More Details about
TR-069
CPE WAN Management Protocol

William.L
wiliwe@gmail.com

2005-11-28

Index

Summary

General Description

The CPE WAN Management Protocol defines a mechanism that encompasses secure auto-configuration of a CPE, and also incorporates other CPE management functions into a common framework.

Terminology

In this seciont, it will list all abbreviations used in this document.

B-NT : Broadband Network Termination

CPE : Customer Primise Equipment

A DSL B-NT is one of form of broadband CPE. This could be ADSL modem

DSLAM : Digital Subscriber Line Access Multiplexer

The DSLAM at the access provider is the equipment that really allows DSL to happen. A DSLAM takes connections from many customers and aggregates them onto a single, high-capacity connection to the Internet. It **may** provide additional functions including **routing** or **dynamic IP address assignment** for the customers.

BRAS : Broadband Remote Access Server

This is the connection point to the network(Internet, WAN) and application service providers(ISP, Coporate Network…etc)

ACS : Auto-Configuration Server

This is a component in the broadband network responsible for auto-configuration of the CPE for advanced services.

IGD : Internet Gateway Device

A CPE device that is either a B-NT or a broadband router.

RPC : Remote Procedure Call

Parameter

A name-value pari representing a manageable CPE parameter made accessible to an ACS for reading and/or writing.

Session

A contiguous sequence of transactions(**one request** and **one response**) between a CPE and an ACS.

Voucher

A digitally signed data structure that instructs a particular CPE to enable or disable Options, and characteristics that determine under what conditions the Options persist.

OUI : Organizationally Unique Identifier

An OUI is a **24-bit** globally unique assigned number referenced by various standards. For example, the OUI is used in the family of **802 LAN** standards: **Ethernet**, **Token Ring**, etc. The OUI is usually concatenated with another 24 bits that are assigned by that Organization in order to make a 48-bit number that is unique to a particular piece of hardware, to make it possible to uniquely address that hardware. For example, the Ethernet MAC Address is such a 48-bit number, unique to one particular Ethernet interface. The OUI is usually concatenated with 24 or 40 bits to form an EUI-48 or an EUI-64.

The other names for OUI is : **MAC Address**, **Vendor Address**, **Vendor ID**, **NIC Address**, **Ethernet Address** and others.

Functional Components

The CPE WAN Management Protocol is intended to support a variety of functionalities to manage a collection of CPEs lie in LAN through WAN. It provides the following main capability :

Auto-Configuration and Dynamic Service Provisioning

The protocol allows an ACS to provision a CPE or collection of CPE based on a variety of criteria. The provisioning mechanism includes specific provisioning parameters and a general mechanism for adding vendor-specific provisioning capabilities as needed.

Software/Firmware Image Management

The CPE WAN Management Protocol provides tools to manage downloading of CPE **software/firmware image** files. The protocol provides mechanisms for version identification, file download initiation (ACS initiated downloads and optional CPE initiated downloads), and notification of the ACS of the success or failure of a file download.

Status and Performance Monitoring

The protocol provides support for a CPE to make available information that the ACS may use to monitor the CPE's status and performance statistics. The protocol defines a common set of such parameters, and provides a standard syntax for vendors to define additional non-standard

Diagnostics

The protocol provides support for a CPE to make available information that the ACS may use to diagnose **connectivity** or **service issues**. The protocol defines a common set of such parameters and a general mechanism for adding vendor-specific diagnostic capabilities.

Positioning in the Auto-Configuration Architecture

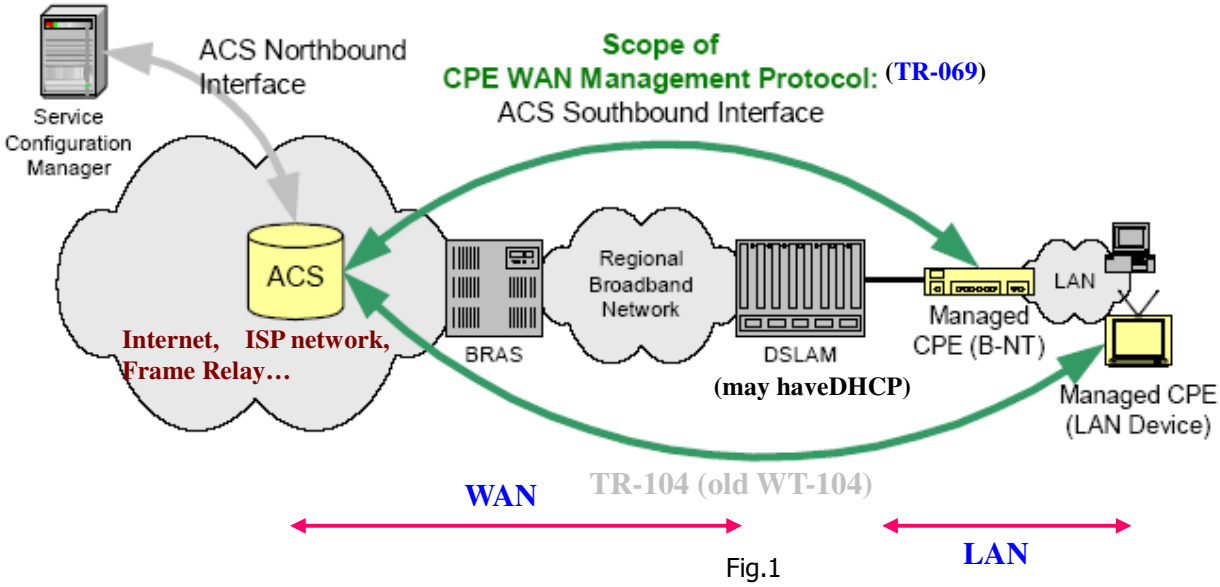


Fig.1

Architectural Goals

The protocol is intended to provide flexible support for various business models for **distributing and managing CPE**, including:

- CPE provided and managed by the **network provider**.
- CPE purchased in retail with **pre-registration to associate** the specific CPE with a service provider and customer account (a mobile-phone like model) .
- CPE purchased in retail **with post-installation user registration** with a service provider.

