
4 Profile Definitions 46
4.1 Notation 46
4.2 Baseline Profile 46
4.3 GatewayInfo Profile 47
4.4 Time Profile 47
4.5 LAN Profile 47
4.6 IPPing Profile 48
4.7 TraceRoute Profile 48
4.8 Download Profile 48
4.9 DownloadTCP Profile 49
4.10 Upload Profile 49
4.11 UploadTCP Profile 50
4.12 UDPEcho Profile 50
4.13 UDPEchoPlus Profile 50
4.14 UDPConnReq Profile 50
Normative References 52
Annex A. CWMP Data Model Definition XML Schema 53
A.1 Introduction 53
A.2 Normative Information 53
A.2.1 URI Conventions 55
A.2.2 Descriptions 56
A.2.3 Data Types 61
A.2.4 Bibliography 70
A.2.5 Components 72
A.2.6 Root and Service Objects 73
A.2.7 Parameters 74
A.2.8 Objects 75
A.2.9 Profiles 75
A.2.10 Modifications 76

A.3 DM Schema 77
Appendix I. "Device" Root Object, Common Objects and Components 99

List of Figures

Figure 1 – Positioning in the End-to-End Architecture 8

List of Tables

Table 1 – Data Types 19
Table 2 – Summary of Common Data Objects 21
Table 3 – Common Object definitions for Device:1 22
Table 4 – Forced Inform parameters 42
Table 5 – Parameters for which Active Notification MAY be denied by the CPE 42
Table 6 – Baseline:1 Profile definition for Device:1 46
Table 7 – GatewayInfo:1 Profile definition for Device:1 47
Table 8 – Time:1 Profile definition for Device:1 47
Table 9 – LAN:1 Profile definition for Device:1 47
Table 10 – IPPing:1 Profile definition for Device:1 48
Table 11 – TraceRoute:1 Profile definition for Device:1 48
Table 12 – Download:1 profile definition for Device:1 48
Table 13 – DownloadTCP:1 profile definition for Device:1 49
Table 14 – Upload:1 profile definition for Device:1 49
Table 15 – UploadTCP:1 profile definition for Device:1 50
Table 16 – UDPEcho:1 profile definition for Device:1 50
Table 17 – UDPEchoPlus:1 profile definition for Device:1 50
Table 18 – UDPConnReq:1 Profile definition for Device:1 50
Table 19 – XML Description Markup 57
Table 20 – XML Description Templates 58
Table 21 – XML Named Data Types 62
Table 22 – XML Data Type Facets 63
Table 23 – Path Name Scope Definition 64
Table 24 – PathRef Facet Definition 66
Table 25 – InstanceRef Facet Definition 67
Table 26 – EnumerationRef Facet Definition 67
Table 27 – XML Facet Inheritance Rules 69
Table 28 – XML Bibliographic References 71
Table 29 – XML Component Definition 72
Table 30 – XML Root and Service Objects 73
Table 31 – XML Parameter Definition 74
Table 32 – XML Parameter Syntax 74
Table 33 – XML Object Definition 75
Table 34 – XML Table Definition 75
Table 35 – XML Profile Definition 75
Table 36 – XML Parameter Modification 76
Table 37 – XML Object Modification 77
Table 38 - XML Profile Modification 77

Summary

This Technical Report specifies data model guidelines to be followed by all TR-069-capable devices. These guidelines include structural requirements for the data hierarchy, requirements for versioning of data models, requirements for defining profiles, a set of common data objects, and a baseline profile for any device supporting these common data objects. In addition, this Technical Report defines an XML Schema that as far as possible embodies these guidelines, and which is to be used for defining TR-069 data models.

1 Introduction

This Technical Report 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 CPE WAN Management Protocol (CWMP) 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

The ability to manage the home network remotely has a number of benefits including reducing the costs associated with activation and support of broadband services, improving time-to-market for new products and services, and improving the user experience.

If TR-069 defines the generic methods for any device, other documents (such as this one) specify the managed objects, or data models, which are collections of objects and parameters 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:

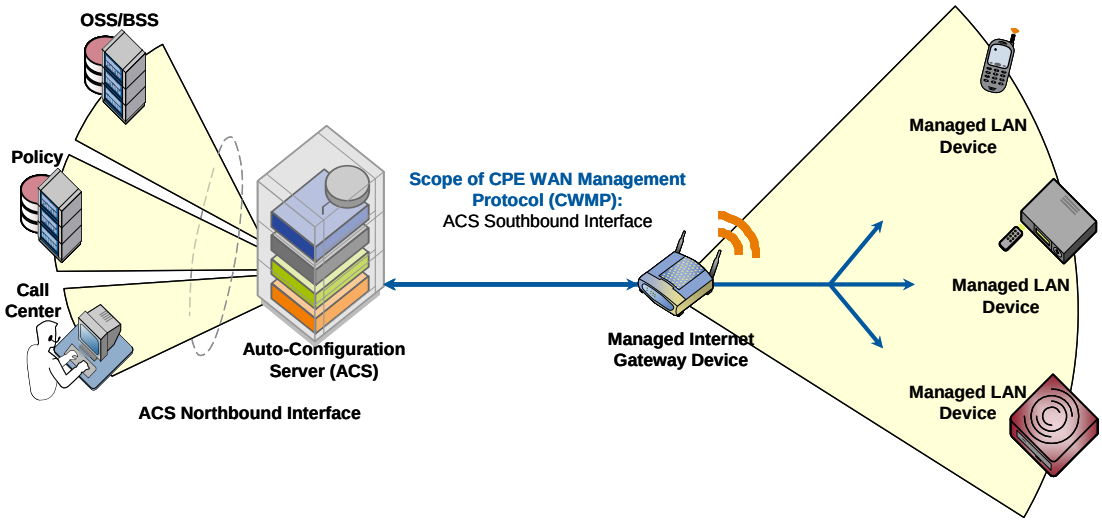

Figure 1 – Positioning in the End-to-End Architecture

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 Technical Report 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

In addition, this document defines an XML Schema that as far as possible embodies these guidelines, and which is to be used for defining TR-069 data models.

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.
BBF	Broadband Forum.
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.
Component	A named collection of Parameters and/or Objects that can be included anywhere within a data model. A Common Object can be thought of as a Component.
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.
DM Instance	Data Model Schema instance document. This is an XML document that conforms to the DM Schema and to any additional rules specified in or referenced by the DM Schema.
DM Schema	Data Model Schema. This is the XML Schema [16] that is used for defining data models for use with the CPE WAN Management Protocol.
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.
MediaWiki	A software application that is used by Wikipedia and other projects. http://en.wikipedia.org/wiki/MediaWiki.
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.
URI	Uniform Resource Identifier [8].

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]

[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.

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**
  - 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

ProfileName MUST start with a letter or underscore, and subsequent characters MUST be letters, digits, underscores or hyphens. The terms “letter” and “digit” are as defined in Appendix B of [10].

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, base64 or hexBinary 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, base64 or hexBinary 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, base64 or hexBinary 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. The name of each node in the hierarchy MUST start with a letter or underscore, and subsequent characters MUST be letters, digits, underscores or hyphens. The terms "letter" and "digit" are as defined in Appendix B of [10].

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 minimum and maximum allowed length can be listed using the form string(Min:Max), where Min and Max are the minimum and maximum string length in characters. If either Min or Max are missing, this indicates no limit, and if Min is missing the colon can also be omitted, as in string(Max). Multiple comma-separated ranges can be specified, in which case the string length MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768. 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. When transporting a string value within an XML document, any characters which are special to XML MUST be escaped as specified by the XML specification [10]. Additionally, any characters other than printable ASCII characters, i.e. any characters whose decimal ASCII representations are outside the (inclusive) ranges 9-10 and 32-126, SHOULD be escaped as specified by the XML specification.
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. Multiple comma-separated ranges can be specified, in which case the value MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768.
long	Long integer in the range –9223372036854775808 to 9223372036854775807, inclusive. For some long types listed, a value range is given using the form long[Min:Max], where the Min and Max values are inclusive. If either Min or Max are missing, this indicates no limit. Multiple comma-separated ranges can be specified, in which case the value MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768.
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. Multiple comma-separated ranges can be specified, in which case the value MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768.
unsignedLong	Unsigned long integer in the range 0 to 18446744073709551615, inclusive. For some unsignedLong types listed, a value range is given using the form unsignedLong[Min:Max], where the Min and Max values are inclusive. If either Min or Max are missing, this indicates no limit. Multiple comma-separated ranges can be specified, in which case the value MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768.
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*.

Type	Description
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 (no line-length limitation). A minimum and maximum allowed length can be listed using the form base64(Min:Max), where Min and Max are the minimum and maximum length in characters before Base64 encoding. If either Min or Max are missing, this indicates no limit, and if Min is missing the colon can also be omitted, as in base64(Max). Multiple comma-separated ranges can be specified, in which case the length MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768. Note that data models defined prior to the introduction of the DM Schema specified the length after Base64 encoding. If the length after encoding is n (which is always a multiple of 4), the length before encoding is m = (n/4)*3, m-1 or m-2.
hexBinary	Hex encoded binary. A minimum and maximum allowed length can be listed using the form hexBinary(Min:Max), where Min and Max are the minimum and maximum length in characters before Hex Binary encoding. If either Min or Max are missing, this indicates no limit, and if Min is missing the colon can also be omitted, as in hexBinary(Max). Multiple comma-separated ranges can be specified, in which case the length MUST be in one of the ranges. A "k" or "K" suffix is interpreted as a 1024 (not 1000) multiplier, e.g. 32k means 32768.

All IPv4 addresses and subnet masks are represented as strings in IPv4 dotted-decimal notation. All IPv6 addresses and subnet masks MUST be represented using any of the 3 standard textual representations as defined in RFC 3513 [7], sections 2.2.1, 2.2.2 and 2.2.3. Both lower-case and upper-case letters can be used. Use of the lower-case letters is RECOMMENDED. Examples of valid IPv6 address textual representations:

- 1080:0:0:800:ba98:3210:11aa:12dd
- 1080::800:ba98:3210:11aa:12dd
- 0:0:0:0:0:0:13.1.68.3

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. The term "packet" is to be interpreted as the transmission unit appropriate to the protocol layer in question, e.g. an IP packet or an Ethernet frame.

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
Capabilities	Root and Service Objects	Device capabilities.
DeviceInfo	Root and Service Objects	General information about the device, including its identity and version information.

Object Name	Allowed Location in Hierarchy	Description
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.
DownloadDiagnostics	Root and Service Objects	HTTP / FTP download test.
UploadDiagnostics	Root and Service Objects	HTTP / FTP upload test.
UDPEchoConfig	Root and Service Objects	UDP echo test.

Table 3 lists the Common Objects and their associated parameters defined for "Device", version 1.2. This definition is a superset of previously defined versions, 1.0 and 1.1.

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 1.2, then it will indicate support for version 1.2. 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
.Capabilities.	object	-	The capabilities of the device. This is a constant read-only object, meaning that only a firmware upgrade will cause these values to be altered.	-	1.2
.Capabilities.PerformanceDiagnostic.	object	-	The capabilities of the Performance Diagnostics (DownloadDiagnostics and UploadDiagnostics) for the device.	-	1.2
DownloadTransports	string	-	Comma-separated list of supported Download-Diagnostics transport protocols for a CPE device. Each item in the list is an enumeration of: "HTTP" "FTP" (OPTIONAL)	-	1.2

[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 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]
UploadTransports	string	-	Comma-separated list of supported Upload-Diagnostics transport protocols for a CPE device. Each item in the list is an enumeration of: "HTTP" "FTP" (OPTIONAL)	-	1.2
.DeviceInfo.	object	-	This object contains general device information.	-	1.0
Manufacturer	string(64)	-	The manufacturer of the CPE (human readable string).	-	1.0
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]. This value MUST remain fixed over the lifetime of the device, including across firmware updates.	-	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. This value MUST remain fixed over the lifetime of the device, including across firmware updates.	-	1.0
SerialNumber	string(64)	-	Serial number of the CPE. This value MUST remain fixed over the lifetime of the device, including across firmware updates.	-	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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
.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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
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 needs 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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
UpgradesManaged	boolean	W	Indicates whether or not the ACS will manage upgrades for the CPE. If True, the CPE SHOULD NOT use other means other than the ACS to seek out available upgrades. If False, the CPE MAY use other means for this purpose. Note that an autonomous upgrade (reported via an "10 AUTONOMOUS TRANSFER COMPLETE" Inform Event code) SHOULD be regarded as a managed upgrade if it is performed according to ACS-specified policy.	-	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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
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. An alternative to this parameter, e.g. when the configuration file is larger than the parameter size limit, is to use the Upload and Download RPCs with a FileType of "1 Vendor Configuration File".	-	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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
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.	-	1.0
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.	-	1.0
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

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
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
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(340)	W	Base64 encoded octet string to be used as the Value of the DHCP option if Request is False.	<Empty>	1.0

