WANCableLinkConfig:1 Service Template Version 1.01

For UPnP™ Version 1.0
Status: Standardized DCP
Date: November 12, 2001

Authors	Company
Matthew Schmitz, Waseem Siddiqi	Cisco Systems, Inc.

Contents

List of Tables

1. Overview and Scope

This service definition is compliant with the UPnP Device Architecture version *1.0*.

This service-type encapsulates physical and link layer properties that are specific to a cable connection used for Internet access. These properties are specific to a cable interface but are common across the different instances of ***WANIPConnection*** service. Please see the Deployment Scenario section for more detail.

The service is OPTIONAL if there is a cable modem WAN interface in the Internet Gateway.
It is specified in
urn:schemas-upnp-org:device:*WANConnectionDevice* in
urn:schemas-upnp-org:device:*WANDevice* under the root device
urn:schemas-upnp-org:device:*InternetGatewayDevice.*

The ***WANDevice*** also provides a ***WANCommonInterfaceConfig*** service that encapsulates Internet access properties common to a WAN Interface.

1.1. Change Log

Changes from *WANCableInterfaceConfig:0.1*
- Added 'Get' actions per Technical Committee recommendation to not use QueryStateVariable for reading state variables.

Changes from *WANCableInterfaceConfig:0.2*
- Removed CableConnectionStatus. Covered by CommonInterfaceConfig
- Removed SignalNoise
- Added a LinkStatus variable
- Added UpstreamPowerLevel
- Fixed GetUpstreamInfo errors
- Added a reference section
- The WANCableInterfaceConfig Service is optional, not mandatory

Changes from *WANCableInterfaceConfig:0.3*
- Renamed the document WANCableLinkConfig in line with overall framework change
- Added BPIEncryptionEnabled, ConfigFile, TFPServer variable from the now defunct WANCableConnection service. Made BPIEncryptionEnabled eventable.
- Renamed LinkStatus to CableLinkConfigState to track DOCSIS connection status on the cable interface
- Added RestartCableInterface, GetBPIEncryptionEnabled, GetCableConfigInfo (renamed from GetHostConfigInfo) from the old WANCableConnection service
- Added a Deployment Scenario section

Changes from *WANCableInterfaceConfig:.0.4*
- Removed RestartCableInterface action
- Changed Boolean values to 1 and 0.
- Removed white spaces from XML section.
- Changed default value for empty strings to tags with no element values
- Removed spaces from strings in AllowedValueLists

Changes from *WANCableInterfaceConfig:.0.5*
- BPIEncryptionEnabled is not eventable.
- Added cable minimum requirements for UPnP compliance in the Theory of Operation section
- Changed default value for empty strings to tags with no element values
- Modified names of formal parameters of actions to be different from 'Related State Variable'.
- Removed 'retval' and empty defaultvalue tags from the XML specification.
- Added LinkType variable to be extensible and consistent with other LinkConfig specs

Changes from *WANCableInterfaceConfig:.0.51*

- Updated to service template v1.01.
- Changed action set to make individual actions for ones that deal with optional variables.
- Renamed GetCableLinkConfigState action name to GetCableLinkConfigInfo (includes LinkType).
- Added error code 501 (Action Failed) for optional actions.
- Changed service from Required to Optional

Changes from *WANCableInterfaceConfig:.0.8*

- Removed default values for SST variables and updated XML template accordingly.
- Deleted Vendor Defined rows from allowedValueList tables

Changes from *WANCableInterfaceConfig:.0.81*

- Added XML comment tags to comments text in XML template
- Fixed case typos for UpstreamPowerLevel, UpstreamChannelID
- Added <action> tag for GetTFTPServer in XML template

Changes from *WANCableInterfaceConfig:.0.81*

- Added accessDenied to table 1.1
- Changed allowed values for UpstreamPowerLevel to >= 0
- Deleted some allowedvaluerange sections from the XML template
- Textual clarifications in the theory of operation section

Changes from *WANCableInterfaceConfig:.0.9*

- Corrected action and parameter names regarding UpstreamPowerLevel for case consistency in XML spec
- Changed all occurrences of “Stream” in variables and action names to “stream” for case consistency

Changes from *WANCableInterfaceConfig:.0.99*

- Version updated to reflect 45-day review completion. No other changes to this draft.