The protocol is intended to provide flexibility in the **connectivity model**. The protocol is intended to provide the following:

- Allow both CPE and ACS initiated connection establishment, **avoiding the need for a persistent connection to be maintained between each CPE and an ACS**.
- The functional interactions between the ACS and CPE should be **independent of which end initiated the establishment of the connection**. In particular, even where ACS initiated connectivity is not supported, all ACS initiated transactions **should be able to take place over a connection initiated by the CPE**.
- Allow one or more ACS servers to serve a population of CPE, which may be associated with one or more service providers.
- Optimize the use of connections that are established to **minimize connection multiple bi-directiona**l transactions to occur over **a single connection**.

The protocol is intended to support **discovery and association** of ACS and CPE :

- Provide mechanisms for **a CPE to discover the appropriate ACS** for **a given service provider**.
- Provide mechanisms to **allow an ACS to securely identify a CPE** (using **OUI**) and **associate it with** a **user/customer**. Processes to support such association should support models that incorporate user interaction as well as those that are fully automatic.

The protocol model to allow an ACS access to **control and monitor** various parameters associated with a CPE. The mechanisms provided to access these parameters is designed with the following premises :

- Different CPE may have differing capability levels, implementing different subsets of optional functionality. As a result, **an ACS must be able to discover the capabilities of a particular CPE**.
- **An ACS must be able to control and monitor the current configuration of a CPE**.
- Other control entities besides an ACS may be able to control some parameters of a CPE's configuration (e.g., via LAN-side auto-configuration). As a result, the protocol must allow an ACS to account for external changes to a CPE's configuration. The ACS should also be able to control which configuration parameters can be controlled via means other than by the ACS.
- The protocol should **allow vendor-specific parameters to be defined and accessed**.

Security

The protocol is designed to provide a high degree of security. The security model is also designed to be **scalable**. It is intended to allow basic security to accommodate less robust CPE implementations, while allowing greater security for those that can support more advanced security mechanisms.

Assumptions

Some assumptions made in defining the CPE WAN Management Protocol are listed below :

- In the case of a B-NT, prior to use of the CPE WAN Management Protocol, **a connection** has been **established to a WAN from which an ACS is accessible**.
- All CPE regardless of type (bridge1, router, or other) obtain an IP address in order to communicate with an ACS.
- **A CPE can interact with a single ACS at a time**. At any time, a CPE is aware of exactly one ACS with which it can connect. An ACS can hand off a CPE to another ACS only by explicitly altering the ACS contact and authentication information.

Use of Protocols

Protocol Components

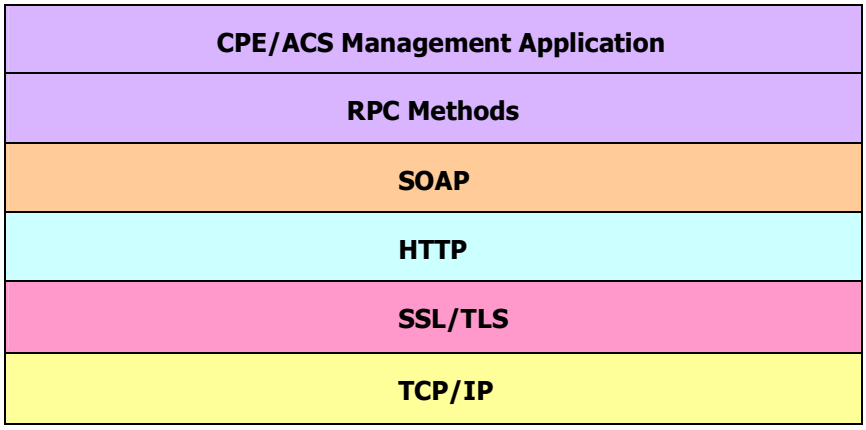


Fig.2

Protocol Layer	Description
CPE/ACS Application	The application uses the CPE WAN Management Protocol on the CPE and ACS, respectively. The application is locally defined and not specified as part of the CPE WAN Management Protocol
RPC Methods	The specific RPC methods that are defined by the CPE WAN Management Protocol (TR-69)
SOAP	A standard XML-based syntax used here to encode remote procedure calls. Specifically **SOAP 1.1** [4]
HTTP	**HTTP 1.1** [2]
SSL/TLS	The standard Internet transport layer security protocols. Specifically, either **SSL 3.0** [4] (Secure Socket Layer), **or TLS 1.0** [5] (Transport Layer Security). Use of SSL/TLS is RECOMMENDED but is not required
TCP/IP	Standard TCP/IP

Fig.3

ACS Discovery

- The CPE may be **configured locally** with the **URL of the ACS**.
- As part of the **IP layer** auto-configuration, a DHCP server on the **access network** may be configured to include the ACS URL as **a DHCP option**.
- The CPE may have **a default ACS URL** that it may use if no other URL is provided to it.
- The CPE would use **DNS(DomainName Server)** to resolve the IP address of the ACS from the **host name** component of the URL.

Use of SSL/TLS and TCP

- The use of SSL/TLS to transport the CPE WAN Management Protocol is **RECOMMENDED**
- If SSL/TLS is supported, support for encryption algorithms with key lengths **greater than or equal to 128 bits** should be supported.
- A CPE **must** be able to initiate **outgoing connections** to the ACS.
- An ACS must be able to accept **CPE-initiated** connections.
- If SSL/TLS is used, the **CPE must authenticate** the ACS using the **ACS-provided certificate**.
- If SSL/TLS is used, the ACS MAY accept a validated CPE-provided certificate to authenticate the CPE, but the ACS MUST allow the SSL/TLS connection to be established if the CPE does not provide a certificate.

Use of HTTP

- A SOAP **request from** an **ACS to a CPE** is sent over an HTTP **response**, while the **CPE**'s SOAP **response to an ACS request** is sent over a subsequent HTTP **post**.
- **Each HTTP post or response** may contain **more than one SOAP envelope** (within the negotiated limits). Each envelope may contain a SOAP request or response, independent from any other envelope.

Authentication

If the **CPE** is not authenticated using SSL/TLS, the **ACS must** authenticate the CPE using HTTP.

If SSL/TLS is being used for encryption, the ACS **may** use either **basic** or **digest authentication**[3]. **If SSL/TLS is not being used**, then the **ACS must use digest authentication**.

The ACS may issue the authentication **once** as part of the **first HTTP transaction**, and assume the authentication to hold for the duration of the TCP connection.

If any form of HTTP authentication is used to authenticate the CPE, the CPE should use a **username** / **userid** that is globally unique among all CPE manufacturers. Specifically it should be a multi-part string comprising a manufacturer identifier and a serial number unique within that manufacturer.