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
.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
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 does not need 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 does not need 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
.DownloadDiagnostics.	object	-	This object defines the diagnostics configuration for a HTTP and FTP DownloadDiagnostics Test. Files received in the DownloadDiagnostics do not require file storage on the CPE device.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
DiagnosticsState	string	W	Indicate the availability of diagnostic data. One of: "None" "Requested" "Completed" "Error_InitConnectionFailed" "Error_NoResponse " "Error_TransferFailed" "Error_PasswordRequestFailed" "Error_LoginFailed" "Error_NoTransferMode" "Error_NoPASV" "Error_IncorrectSize" "Error_Timeout" 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 Completed (if the test completed successfully), or one of the Error values listed above. If the value of this parameter is anything other than Completed, 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.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
Interface	string(256)	W	Specifies the IP-layer interface over which the test is to be performed. The content is the full hierarchical parameter name of the interface. The value of this parameter MUST be either a valid interface or an empty string. An attempt to set this parameter to a different value MUST be rejected as an invalid parameter value. If an empty string is specified, the CPE MUST use the default routing interface.	-	1.2
DownloadURL	string(256)	W	The URL, as defined in [8], for the CPE to perform the download on. This parameter MUST be in the form of a valid HTTP [13] or FTP [12] URL. When using FTP transport, FTP binary transfer MUST be used. When using HTTP transport, persistent connections MUST be used and pipelining MUST NOT be used. When using HTTP transport the HTTP Authentication MUST NOT be used.	-	1.2
DSCP	unsignedInt[0:63]	W	The DiffServ code point for marking packets transmitted in the test. The default value SHOULD be zero.	-	1.2
EthernetPriority	unsignedInt[0:7]	W	Ethernet priority code for marking packets transmitted in the test (if applicable). The default value SHOULD be zero.	-	1.2
ROMTime	dateTime	-	Request time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the client sends the GET command. For FTP this is the time at which the client sends the RTRV command.	-	1.2
BOMTime	dateTime	-	Begin of transmission time in UTC, which MUST be specified to microsecond precision For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the first data packet is received. For FTP this is the time at which the client receives the first data packet on the data connection.	-	1.2
EOMTime	dateTime	-	End of transmission in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the last data packet is received. For FTP this is the time at which the client receives the last packet on the data connection.	-	1.2
TestBytesReceived	unsignedInt	-	The test traffic received in bytes during the FTP/HTTP transaction including FTP/HTTP headers, between BOMTime and EOMTime,	-	1.2
TotalBytesReceived	unsignedInt	-	The total number of bytes received on the Interface between BOMTime and EOMTime.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
TCPOpenRequestTime	dateTime	-	Request time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the TCP socket open (SYN) was sent for the HTTP connection. For FTP this is the time at which the TCP socket open (SYN) was sent for the data connection. Note: Interval of 1 microsecond SHOULD be supported.	-	1.2
TCPOpenResponseTime	dateTime	-	Response time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the TCP ACK to the socket opening the HTTP connection was received. For FTP this is the time at which the TCP ACK to the socket opening the data connection was received. Note: Interval of 1 microsecond SHOULD be supported.	-	1.2
.UploadDiagnostics.	object	-	This object defines the diagnostics configuration for a HTTP or FTP UploadDiagnostics test. Files sent by the UploadDiagnostics do not require file storage on the CPE device, and MAY be an arbitrary stream of bytes.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
DiagnosticsState	string	W	Indicate the availability of diagnostic data. One of: "None" "Requested" "Completed" "Error_InitConnectionFailed" "Error_NoResponse" "Error_PasswordRequestFailed" "Error_LoginFailed" "Error_NoTransferMode" "Error_NoPASV" "Error_NoCWD" "Error_NoSTOR" "Error_NoTransferComplete" 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 Completed (if the test completed successfully), or one of the Error values listed above. If the value of this parameter is anything other than Completed, 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.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
Interface	string(256)	W	IP-layer interface over which the test is to be performed. The content is the full hierarchical parameter name of the interface. The value of this parameter MUST be either a valid interface or an empty string. An attempt to set this parameter to a different value MUST be rejected as an invalid parameter value. If an empty string is specified, the CPE MUST use the default routing interface.	-	1.2
UploadURL	string(256)	W	The URL, as defined in [8], for the CPE to Upload to. This parameter MUST be in the form of a valid HTTP [13] or FTP [12] URL. When using FTP transport, FTP binary transfer MUST be used. When using HTTP transport, persistent connections MUST be used and pipelining MUST NOT be used. When using HTTP transport the HTTP Authentication MUST NOT be used.	-	1.2
DSCP	unsignedInt[0:63]	W	DiffServ code point for marking packets transmitted in the test. The default value SHOULD be zero.	-	1.2
EthernetPriority	unsignedInt[0:7]	W	Ethernet priority code for marking packets transmitted in the test (if applicable). The default value SHOULD be zero.	-	1.2
TestFileLength	unsignedInt	W	The size of the file (in bytes) to be uploaded to the server. The CPE MUST insure the appropriate number of bytes are sent.	-	1.2
ROMTime	dateTime	-	Request time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the client sends the PUT command For FTP this is the time at which the STOR command is sent.	-	1.2
BOMTime	dateTime	-	Begin of transmission time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the first data packet is sent. For FTP this is the time at which the client receives the ready for transfer notification.	-	1.2
EOMTime	dateTime	-	End of transmission in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time when the HTTP successful response code is received. For FTP this is the time when the client receives a transfer complete.	-	1.2
TotalBytesSent	unsignedInt	-	The total number of bytes sent on the Interface between BOMTime and EOMTime.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
TCPOpenRequestTime	dateTime	-	Request time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the time at which the TCP socket open (SYN) was sent for the HTTP connection. For FTP this is the time at which the TCP socket open (SYN) was sent for the data connection Note: Interval of 1 microsecond SHOULD be supported.	-	1.2
TCPOpenResponseTime	dateTime	-	Response time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 For HTTP this is the Time at which the TCP ACK to the socket opening the HTTP connection was received. For FTP this is the Time at which the TCP ACK to the socket opening the Data connection was received. Note: Interval of 1 microsecond SHOULD be supported.	-	1.2
.UDPEchoConfig.	object	-	This object allows the CPE to be configured to perform the UDP Echo Service defined in [11] and UDP Echo Plus Service defined in Appendix A.1 of [15].	-	1.2
Enable	boolean	W	MUST be enabled to receive UDP echo. When enabled from a disabled state all related timestamps, statistics and UDP Echo Plus counters are cleared.	-	1.2
Interface	string(256)	W	IP-layer interface over which the CPE MUST listen and receive UDP echo requests on. The content is the full hierarchical parameter name of the interface. The value of this parameter MUST be either a valid interface or an empty string. An attempt to set this parameter to a different value MUST be rejected as an invalid parameter value. If an empty string is specified, the CPE MUST listen and receive UDP echo requests on all interfaces. Note: Interfaces behind a NAT MAY require port forwarding rules configured in the Gateway to enable receiving the UDP packets.	-	1.2
SourceIPAddress	string	W	The Source IP address of the UDP echo packet. The CPE MUST only respond to a UDP echo from this source IP address.	-	1.2
UDPPort	unsignedInt	W	The UDP port on which the UDP server MUST listen and respond to UDP echo requests.	-	1.2
EchoPlusEnabled	boolean	W	If True the CPE will perform necessary packet processing for UDP Echo Plus packets.	-	1.2
EchoPlusSupported	boolean	-	True if UDP Echo Plus is supported.	-	1.2
PacketsReceived	unsignedInt	-	Incremented upon each valid UDP echo packet received.	-	1.2
PacketsResponded	unsignedInt	-	Incremented for each UDP echo response sent.	-	1.2
BytesReceived	unsignedInt	-	The number of UDP received bytes including payload and UDP header after the UDPEchoConfig is enabled.	-	1.2
BytesResponded	unsignedInt	-	The number of UDP responded bytes, including payload and UDP header sent after the UDPEchoConfig is enabled.	-	1.2

Name[2]	Type	Write[3]	Description	Default[4]	Version[5]
TimeFirstPacketReceived	dateTime	-	Time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456, The time that the server receives the first UDP echo packet after the UDPEchoConfig is enabled.	-	1.2
TimeLastPacketReceived	dateTime	-	Time in UTC, which MUST be specified to microsecond precision. For example: 2008-04-09T15:01:05.123456 The time that the server receives the most recent UDP echo packet.	-	1.2

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

[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]
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
.LAN.TraceRouteDiagnostics.
DiagnosticsState
ResponseTime
NumberOfRouteHops
.LAN.TraceRouteDiagnostics.RouteHops.{i}.
HopHost
.DownloadDiagnostics.
DiagnosticsState
ROMTime
BOMTime
EOMTime
TestBytesReceived
TotalBytesReceived
TCPOpenRequestTime
TCPOpenResponseTime
.UploadDiagnostics.
DiagnosticsState

Parameter[6]
ROMTime
BOMTime
EOMTime
TotalBytesSent
TCPOpenRequestTime
TCPOpenResponseTime
.UDPEchoConfig.
PacketsReceived
PacketsResponded
BytesReceived
BytesResponded
TimeFirstPacketReceived
TimeLastPacketReceived

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 Download Profile

Table 12 defines the Download:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 12 – Download:1 profile definition for Device:1

Name	Requirement
Device.Capabilities.PerformanceDiagnostic.	P
DownloadTransports	R
Device.DownloadDiagnostics.	P
DiagnosticsState	W

Name	Requirement
Interface	W
DownloadURL	W
DSCP	W
EthernetPriority	W
ROMTime	R
BOMTime	R
EOMTime	R
TestBytesReceived	R
TotalBytesReceived	R

4.9 DownloadTCP Profile

Table 13 defines the DownloadTCP:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 13 – DownloadTCP:1 profile definition for Device:1

Name	Requirement
Device.DownloadDiagnostics.	P
TCPOpenRequestTime	R
TCPOpenResponseTime	R

4.10 Upload Profile

Table 14 defines the Upload:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 14 – Upload:1 profile definition for Device:1

Name	Requirement
Device.Capabilities.PerformanceDiagnostic.	P
UploadTransports	R
Device.UploadDiagnostics.	P
DiagnosticsState	W
Interface	W
UploadURL	W
DSCP	W
EthernetPriority	W
ROMTime	R
BOMTime	R
EOMTime	R
TestFileLength	R
TotalBytesSent	R

4.11 UploadTCP Profile

Table 15 defines the UploadTCP:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 15 – UploadTCP:1 profile definition for Device:1

Name	Requirement
Device.UploadDiagnostics.	P
TCPOpenRequestTime	R
TCPOpenResponseTime	R

4.12 UDPEcho Profile

Table 16 defines the UDPEcho:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 16 – UDPEcho:1 profile definition for Device:1

Name	Requirement
Device.UDPEchoConfig.	P
Enable	W
Interface	W
SourceIPAddress	W
UDPPort	W
PacketsReceived	R
PacketsResponded	R
BytesReceived	R
BytesResponded	R
TimeFirstPacketReceived	R
TimeLastPacketReceived	R
EchoPlusSupported	R

4.13 UDPEchoPlus Profile

Table 17 defines the UDPEchoPlus:1 profile for the Device:1 object. The minimum REQUIRED version for this profile is Device:1.2.

Table 17 – UDPEchoPlus:1 profile definition for Device:1

Name	Requirement
Device.UDPEchoConfig.	P
EchoPlusEnabled	W

4.14 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 18. The minimum REQUIRED version for this profile is Device:1.1.

Table 18 – 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

A list of the currently valid Broadband Forum Technical Reports is published at http://www.broadband-forum.org. 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 2, *CPE WAN Management Protocol*, Broadband Forum Technical Report

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

[10] *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, http://www.w3.org/TR/REC-xml

[11] RFC 862, *Echo Protocol*, http://www.ietf.org/rfc/rfc862.txt

[12] RFC 959, *File Transfer Protocol*, http://www.ietf.org/rfc/rfc959.txt

[13] RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt

[14] RFC 2648, *A URN Namespace for IETF Documents*, http://www.ietf.org/rfc/rfc2648.txt

[15] TR-143. *Enabling Network Throughput Performance Tests and Statistical Monitoring*, Broadband Forum Technical Report,

[16] *XML Schema Part 0: Primer Second Edition*, http://www.w3.org/TR/xmlschema-0

Annex A. CWMP Data Model Definition XML Schema

A.1 Introduction

The CWMP Data Model Definition XML Schema [16], or DM Schema, is used for defining TR-069 data models, and is specified in A.3.

DM Schema instance documents can contain any or all of the following:

- Data type definitions
- Root Object definitions (including profiles)
- Service Object definitions (including profiles)
- Component definitions
- Vendor extension definitions

A.2 Normative Information

It is possible to create instance documents that conform to the DM Schema but nevertheless are not valid data model definitions. This is because it is not possible to specify all the normative data model definition requirements using the XML Schema language. Therefore, the schema contains additional requirements written using the usual normative language. Instance documents that conform to the DM Schema and meet these additional requirements are referred to as DM Instances.

For example, the definition of the parameter element includes the following additional requirements on the name and base attributes:


```
  <xs:complexType name="ModelParameter">
    <xs:annotation>
      <xs:documentation>Parameter definition and reference.</xs:documentation>
    </xs:annotation>
    …
    <xs:attribute name="name" type="tns:ParameterName">
      <xs:annotation>
        <xs:documentation>MUST be unique within the parent object (this is checked by schema
validation).
MUST be present if and only if defining a new parameter.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="base" type="tns:ParameterName">
      <xs:annotation>
        <xs:documentation>MUST be present if and only if modifying an existing
parameter.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    …
  </xs:complexType>
```

In some cases, a requirement that is in fact implied by the DM Schema is emphasized within the schema via the xs:documentation element (the uniqueness requirement on the name is an example of this).

In other cases, a schema-implied requirement is not highlighted. For example, the name and base attributes are of type tns:ParameterName:

```
<!DOCTYPE cwmp-datamodel [
  …
  <!ENTITY name "([\i-[:]][\c-[:\.]]*)">
  …
]>
  …
  <xs:simpleType name="ParameterName">
    <xs:annotation>
      <xs:documentation>Parameter name (maximum length 256); the same as xs:NCName except that
periods are not permitted.  This name MUST in addition follow the vendor-specific parameter name
requirements of section 3.3.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:maxLength value="256"/>
      <xs:pattern value="&name;"/>
    </xs:restriction>
  </xs:simpleType>
```

This states that the parameter name is a string that follows the following rules:

- It is derived from xs:token, which has a whitespace facet of "collapse", meaning that any leading whitespace in the name will be ignored.
- It has a maximum length of 256 characters.
- Its first character matches the pattern "[\i-[:]]", which means "any character permitted as the first character of an XML name, except for a colon", and any subsequent characters match the pattern "[\c-[:\.]]", which means "any character permitted in an XML name, except for a colon and a dot".
- It follows the vendor-specific parameter name requirements of section 3.3.

The question of the location of the definitive normative information therefore arises. The answer is as follows:

- All the normative information in the main part of the document remains normative.
- The DM Schema, and the additional requirements therein, are normative. Some of these additional requirements are duplicated (for emphasis) in this Annex.
- The DM Schema references additional material in this Annex. Such material is normative.
- If the DM Schema conflicts with a normative requirement in the main part of the document, this is an error in the DM Schema, and the requirement in the main part of the document takes precedence.

A.2.1 URI Conventions

The top-level spec attribute contains the URI of the associated specification document, e.g. the BBF Technical Report.

This URI SHOULD uniquely identify the specification. More than one DM Schema instance document MAY reference the same specification.

The following rules apply to the value of the top-level spec attribute:

- For a BBF Technical Report, it MUST be of the form "urn:broadband-forum-org:tr-nnn-i-a-c", where nnn is the specification number (including leading zeros), i is the issue number, a is the amendment number, and c is the corrigendum number. The issue, amendment and corrigendum numbers do not include leading zeros. For example, "urn:broadband-forum-org:tr-106-1-0" refers to TR-106 (Issue 1 Amendment 0), and "urn:broadband-forum-org:tr-106-1-2" refers to TR-106 (Issue 1) Amendment 2. If the corrigendum number (including the preceding hyphen) is omitted, the most recent corrigendum is assumed.
- For specifications issued by other standards organizations, or by vendors, it SHOULD be of a standard form if one is defined. For example, RFC 2648 [14] specifies a syntax for referencing RFCs.
- Note that processing tools are likely to assume that all files that share a spec value are related to each other. Therefore, use of meaningful spec values is RECOMMENDED.