Changes from *WANCableInterfaceConfig:.0.99*

- Copyright messages and document status updated.

2. Service Modeling Definitions

2.1. ServiceType

The following service type identifies a service that is compliant with this template:

urn:schemas-upnp-org:service:*WANCableLinkConfig:1*.

2.2. State Variables

Table 1: State Variables

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
`CableLinkConfigState`	R	string	See Table 1.1	Not specified	N/A
`LinkType`	R	string	See Table 1.2	Not specified	N/A
`DownstreamFrequency`	O	ui4	>=0	Undefined - Depends on Service Provider	N/A
`DownstreamModulation`	O	string	See Table 1.3	Empty string (Depends on Service Provider)	N/A
`UpstreamFrequency`	O	ui4	>=0	Empty string (Depends on Service Provider)	N/A
`UpstreamModulation`	O	string	See Table 1.4	Empty string (Depends on Service Provider)	N/A
`UpstreamChannelID`	O	ui4	>=0	Undefined - Depends on Service Provider	N/A
`UpstreamPowerLevel`	O	ui4	>=0	Undefined - Depends on Service Provider	N/A
`BPIEncryptionEnabled`	O	boolean	1, 0	Not specified	N/A

Variable Name	Req. or Opt.[1]	Data Type	Allowed Value [2]	Default Value [2]	Eng. Units
`ConfigFile`	O	string	N/A	Empty string	N/A
`TFTPServer`	O	string	N/A	Empty string	N/A
Non-standard state variables implemented by a UPnP vendor go here.	*X*	*TBD*	*TBD*	*TBD*	*TBD*

[1] R = Required, O = Optional, X = Non-standard.

[2] Values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, you must reference a specific instance of an appropriate table below.

NOTE: Default values are not specified in the DCP. A vendor may however choose to provide default values for SST variables where appropriate.

Table 1.1: allowedValueList for `CableLinkConfigState`

Value	Req. or Opt. [3]
notReady	R
dsSyncComplete	R
usParamAcquired	R
rangingComplete	R
ipComplete	R
todEstablished	R
paramTransferComplete	R
registrationComplete	R
operational	R
accessDenied	R

[3]

Table 1.2: allowedValueList for `LinkType`

Value	Req. or Opt.
Ethernet	R

Table 1.3: allowedValueList for `DownstreamModulation`

Value	Req. or Opt.
64QAM	R
256QAM	R

Table 1.4: allowedValueList for `UpstreamModulation`

Value	Req. or Opt.
QPSK	R
16QAM	R

2.2.1. `CableLinkConfigState`

This variable represents the current status of the DOCSIS cable connection. Possible string values are:

- ***notReady***: This is the default value and indicates that other variables in the service table are not in a valid state.
- ***dsSyncComplete**:* The cable interface of the Internet Gateway has successfully locked on to the downstream channel.
- ***usParamAcquired**:* The upstream parameters have been received by the cable interface.
- ***rangingComplete**:* The cable interface has completed ranging and all automatic adjustments.
- ***ipComplete***: The cable interface has received its IP address from the DHCP server.
- ***todEstablished***: The cable interface has received the IP address for the TOD server.
- ***paramTransferComplete***: The cable interface has downloaded its operational parameters
- ***registrationComplete***: The cable interface has registered with the CMTS.
- ***operational***: The cable interface is now operational.
- ***accessDenied***: The cable interface of the Internet Gateway has been set in a deny state by the ISP.

2.2.2. `LinkType`

This variable indicates the type of Cable Link used for connection to the Internet. It cannot be set by a UPnP control point and is a read-only variable.