The recommended format for this string is:

OUI-SERIAL

where OUI is a six hexadecimal-digit value using **all upper-case** letters and including any **leading zeros**. The OUI value MUST be a valid OUI as defined in [5]. SERIAL is a **string** that uniquely identifies the CPE from the **particular manufacturer**.

If the manufacturer has multiple CPE products with overlapping serial number ranges, the SERIAL string **must** include **additional distinguishing characters** to ensure that the entire string is unique.

Example: "00D09E-0123456789" The password used in either form of HTTP authentication should be a unique value for each CPE. That is, **multiple CPE should not share the same password**. This password is a shared secret, and thus **must be known by both CPE and ACS**.

Both CPE and ACS should take appropriate steps to prevent unauthorized access to the password, or list of passwords in the case of an ACS.

Use of SOAP

The CPE WAN Management Protocol defines **SOAP 1.1** as the **encoding syntax** to transport the RPC method calls and responses.

The encoding must use the standard SOAP 1.1 envelope and serialization namespaces :

- Envelope namespace identifier "**http://schemas.xmlsoap.org/soap/envelope/**"
- Serialization namespace identifier "**http://schemas.xmlsoap.org/soap/encoding/**"

All elements and attributes defined as part of this version of the CPE WAN Management Protocol are associated with the following namespace identifier:

- "**urn:dslforum-org:cwmp-1-0**"

TR-69 specified RPCs

Method name	CPE requirement	Server requirement
CPE methods	Responding	Calling
GetRPCMethods	Required	Optional
SetParameterValues	Required	Required
GetParameterValues	Required	Required
GetParameterNames	Required	Required
SetParameterAttributes	Required	Optional
GetParameterAttributes	Required	Optional
AddObject	Required	Optional
DeleteObject	Required	Optional
Reboot	Required	Optional
Download	Required	Required
Upload	Optional	Optional
FactoryReset	Optional	Optional
GetQueuedTransfers	Optional	Optional
ScheduleInform	Optional	Optional
SetVouchers	Optional	Optional
GetOptions	Optional	Optional
Server methods	Calling	Responding
GetRPCMethods	Optional	Required
Inform	Required	Required
TransferComplete	Required	Required
RequestDownload	Optional	Optional
Kicked	Optional	Optional

Fig.4

Fault Handling

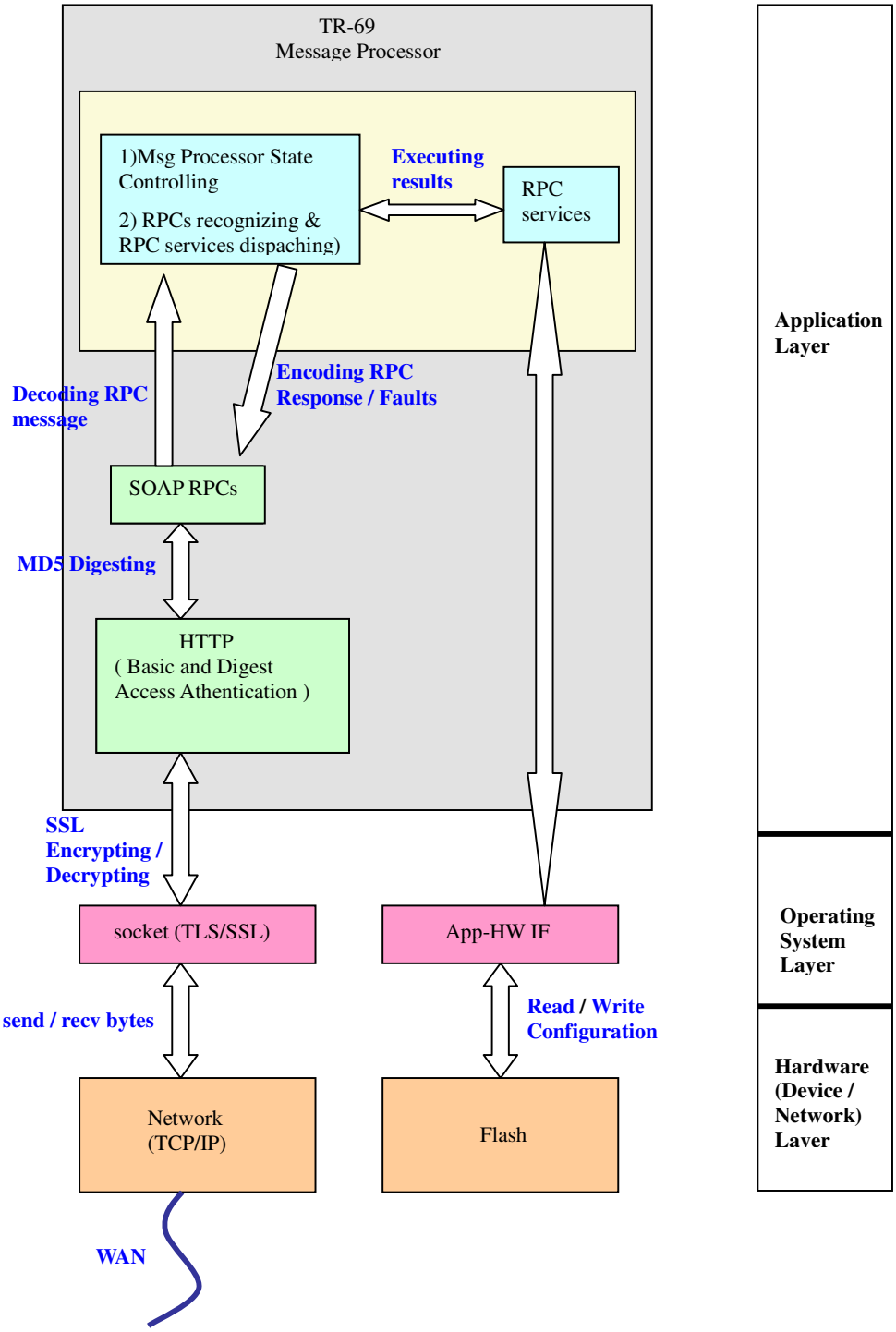
Fig.5 lists the fault codes that can be returned by a CPE.