Formally, the value of the spec attribute is defined as follows:


```
SpecURI = BBFURI
        | OtherURI

BBFURI = "urn:broadband-forum-org:" BBFDoc

BBFDoc = "tr-" BBFNumber BBFIssue BBFAmendment BBFCorrigendum

BBFNumber = [DIGIT]{3,}     // including leading zeros, e.g. 069

BBFIssue = "-" NoLeadingZeroPositiveNumber

BBFAmendment = "-" NoLeadingZeroNumber

BBFCorrigendum = "-" NoLeadingZeroPositiveNumber
               | ""         // if omitted, most recent corrigendum is assumed

NoLeadingZeroNumber = [DIGIT]
                    | [NONZERODIGIT] [DIGIT]*

NoLeadingZeroPositiveNumber = [NONZERODIGIT] [DIGIT]*

OtherURI = <of a standard form if one is defined>
```

Standard BBF DM Instances can be accessed at the following URL:

```
BBFURL = "http://www.broadband-forum.org/cwmp/" BBFDoc BBFSubDoc ".xml"

BBFDoc = <as before>

BBFSubDoc = "-" LABEL      // distinguishing label (not beginning with a digit)
          | ""             // not needed if only one DM Instance is associated with spec
```

For example, the DM Instance associated with TR-106 Amendment 2 can be accessed at http://www.broadband-forum.org/cwmp/tr-106-1-2.xml. If two DM Instances had been associated with TR-106 Amendment 2, they might perhaps have been accessible at http://www.broadband-forum.org/cwmp/tr-106-1-2-types.xml and http://www.broadband-forum.org/cwmp/tr-106-1-2-objects.xml.

A.2.2 Descriptions

Many elements have descriptions, and the same rules apply to all description elements in the DM Schema. A description is free text which can contain a limited amount of MediaWiki-like markup as specified in A.2.2.3.

A.2.2.1 Character Set

For BBF standards, the character set MUST be restricted to printable characters in the Basic Latin Unicode block, i.e. to characters whose decimal ASCII representations are in the (inclusive) ranges 9-10 and 32-126.

A.2.2.2 Pre-processing

All DM Instance processing tools MUST conceptually perform the following pre-processing before interpreting the markup:

1) Remove any leading whitespace up to and including the first line break[7].

2) Remove the longest common whitespace prefix (i.e. that occurs at the start of every line) from each line. See the example below, where three lines start with four spaces and one line starts with five spaces, so the longest whitespace prefix that occurs at start of each line is four spaces. In this

[7] It can be assumed that all line breaks are represented by a single line feed, i.e. ASCII 10. See [10] section 2.11.

calculation, a tab character counts as a single character. To avoid confusion, the description SHOULD NOT contain tab characters.

3) Remove all trailing whitespace, including line breaks.

This pre-processing is designed to permit a reasonable variety of layout styles while still retaining predictable behavior. For example, both the following:

```
<description>This is the first line.
This is the second line.
 This is the indented third line.
This is the fourth line.</description>
```

```
<description>
    This is the first line.
    This is the second line.
     This is the indented third line.
    This is the fourth line.
</description>
```

…result in the following:

```
This is the first line.
This is the second line.
 This is the indented third line.
This is the fourth line.
```

A.2.2.3 Markup

The pre-processed description can contain the following markup, which is inspired by, but is not identical to, MediaWiki markup. All DM Instance processing tools SHOULD support this markup to the best of their ability.

Table 19 – XML Description Markup

Name	Markup Example	Description
Italics	''italic text''	Two apostrophes on each side of some text will result in the contained text being emphasized in italics.
Bold	'''bold text'''	Three apostrophes on each side of some text will result in the contained text being emphasized in bold.
Bold italics	'''''b+i text'''''	Five apostrophes on each side of some text will result in the contained text being emphasized in bold italics.
Paragraph	This paragraph just ended.	A line break is interpreted as a paragraph break.
Bulleted lists	* level one ** level two * level one again ** level two again *** level three *: level one continued outside of list	A line starting with one or more asterisks (*) denotes a bulleted list entry, whose indent depth is proportional to the number of asterisks specified. If the asterisks are followed by a colon (:), the previous item at that level is continued, as shown. An empty line, or a line that starts with a character other than an asterisk, indicates the end of the list.
Numbered lists	# level one ## level two # level one again ## level two again ### level three #: level one continued outside of list	A line starting with one or more number signs (#) denotes a numbered list entry. All other conventions defined for bulleted lists apply here (using # rather than *), except that numbered list entries are prefixed with an integer decoration rather than a bullet.

Name	Markup Example	Description

Indented lists	`: level one`
	`:: level two`
	`: level one again`
	`:: level two again`
	`::: level three`
	`outside of list`
		A line starting with one or more colons (:) denotes an indented list entry.
		All other conventions defined for bulleted lists apply here (using : rather than *), except that indented list entries have no prefix decoration, and item continuation is not needed.

Verbatim	`code example:`
	` if (something) {`
	`  /* do something */`
	` } else {`
	`  /* do other */`
	` }`
		A block of lines each of which starts with a space is to be formatted exactly as typed, preferably in a fixed width font.
		This allows code fragments, simple tables etc. to be included in descriptions.
		Note that the pre-processing rules of A.2.2.2 imply that it is not possible to process an entire description as verbatim text (because all the leading whitespace would be removed). This is not expected to be a problem in practice.

Hyperlinks	`http://www.broadband-forum.org`	URL links are specified as plain old text (no special markup).

Templates	`{{bibref|1|section 2}}`
	`{{section|table}}`
	`{{param|Enable}}`
	`{{enum|Error}}`
		Text enclosed in double curly braces ({}) is a template reference, which is replaced by template-dependent text.
		A.2.2.4 specifies the standard templates.

A.2.2.4 Templates

A template invocation is encoded as two curly braces on either side of the template name and arguments. Arguments can follow the template name, separated by vertical pipe (|) characters. All whitespace is significant. For example:

```
{{someTemplate|arg1|arg2|…|argN}}
```

The following standard templates are defined. Any vendor-specific template names MUST obey the rules of section 3.3.

Table 20 – XML Description Templates

Name	Markup Definition	Description

Bibliographic reference	`{{bibref|id}}`
	`{{bibref|id|section}}`
		A bibliographic reference.
		The id argument MUST match the id attribute of one of the current file's (or an imported file's) top-level bibliography element's reference elements (A.2.4).
		The OPTIONAL section argument specifies the section number, including any leading "section", "annex" or "appendix" text.
		Typically, the processing tool will (a) validate the id, and (b) replace the template reference with something like "[id] section".
		Markup examples:
		`{{bibref|1}}`
		`{{bibref|1|section 3}}`

Name	Markup Definition	Description
Section separator	{{section|category}} {{section}}	The beginning or end of a section or category. This is a way of splitting the description into sections. If the category argument is present, this marks the end of the previous section (if any), and the beginning of a section of the specified category. The "table", "row" and "examples" categories are reserved for the obvious purposes. If the category argument is absent, this marks the end of the previous section (if any). Typically, the processing tool will (a) validate the category, and (b) replace the template reference with a section marker. Markup examples: {{section|table}} {{section|row}} {{section|examples}}
Parameter and object reference	{{param|ref}} {{param}} {{object|ref}} {{object}}	A reference to the specified parameter or object. The OPTIONAL ref argument references a parameter or object. Parameter and object names SHOULD adhere to the rules of A.2.3.4 with object scope. Typically, the processing tool will (a) validate the reference, and (b) replace the template reference with the ref argument or, if it is omitted, the current parameter or object name, possibly rendered in a distinctive font. Markup examples: {{param|Enable}} {{object|Stats.}}
Enumeration reference	{{enum|value}} {{enum|value|param}} {{enum}}	A reference to the specified enumeration value. The OPTIONAL value argument specifies one of the enumeration values for the referenced parameter. If present, it MUST be a valid enumeration value for that parameter. The OPTIONAL param argument identifies the referenced parameter. If present, it SHOULD adhere to the rules of A.2.3.4 with object scope. If omitted, the current parameter is assumed. If the arguments are omitted, this is a hint to the processing tool to replace the template reference with a list of the parameter's enumerations, possibly preceded by text such as "Enumeration of:". This overrides the processing tool's expected default behavior of listing the parameter's enumerations after the description. Otherwise, typically the processing tool will (a) validate that the enumeration value is valid, and (b) replace the template reference with the value and/or param arguments, appropriately formatted and with the value possibly rendered in a distinctive font. Markup examples: {{enum|None}} {{enum|None|OtherParam}}
Pattern reference	{{pattern}}	This is a hint to the processing tool to replace the template reference with a list of the parameter's patterns, possibly preceded by text such as "Possible patterns:". This overrides the processing tool's expected default behavior of listing the parameter's patterns after the description.
Units reference	{{units}}	The parameter's units string. Typically, the processing tool will (a) check that the parameter has a units string, and (b) substitute the value of its units string.
Boolean values	{{false}} {{true}}	Boolean values. Typically, the processing tool will substitute the value False or True, possibly rendered in a distinctive font.

Name	Markup Definition	Description
Miscellaneous	{{empty}}	Typically, the processing tool will render such values in a distinctive font, possibly using standard wording, such as <Empty> or “an empty string”.

A.2.2.5 HTML Example

This includes examples of most of the markup and templates.

```
  <model name="Goo:1.1" base="Goo:1.0">
    <object name="GooTop." access="readOnly" minEntries="1" maxEntries="1">
      <parameter name="ExampleParam" access="readOnly">
      <description>
        {{section|Introduction}}This is an ''example'' parameter that illustrates many of the
'''formatting''' templates.  For '''''example''''', this references {{bibref|TR-106a1|section
3.2}}.
        {{section|Usage}}This parameter is called {{object}}{{param}}.  One can also reference
other parameters in the same object, such as {{param|OtherParameter}}, and indicate that the
parameter value is measured in {{units}}.
        One can also include bulleted lists:
        * level one
        ** level two
        * level one again
        ** level two again
        *** level three
        *: level one continued
        and numbered lists:
        # level one
        ## level two
        # level one again
        ## level two again
        ### level three
        #: level one continued
        and indented lists
        : level one
        :: level two
        : level one again
        :: level two again
        ::: level three
        and hyperlinks such as http://www.google.com
        and code examples:
          if (something) {
            /* do something */
          } else {
            /* do other */
          }
        If the parameter was Boolean, one could refer to its values {{false}} and {{true}}.
        One can refer to its enumerations individually, e.g. {{enum|Disabled}}, or to other
parameters' enumerations, such as {{enum|Value|OtherParam}}, or can list them all.  {{enum}}
        Finally, if there were any patterns they could be listed too.  {{pattern}}
      </description>
      <syntax>
        <string>
          <enumeration value="A"/>
          <enumeration value="B"/>
          <units value="packets"/>
        </string>
      </syntax>
    </parameter>
```

The resulting HTML would look something like this:

This is an *example* parameter that illustrates many of the **formatting** templates. For ***example***, this references [TR-106a1] section 3.2.

This parameter is called *ParentObject.ExampleParam*. One can also reference other parameters in the same object, such as *OtherParameter*, and indicate that the parameter value is measured in packets.

One can also include bulleted lists:

- level one
  - level two
- level one again
  - level two again
    - level three

  level one continued

and numbered lists:

1. level one
   1. level two
2. level one again
   1. level two again
      1. level three

   level one continued

and indented lists

    level one
        level two
    level one again
        level two again
            level three

and hyperlinks such as http://www.google.com

and code examples:

```
if (something) {
  /* do something */
} else {
  /* do other */
}
```

If the parameter was Boolean, one could refer to its values *false* and *true*.

One can refer to its enumerations individually, e.g. *A*, or to other parameters' enumerations, such as *Value*, or can list them all. Possible values:

- *Disabled*
- *Enabled*
- *Error* (OPTIONAL)

Finally, if there were any patterns they could be listed too.

A.2.3 Data Types

TR-069 data models support only the Table 1 primitive data types "on the wire". However, the DM Schema allows data types to be derived from the primitive types or from other named data types. Such derived types can be named or anonymous.

A.2.3.1 Named Data Types

Named data types are defined using the top-level dataType element. A DM Instance can contain zero or more top-level dataType elements.

When defining a new named data type, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 21 – XML Named Data Types

Name	Description
name	The data type name.
base	The base type name, i.e. name of the data type from which this data type is derived. This is used only where the base type is itself a named data type, not a primitive type.
status	The data type's {current, deprecated, obsoleted, deleted} status. This defaults to current, and so is not likely to be specified for a new data type.
description	The data type's description (A.2.2).
size pathRef instanceRef range enumeration enumerationRef pattern units	Data type facets (A.2.3.3). These are permitted only when the base type is a named data type, i.e. when the base attribute is specified.
base64 boolean dateTime hexBinary int long string unsignedInt unsignedLong	Primitive data type definition. These are permitted only when the base type is primitive. There is an element for each primitive data type, and each element supports only the facets (A.2.3.3) that are appropriate to that data type.

For example:

```
<dataType name="String255">
  <string>
    <size maxLength="255"/>
  </string>
</dataType>

<dataType name="String127" base="String255">
  <size maxLength="127"/>
</dataType>
```

A.2.3.2 Anonymous Data Types

Anonymous data types are defined within parameter syntax elements (A.2.7.1), and can apply only to the parameters within which they are defined. For example:

```
<parameter name="Example1" access="readOnly">
  <syntax>
    <string>
      <size maxLength="127"/>
    </string>
  </syntax>
</parameter>

<parameter name="Example2" access="readOnly">
  <syntax>
    <dataType base="String255">
      <size maxLength="127"/>
    </dataType>
  </syntax>
</parameter>
```

If an anonymous data type is modified in a later version of a data model, the modified anonymous data type

is regarded as being derived from the original anonymous data type. Therefore the base type restriction rules of A.2.3.8 MUST be obeyed.

A.2.3.3 Data Type Facets

A facet specifies some aspect of a data type, e.g. its size, range or units.

Note that XML Schema [16] also associates facets with data types. The XML Schema and DM Schema concepts are the same, but the set of facets is not identical.