2.2.3. `DownstreamFrequency`

This variable specifies the center frequency for the downstream channel to be used by the modem. It cannot be set by a UPnP control point and is a read-only variable.

2.2.4. `DownstreamModulation`

This variable indicates the modulation type associated with the downstream channel. It cannot be set by a UPnP control point and is a read-only variable.

2.2.5. `UpstreamFrequency`

This variable specifies the center frequency for the upstream channel to be used by the modem. It cannot be set by a UPnP control point and is a read-only variable.

2.2.6. `UpstreamModulation`

This variable indicates the modulation type associated with the upstream channel. It cannot be set by a UPnP control point and is a read-only variable.

2.2.7. `UpstreamChannelID`

This variable specifies channel ID for the upstream channel to be used by the modem. It cannot be set by a UPnP control point and is a read-only variable.

2.2.8. `UpstreamPowerLevel`

This variable indicates the upstream power level used by the modem in dBmV. It cannot be set by a UPnP control point and is a read-only variable.

2.2.9. `BPIEncryptionEnabled`

This variable indicates whether BPI encryption is enabled on the cable interface. It cannot be set by a UPnP control point and is a read-only variable.

2.2.10. `ConfigFile`

This variable indicates the name of the configuration file used.

2.2.11. `TFTPServer`

This variable indicates the IP Address of the TFTP server.

2.2.12.Relationships Between State Variables

The variables in the SST have no dependencies or relationship other than what is mandated by relevant Cable modem standards and protocols.

2.3. Eventing and Moderation

Table 2: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
`CableLinkConfigState`	No	No	N/A	N/A	N/A
`LinkType`	No	No	N/A	N/A	N/A
`DownstreamFrequency`	No	No	N/A	N/A	N/A
`DownstreamModulation`	No	No	N/A	N/A	N/A
`UpstreamFrequency`	No	No	N/A	N/A	N/A

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
`UpstreamModulation`	No	No	N/A	N/A	N/A
`UpstreamChannelID`	No	No	N/A	N/A	N/A
`UpstreamPowerLevel`	No	No	N/A	N/A	N/A
`TFTPServer`	No	No	N/A	N/A	N/A
`ConfigFile`	No	No	N/A	N/A	N/A
`BPIEncryptionEnabled`	No	No	N/A	N/A	N/A
Non-standard state variables implemented by an UPnP vendor go here.	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

2.3.1. Event Model

None of the variables are evented.

2.4. Actions

Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 3: Actions

Name	Req. or Opt. [1]
`GetCableLinkConfigInfo`	R
`GetDownstreamFrequency`	O
`GetDownstreamModulation`	O
`GetUpstreamFrequency`	O
`GetUpstreamModulation`	O
`GetUpstreamChannelID`	O
`GetUpstreamPowerLevel`	O
`GetBPIEncryptionEnabled`	O
`GetConfigFile`	O
`GetTFTPServer`	O
Non-standard actions implemented by an UPnP vendor go here.	X

[1] R = Required, O = Optional, X = Non-standard.

2.4.1.GetCableLinkConfigInfo

This action retrieves the status of the cable interface, including the Link Type.

2.4.1.1. Arguments

Table 4: Arguments for `GetCableLinkConfigInfo`

Argument	Direction	relatedStateVariable
`NewCableLinkConfigState`	*OUT*	`CableLinkConfigState`
`NewLinkType`	*OUT*	`LinkType`

2.4.1.2. Dependency on State (if any)

2.4.1.3. Effect on State (if any)

None

2.4.1.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13

2.4.2.GetDownstreamFrequency

This action retrieves the center frequency associated with downstream channel.

2.4.2.1. Arguments

Table 5: Arguments for `Get GetDownstreamFrequency`

Argument	Direction	relatedStateVariable
`NewDownstreamFrequency`	*OUT*	`DownstreamFrequency`

2.4.2.2. Dependency on State (if any)

2.4.2.3. Effect on State (if any)

None

2.4.2.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.3. `GetDownstreamModulation`

This action retrieves the modulation type associated with downstream channel.