Fault code	Description
9000	Method not supported
9001	Request denied (no reason specified)
9002	Internal error
9003	Invalid arguments
9004	Resources exceeded (when used in association with SetParameterValues, this MUST not be used to indicate parameters in error)
9005	Invalid parameter name (associated with Set/GetParameterValues, GetParameterNames, Set/GetParameterAttributes)
9006	Invalid parameter type (associated with SetParameterValues)
9007	Invalid parameter value (associated with SetParameterValues)
9008	Attempt to set a non-writable parameter (associated with SetParameterValues)
9009	Notification request rejected (associated with SetParameterAttributes method).
9010	Download failure (associated with Download or TransferComplete methods).
9011	Upload failure (associated with Upload or TransferComplete methods).
9012	File transfer server authentication failure (associated with Upload, Download, or TransferComplete methods).
9013	Unsupported protocol for file transfer (associated with Upload and Download methods).
9800 – 9899	Vendor defined fault codes

Fig.6 lists the fault codes that can be returned by a server.

Fault code	Description
8000	Method not supported
8001	Request denied (no reason specified)
8002	Internal error
8003	Invalid arguments
8004	Resources exceeded
8005	Retry request
8800 – 8899	Vendor defined fault codes

Protocols' Interoperability



Transaction Session

- Since the CPE is driving the HTTP connection to the ACS, **only** the CPE is responsible for connection **initiation** and **teardown**.
- All transaction sessions **must** begin with an Inform message **from** the **CPE** contained in the initial **HTTP post**.
- CPE is responsible for **terminating** the transaction session, while the ACS could just **consider** the session terminated.
- This specification does not say that there should be a persistent connection between the CPE and the ACS.

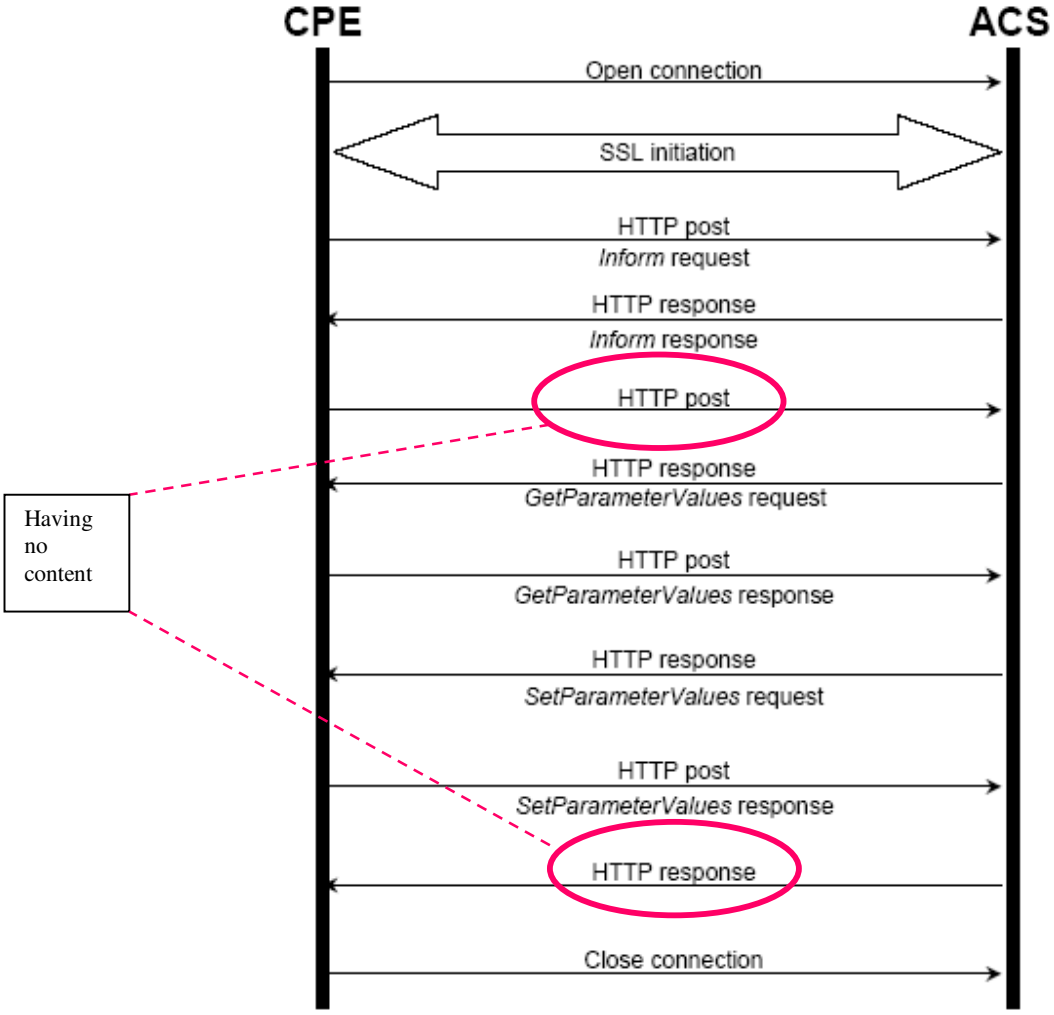


Fig.7

RPCs

The following are the RPPs specified in the TR-069 specification.

Method name	CPE requirement	Server requirement
CPE methods	Responding	Calling
GetRPCMethods	Required	Optional
SetParameterValues	Required	Required
GetParameterValues	Required	Required
GetParameterNames	Required	Required
SetParameterAttributes	Required	Optional
GetParameterAttributes	Required	Optional
AddObject	Required	Optional
DeleteObject	Required	Optional
Reboot	Required	Optional
Download	Required	Required
Upload	Optional	Optional
FactoryReset	Optional	Optional
GetQueuedTransfers	Optional	Optional
ScheduleInform	Optional	Optional
SetVouchers	Optional	Optional
GetOptions	Optional	Optional
Server methods	Calling	Responding
GetRPCMethods	Optional	Required
Inform	Required	Required
TransferComplete	Required	Required
RequestDownload	Optional	Optional
Kicked	Optional	Optional

Inform

A CPE must call the Inform method to initiated a transaction sequence whenever a connection to an ACS is established.

GetRPCMethods

This method may be used by a CPE or Server to discover the set of methods supported by the Server or CPE it is in communication with. This list may include both standard methods (those defined in this specification or a subsequent version) and vendor-specific methods. The receiver of the response MUST ignore any unrecognized methods.

GetParameterNames

This method may be used by a Server to discover the Parameters accessible on a particular CPE.