The DM Schema defines the following facets (normative requirements are specified in the schema):

Table 22 – XML Data Type Facets

Name	Description
size	Size ranges for the data type (applies to string, base64, hexBinary and their derived types). Note that the size facet always refers to the actual value, not to the base64- or hexBinary-encoded value. Prior to the definition of the DM Schema, the maximum sizes of base64 parameters referred to the base64-encoded values. Processing tools that generate reports from DM Instances SHOULD include explicit clarification of whether the size ranges refer to the actual or encoded values.
pathRef	Details of how to reference parameters and objects via their path names (applies to string and its derived types: A.2.3.7).
instanceRef	Details of how to reference object instances (table rows) via their instance numbers (applies to int, unsignedInt and their derived types; A.2.3.7).
range	Value ranges for the data type (applies to numeric data types and their derived types).
enumeration	Enumerations for the data type (applies to string and its derived types).
enumerationRef	Enumerations for the data type, obtained at run-time from the value of a specified parameter (applies to string and its derived types; A.2.3.7).
pattern	Patterns for the data type (applies to string and its derived types).
units	Units for the data type (applies to numeric data types and their derived types).

It is important to note that the enumeration facet does not necessarily define all the valid values for a data type. This is for the following reasons:

- As specified in section 3.3, vendors are allowed to add additional enumeration values.
- A future version of a data model may need to add additional enumerations values.

A.2.3.4 Reference Path Names

Some description templates (A.2.2.4), and all reference facets (A.2.3.7), need to specify parameter or object names. It is always possible to specify a full path name, but it is frequently necessary or convenient to specify a relative path name. For example, it might be necessary to reference another parameter in the current object. Any instance numbers in the parameter's full path name cannot be known at data model definition time, so this can only be done using a relative path name.

The following rules apply to all path names that are used in data model definitions for referencing parameters or objects:

- Path names MAY contain "{i}" placeholders, which MUST be interpreted as wild cards matching all instance numbers, e.g. "InternetGatewayDevice.WANDevice.{i}." references all WANDevice instances.
- Path names MUST NOT contain instance numbers.

A path name is always associated with a path name scope, which defines the point in the naming hierarchy relative to which the path name applies.

Table 23 – Path Name Scope Definition

Name	Description
normal	This is a hybrid scope which usually gives the desired behavior: • If the path begins with a "Device" or "InternetGatewayDevice" component, it is relative to the top of the naming hierarchy. • If the path begins with a dot, it is relative to the Root or Service Object (c.f. scope=model). • Otherwise, the path is relative to the current object (c.f. scope=object).
model	The path is relative to the Root or Service Object.
object	The path is relative to the current object.

Formally, if the path name scope is normal:

- If the path is empty, it MUST be regarded as referring to the top of the naming hierarchy.
- Otherwise, if the path begins with a "Device" or "InternetGatewayDevice" component, it MUST be regarded as a full path name (these are the only two possible Root Device names).
- Otherwise, if the path begins with a dot, it MUST be regarded as a path relative to the Root or Service Object. For example, in the Device Root Object ".DeviceInfo." means "Device.DeviceInfo.", and in the Device.Services.ABCService.1 Service Object it means "Device.Services.ABCService.1.-DeviceInfo.".
- Otherwise, it MUST be regarded as a path relative to the current object. For example, if the current object is "Device.LAN.", "IPAddress" means "Device.LAN.IPAddress" and "Stats." means "Device.LAN.Stats.". Within a parameter definition, the current object is the parameter's parent object, so within the "Device.LAN.IPAddress" definition, "SubnetMask" means "Device.LAN.Subnet-Mask".

If the path name scope is model:

- If the path is empty, it MUST be regarded as referring to the Root or Service Object.
- Otherwise, it MUST be regarded as a path relative to the Root or Service Object. Any leading dot MUST be ignored.

If the path name scope is object:

- If the path is empty, it MUST be regarded as referring to the current object.
- Otherwise, it MUST be regarded as a path relative to the current object. Any leading dot MUST be ignored.

Note that the term "Root or Service Object", which is used several times above, means "if within a Service Object instance, the Service Object instance; otherwise, the Root Object".

For example, the pathRef and instanceRef facets (A.2.3.7) have a targetParent attribute which specifies the possible parent(s) of the referenced parameter or object, and a targetParentScope attribute (defaulted to normal) which specifies targetParent's scope. If the current object is within a Service Object instance, setting targetParentScope to model forces the referenced parameter or object to be in the same Service Object instance. Similarly, setting targetParentScope to object forces the referenced parameter or object to be in the same object or in a sub-object.

String parameters whose values are path names are subject to the rules of section 3.2, so object names do not include a trailing dot. The parameter value (or each list item if the parameter is list-valued) always behaves as described above for normal path name scope, regardless of the path name scope in the data model definition. For example, in the Device Root Object, a parameter value of ".DeviceInfo"always means "Device.DeviceInfo".

In order to be able to use reference parameters as unique keys (A.2.8.1), path names in parameter values MUST conceptually be converted to full path names before being compared. For example, in the Device

Root Object, ".DeviceInfo." and "Device.DeviceInfo." would compare as equal. If a reference parameter is list-valued, i.e. it is a list of path names or instance numbers, the parameter value MUST conceptually be regarded as a set when being compared, i.e. the comparison has to ignore the item order and any repeated items. For example, "1,2,1" and "2,1" would compare as equal because both reference instances 1 and 2.

A.2.3.5 Null References

A null reference indicates that a reference parameter is not currently referencing anything. The value that indicates a null reference depends on the reference parameter's base data type:

- **string**: a null reference MUST be indicated by an empty string.
- **unsignedInt**: a null reference MUST be indicated by the value 0.
- **int**: a null reference MUST be indicated by the value -1.

A.2.3.6 Reference Types

A reference to another parameter or object can be weak or strong:

- **weak**: it doesn't necessarily reference an existing parameter or object. For example, if the referenced parameter or object is deleted, the value of the reference parameter might not get updated.
- **strong**: it always either references a valid parameter or object, or else is a null reference (A.2.3.5). If the referenced parameter or object is deleted, the value of the reference parameter is always set to a null reference.

The following requirements relate to reference types and the associated CPE behavior.

- All read-only reference parameters MUST be declared as strong references.
- A CPE MUST reject an attempt to set a strong reference parameter if the new value does not reference an existing parameter or object.
- A CPE MUST NOT reject an attempt to set a weak reference parameter because the new value does not reference an existing parameter or object.
- A CPE MUST change the value of a non-list-valued strong reference parameter to a null reference when a referenced parameter or object is deleted.
- A CPE MUST remove the corresponding list item from a list-valued strong reference parameter when a referenced parameter or object is deleted.
- A CPE MUST NOT change the value of a weak reference parameter when a referenced parameter or object is deleted.

A.2.3.7 Reference Facets

A reference facet specifies how a parameter can reference another parameter or object. There are three sorts of reference:

- **Path reference**: references another parameter or object via its path name. Details are specified via the pathRef facet, which applies to string and its derived types.
- **Instance reference**: references an object instance (table row) via its instance number. Details are specified via the instanceRef facet, which applies to int, unsignedInt and their derived types.
- **Enumeration reference**: references a list-valued parameter via its path name. The current value of the referenced parameter indicates the valid enumerations for this parameter. Details are specified via the enumerationRef facet, which applies to string and its derived types.

When defining a path reference, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 24 – PathRef Facet Definition

Name	Description
targetParent	An XML list of path names that can restrict the set of parameters and objects that can be referenced. If the list is empty (the default), then anything can be referenced. Otherwise, only the immediate children of one of the specified objects can be referenced, A "{i}" placeholder in a path name acts as a wild card, e.g. "InternetGatewayDevice.-WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.". Path names cannot contain explicit instance numbers.
targetParentScope	Specifies the point in the naming hierarchy relative to which targetParent applies (A.2.3.4): normal (default), model or object.
targetType	Specifies what types of item can be referenced: • **any**: any parameter or object can be referenced (default) • **parameter**: any parameter can be referenced • **object**: any object can be referenced • **single**: any single-instance object can be referenced • **table**: any multi-instance object (table) can be referenced • **row**: any multi-instance object (table) instance (row) can be referenced
targetDataType	Specifies the valid data types for the referenced parameter. Is relevant only when targetType is any or parameter. Possible values are as follows: • **any**: a parameter of any data type can be referenced (default) • **base64**: only a base64 parameter can be referenced • **boolean**: only a boolean parameter can be referenced • **dateTime**: only a dateTime parameter can be referenced • **hexBinary**: only a hexBinary parameter can be referenced • **integer**: only an integer (int, long, unsignedInt or unsignedLong) parameter can be referenced • **int**: only an int parameter can be referenced • **long**: only a long (or int) parameter can be referenced • **string**: only a string parameter can be referenced • **unsignedInt**: only an unsignedInt parameter can be referenced • **unsignedLong**: only an unsignedLong (or unsignedInt) parameter can be referenced • **<named data type>**: only a parameter of the named data type can be referenced In addition, a parameter whose data type is derived from the specified data type can be referenced. The built-in type hierarchy (a simplified version of the XML Schema type hierarchy) is as follows: *any* base64 boolean dateTime hexBinary *integer* long int unsignedLong unsignedInt string Note that *any* and *integer* are not valid parameter data types. They are included in order to support "can reference any data type" and "can reference any numeric data type".
refType	Specifies the reference type (A.2.3.6): weak or strong.

When defining an instance reference, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 25 – InstanceRef Facet Definition

Name	Description
targetParent	Specifies the path name of the multi-instance object (table) of which an instance (row) is being referenced. "{i}" placeholders and explicit instance numbers are not permitted in the path name. targetParentScope can be used to specify path names relative to the Root or Service Object or the current object.
targetParentScope	Specifies the point in the naming hierarchy relative to which targetParent applies (A.2.3.4): normal (default), model or object.
refType	Specifies the reference type (A.2.3.6): weak or strong.

When defining an enumeration reference, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 26 – EnumerationRef Facet Definition

Name	Description
targetParam	Specifies the path name of the list-valued parameter whose current value indicates the valid enumerations for this parameter.
targetParamScope	Specifies the point in the naming hierarchy relative to which targetParam applies (A.2.3.4): normal (default), model or object.
nullValue	Specifies the parameter value that indicates that none of the values of the referenced parameter currently apply (if not specified, no such value is designated). Note that if this parameter is list-valued then nullValue is not relevant, because this condition will be indicated by an empty list.

The following examples illustrate the various possible types of reference.

```
    <object name="PeriodicStatistics.SampleSet.{i}.Parameter.{i}." …>
      …
      <parameter name="Reference" access="readWrite">
        <description>Reference to the parameter that is associated with this object instance.
This MUST be the parameter's full path name.</description>
        <syntax>
          <string>
            <size maxLength="256"/>
            <pathRef targetType="parameter" refType="weak"/>
          </string>
          <default type="object" value=""/>
        </syntax>
      </parameter>
```

```
    <object name="StorageService.{i}.StorageArray.{i}." …>
      …
      <parameter name="PhysicalMediumReference" access="readWrite">
        <description>A comma-separated list of Physical Medium references.  Each Physical Medium
referenced by this parameter MUST exist within the same StorageService instance. A Physical
Medium MUST only be referenced by one Storage Array instance. Each reference can be either in
the form of ".PhysicalMedium.{i}" or a fully qualified object name…</description>
        <syntax>
          <list>
            <size maxLength="1024"/>
          </list>
          <string>
            <pathRef targetParent=".PhysicalMedium." targetParentScope="model" targetType="row"
refType="strong"/>
          </string>
        </syntax>
      </parameter>
```

```
    <object name="InternetGatewayDevice.QueueManagement.Classification.{i}." access="readWrite"
minEntries="0" maxEntries="unbounded" entriesParameter="ClassificationNumberOfEntries">
```



```
      <description>Classification table.</description>
      <parameter name="ClassQueue" access="readWrite">
        <description>Classification result.  Instance number…</description>
        <syntax>
          <int>
            <instanceRef targetParent=".QueueManagement.Queue." refType="strong"/>
          </int>
        </syntax>
      </parameter>
```

```
    <object name="STBService.{i}.Components.FrontEnd.{i}.IP.Inbound.{i}." …>
      …
      <parameter name="StreamingControlProtocol" access="readOnly">
        <description>Network protocol currently used for controlling streaming of the source
content, or an empty string if the content is not being streamed or is being streamed but is not
being controlled.
If non-empty, the string MUST be one of the .Capabilities.FrontEnd.IP.StreamingControlProtocols
values.</description>
        <syntax>
          <string>
            <enumerationRef targetParam=".Capabilities.FrontEnd.IP.StreamingControlProtocols"
nullValue=""/>
          </string>
        </syntax>
      </parameter>

      <parameter name="StreamingTransportProtocol" access="readOnly">
        <description>Network protocol currently used for streaming the source content, or an
empty string if the content is not being streamed.
If non-empty, the string MUST be one of the
.Capabilities.FrontEnd.IP.StreamingTransportProtocols values.</description>
        <syntax>
          <string>
            <enumerationRef targetParam=".Capabilities.FrontEnd.IP.StreamingTransportProtocols"
nullValue=""/>
            <string/>
          </string>
        </syntax>
      </parameter>
```

```
    <object name="InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.WPS." …>
      …
      <parameter name="ConfigMethodsEnabled" access="readWrite">
        <description>Comma-separated list of the WPS configuration methods enabled on the
device.  Each entry in the list MUST be a member of the list reported by the
ConfigMethodsSupported parameter…</description>
        <syntax>
          <list/>
          <string>
            <enumerationRef targetParam="ConfigMethodsSupported"/>
          </string>
        </syntax>
      </parameter>
```

A.2.3.8 Base Type Restriction

A new data type MUST always be a restriction of its base type, meaning that a valid value of the new data type will always be a valid value for its base type. This is the case for the examples of A.2.3.1, which involve three different data types:

- string of unlimited length
- string of maximum length 255
- string of maximum length 127

Clearly a string of length 100 is valid for all three data types, but a string of length 200 is only valid for the first two data types.

The examples of A.2.3.1 considered only the size facet, but in general all facets that are applicable to the data type have to be considered. The base type restriction requirements for each facet are as follows:

Table 27 – XML Facet Inheritance Rules

Facet	Requirements
size	The derived data type can define sizes in any way, provided that the new sizes do not permit any values that are not valid for the base type.
pathRef	The derived data type can modify the data type in the following ways: • By "promoting" status to a "higher" value, where the lowest to highest ordering is: current, deprecated, obsoleted, deleted. For example, current can be changed to deprecated, and obsoleted can be changed to deleted, but deleted can't be changed back to obsoleted. When promoting status, the deprecation, obsoletion and deletion rules of section 2.4 MUST be obeyed. • By changing targetParent to narrow the set of possible parent objects. • By changing targetType to narrow the set of possible target types. • By changing targetDataType to narrow the set of possible target data types.
instanceRef	The derived data type can modify the data type in the following ways: • By "promoting" status to a "higher" value, as described for pathRef. • By changing targetParent to narrow the set of possible parent objects.
range	The derived data type can define ranges in any way, provided that the new ranges do not permit any values that are not valid for the base type.
enumeration	The derived data type can modify existing enumeration values in the following ways: • By "promoting" access from readOnly to readWrite. • By "promoting" status to a "higher" value, as described for pathRef. • By "promoting" optional from False to True. • By adding a code, if none was previously specified. • By using the action attribute to extend or replace the description (see below). The derived data type can add new enumeration values.
enumerationRef	The derived data type can modify the data type in the following ways: • By "promoting" status to a "higher" value, as described for pathRef.
pattern	The derived data type can modify existing pattern values by changing access, status, optional and description exactly as for enumerations. The derived data type can add new patterns and/or replace existing patterns with new patterns, provided that the new patterns do not permit any values that are not valid for the base type. For example a single pattern "[AB]" could be replaced with "A" and "B", but "C" could not be added.
units	The derived data type can add units if the base type didn't specify any.