2.4.3.1. Arguments

Table 6: Arguments for `GetDownstreamModulation`

Argument	Direction	relatedStateVariable
`NewDownstreamModulation`	*OUT*	`DownstreamModulation`

2.4.3.2. Dependency on State (if any)

2.4.3.3. Effect on State

None

2.4.3.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.4.`GetUpstreamFrequency`

This action retrieves the center frequency associated with the upstream channel**.**

2.4.4.1. Arguments

Table 7: Arguments for `GetUpstreamFrequency`

Argument	Direction	relatedStateVariable
`NewUpstreamFrequency`	*OUT*	`UpstreamFrequency`

2.4.4.2. Dependency on State (if any)

2.4.4.3. Effect on State

None

2.4.4.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.5. `GetUpstreamModulation`

This action retrieves the modulation type associated with the upstream channel.

2.4.5.1. Arguments

Table 8: Arguments for `GetUpstreamModulation`

Argument	Direction	relatedStateVariable
`NewUpstreamModulation`	*OUT*	`UpstreamModulation`

2.4.5.2. Dependency on State (if any)

2.4.5.3. Effect on State

None

2.4.5.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.6. `GetUpstreamChannelID`

This action retrieves the channel ID associated with the upstream channel.

2.4.6.1. Arguments

Table 9: Arguments for `GetUpstreamChannelID`

Argument	Direction	relatedStateVariable
`NewUpstreamChannelID`	*OUT*	`UpstreamChannelID`

2.4.6.2. Dependency on State (if any)

2.4.6.3. Effect on State

None

2.4.6.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.7. `GetUpstreamPowerLevel`

This action retrieves the power level associated with the upstream channel.

2.4.7.1. Arguments

Table 10: Arguments for `GetUpstreamPowerLevel`

Argument	Direction	relatedStateVariable
`NewUpstreamPowerLevel`	*OUT*	`UpstreamPowerLevel`

2.4.7.2. Dependency on State (if any)

2.4.7.3. Effect on State

None

2.4.7.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.8. `GetBPIEncryptionEnabled`

This action retrieves a value that indicates whether BPI encryption is currently enabled on the cable interface.

2.4.8.1. Arguments

Table 11: Arguments for `GetBPIEncryptionEnabled`

Argument	Direction	relatedStateVariable
`NewBPIEncryptionEnabled`	*OUT*	`BPIEncryptionEnabled`

2.4.8.2. Dependency on State (if any)

2.4.8.3. Effect on State

None

2.4.8.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.9.`GetConfigFile`

This action retrieves the name of the configuration file used for the gateway by the CMTS.

2.4.9.1. Arguments

Table 12: Arguments for `GetConfigFile`

Argument	Direction	relatedStateVariable
`NewConfigFile`	OUT	`ConfigFile`

2.4.9.2. Dependency on State (if any)

2.4.9.3. Effect on State

None

2.4.9.4. Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.10.GetTFTPServer

This action retrieves IP Address of the TFTP server used for the Gateway by the CMTS.

2.4.10.1. Arguments

Table 13: Arguments for `GetTFTPServer`

Argument	Direction	relatedStateVariable
`NewTFTPServer`	OUT	`TFTPServer`

2.4.10.2.Dependency on State (if any)

2.4.10.3.Effect on State

None

2.4.10.4.Errors

errorCode	errorDescription	Description
402	Invalid args	See Table 2.4.13
501	Action Failed	See Table 2.4.13

2.4.11.Non-Standard Actions Implemented by a UPnP Vendor

To facilitate certification, non-standard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for non-standard actions (see the section on Description).

2.4.12.Relationships Between Actions

All actions defined are for querying state variables, and have no specific relationship between them.