GetParameterValues

This method may be used by a Server to obtain the value of one or more CPE Parameters.

SetParameterValues

This method may be used by a Server to modify the value of one or more CPE Parameters.

On successful receipt of a SetParameterValues RPC, the CPE **must apply the changes** to each of the specified Parameters **immediately and atomically**. The order of Parameters listed in the ParameterList has no significance—the application of value changes to the CPE must be independent from the order in which they are listed.

A successful response to this RPC should occur **only after the new Parameter values have been successfully applied**. If the CPE requires a reboot before applying the Parameter values, the **CPE must reply before such a reboot**, and thus before the Parameter values have been applied. In this case, the reply must come only after all validation of the request has been completed and the **new values have been appropriately saved** such that they will definitely **be applied immediately following the reboot**.

GetParameterAttributes

This method may be used by a Server to read the attributes associated with one or more CPE Parameter.

SetParameterAttributes

This method may be used by a Server to modify attributes associated with one or more CPE Parameter.

AddObject

This method may be used by the Server to create a new instance of a multi-instance object — **a collection of Parameters** and/or **other objects** for which multiple instances are defined.

The method call takes as an argument the path name of the collection of objects for which a new instance is to be created. For example :

Top.Group.Object

This **path name does not include an instance number** for the object to be created. That **instance number is assigned by the CPE** and returned in the response. Once assigned the instance number of an object cannot be changed and persists until the object is deleted using the DeleteObject method.

After creation, Parameters or sub-objects within the object are refereced by the path name **appended with the instance number**. For example, if the AddObject method returned an instance number of 2, a Parameter within this instance may then be referred to by the path:

Top.Group.Object.2.Parameter

On creation of an object using this method, the Parameters contained within the object are set to their default values and the associated attributes are set to the following :

- Notification is set to zero (notification off)
- AccessList includes all defined entities

After doing this operation, it should save the modified configuration into the flash.

DeleteObject

This method is used to **remove** a particular **instance of an object**. This method call takes as an argument the path name of the object instance including the instance number. For example:

Top.Group.Object.2.

If this method call is successful, the specified instance of this object is subsequently unavailable for access and the CPE may discard the state previously associated with any Parameter or sub-object contained within this instance.

When an object instance is deleted, the instance numbers associated with any other instances of the same collection of objects remain unchanged. Thus, the instance numbers of object instances in a collection might not be consecutive.

After doing this operation, it should save the modified configuration into the flash.

Download

This method may be used by the Server to cause the CPE to download a specified file from the designated location.

TransferComplete

This method informs the server of the completion (either successful or unsuccessful) of a file transfer initiated by an earlier Download or Upload method call.

This must be called **only** when the associated **Download** or **Upload** response indicated that the transfer **had not yet completed at that time** (indicated by a non-zero value of the Status argument in the response). In such cases, it may be called either later in the same session in which the transfer was initiated or in any subsequent session.

When used, this method should be called only after the transfer has **completed** (or **failed**). The criteria used by a CPE to determine when a transfer is considered complete are specific to the implementation of the CPE.

RequestDownload

This method allows the CPE to request a file download from the Server. On reception of this request, the Server may call the Download method to initiate the download.

FactoryReset

This method resets the CPE to its factory default state. This method should be used with extreme caution.

Reboot

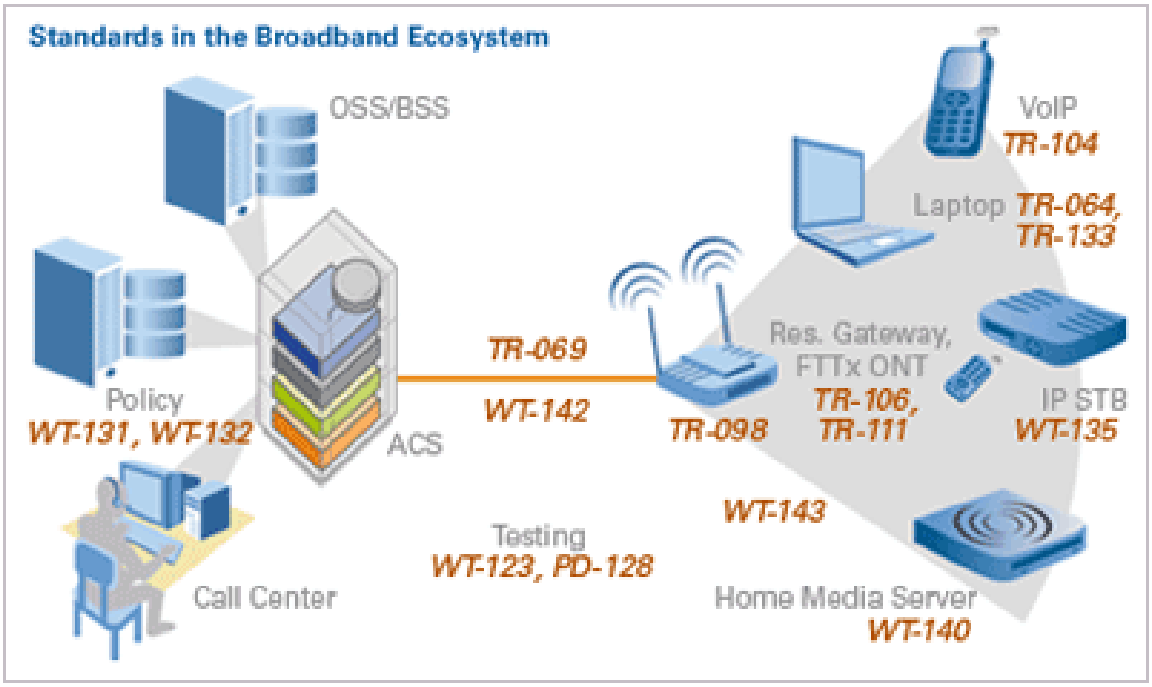
This method causes the CPE to reboot. The CPE MUST send the method response prior to rebooting. It should be used with extreme caution.

This method is primarily intended for **diagnostic** purposes. This method is **not intended for** use by an ACS to **initiate a reboot after setting CPE parameters** or **initiating a download**.

If a CPE requires a reboot in those cases, **it is responsible for initiating that reboot on its own after the termination of the session**.

Because some CPE may not require a reboot under these circumstances, an ACS should not call this method in these cases, which would result in an unnecessary reboot.