Most of the above requirements are non-normative, because it has to be possible to correct errors. For example, if the base type supports a range of [-1:4095] but the values 0 and 4095 were included in error, it would be permissible for a derived type to support ranges of [-1:-1] and [1:4094]. Processing tools SHOULD be able to detect and warn about such cases.

When defining a new data type, if a facet is omitted, the new data type will inherit that facet from its base type. If a facet is present, it MUST be fully specified (except that special rules apply to descriptions; see below). For example, this means that a derived type that adds additional enumeration values has also to re-declare the enumeration values of the base type.

For example, in the following, the derived type inherits the units facet from its parent but it does not inherit the range facet, so the PacketCounter range is [10:] and the PacketCounter2 range is [15:20].

```
<dataType name="PacketCounter">
  <unsignedLong>
    <range minInclusive="10"/>
    <units value="packets"/>
  </unsignedLong>
</dataType>

<dataType name="PacketCounter2" base="PacketCounter">
  <range minInclusive="15" maxInclusive="20"/>
</dataType>
```

Similarly, in the following, the enumeration values for ABCD are not A, B, C and D, but are just C and D. This is an error (because the derived type cannot remove enumeration values), and processing tools SHOULD detect and warn about such cases.

```
<dataType name="AB">
  <string>
    <enumeration value="A"/>
    <enumeration value="B"/>
  </string>
</dataType>

<dataType name="ABCD" base="AB">
  <string>
    <enumeration value="C"/>
    <enumeration value="D"/>
  </string>
</dataType>
```

A derived data type and any of its facets that support descriptions will inherit those descriptions from the base type. Facet descriptions are inherited regardless of whether the facet is present in the derived type. For any descriptions that are explicitly specified in the derived type, the action attribute controls whether they will be extended or replaced.

For example, in the following, the description of Y (which is not changed) does not have to be repeated.

```
<dataType name="XY">
  <description>This is XY.</description>
  <string>
    <enumeration value="X">
      <description>This is X.</description>
    </enumeration>
    <enumeration value="Y">
      <description>This is Y.</description>
    </enumeration>
  </string>
</dataType>

<dataType name="XY2" base="XY">
  <description action="replace">This is all about XY.</description>
  <enumeration value="X">
    <description action="append">This is more about X.</description>
  </enumeration>
  <enumeration value="Y"/>
</dataType>
```

A.2.4 Bibliography

The bibliography is defined using the top-level bibliography element, which can contain zero or more (bibliographic) reference elements.

When defining a new bibliographic reference, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 28 – XML Bibliographic References

Name	Description
id	The bibliographic reference ID.
name	The name by which the referenced document is usually known.
title	The document title.
organization	The organization that published the referenced document, e.g. BBF, IEEE, IETF.
category	The document category, e.g. TR (BBF), RFC (IETF).
date	The publication date.
hyperlink	Hyperlink(s) to the document.

The bibliographic reference ID is intended to uniquely identify this reference across all instance documents. Therefore, for instance documents that are published by the BBF, IDs MUST obey the following rules:

- For a BBF Technical Report, it MUST be of the form "TR-nnnixaycz", where TR is the literal "TR", nnn is the Technical Report number (including leading zeros), i, a and c are literal letters, and x, y, and z are the issue, amendment and corrigendum numbers respectively. The issue number (ix) is omitted if it is issue 1 and the amendment number (ay) is omitted if it is amendment 0. For example, "TR-106a2" refers to TR-106 (Issue 1) Amendment 2. If the corrigendum number (cz) is omitted, the most recent corrigendum is assumed.
- For an IETF RFC, it MUST be of the form "RFCnnn", where RFC is the literal "RFC" and nnn is the RFC number (no leading zeros).
- For an IEEE specification, it SHOULD be of the form "nnn.ml-dddd", where nnn.m is the IEEE group, l is the spec letter(s), and dddd is the publication year. For example, "802.1D-2004".
- For an ETSI specification (which includes DVB specifications), it SHOULD be of the form "TTnnnnnnva.b.c" where TT is the specification type, usually "TS" (Technical Specification), nnnnnn is the specification number, and a.b.c is the version number.
- For specifications issued by other standards organizations, or by vendors, it SHOULD be of a standard form if one is defined.

Processing tools SHOULD be lenient when comparing bibliographic reference IDs. Specifically, they SHOULD ignore all whitespace, punctuation, leading zeros in numbers, and upper / lower case. So, for example, "rfc 1234" and "RFC1234" would be regarded as the same ID, as would "TR-069" and "TR69".

Processing tools SHOULD detect and report inconsistent bibliographic references, e.g. a reference with the same ID (i.e. an ID that compares as equal) as one that was encountered in a different file, but with a different name or hyperlink.

Formally, bibliographic reference IDs in instance documents that are published by the BBF and the other organizations mentioned above are defined as follows:


```
ReferenceID = BBFID
            | RFCID
            | IEEEID
            | ETSIID
            | OtherID

BBFID = "TR-" BBFNumber BBFIssue BBFAmendment BBFCorrigendum

BBFNumber = [DIGIT]{3,}         // including leading zeros, e.g. 069

BBFIssue = "i" <number greater than one>
         | ""                   // empty means Issue 1

BBFAmendment = "a" <number greater than zero>
             | ""               // empty means Amendment 0

BBFCorrigendum = "c" <number greater than zero>
               | ""             // empty means the most recent Corrigendum

RFCID = "RFC" RFCNumber

RFCNumber = NONZERODIGIT [DIGIT]*
                                // no leading zeros, e.g. 123

IEEEID = IEEEGroup IEEESpec IEEEDate
       | <for other IEEE specifications, of a standard form if one is defined>

IEEEGroup = <group number> "." <group sub-number>
                                // e.g. 802.1

IEEESpec = <spec letter(s)>     // e.g. D

IEEEDate = "-" <publication year>
                                // e.g. -2004
         | ""                   // can be empty

ETSIID = ETSISpecType ETSINumber ETSIVersion
       | <for other ETSI specifications, of a standard form if one is defined>

ETSISpecType = "TR"             // Technical Report
             | "TS"             // Technical Specification
             | "ES"             // ETSI Specification
             | "EN"             // European Standard

ETSINumber = [DIGIT]{6}         // e.g. 102034

ETSIVersion = "v" <version number as specified by ETSI>
            | ""                // can be empty

OtherURI = <of a standard form if one is defined>
```

A.2.5 Components

A component is a way of defining a named set of parameters, objects and/or profiles to be used wherever such a group is needed in more than one place (or just to structure the definitions). A DM Instance can contain zero or more top-level component elements.

When defining a new component, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 29 – XML Component Definition

Name	Description
name	The component name.
description	The component's description (A.2.2).
component	The other components that are referenced (included) by this component.

Name	Description
parameter	The component's top-level parameter definitions (A.2.7).
object	The component's object definitions (A.2.8).
profile	The component's profile definitions (A.2.9).

Referencing (including) a component can be thought of as textual substitution. A component has no version number and isn't tied to a particular Root or Service Object.

The following is a simple example of component definition and reference.

```
<component name="ByteStats">
  <parameter name="BytesSent" access="readOnly">
    <description>Number of bytes sent.</description>
    <syntax><unsignedInt/></syntax>
  </parameter>
  <parameter name="BytesReceived" access="readOnly">
    <description>Number of bytes received.</description>
    <syntax><unsignedInt/></syntax>
  </parameter>
</component>

<model name="InternetGatewayDevice:1.4">
  <object name="InternetGatewayDevice." access="readOnly" minEntries="1" maxEntries="1">
    …
    <component ref="ByteStats"/>
    …
  </object>
  …
</model>
```

Here the component is referenced from within an object definition. Components can be referenced from within component, model and object definitions. Parameter, object and profile definitions within components are relative to the point of inclusion unless overridden using the path attribute.

A.2.6 Root and Service Objects

Root and Service Objects are defined using the model element and an associated top-level object element. A DM Instance can contain zero or more top-level model elements.

When defining a new model, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 30 – XML Root and Service Objects

Name	Description
name	The model name, including its major and minor version numbers (3.7).
base	The name of the previous version of the model (for use when the model version is greater than 1.0).
isService	Whether it's a Service Object. This defaults to False and so can be omitted for Root Objects.
description	The model's description (A.2.2).
component	The components that are referenced (included) by the model.
parameter	The model's top-level parameter definitions (A.2.7).
object	The model's top-level and other object definitions (A.2.8).
profile	The model's profile definitions (A.2.9).

Once a given version has been defined, it cannot be modified; instead, a new version of the object has to be defined. For example, the following example defines v1.0 and v1.1 of a notional Service Object.


```
<model name="DemoService:1.0" isService="true">
  <parameter name="DemoServiceNumberOfEntries" access="readOnly"/>
  <object name="DemoService.{i}." access="readOnly" minEntries="0" maxEntries="unbounded"
                                                  entriesParameter="DemoServiceNumberOfEntries"/>
</model>

<model name="DemoService:1.1" base="DemoService:1.0" isService="true">
  <object base="DemoService.{i}." access="readOnly" minEntries="0" maxEntries="unbounded"/>
</model>
```

A.2.7 Parameters

Parameters are defined using the parameter element, which can occur within component, model and object elements. When defining a new parameter, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 31 – XML Parameter Definition

Name	Description
name	The parameter name (3.1).
access	Whether the parameter can be writable (readWrite) or not (readOnly).
status	The parameter's {current, deprecated, obsoleted, deleted} status. This defaults to current, and so is not likely to be specified for a new parameter.
activeNotify	The parameter's {normal, forceEnabled, ForceDefault, canDeny} Active Notification status. This defaults to normal, and so is not often specified for a new parameter.
forcedInform	The parameter's Forced Inform status. This defaults to False, and so is not often specified for a new parameter.
description	The parameter's description (A.2.2).
syntax	The parameter's syntax (A.2.7.1).

A.2.7.1 Parameter Syntax

Parameter syntax is defined using the syntax element, which can occur only within parameter elements. When defining a new parameter, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 32 – XML Parameter Syntax

Name	Description
hidden	Whether the value is hidden on readback. This defaults to False, and so is not often specified for a new parameter.
list	If the parameter is list-valued, details of the list value (3.2). This allows specification of the maximum and minimum number of items in the list, and also supports a size facet for the list (A.2.3.3). Note that a list-valued parameter is always a string as far as TR-069 is concerned. For a list, the rest of the syntax specification refers to the individual list items, not to the parameter value.
base64 boolean dateTime hexBinary int long string unsignedInt unsignedLong	If the parameter is of a primitive data type, specifies a primitive data type reference, e.g. <int/>. If the parameter data type is derived from a primitive data type, specifies an anonymous primitive data type definition (A.2.3.2), e.g. <int><range maxInclusive="255"/></int>.
dataType	If the parameter is of a named data type, specifies a named data type (A.2.3.1) reference, e.g. <dataType ref="IPAddress"/>. If the parameter data type is derived from a named data type, specifies an anonymous named data type (A.2.3.2) definition, e.g. <dataType base="IPAddress"><size maxLength="15"/></dataType>.

A.2.8 Objects

Objects are defined using the object element, which can occur within component and model elements. When defining a new object, the following attributes and elements are relevant (normative requirements are specified in the schema).

Table 33 – XML Object Definition

Name	Description
name	The object name, specified as a partial path (3.1).
access	Whether object instances can be Added or Deleted (readWrite) or not (readOnly). Adding or deleting instances is meaningful only for a multi-instance (table) object.
minEntries	The minimum number of instances of this object (always less than or equal to maxEntries).
maxEntries	The maximum number of instances of this object (can be "unbounded"). minEntries and maxEntries allow the object to be placed into one of three categories: • **minEntries=0, maxEntries=1:** single-instance object which might not be allowed to exist, e.g. because only one of it and another object can exist at the same time. • **minEntries=1, maxEntries=1:** single-instance object that is always allowed to exist. • **All other cases:** multi-instance (table) object (A.2.8.1).
status	The object's {current, deprecated, obsoleted, deleted} status. This defaults to current, and so is not likely to be specified for a new object.
description	The object's description (A.2.2).
component	The components that are referenced (included) by the object.
parameter	The object's parameter definitions (A.2.7).

A.2.8.1 Tables

If an object is a table, several other attributes and elements are relevant (normative requirements are specified in the schema).

Table 34 – XML Table Definition

Name	Description
name	For a table, the last part of the name has to be "{i}" (3.1).
entriesParameter	The name of the parameter (in the parent object) that contains the number of entries in the table. Such a parameter is needed whenever there is a variable number of entries, i.e. whenever maxEntries is unbounded or is greater than minEntries.
enableParameter	The name of the parameter (in each table entry) that enables and disables that table entry. Such a parameter is needed whenever access is readWrite (so the ACS might be able to create entries) and the uniqueKey element is present.
uniqueKey	An element that specifies a unique key by referencing those parameters that constitute the unique key. For a table in which there is an enableParameter, the uniqueness requirement applies only to enabled table entries.

A.2.9 Profiles

Profiles are defined using the profile element, which can occur within component and model elements. When defining a new profile, the following attributes and elements are always relevant (normative requirements are specified in the schema).

Table 35 – XML Profile Definition

Name	Description
name	The profile name, including its version number (2.3.3).
base	The name of the previous version of the profile (for use when the profile version is greater than 1).
extends	A list of the names of the profiles that this profile extends.

Name	Description
description	The profile's description (A.2.2).
parameter	The profile's parameter requirements, which can include descriptions, references to the parameters in question, and the parameter access requirement.
object	The profile's object requirements, which can include descriptions, references to the objects in question, the object access requirements, and requirements for the object's parameters.

A.2.10 Modifications

New data types, components, models and profiles can be created based on existing items. This doesn't modify the existing item.