2.4.13.Common Error Codes

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 14: Common Error Codes

errorCode	errorDescription	Description
401	Invalid Action	See UPnP Device Architecture section on Control.
402	Invalid Args	One of following: not enough IN arguments, too many IN arguments, no IN argument by that name, one or more IN arguments are of the wrong data type. See UPnP Device Architecture section on Control.
404	Invalid Var	See UPnP Device Architecture section on Control.
501	Action Failed	May be returned in current state if service prevents invoking of that action. See UPnP Device Architecture section on Control.
600-699	TBD	Common action errors. Defined by UPnP Forum Technical Committee.
701-799		Common action errors defined by the UPnP Forum working committees.
800-899	*TBD*	*(Specified by UPnP vendor.)*

2.5. Theory of Operation

There are certain minimum requirements that a gateway device with a cable interface needs to support to be able to be UPnP compliant. This also applies to a cable modem that acts as a gateway device. The device needs to expose an IP address to the in-home LAN. Therefore the gateway should be reachable to the control point from the in-home LAN before, during and after initialization and connection procedures are completed with the Cable Modem Termination System (CMTS).

A ***WANDevice*** that has a cable interface MUST implement a ***WANConnectionDevice*** device and optionally the ***WANCableLinkConfig*** service. The SST variables in this service give information on specific properties of a Cable modem used for WAN Internet access.

Please refer to the DOCSIS specifications (http://www.docsis.org/) for more details on the physical and link layer operation of DOCSIS-compliant cable interfaces.

When a ***WANDevice*** and a ***WANConnectionDevice*** is initialized, one instance of ***WANCableLinkConfig*** service (if implemented) will be initialized. In addition, one or more instances of ***WANIPConnection*** service will be initialized depending on the number of configurations supported on the ***WANDevice***. The gateway implementation may choose to initialize more than one instance of ***WANIPConnection*** service even though the cable interface may support only one connection at a time.

The state variables in this service are DOCSIS oriented. However with the exception of `CableLinkConfigState` and `LinkType`, all the other states are optional. `CableLinkConfigState` values are taken from DOCSIS initialization states but it is expected that non-DOCSIS cable modems would have similar states. Non-DOCSIS cable interfaces can also be supported in UPnP through a combination of these variables and other optional states and use of vendor extensions means that non-DOCSIS cable interfaces can also be supported in UPnP.

2.5.1. Connection Initiation

When an Internet Gateway has a cable interface, the connection is considered "always on." But each time the Internet Gateway is powered on, the cable interface has to go through an initial configuration and registration. The cable interface has a number of parameters that are passed to it by the CMTS. These parameters and the initialization process are detailed in the DOCSIS standard [1]. Please refer to it for a more detailed theory of operation. At the end of the DOCSIS initialization process the cable interface of the Internet Gateway is online and operational in the basic DOCSIS bridging ("plug and play") mode.

The outline of the DOCSIS initialization process is shown in Figure 1 with brief description provided below:

1. Scan for a downstream channel and establish synchronization with the CMTS.
The cable interface of an Internet Gateway acquires a downstream channel by matching the clock sync signal that is regularly sent out by the CMTS on the downstream channel. The cable interface of the Internet Gateway saves the last operational frequency in non-volatile memory and tries to reacquire the saved downstream channel the next time a request is made.

2. Obtain upstream channel parameters.
The cable interface of the Internet Gateway waits for a message from the CMTS and configures itself for the upstream frequency specified in that message.

3. Start ranging for power adjustments.
As needed, the cable interface adjusts its transmit power levels using the power increment value given by the CMTS in its ranging response message.

Note At this point, the cable interface of the Internet Gateway has established connectivity with the CMTS but is not yet online. The next steps are the configuration required for IP network connectivity. In the future it is assumed that there will be different IP connectivity providers so the architecture is created so that there could be multiple ***WANIPConnection***.