TR-069 Relevant TRs



The followings are explanations for all TR-069 related TRs.

TR-069 : CPE WAN Management Protocol
Describes a bi-directional remote management protocol for CPE, intended for communication between CPE and management entity known as an Auto-Configuration Server (ACS). It is intended to support a variety of functionalities to manage CPE, including auto-configuration and dynamic service provisioning, software/firmware image management, status and performance monitoring and diagnostics.
Amendment 1 includes interoperability-related clarifications and errata.
Amendment 2 includes extensions to enable multi-cast and autonomous file downloads, especially important for many video environments.

TR-098 : Internet Gateway Device Data Model
Defines the Internet Gateway Device (IGD) data model for managing various types of broadband CPE, or gateways, using TR-069. Includes functionality such as managing NAT, DHCP, WAN and LAN interfaces.

TR-104 : Provisioning Parameters for VoIP CPE
Defines the data model required for managing a VoIP-capable device (including terminal adapters and VoIP endpoints) using TR-069.

TR-106 : Data Model Template for TR-069-enabled Devices
Includes both common management objects for TR-069 managed end devices and rules for combining modular functionalities described in various specifications (e.g., how to combine TR-104 and TR-098 object for an ATA embedded in a gateway).

WT-107 : TR-098 Issue 2
Contains additional management objects and parameters for the gateway data model.

TR-111 : Applying TR-069 to Remote Management of Home Networking Devices
Defines extensions to the TR-069 protocol and TR-098 IGD and TR-106 data models to enhance the ability of an ACS to manage devices in customer premises. Describes both a mechanism to enable IGD and end device association and the ability for an ACS to initiate a management session on a device behind a NAT gateway. Note that the content originally specified in TR-111 has been incorporated into the Amendment 1 versions of TR-069, TR-106, and TR-098. PD-128Interoperability Test Plan for TR-069 Plugfests Interoperability test plan used at plugfests for TR-069-capable ACSes and CPE. Defines interoperability tests at for the underlying protocols on which TR-069 is built, the TR-069 methods, as well as real-world application tests.

WT-131 : ACS Northbound Interface Requirements
Requirements for an ACS northbound API to backend OSS/BSS, support and policy systems.

TR-135 : Data Model for a TR-069-enabled STB
Defines the data model required for managing a set-top box (STB) using TR-069, including satellite, cable, terrestrial and IPTV STBs—with or without PVR and other optional functionality.

TR-140 : Data Model for TR-069-enabled Network Attached Storage Device
Defines a TR-069 data model for provisioning and management of devices delivering content storage capabilities. Storage-enabled devices could include NAS, media servers, or PVR-related storage.

WT-142 : Framework for TR-069-enabled PON Devices
Outlines a general framework for applying TR-069 to Optical Network Terminals (ONTs) in a fiber network.

WT-143 : Network Service Provider Initiated Throughput Performance Test
Defines a data model to enable throughput testing for TR-069-enabled CPE.

PD-154 : XML Data Model Definition
Defines XML schema for data models. Provides the architecture of expression of data model specifications in XML as well as a CPE's overall management capabilities in XML.

PD-157TR-069 Data Model Common Objects
Contains additional common management objects and parameters for end devices.

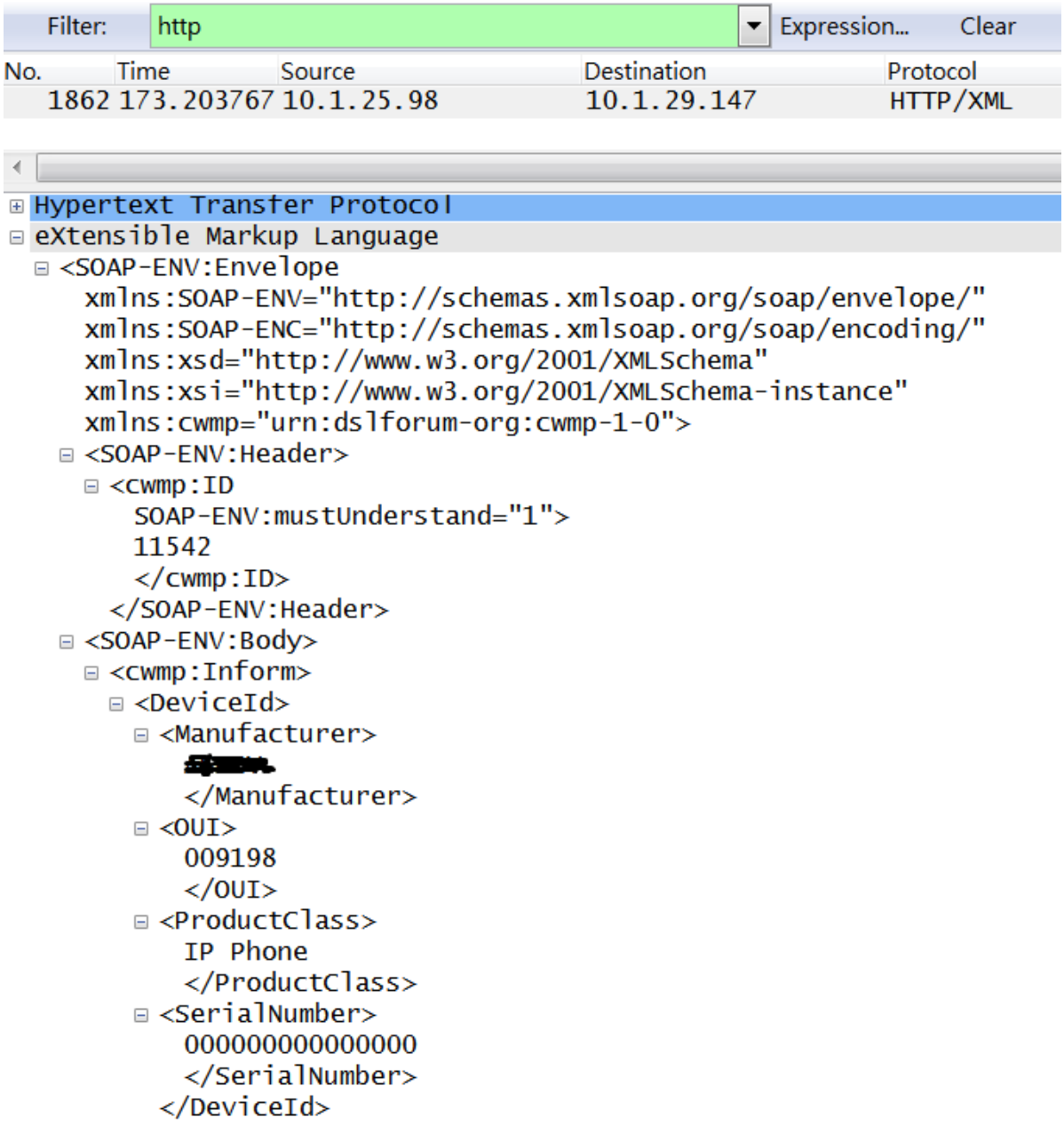
PD-158 : TR-069 Data Model for Email and Browser Services
Defines a data model to enable management of device email and browser services.