Parameters, objects and profiles can be modified "in place", i.e. without creating a new item. This still uses the parameter, object and profile elements, and is indicated by using the base, rather than the name, attribute. The base attribute specifies the name of the existing item that is to be modified.

The syntax for modifying an item is the same as for creating an item, but there are rules. These rules are not specified in the DM Schema.

A.2.10.1 Parameter Modifications

The following rules govern parameter modifications.

Table 36 – XML Parameter Modification

Name	Description
access	Can be "promoted" from readOnly to readWrite.
status	Can be "promoted" to a "higher" value, where the lowest to highest ordering is: current, deprecated, obsoleted, deleted. For example, current can be changed to deprecated, and obsoleted can be changed to deleted, but deleted can't be changed back to obsoleted. When promoting status, the deprecation, obsoletion and deletion rules of section 2.4 MUST be obeyed.
activeNotify	Can be changed from forceEnabled to forceDefaultEnabled. No other changes are permitted.
forcedInform	Cannot be changed.
description	Can be extended or replaced via use of the action attribute. When changing the description, behavioral backwards compatibility MUST be preserved.
syntax/hidden	Cannot be changed.
syntax/list	Can add or modify the list element in the following ways: • Can convert a non-list string parameter to a list provided that an empty string was already a valid value with the appropriate meaning. • Can adjust limits on numbers of items, and on the list size, provided that the new rules do not permit any values that were not valid for the previous version of the parameter.
syntax/int etc syntax/dataType	Can make any change that follows the base type restriction rules of A.2.3.8, e.g. can add enumerations.
syntax/default	A default can be added if the parameter didn't already have one.

Most of the above requirements are non-normative, because it has to be possible to correct errors in a previous version of a parameter. Processing tools SHOULD be able to detect and warn when a parameter is modified in a way that contravenes the above rules.

A.2.10.2 Object Modifications

The following rules govern object modifications.

Table 37 – XML Object Modification

Name	Description
access	Can be "promoted" from readOnly to readWrite.
minEntries	Cannot be changed.
maxEntries	Cannot be changed.
entriesParameter	Cannot be changed, unless was previously missing, in which case can be added.
enableParameter	Cannot be changed, unless was previously missing, in which case can be added.
status	Can be "promoted" to a "higher" value, where the lowest to highest ordering is: current, deprecated, obsoleted, deleted. For example, current can be changed to deprecated, and obsoleted can be changed to deleted, but deleted can't be changed back to obsoleted. When promoting status, the deprecation, obsoletion and deletion rules of section 2.4 MUST be obeyed.
description	Can be extended or replaced via use of the action attribute. When changing the description, behavioral backwards compatibility MUST be preserved.
uniqueKey	Cannot be changed.
component	Can reference (include) new components.
parameter	Can add new parameters.

Most of the above requirements are non-normative, because it has to be possible to correct errors in a previous version of an object. Processing tools SHOULD be able to detect and warn when an object is modified in a way that contravenes the above rules.

A.2.10.3 Profile Modifications

The following rules govern profile modifications. They apply to the profile element, and to its nested parameter and object elements.

Table 38 - XML Profile Modification

Name	Description
status	Can be "promoted" to a "higher" value, where the lowest to highest ordering is: current, deprecated, obsoleted, deleted. For example, current can be changed to deprecated, and obsoleted can be changed to deleted, but deleted can't be changed back to obsoleted. When promoting status, the deprecation, obsoletion and deletion rules of section 2.4 MUST be obeyed.
description	Can be extended or replaced via use of the action attribute. When changing the description, behavioral backwards compatibility MUST be preserved.

Most of the above requirements are non-normative, because it has to be possible to correct errors in a profile. Indeed, since profiles are immutable, the only valid reason for changing a profile is to correct errors. Processing tools SHOULD be able to detect and warn when a profile is modified in a way that contravenes the above rules.