4. Establish IP connectivity.
The cable interface invokes the Dynamic Host Configuration Protocol (DHCP) to establish IP connectivity with the TCP/IP network at the headend. The DHCP server sends a response containing the cable interface's IP address as well as the IP addresses for the default gateway, time of day (TOD) server, and Trivial File Transfer Protocol (TFTP) server, and the DOCSIS configuration file to be downloaded.

5. Establish the time of day.
The cable interface accesses the TOD server for the current date and time, which is used to create time stamps for logged events.

6. Transfer operational parameters.
Using TFTP, the cable interface downloads the specified DOCSIS configuration file and configures itself for the appropriate parameters. The DOCSIS configuration file defines the operating mode of the cable interface, such as the provisioned downstream and upstream service assignments, including assigned frequencies, data rates, modulation schemes, Class of Service, Quality of Service and other parameters.

7. Perform registration.
The cable interface completes its secondary ranging and is then online, passing data between the HFC network and the PCs and other CPE devices that are connected to the Internet Gateway.

8. Comply with baseline privacy.
If baseline privacy (BPI) is configured and enabled on both the cable interface and CMTS, the cable interface and CMTS negotiate the appropriate encryption/decryption parameters and exchange keys for privacy. After encryption is enabled, all information sent within Ethernet packets is encrypted to prevent interception or modification by an unauthorized party.

9. *Enter in operational state.*

As soon as the cable interface has successfully completed the above sequence, it enters operational state.

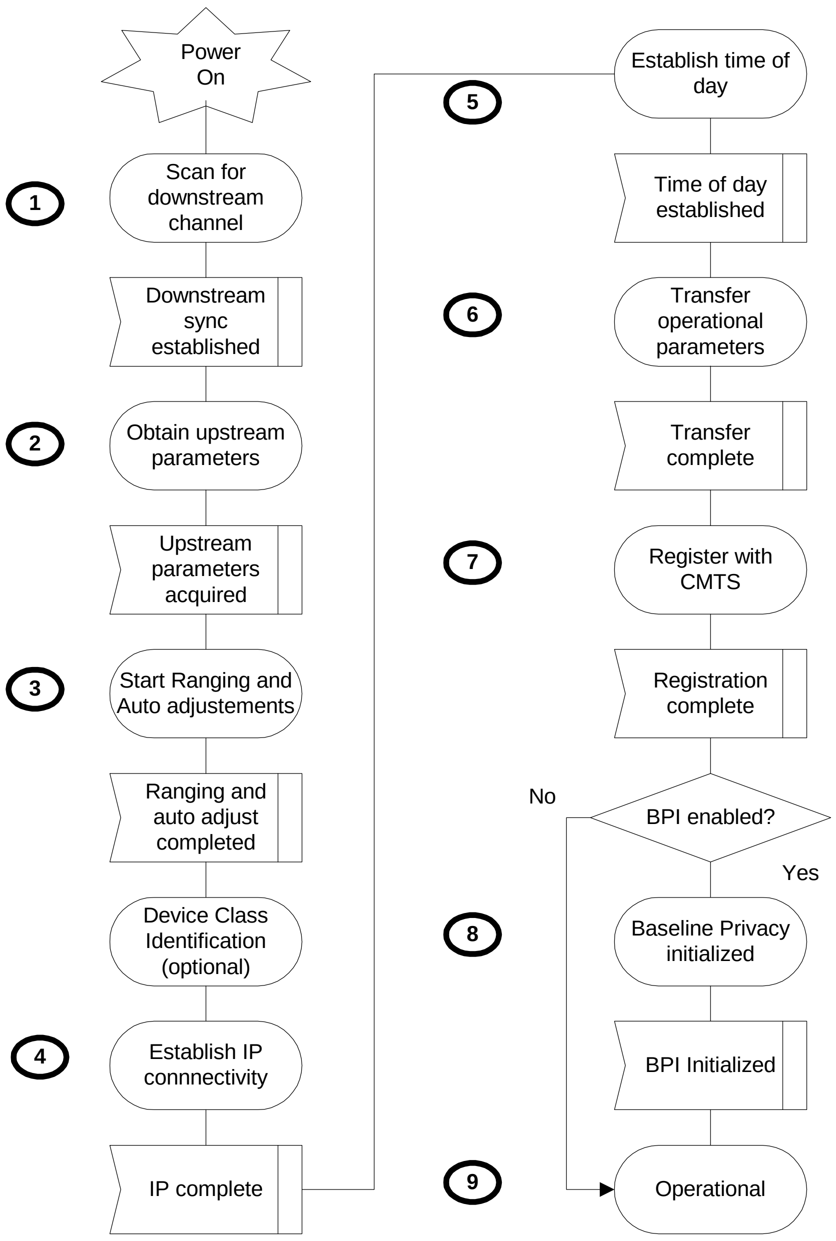
Power On
1
Scan for downstream channel
Downstream sync established
2
Obtain upstream parameters
Upstream parameters acquired
3
Start Ranging and Auto adjustements
Ranging and auto adjust completed
Device Class Identification (optional)
4
Establish IP connnectivity
IP complete
5
Establish time of day
Time of day established
6
Transfer operational parameters
Transfer complete
7
Register with CMTS
Registration complete
No
BPI enabled?
Yes
8
Baseline Privacy initialized
BPI Initialized
9
Operational