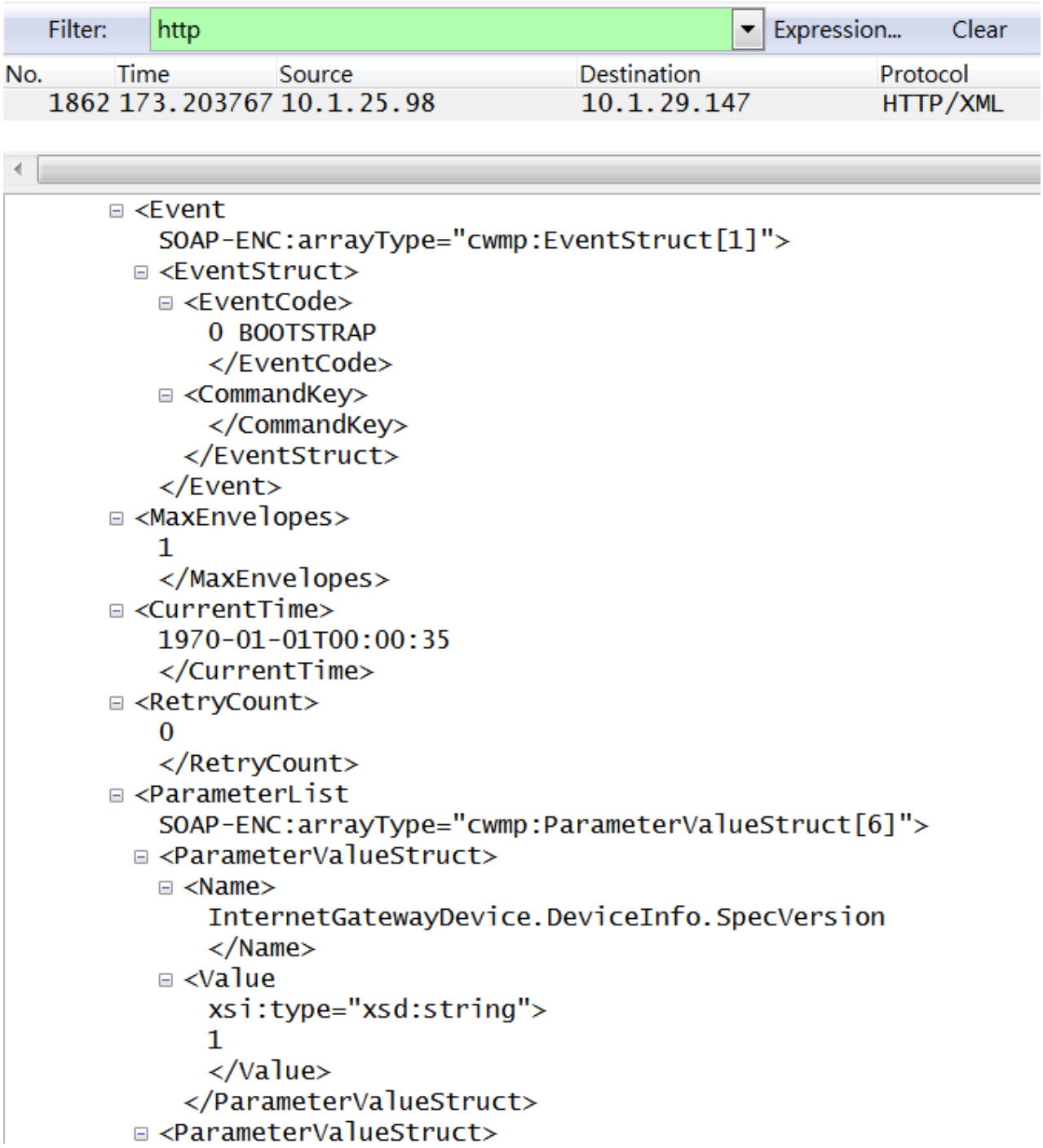
References

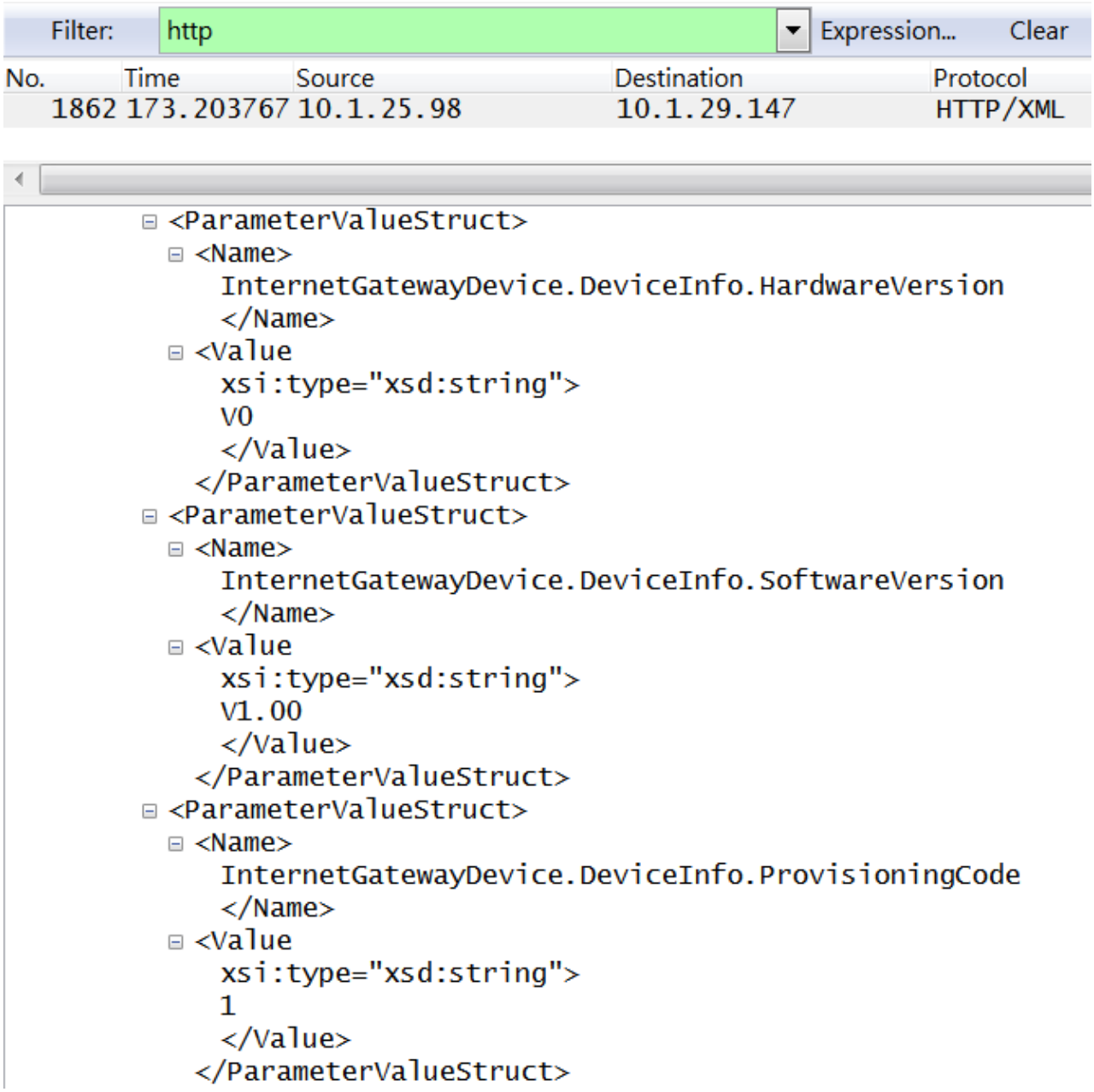
1. DSL Forum, http://www.dslforum.org
2. RFC 2616, Hypertext Transfer Protocol—HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt
3. RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt
4. Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/2000/NOTE-SOAP-20000508
5. Orginationally Unique Identifiers (OUIs), http://standards.ieee.org/faqs/OUI.html
6. The SSL Protocol, Version 3.0, http://www.netscape.com/eng/ssl3/draft302.txt
7. RFC 2246, The TLS Protocol, Version 1.0, http://www.ietf.org/rfc/rfc2246.txt
8. RFC 2132, DHCP Options and BOOTP Vendor Extensions, http://www.ietf.org/rfc/rfc2132.txt
9. Technical Report 037(TR-037) : Auto-Configuration for the Conncetion Between the DSL Broadband Network termination(B-NT) and the Network using ATM, DSL Forum, March 2001.
10. Technical Report 069(TR-069) : CPE WAN Management Protocol, Jeff Bernstein, Tim Spets, May 2004.

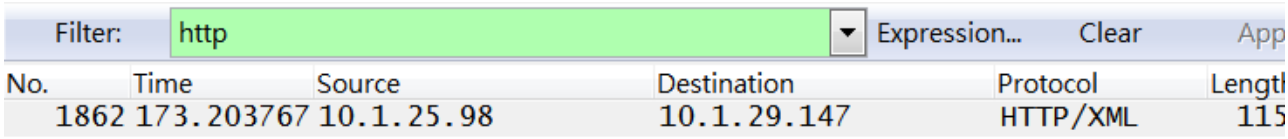
Appendix: Inform message Ethereal PCAP

Captured on **2005-12-26**




Filter:
http
Expression...
Clear
No.
Time
Source
Destination
Protocol
1862 173.203767 10.1.25.98
10.1.29.147
HTTP/XML
<Event
SOAP-ENC:arrayType="cwmp:EventStruct[1]">
<EventStruct>
<EventCode>
0 BOOTSTRAP
</EventCode>
<CommandKey>
</CommandKey>
</EventStruct>
</Event>
<MaxEnvelopes>
1
</MaxEnvelopes>
<CurrentTime>
1970-01-01T00:00:35
</CurrentTime>
<RetryCount>
0
</RetryCount>
<ParameterList
SOAP-ENC:arrayType="cwmp:ParameterValueStruct[6]">
<ParameterValueStruct>
<Name>
InternetGatewayDevice.DeviceInfo.SpecVersion
</Name>
<Value