A.3 DM Schema

The DM Schema is specified below. The normative version can be found at http://www.broadband-forum.org/cwmp/cwmp-datamodel-1-0.xsd. Any conflict MUST be resolved in favor of the normative version.


```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  TR-069 Data Model Definition Schema (DM Schema) v1.0

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

  This document is provided "as is," with all faults.  Any person holding a
  copyright in this document, or any portion thereof, disclaims to the fullest
  extent permitted by law any representation or warranty, express or implied,
  including, but not limited to,
  (a) any warranty of merchantability, fitness for a particular purpose,
      non-infringement, or title;
  (b) any warranty that the contents of the document are suitable for any
      purpose, even if that purpose is known to the copyright holder;
  (c) any warranty that the implementation of the contents of the documentation
      will not infringe any third party patents, copyrights, trademarks or other
      rights.

  This publication may incorporate intellectual property. The Broadband Forum
  encourages but does not require declaration of such intellectual property.
  For a list of declarations made by Broadband Forum member companies, please see
  http://www.broadband-forum.org.

  Copyright The Broadband Forum. All Rights Reserved.

  Broadband Forum XML Schemas may be copied, downloaded, stored on a server or
  otherwise re-distributed in their entirety only. The text of this
  notice must be included in all copies.

  Summary:
  TR-069 Data Model Definition Schema (DM Schema).  DM Instances define TR-069
  data models.  Within the schema, elements are grouped by category (simple
  types, complex types etc), and are in alphabetical order within each category.

  Version History:
  November 2008: cwmp-datamodel-1-0.xsd, corresponds to TR-106 Amendment 2
-->
<!DOCTYPE cwmp-datamodel [
  <!ENTITY colon ":">
  <!ENTITY dot "\.">
  <!ENTITY inst "(\{i\})">
  <!ENTITY name "([\i-[:]][\c-[:\.]]*)">
  <!ENTITY Name "([\i-[a-z:]][\c-[:\.]]*)">
  <!ENTITY num "(\d+)">
]>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="urn:broadband-forum-
                      org:cwmp:datamodel-1-0" targetNamespace="urn:broadband-forum-
                      org:cwmp:datamodel-1-0" elementFormDefault="unqualified"
                      attributeFormDefault="unqualified">
  <!-- Simple types -->
  <xs:simpleType name="ActiveNotify">
    <xs:annotation>
      <xs:documentation>Parameter active notify support.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="normal"/>
      <xs:enumeration value="forceEnabled"/>
      <xs:enumeration value="forceDefaultEnabled"/>
      <xs:enumeration value="canDeny"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AnyTypeName">
    <xs:annotation>
      <xs:documentation>Built-in or derived type name.</xs:documentation>
    </xs:annotation>
```



```
    <xs:union memberTypes="tns:BuiltinTypeName tns:DataTypeName"/>
  </xs:simpleType>
  <xs:simpleType name="BibrefId">
    <xs:annotation>
      <xs:documentation>Bibliographic reference ID; SHOULD uniquely identify this
                        reference across all instance documents.
For BBF DM Instances, the bibliographic reference ID rules specified in A.2.4 MUST be
                        used.  For example, to reference TR-106 Issue 1 Amendment 2, the
                        value of this attribute would be TR-106a2.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token"/>
  </xs:simpleType>
  <xs:simpleType name="BuiltinTypeName">
    <xs:annotation>
      <xs:documentation>Built-in type name.
The type hierarchy is as for XML Schema, with "any" and "base64" mapping to the
                        "anySimpleType" and "base64Binary" XML Schema types
                        respectively.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="any"/>
      <xs:enumeration value="base64"/>
      <xs:enumeration value="boolean"/>
      <xs:enumeration value="dateTime"/>
      <xs:enumeration value="hexBinary"/>
      <xs:enumeration value="integer"/>
      <xs:enumeration value="int"/>
      <xs:enumeration value="long"/>
      <xs:enumeration value="string"/>
      <xs:enumeration value="unsignedInt"/>
      <xs:enumeration value="unsignedLong"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ComponentName">
    <xs:annotation>
      <xs:documentation>Component name; the same as xs:NCName except that dots are not
                        permitted.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:pattern value="&name;"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DataTypeName">
    <xs:annotation>
      <xs:documentation>Data type name; the same as xs:NCName except that cannot start
                        with lower-case letter (to avoid conflict with built-in data type
                        names) and dots are not permitted.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:pattern value="&Name;"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DefaultType">
    <xs:annotation>
      <xs:documentation>Type of default.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="factory">
        <xs:annotation>
          <xs:documentation>Default from standard, e.g. RFC.  Also applies on object
                        creation.</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="object">
        <xs:annotation>
          <xs:documentation>Default on object creation.</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DescriptionAction">
```



```
      <xs:annotation>
        <xs:documentation>Description action.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:enumeration value="create"/>
        <xs:enumeration value="append"/>
        <xs:enumeration value="replace"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="MaxEntries">
      <xs:annotation>
        <xs:documentation>Positive integer or "unbounded".</xs:documentation>
      </xs:annotation>
      <xs:union memberTypes="xs:positiveInteger">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="unbounded"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:simpleType name="ModelName">
      <xs:annotation>
        <xs:documentation>Model name, including major and minor versions.  The name part is
                          the same as xs:NCName except that dots are not
                          permitted.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:pattern value="&name;&colon;&num;&dot;&num;"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ObjectName">
      <xs:annotation>
        <xs:documentation>Object name (maximum length 256).  Each component is the same as
                          xs:NCName except that dots are not permitted.  This name MUST in
                          addition follow the vendor-specific object name requirements of
                          section 3.3.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:maxLength value="256"/>
        <xs:pattern value="(&name;&dot;(&inst;&dot;)?)+"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ObjectReference">
      <xs:annotation>
        <xs:documentation>Object path that cannot contain "{i}" placeholders and that
                          therefore references a single object.  The path MUST follow the
                          requirements of A.2.3.4 (its scope will typically be specified via
                          an attribute of type PathScope).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:pattern value="&dot;?(&name;&dot;)*"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ObjectReferencePattern">
      <xs:annotation>
        <xs:documentation>Object path that can contain "{i}" placeholders and that can
                          therefore references multiple objects.  The path MUST follow the
                          requirements of A.2.3.4 (its scope will typically be specified via
                          an attribute of type PathScope).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:pattern value="&dot;?(&name;&dot;(&inst;&dot;)?)*"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ObjectReferencePatterns">
      <xs:annotation>
        <xs:documentation>List of object paths, each of which can contain "{i}"
                          placeholders.</xs:documentation>
      </xs:annotation>
      <xs:list itemType="tns:ObjectReferencePattern"/>
```



```
  </xs:simpleType>
  <xs:simpleType name="OpaqueID">
    <xs:annotation>
      <xs:documentation>Opaque ID.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ParameterName">
    <xs:annotation>
      <xs:documentation>Parameter name (maximum length 256); the same as xs:NCName except
                    that dots are not permitted.  This name MUST in addition follow the
                    vendor-specific parameter name requirements of section
                    3.3.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:maxLength value="256"/>
      <xs:pattern value="&name;"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ParameterReference">
    <xs:annotation>
      <xs:documentation>Parameter path that cannot contain "{i}" placeholders and that
                    therefore references a single parameter.  The path MUST follow the
                    requirements of A.2.3.4 (its scope will typically be specified via
                    an attribute of type PathScope).</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:pattern value="&dot;?(&name;&dot;)*&name;?"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="PathScope">
    <xs:annotation>
      <xs:documentation>Object / parameter path name scope (A.2.3.4).</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="normal"/>
      <xs:enumeration value="model"/>
      <xs:enumeration value="object"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ProfileName">
    <xs:annotation>
      <xs:documentation>Profile name, including version.  The name part is the same as
                    xs:NCName except that dots are not permitted.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:pattern value="&name;&colon;&num;"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ProfileNames">
    <xs:annotation>
      <xs:documentation>List of profile names.</xs:documentation>
    </xs:annotation>
    <xs:list itemType="tns:ProfileName"/>
  </xs:simpleType>
  <xs:simpleType name="ProfileObjectAccess">
    <xs:annotation>
      <xs:documentation>Object access (within profile).</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="notSpecified"/>
      <xs:enumeration value="present"/>
      <xs:enumeration value="create"/>
      <xs:enumeration value="delete"/>
      <xs:enumeration value="createDelete"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ReadWriteAccess">
    <xs:annotation>
```



```
        <xs:documentation>Read-write access.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:enumeration value="readOnly"/>
        <xs:enumeration value="readWrite"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="ReferenceType">
      <xs:annotation>
        <xs:documentation>Reference type (A.2.3.6).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:enumeration value="weak"/>
        <xs:enumeration value="strong"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Status">
      <xs:annotation>
        <xs:documentation>Item status (applies to most types of item).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:enumeration value="current"/>
        <xs:enumeration value="deprecated"/>
        <xs:enumeration value="obsoleted"/>
        <xs:enumeration value="deleted"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="TargetType">
      <xs:annotation>
        <xs:documentation>(Reference) target type (used in path
                        references).</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:enumeration value="any"/>
        <xs:enumeration value="parameter"/>
        <xs:enumeration value="object"/>
        <xs:enumeration value="single"/>
        <xs:enumeration value="table"/>
        <xs:enumeration value="row"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="UnitsString">
      <xs:annotation>
        <xs:documentation>Units string.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:token">
        <xs:maxLength value="32"/>
      </xs:restriction>
    </xs:simpleType>
    <!-- Model groups -->
    <xs:group name="AllBuiltinDataTypes">
      <xs:annotation>
        <xs:documentation>All built-in data types.</xs:documentation>
      </xs:annotation>
      <xs:choice>
        <xs:element name="base64">
          <xs:complexType>
            <xs:sequence>
              <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="size" type="tns:SizeFacet">
                  <xs:annotation>
                    <xs:documentation>Length is that of the actual string, not the base64-
                        encoded string.  See A.2.3.3.</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:choice>
              <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute namespace="##other"/>
          </xs:complexType>
        </xs:element>
```



```
      <xs:element name="boolean">
        <xs:complexType>
          <xs:sequence>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="dateTime">
        <xs:complexType>
          <xs:sequence>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="hexBinary">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="size" type="tns:SizeFacet">
                <xs:annotation>
                  <xs:documentation>Length is that of the actual string, not the
                      hexBinary-encoded string.  See A.2.3.3.</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="int">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="instanceRef" type="tns:InstanceRefFacet"/>
              <xs:element name="range" type="tns:RangeFacet"/>
              <xs:element name="units" type="tns:UnitsFacet"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="long">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="range" type="tns:RangeFacet"/>
              <xs:element name="units" type="tns:UnitsFacet"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="string">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="size" type="tns:SizeFacet"/>
              <xs:element name="pathRef" type="tns:PathRefFacet"/>
              <xs:element name="enumeration" type="tns:EnumerationFacet"/>
              <xs:element name="enumerationRef" type="tns:EnumerationRefFacet"/>
              <xs:element name="pattern" type="tns:PatternFacet"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
```



```
        <xs:unique name="stringEnumerationValue">
          <xs:selector xpath="enumeration"/>
          <xs:field xpath="@value"/>
        </xs:unique>
        <xs:unique name="stringPatternValue">
          <xs:selector xpath="pattern"/>
          <xs:field xpath="@value"/>
        </xs:unique>
      </xs:element>
      <xs:element name="unsignedInt">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="instanceRef" type="tns:InstanceRefFacet"/>
              <xs:element name="range" type="tns:RangeFacet"/>
              <xs:element name="units" type="tns:UnitsFacet"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="unsignedLong">
        <xs:complexType>
          <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="range" type="tns:RangeFacet"/>
              <xs:element name="units" type="tns:UnitsFacet"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:group>
  <xs:group name="AllFacets">
    <xs:annotation>
      <xs:documentation>All facets.</xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:element name="size" type="tns:SizeFacet">
        <xs:annotation>
          <xs:documentation>Size facets, taken together, define the valid size ranges,
                    e.g. (0:0) and (6:6) mean that the size has to be 0 or 6.
The size facet MUST NOT be specified for non-string data types, i.e. data types that are
                    not derived from base64, hexBinary or string.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="instanceRef" type="tns:InstanceRefFacet">
        <xs:annotation>
          <xs:documentation>InstanceRef facets specify how a parameter can reference an
                    object instance (table row) via its instance number.
The instanceRef facet MUST NOT be specified for data types that are not derived from int
                    or unsignedInt.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="pathRef" type="tns:PathRefFacet">
        <xs:annotation>
          <xs:documentation>PathRef facets specify how a parameter can reference a
                    parameter or object via its path name.
The pathRef facet MUST NOT be specified for data types that are not derived from
                    string.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="range" type="tns:RangeFacet">
        <xs:annotation>
          <xs:documentation>Range facets, taken together, define the valid value  ranges,
                    e.g. [-1:-1] and [1:4094] mean that the value has to be -1 or 1:4094
                    (it cannot be 0).
```



```
The range facet MUST NOT be specified for non-numeric data types, i.e. data types that are
                    not derived from one of the integer types.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="enumeration" type="tns:EnumerationFacet">
        <xs:annotation>
          <xs:documentation>Enumeration facets, taken together, define the valid values,
                    e.g. "a" and "b" mean that the value has to be a or b.
The enumeration facet MUST NOT be specified for data types that are not derived from
                    string.
Derived types MAY add additional enumeration values.  See A.2.5.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="enumerationRef" type="tns:EnumerationRefFacet">
        <xs:annotation>
          <xs:documentation>EnumerationRef facets allow a parameter's valid values to be
                    obtained from another parameter.
The enumerationRef facet MUST NOT be specified for data types that are not derived from
                    string.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="pattern" type="tns:PatternFacet">
        <xs:annotation>
          <xs:documentation>Pattern attributes, taken together, define valid patterns,
                    e.g. "" and "[0-9A-Fa-f]{6}" means that the value has to be empty or
                    a 6 digit hex string.
The pattern facet MUST NOT be specified for data types that are not derived from string.
Pattern syntax is the same as for XML Schema regular expressions.  See
                    http://www.w3.org/TR/xmlschema-2/#regexs.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="units" type="tns:UnitsFacet">
        <xs:annotation>
          <xs:documentation>Multiple units facets MUST NOT be specified.
The units facet MUST NOT be specified for data types that are not numeric, i.e. data types
                    that are not derived from one of the integer
                    types.</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:choice>
  </xs:group>
  <!-- Complex types -->
  <xs:complexType name="BaseStatusFacet" abstract="true">
    <xs:annotation>
      <xs:documentation>Base facet (status attribute).</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="status" type="tns:Status" default="current"/>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="BaseAccessFacet" abstract="true">
    <xs:annotation>
      <xs:documentation>Base facet (access, status and optional
                    attributes).</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="tns:BaseStatusFacet">
        <xs:attribute name="access" type="tns:ReadWriteAccess" default="readWrite"/>
        <xs:attribute name="optional" type="xs:boolean" default="false"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Bibliography">
    <xs:annotation>
      <xs:documentation>Bibliography definition.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
```



```
      <xs:element name="reference" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:token">
              <xs:annotation>
                <xs:documentation>Name by which the referenced document is usually known,
                      e.g. TR-069, RFC 2863.</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="title" type="xs:token" minOccurs="0"/>
            <xs:element name="organization" type="xs:token" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Organization that published the referenced document,
                      e.g. BBF, IEEE, IETF.</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="category" type="xs:token" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Document category, e.g. TR (BBF), RFC
                      (IETF).</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="date" type="xs:token" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Publication date.</xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="hyperlink" type="xs:anyURI"/>
            </xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="id" type="tns:BibrefId" use="required">
            <xs:annotation>
              <xs:documentation>Uniquely identifies the reference (this is checked by
                      schema validation).  Can be referenced from descriptions by using
                      the {{bibref}} template.  See A.2.2.4.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ComponentDefinition">
    <xs:annotation>
      <xs:documentation>Component definition.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="component" type="tns:ComponentReference"/>
        <xs:element name="parameter" type="tns:ModelParameter"/>
        <xs:element name="object" type="tns:ModelObject">
          <xs:unique name="componentObjectParameterName">
            <xs:selector xpath="parameter"/>
            <xs:field xpath="@name"/>
          </xs:unique>
          <xs:keyref name="componentEnableParameterRef"
                     refer="tns:componentObjectParameterName">
            <xs:selector xpath="."/>
            <xs:field xpath="@enableParameter"/>
          </xs:keyref>
          <xs:keyref name="componentUniqueKeyRef"
                     refer="tns:componentObjectParameterName">
            <xs:selector xpath="uniqueKey/parameter"/>
            <xs:field xpath="@ref"/>
          </xs:keyref>
        </xs:element>
      </xs:choice>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
```



```
        <xs:element name="profile" type="tns:Profile"/>
      </xs:choice>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="tns:ComponentName" use="required">
      <xs:annotation>
        <xs:documentation>MUST be unique within the document, including imported
                components (this is checked by schema
                validation).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="status" type="tns:Status" default="current"/>
    <xs:attribute name="id" type="tns:OpaqueID"/>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="ComponentReference">
    <xs:annotation>
      <xs:documentation>Component reference.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="ref" type="tns:ComponentName" use="required">
      <xs:annotation>
        <xs:documentation>Name of component to be referenced
                (included).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="path" type="tns:ObjectName">
      <xs:annotation>
        <xs:documentation>If specified, is relative path between point of reference
                (inclusion) and the component's items.  If not specified, behavior
                is as if an empty relative path was specified.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="DataTypeDefinition">
    <xs:annotation>
      <xs:documentation>Parameter data type definition.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
      <xs:choice>
        <xs:group ref="tns:AllFacets" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Facets MUST NOT be specified if the base attribute is
                    omitted.</xs:documentation>
          </xs:annotation>
        </xs:group>
        <xs:group ref="tns:AllBuiltinDataTypes" minOccurs="0">
          <xs:annotation>
            <xs:documentation>A built-in data type element MUST NOT be specified if the
                    base attribute is present.
See tns:AllFacets for notes and requirements on individual facets.</xs:documentation>
          </xs:annotation>
        </xs:group>
      </xs:choice>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="tns:DataTypeName" use="required">
      <xs:annotation>
        <xs:documentation>MUST be unique within the document, including imported data
                types (this is checked by schema validation).
Cannot begin with a lower-case letter, in order to avoid confusion with built-in data
                types.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="base" type="tns:DataTypeName">
      <xs:annotation>
        <xs:documentation>MUST be present if and only if deriving from a non-built-in data
                type.  See A.2.3.1.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
```



```
      <xs:attribute name="status" type="tns:Status" default="current"/>
      <xs:attribute name="id" type="tns:OpaqueID"/>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="DataTypeReference">
      <xs:annotation>
        <xs:documentation>Parameter data type reference or anonymous restriction /
                        extension.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:group ref="tns:AllFacets" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ref" type="tns:DataTypeName">
        <xs:annotation>
          <xs:documentation>If specified, content MUST be empty.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="base" type="tns:DataTypeName">
        <xs:annotation>
          <xs:documentation>If specified, content MUST NOT be empty.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="DefaultFacet">
      <xs:annotation>
        <xs:documentation>Default facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseStatusFacet">
          <xs:attribute name="type" type="tns:DefaultType" use="required"/>
          <xs:attribute name="value" type="xs:string" use="required">
            <xs:annotation>
              <xs:documentation>Value MUST be valid for the data type.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Description">
      <xs:annotation>
        <xs:documentation>Description: free text which MAY contain a limited amount of
                        mediawiki-like markup as specified in A.2.2.  For example, use "*"
                        at the start of a line to indicate a bulleted list.
  To avoid confusion, the description SHOULD NOT contain tab characters.
  For BBF standards, the character set MUST be restricted to printable characters in the
                        Basic Latin Unicode block, i.e. to characters whose decimal ASCII
                        representations are in the (inclusive) ranges 9-10 and 32-
                        126.</xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="action" type="tns:DescriptionAction" default="create">
            <xs:annotation>
              <xs:documentation>This MUST be specified when the description modifies that of
                        a previously defined item.
  Specify "append" to append to the previous description, or "replace" to replace the
                        previous description.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:anyAttribute namespace="##other"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="EnumerationFacet">
      <xs:annotation>
        <xs:documentation>Enumeration facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseAccessFacet">
```



```
          <xs:attribute name="value" type="xs:string" use="required"/>
          <xs:attribute name="code" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="EnumerationRefFacet">
      <xs:annotation>
        <xs:documentation>Enumeration reference facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseStatusFacet">
          <xs:attribute name="targetParam" type="tns:ParameterReference" use="required">
            <xs:annotation>
              <xs:documentation>MUST reference a list-valued parameter.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="targetParamScope" type="tns:PathScope" default="normal">
            <xs:annotation>
              <xs:documentation>Specifies the point in the naming hierarchy relative to
                        which targetParam applies (A.2.3.4).</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="nullValue" type="xs:token">
            <xs:annotation>
              <xs:documentation>Specifies the value that indicates that none of the values
                        of the referenced parameter currently apply.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Import">
      <xs:annotation>
        <xs:documentation>Import data types, components and models (Root and Service
                        Objects) from external documents.  All such items MUST be imported
                        (this is checked by schema validation).
  The optional ref attribute MAY be used in order to avoid name conflicts between imported
                        and locally-defined items.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="dataType">
            <xs:complexType>
              <xs:attribute name="name" type="tns:DataTypeName" use="required"/>
              <xs:attribute name="ref" type="tns:DataTypeName">
                <xs:annotation>
                  <xs:documentation>If omitted, data type is known by the same name in both
                        this and the referenced document.</xs:documentation>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
          <xs:element name="component">
            <xs:complexType>
              <xs:attribute name="name" type="tns:ComponentName" use="required"/>
              <xs:attribute name="ref" type="tns:ComponentName">
                <xs:annotation>
                  <xs:documentation>If omitted, component is known by the same name in both
                        this and the referenced document.</xs:documentation>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
          <xs:element name="model">
            <xs:complexType>
              <xs:attribute name="name" type="tns:ModelName" use="required"/>
              <xs:attribute name="ref" type="tns:ModelName">
                <xs:annotation>
                  <xs:documentation>If omitted, model is known by the same name in both this
                        and the referenced document.</xs:documentation>
                </xs:annotation>
```



```
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="file" type="xs:string" use="required">
      <xs:annotation>
        <xs:documentation>Only the file name part SHOULD be specified (the processing tool
                  is assumed to support a search path).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="spec" type="xs:anyURI">
      <xs:annotation>
        <xs:documentation>If specified, the processing tool MUST regard a mismatch between
                  this and the external document's spec attribute as an
                  error.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="InstanceRefFacet">
    <xs:annotation>
      <xs:documentation>Instance number reference facet.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="tns:BaseStatusFacet">
        <xs:attribute name="refType" type="tns:ReferenceType" use="required">
          <xs:annotation>
            <xs:documentation>Specifies the type of reference
                  (A.2.3.6).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetParent" type="tns:ObjectReference" use="required">
          <xs:annotation>
            <xs:documentation>MUST reference a multi-instance object (table)
                  (A.2.3.4).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetParentScope" type="tns:PathScope" default="normal">
          <xs:annotation>
            <xs:documentation>Specifies the point in the naming hierarchy relative to
                  which targetParent applies (A.2.3.4).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ListFacet">
    <xs:annotation>
      <xs:documentation>List facet.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="tns:BaseStatusFacet">
        <xs:sequence>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="size" type="tns:SizeFacet">
              <xs:annotation>
                <xs:documentation>This specifies the size of the TR-069 list-valued
                      parameter, not of the individual list items.</xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:choice>
        </xs:sequence>
        <xs:attribute name="minItems" type="xs:nonNegativeInteger" default="0"/>
        <xs:attribute name="maxItems" type="tns:MaxEntries" default="unbounded"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Model">
    <xs:annotation>
```



```
      <xs:documentation>Model (Root or Service Object) definition and
                        reference.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="component" type="tns:ComponentReference"/>
        <xs:element name="parameter" type="tns:ModelParameter"/>
        <xs:element name="object" type="tns:ModelObject">
          <xs:unique name="objectParameterName">
            <xs:selector xpath="parameter"/>
            <xs:field xpath="@name"/>
          </xs:unique>
          <xs:keyref name="objectEnableParameterRef" refer="tns:objectParameterName">
            <xs:selector xpath="."/>
            <xs:field xpath="@enableParameter"/>
          </xs:keyref>
          <xs:keyref name="objectUniqueKeyRef" refer="tns:objectParameterName">
            <xs:selector xpath="uniqueKey/parameter"/>
            <xs:field xpath="@ref"/>
          </xs:keyref>
        </xs:element>
      </xs:choice>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="profile" type="tns:Profile"/>
      </xs:choice>
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="tns:ModelName" use="required">
      <xs:annotation>
        <xs:documentation>MUST be unique within the document, including imported models
                        (this is checked by schema validation).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="base" type="tns:ModelName">
      <xs:annotation>
        <xs:documentation>MUST be present if and only if extending an existing model.  See
                        A.2.10.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="isService" type="xs:boolean" default="false"/>
    <xs:attribute name="status" type="tns:Status" default="current"/>
    <xs:attribute name="id" type="tns:OpaqueID"/>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="ModelObject">
    <xs:annotation>
      <xs:documentation>Object definition and reference.  See A.2.8.1 for details of how
                        tables are represented.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="description" type="tns:Description" minOccurs="0"/>
      <xs:element name="uniqueKey" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>MUST NOT be present if the object is not a table (see
                        maxEntries).
The parameters referenced by each unique key element MUST constitute a unique key.
For a table in which there is an enableParameter, the uniqueness requirement applies only
                        to enabled table entries.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="parameter" maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="ref" type="tns:ParameterName" use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
        <xs:unique name="uniqueKeyParameterRef">
          <xs:selector xpath="parameter"/>
```



```
            <xs:field xpath="@ref"/>
          </xs:unique>
        </xs:element>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="component" type="tns:ComponentReference"/>
          <xs:element name="parameter" type="tns:ModelParameter"/>
        </xs:choice>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="tns:ObjectName">
        <xs:annotation>
          <xs:documentation>MUST be unique within the component or model (this is checked by
                        schema validation).
MUST be present if and only if defining a new object.
If the object is a table (see maxEntries), the final part of the name MUST be
                        "{i}.".</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="base" type="tns:ObjectName">
        <xs:annotation>
          <xs:documentation>MUST be present if and only if modifying an existing
                        object.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="access" type="tns:ReadWriteAccess" use="required"/>
      <xs:attribute name="minEntries" type="xs:nonNegativeInteger" use="required">
        <xs:annotation>
          <xs:documentation>minEntries MUST be less than or equal to maxEntries (all values
                        are regarded as being less than "unbounded").</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="maxEntries" type="tns:MaxEntries" use="required">
        <xs:annotation>
          <xs:documentation>minEntries and maxEntries indicate whether the object is a
                        table:
* minEntries=0, maxEntries=1 : single-instance object which might not be allowed to exist,
                        e.g. because only one of it and another object can exist at the same
                        time
* minEntries=1, maxEntries=1 : single-instance object that is always allowed to exist
* all other cases : object is a table</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="numEntriesParameter" type="tns:ParameterName">
        <xs:annotation>
          <xs:documentation>MUST be specified for a table with a variable number of entries,
                        i.e. for which maxEntries is greater than minEntries ("unbounded" is
                        regarded as being greater than all values).</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="enableParameter" type="tns:ParameterName">
        <xs:annotation>
          <xs:documentation>MUST be specified for a table in which the ACS can create
                        entries and which has one or more uniqueKey
                        elements.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="status" type="tns:Status" default="current"/>
      <xs:attribute name="id" type="tns:OpaqueID"/>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="ModelParameter">
      <xs:annotation>
        <xs:documentation>Parameter definition and reference.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="description" type="tns:Description" minOccurs="0"/>
        <xs:element name="syntax" type="tns:Syntax" minOccurs="0"/>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="tns:ParameterName">
        <xs:annotation>
```



```
        <xs:documentation>MUST be unique within the parent object (this is checked by
                  schema validation).
MUST be present if and only if defining a new parameter.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="base" type="tns:ParameterName">
      <xs:annotation>
        <xs:documentation>MUST be present if and only if modifying an existing
                  parameter.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="access" type="tns:ReadWriteAccess" use="required"/>
    <xs:attribute name="status" type="tns:Status" default="current"/>
    <xs:attribute name="activeNotify" type="tns:ActiveNotify" default="normal"/>
    <xs:attribute name="forcedInform" type="xs:boolean" default="false"/>
    <xs:attribute name="id" type="tns:OpaqueID"/>
    <xs:anyAttribute namespace="##other"/>
  </xs:complexType>
  <xs:complexType name="PathRefFacet">
    <xs:annotation>
      <xs:documentation>Path name reference facet.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="tns:BaseStatusFacet">
        <xs:attribute name="refType" type="tns:ReferenceType" use="required">
          <xs:annotation>
            <xs:documentation>Specifies the type of reference
                  (A.2.3.6).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetParent" type="tns:ObjectReferencePatterns" default="">
          <xs:annotation>
            <xs:documentation>If the list is non-empty, this parameter MUST only reference
                  immediate children of matching objects (A.2.3.4).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetParentScope" type="tns:PathScope" default="normal">
          <xs:annotation>
            <xs:documentation>Specifies the point in the naming hierarchy relative to
                  which targetParent applies (A.2.3.4).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetType" type="tns:TargetType" default="any">
          <xs:annotation>
            <xs:documentation>Specifies the type of item that can be
                  referenced.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="targetDataType" type="tns:AnyTypeName" default="any">
          <xs:annotation>
            <xs:documentation>Specifies the valid data types for a referenced
                  parameter.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="PatternFacet">
    <xs:annotation>
      <xs:documentation>Pattern facet (pattern syntax is as in XML
                  Schema).</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="tns:BaseAccessFacet">
        <xs:attribute name="value" type="xs:string" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Profile">
    <xs:annotation>
      <xs:documentation>Profile definition and reference.</xs:documentation>
```



```
      </xs:annotation>
      <xs:sequence>
        <xs:element name="description" type="tns:Description" minOccurs="0">
          <xs:annotation>
            <xs:documentation>If the extends attribute is insufficient to express general
                        profile requirements, any additional requirements MUST be specified
                        here.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="parameter" type="tns:ProfileParameter"/>
          <xs:element name="object" type="tns:ProfileObject"/>
        </xs:choice>
        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="tns:ProfileName">
        <xs:annotation>
          <xs:documentation>MUST be unique within the model (this is checked by schema
                        validation).
MUST be present if and only if defining a new profile.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="base" type="tns:ProfileName">
        <xs:annotation>
          <xs:documentation>MUST specify base if modifying an existing profile or if the
                        profile version is greater than 1.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="extends" type="tns:ProfileNames">
        <xs:annotation>
          <xs:documentation>MUST specify extends if the profile extends other
                        profile(s).</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="status" type="tns:Status" default="current"/>
      <xs:attribute name="id" type="tns:OpaqueID"/>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="ProfileObject">
      <xs:annotation>
        <xs:documentation>Profile object definition.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="description" type="tns:Description" minOccurs="0">
          <xs:annotation>
            <xs:documentation>If the requirement attribute is insufficient to express the
                        requirement, any additional requirements MUST be specified here and
                        MAY override the attribute.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="parameter" type="tns:ProfileParameter" minOccurs="0"
                        maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ref" type="tns:ObjectName" use="required"/>
      <xs:attribute name="requirement" type="tns:ProfileObjectAccess" use="required"/>
      <xs:attribute name="status" type="tns:Status" default="current"/>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="ProfileParameter">
      <xs:annotation>
        <xs:documentation>Profile parameter definition.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="description" type="tns:Description" minOccurs="0">
          <xs:annotation>
            <xs:documentation>If the requirement attribute is insufficient to express the
                        requirement, any additional requirements MUST be specified here and
                        MAY override the attribute.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
```



```
      <xs:attribute name="ref" type="tns:ParameterName" use="required"/>
      <xs:attribute name="requirement" type="tns:ReadWriteAccess" use="required"/>
      <xs:attribute name="status" type="tns:Status" default="current"/>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="RangeFacet">
      <xs:annotation>
        <xs:documentation>Range facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseAccessFacet">
          <xs:attribute name="minInclusive" type="xs:integer"/>
          <xs:attribute name="maxInclusive" type="xs:integer"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SizeFacet">
      <xs:annotation>
        <xs:documentation>Size facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseAccessFacet">
          <xs:attribute name="minLength" type="xs:nonNegativeInteger" default="0"/>
          <xs:attribute name="maxLength" type="xs:nonNegativeInteger" default="16"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="UnitsFacet">
      <xs:annotation>
        <xs:documentation>Units facet.</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="tns:BaseStatusFacet">
          <xs:attribute name="value" type="tns:UnitsString" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Syntax">
      <xs:annotation>
        <xs:documentation>Parameter syntax specification.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="list" type="tns:ListFacet" minOccurs="0">
          <xs:annotation>
            <xs:documentation>For lists, the TR-069 parameter is always a string and the
                      data type specification applies to individual list items, not to the
                      parameter value.
See section 3.2 for comma-separated list formatting rules.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:choice>
          <xs:group ref="tns:AllBuiltinDataTypes">
            <xs:annotation>
              <xs:documentation>Direct use of built-in data type, possibly modified via use
                      of facets.</xs:documentation>
            </xs:annotation>
          </xs:group>
          <xs:element name="dataType" type="tns:DataTypeReference">
            <xs:annotation>
              <xs:documentation>Use of named data type, possibly modified via use of
                      facets.</xs:documentation>
            </xs:annotation>
            <xs:unique name="dtRefEnumerationValue">
              <xs:selector xpath="enumeration"/>
              <xs:field xpath="@value"/>
            </xs:unique>
            <xs:unique name="dtRefPatternValue">
              <xs:selector xpath="pattern"/>
              <xs:field xpath="@value"/>
            </xs:unique>
          </xs:element>
```



```
        </xs:choice>
        <xs:element name="default" type="tns:DefaultFacet" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="hidden" type="xs:boolean" default="false">
        <xs:annotation>
          <xs:documentation>If true, readback is always false, 0 or empty
                    string.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <!-- Elements -->
    <xs:element name="document">
      <xs:annotation>
        <xs:documentation>CWMP Data Model Definition XML Schema (DM Schema) instance
                    documents can contain any or all of the following:
* Data type definitions
* Root Object definitions (including profiles)
* Service Object definitions (including profiles)
* Component definitions
* Vendor extension definitions</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="description" type="tns:Description" minOccurs="0">
            <xs:annotation>
              <xs:documentation>Top-level description.</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="import" type="tns:Import" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>Imported data types, components and models (Root and Service
                    Objects).</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="dataType" type="tns:DataTypeDefinition" minOccurs="0"
                    maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>Top-level data type definitions.</xs:documentation>
            </xs:annotation>
            <xs:unique name="dtDefEnumerationValue">
              <xs:selector xpath="enumeration"/>
              <xs:field xpath="@value"/>
            </xs:unique>
            <xs:unique name="dtDefPatternValue">
              <xs:selector xpath="pattern"/>
              <xs:field xpath="@value"/>
            </xs:unique>
          </xs:element>
          <xs:element name="bibliography" type="tns:Bibliography" minOccurs="0">
            <xs:annotation>
              <xs:documentation>Bibliographic references.</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="component" type="tns:ComponentDefinition">
              <xs:annotation>
                <xs:documentation>Component definitions.</xs:documentation>
              </xs:annotation>
              <xs:unique name="componentParameterName">
                <xs:selector xpath="parameter"/>
                <xs:field xpath="@name"/>
              </xs:unique>
              <xs:unique name="componentObjectName">
                <xs:selector xpath="object"/>
                <xs:field xpath="@name"/>
              </xs:unique>
              <xs:unique name="componentProfileName">
                <xs:selector xpath="profile"/>
                <xs:field xpath="@name"/>
              </xs:unique>
```



```
            </xs:element>
            <xs:element name="model" type="tns:Model">
              <xs:annotation>
                <xs:documentation>Model (Root and Service Object)
                        definitions.</xs:documentation>
              </xs:annotation>
              <xs:unique name="modelParameterName">
                <xs:selector xpath="parameter"/>
                <xs:field xpath="@name"/>
              </xs:unique>
              <xs:unique name="modelObjectName">
                <xs:selector xpath="object"/>
                <xs:field xpath="@name"/>
              </xs:unique>
              <xs:unique name="modelProfileName">
                <xs:selector xpath="profile"/>
                <xs:field xpath="@name"/>
              </xs:unique>
            </xs:element>
          </xs:choice>
          <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="spec" use="required">
          <xs:annotation>
            <xs:documentation>URI of the associated specification document, e.g. the BBF
                        Technical Report.  This URI SHOULD uniquely identify the
                        specification.  More than one DM Schema instance document MAY
                        reference the same specification.
Where the specification is a BBF document, the URI naming rules specified in A.2.1 MUST be
                        used.  For example, to reference TR-106 Issue 1 Amendment 2, the
                        value of this attribute would be urn:broadband-forum-org:tr-106-1-
                        2.</xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:anyURI">
              <xs:pattern value=".+"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:anyAttribute namespace="##other"/>
      </xs:complexType>
      <xs:unique name="dataTypeName">
        <xs:selector xpath="dataType|import/dataType"/>
        <xs:field xpath="@name"/>
      </xs:unique>
      <xs:unique name="componentName">
        <xs:selector xpath="component|import/component"/>
        <xs:field xpath="@name"/>
      </xs:unique>
      <xs:unique name="modelName">
        <xs:selector xpath="model|import/model"/>
        <xs:field xpath="@name"/>
      </xs:unique>
      <xs:unique name="bibId">
        <xs:selector xpath="bibliography/reference"/>
        <xs:field xpath="@id"/>
      </xs:unique>
      <xs:keyref name="dataTypeBase" refer="tns:dataTypeName">
        <xs:selector xpath="dataType|.//parameter/syntax/dataType"/>
        <xs:field xpath="@base"/>
      </xs:keyref>
      <xs:keyref name="dataTypeRef" refer="tns:dataTypeName">
        <xs:selector xpath=".//parameter/syntax/dataType"/>
        <xs:field xpath="@ref"/>
      </xs:keyref>
      <xs:keyref name="componentRef" refer="tns:componentName">
        <xs:selector xpath=".//component"/>
        <xs:field xpath="@ref"/>
      </xs:keyref>
      <xs:keyref name="modelBase" refer="tns:modelName">
        <xs:selector xpath="model"/>
```



```
        <xs:field xpath="@base"/>
      </xs:keyref>
    </xs:element>
</xs:schema>
```