Figure 1: Cable interface initialization overview

2.5.2. Deployment Scenarios

The following is a guide to implementers on how to model a UPnP Internet Gateway device that involves a cable interface. Only the WAN modeling will be elaborated since the others should be straightforward. Three cases will be elaborated here.

- A Gateway with CM interface

This is a gateway capable of routing with multiple LAN interfaces as well as one or more WAN interfaces, one of which is a cable interface. It will need the following devices and services to model the just the WAN cable interface:

- ***WANDevice*** – one for the cable interface
  - ***WANCommonInterfaceConfig*** service – one corresponding to the ***WANDevice*** above
  - ***WANConnectionDevice*** – for the cable interface
    - ***WANCableLinkConfig*** service (optional) – within the ***WANConnectionDevice***
    - ***WANIPConnection*** service – one within the ***WANConnectionDevice***

- Cable Modem as the Internet Gateway

This is a regular cable modem, most probably DOCSIS compliant. It does Ethernet bridging and has to have an IP stack to be UPnP compliant. It has to expose an IP address to the in-home LAN. Cable modem implementation vary, so it is important to have a mechanism for the control point on the in-home LAN to talk to the cable modem both before and after connecting to the CMTS, in order for the cable modem to be UPnP compliant. If this is present, the Internet Gateway WAN model is exactly the same as the first case.

- Cable modem connected to an Internet Gateway PC

In this case, the PC is acting as the Internet Gateway which is using its Ethernet interface to connect to the CM and hence to the Internet. The cable modem plays no part in the PC's Internet Gateway implementation. The PC is using its Ethernet interface as the WAN connection and so it should be modeled as:

- ***WANDevice*** – one for the Ethernet interface
  - ***WANCommonInterfaceConfig*** service – one corresponding to the ***WANDevice*** above
  - ***WANConnectionDevice*** – for the Ethernet interface
    - ***WANEthernetLinkConfig*** service (optional) – within the ***WANConnectionDevice***
    - ***WANIPConnection*** service – one within the ***WANConnectionDevice***

3. XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
  <action>
    <name>GetCableLinkConfigInfo</name>
      <argumentList>
        <argument>
          <name>NewCableLinkConfigState</name>
          <direction>out</direction>
       <relatedStateVariable>CableLinkConfigState</relatedStateVariable>
        </argument>
        <argument>
          <name>NewLinkType</name>
          <direction>out</direction>
          <relatedStateVariable>LinkType</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetDownstreamFrequency</name>
      <argumentList>
        <argument>
          <name>NewDownstreamFrequency</name>
          <direction>out</direction>
        <relatedStateVariable>DownstreamFrequency</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetDownstreamModulation</name>
      <argumentList>
        <argument>
          <name>NewDownstreamModulation</name>
          <direction>out </direction>
       <relatedStateVariable>DownstreamModulation</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
    <name>GetUpstreamFrequency</name>
      <argumentList>
        <argument>
          <name>NewUpstreamFrequency</name>
          <direction>out</direction>
          <relatedStateVariable>UpstreamFrequency</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
```