xsi:type="xsd:string">
1
</Value>
</ParameterValueStruct>
<ParameterValueStruct>


Filter:
http
Expression...
Clear
No.
Time
Source
Destination
Protocol
1862 173.203767 10.1.25.98
10.1.29.147
HTTP/XML
<ParameterValueStruct>
<Name>
InternetGatewayDevice.DeviceInfo.HardwareVersion
</Name>
<Value
xsi:type="xsd:string">
V0
</Value>
</ParameterValueStruct>
<ParameterValueStruct>
<Name>
InternetGatewayDevice.DeviceInfo.SoftwareVersion
</Name>
<Value
xsi:type="xsd:string">
V1.00
</Value>
</ParameterValueStruct>
<ParameterValueStruct>
<Name>
InternetGatewayDevice.DeviceInfo.ProvisioningCode
</Name>
<Value
xsi:type="xsd:string">
1
</Value>
</ParameterValueStruct>




```
            </ParameterValueStruct>
          <ParameterValueStruct>
            <Name>
              InternetGatewayDevice.DeviceInfo.ProvisioningCode
              </Name>
            <Value
              xsi:type="xsd:string">
              1
              </Value>
            </ParameterValueStruct>
          <ParameterValueStruct>
            <Name>
              InternetGatewayDevice.ManagementServer.ConnectionRequestURL
              </Name>
            <Value
              xsi:type="xsd:string">
              http://10.1.25.98:30005/
              </Value>
            </ParameterValueStruct>
          <ParameterValueStruct>
            <Name>
              InternetGatewayDevice.ManagementServer.ParameterKey
              </Name>
            <Value
              xsi:type="xsd:string">
              </Value>
            </ParameterValueStruct>
          </ParameterList>
        </cwmp:Inform>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```