Appendix I. "Device" Root Object, Common Objects and Components

The "Device" Root Object, Common Objects and Components are defined in several DM Instance documents, which include the following information:

- Data type definitions, the IP address and MAC address conventions of section 3.2.
- Bibliography, containing those items from the Normative References section that are referenced by data model definitions.
- "Device" Root Object definition, including the Common Objects and Components of Table 3 through Table 5 and the profiles of Table 6 through Table 18. The Root Object definition has a model element for each version of the Root Object; each such model element consists of the changes relative to the previous version, if any.

The DM Instance documents are as follows, and can be found at http://www.broadband-forum.org/cwmp/<File>.

Date	Document	File	Description
September 2005	TR-106	tr-106-1-0-types.xml	TR-069 Data Model Data Types.
		tr-106-1-0.xml	TR-069 Device:1.0 Root Object definition.
December 2006	TR-106 Amendment 1	tr-106-1-1.xml	TR-069 Device:1.1 Root Object definition.
May 2008	TR-143	tr-143-1-0.xml	TR-069 Device:1.2 (and InternetGatewayDevice:1.3) Root Object definition.
November 2008	TR-106 Amendment 2	tr-106-1-2.xml	TR-069 Device:1.2 Root Object errata and clarifications; only minor changes, hence not a new version of the Root Object.