```
<name>GetUpstreamModulation</name>
  <argumentList>
    <argument>
      <name>NewUpstreamModulation</name>
      <direction>out</direction>
     <relatedStateVariable>UpstreamModulation</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
<name>GetUpstreamChannelID</name>
  <argumentList>
    <argument>
      <name>NewUpstreamChannelID</name>
      <direction>out</direction>
      <relatedStateVariable>UpstreamChannelID</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
<name>GetUpstreamPowerLevel</name>
  <argumentList>
    <argument>
      <name>NewUpstreamPowerLevel</name>
      <direction>out</direction>
     <relatedStateVariable>UpstreamPowerLevel</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
<name>GetBPIEncryptionEnabled</name>
  <argumentList>
    <argument>
      <name>NewBPIEncryptionEnabled</name>
      <direction>out</direction>
   <relatedStateVariable>BPIEncryptionEnabled</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
<name>GetConfigFile</name>
  <argumentList>
    <argument>
      <name>NewConfigFile</name>
      <direction>out</direction>
      <relatedStateVariable>ConfigFile</relatedStateVariable>
    </argument>
   </argumentList>
</action>
<action>
<name>GetTFTPServer</name>
  <argumentList>
    <argument>
      <name>NewTFTPServer</name>
      <direction>out</direction>
      <relatedStateVariable>TFTPServer</relatedStateVariable>
```

```
        </argument>
       </argumentList>
    </action>
    <!-- Declarations for other actions added by UPnP vendor (if any) go
here -->
  </actionList>
  <serviceStateTable>
  <stateVariable sendEvents="no">
      <name>CableLinkConfigState</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>notReady</allowedValue>
        <allowedValue>dsSyncComplete</allowedValue>
        <allowedValue>usParamAcquired</allowedValue>
        <allowedValue>rangingComplete</allowedValue>
        <allowedValue>ipComplete</allowedValue>
        <allowedValue>todEstablished</allowedValue>
        <allowedValue>paramTransferComplete</allowedValue>
        <allowedValue>registrationComplete</allowedValue>
        <allowedValue>operational</allowedValue>
        <allowedValue>accessDenied</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>LinkType</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>Ethernet</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>DownstreamFrequency</name>
      <dataType>ui4</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>DownstreamModulation</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>64QAM</allowedValue>
        <allowedValue>256QAM</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>UpstreamFrequency</name>
      <dataType>ui4</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>UpstreamModulation</name>
      <dataType>string</dataType>
      <allowedValueList>
        <allowedValue>QPSK</allowedValue>
        <allowedValue>16QAM</allowedValue>
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>UpstreamChannelID</name>
      <dataType>ui4</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>UpstreamPowerLevel</name>
      <dataType>ui4</dataType>

    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ConfigFile</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>TFTPServer</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>BPIEncryptionEnabled</name>
      <dataType>boolean</dataType>
    </stateVariable>
    <!-- Declarations for other state variables added by UPnP vendor (if
any) go here -->
  </serviceStateTable>
</scpd>
```

4. Test

No semantic tests have been defined for this service.

Change History

Change Log for Version 1.0 (10-4-00)

- Revised the Title Page to call out V1.0 of the Service Template
- Changed to be consistent with Sample Designs released to the Technical Committee
- Service State Table: Variable Descriptions removed from the table and are listed in specific sections following the table.
- Actions: Reformatted the information contained in the Action Table:
    - Added overview entry point.
    - Added an Action Summary Table to specify Required or Optional
    - Added enumerated sections to specify each actions: Arguments, Effect on State, and Errors.