POSITRON
Access Solutions

Overview of JSON-RPC API for G.hn Access Multiplexer (GAM)

January 2021

Publication Information

©2021 Positron Access Solutions

Overview of JSON-RPC API for GAM (G.hn Access Multiplexer)

Publication date: January 2021

Printed in Canada

Published By
Positron Access Solutions
5101 Buchan Street, Suite 220
Montreal, Quebec, Canada
H4P 2R9
Telephone US and Canada: 1-888-577-5254
International: +1-514-345-2220

Product names, other than Positron's, mentioned herein may be trademarks and/or registered trademarks their respective companies.

Disclaimer Notice
Although Positron Access Solutions has made every effort to ensure the accuracy of the information contained herein, this document is subject to change.

CONTENTS

Table of Figures

Chapter 1

About the GAM

1.1 Introduction

The GAM is a Fiber to the Distribution Point (FTTDp) solution typically installed inside a wiring closet. Each GAM comes with either one (1) or two (2) 10 Gigabit SFP+ interfaces to support any type of fiber or PON standards (using an SFP-based or external ONT as required that is compatible with the OLT). These SFP+ ports can further support additional GAM devices in medium to large MDUs and share the fiber backhaul link.

The GAM provides extensive support for provisioning via a JSON-RPC API in addition to the WEB GUI and Command Line Interface (CLI) methods.

Chapter 2 provides an overview of the GAM settings typically provisioned from a centralized provisioning system.

Chapter 3 covers the more widely used G.hn provisioning methods for Bandwidth Profiles, Subscriber Profiles and Endpoint management.

Chapter 4 provides an example of how the JSON-RPC API of the Positron GAM is used to provision bandwidth and subscriber profiles.

Chapter 2

Overview of the JSON-RPC API

2.1 Configuring a G.hn Service

When operating in “Endpoint Aware” mode, a G.hn service consists of an association of a subscriber, a bandwidth plan, an endpoint, and a Client VLAN (CVLAN). If an association is missing, user traffic will not work.

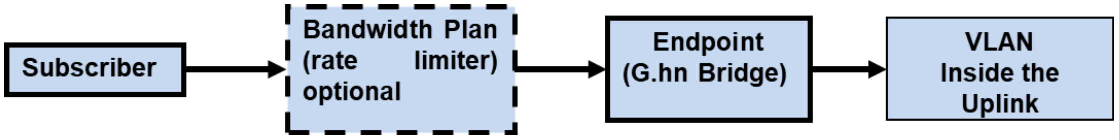


Figure 1: ***Service Configuration – Endpoint Aware mode***

When set to Endpoint-aware mode and upon detecting an Endpoint on a port where it does not match an associated Subscriber, the GAM will automatically assign VLAN 4094 used to force redirection toward a Self-Care Captive Portal attached to the GAM.

By default, a subscriber is assigned to a specific Endpoint (G.hn bridge to Ethernet)

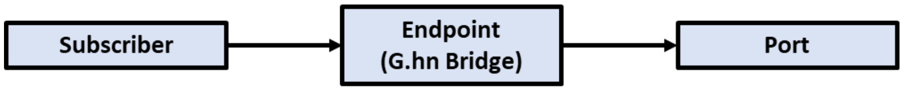


Figure 2: ***Assigning a subscriber to a G.hn endpoint***

As of version 1.4, the GAM now supports “Port Aware” mode. It is now possible to assign a subscriber to a G.hn port and dynamically associate the endpoint detected on the G.hn port to the subscriber profile. This mode of operation is limited to operation over copper (telephone) pairs in MIMO or SISO.

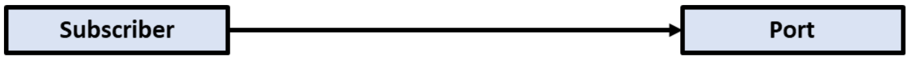
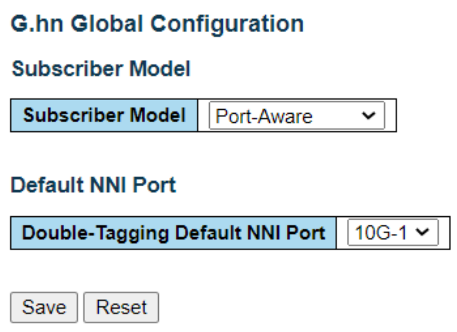


Figure 3: ***Assigning a subscriber to a G.hn port – Port Aware Mode***

IMPORTANT NOTE: *When operating over a point to multipoint COAX infrastructure, a subscriber is always assigned to a specific endpoint since there can be up to sixteen (16) subscribers per coax port.*

2.1.1 Selection G.hn Global Operating Mode (Port Aware or Endpoint Aware)


G.hn Global Configuration

Subscriber Model

Subscriber Model	Port-Aware

Default NNI Port

Double-Tagging Default NNI Port	10G-1

Figure 4: ***Selecting the G.hn Global Operating Mode: Port-Aware or Endpoint-Aware***

Select the **default uplink (NNI) Port** used for double-tagged (Q-in-Q) subscriber traffic. This can be overridden in the Subscriber Profile settings (if required).

Select between **Endpoint-Aware** and **Port-Aware** and then click on the SAVE button to activate the selected mode.

Endpoint-Aware (default mode): Each subscriber is assigned an endpoint (the endpoint 'belongs' to a specific subscriber). The endpoints in turn are bound to a specific G.hn port. That is, you have endpoints belonging to specific subscribers, and bound to specific ports. That is the legacy mode. In this mode, if you have an endpoint that is located on a different port than the one where it is detected, it will be put on hold (quarantined) and will not carry service. That is, the endpoint is assigned to one and only one port. This is useful for example in buildings with pre-installed endpoints. In that case, you assign which subscriber is bound to which endpoint. The process is as follows:

- Configure an endpoint
- Bind the endpoint to a specific port (the endpoint MAC address will be locked to that specific port)
- Configure a Subscriber and assign it to an endpoint.

Port-Aware: Each subscriber is assigned to a specific G.hn port. In this mode, the subscribers are not tied to any specific endpoints. This makes it possible to swap any endpoints, or even send a new or replacement endpoint to a subscriber without previously requiring the endpoint to be defined and configured. In this mode, the endpoints are not assigned to any ports. Instead, service will be given to the subscriber as soon as an endpoint is connected to a port that is assigned to a subscriber. In this mode, the endpoints are 'floating' resources, not tied to any specific subscriber. The subscribers are bound directly to a specific G.hn port. The subscribers are NOT configured to use a specific endpoint. This mode is useful in situations where the endpoint for a subscriber is not yet known. Upon installation of an Endpoint on the G.hn port assigned as part of a subscriber profile, the G.hn link will go up and the endpoint will automatically be added to the subscriber profile. This means that when a subscriber is configured and assigned to a specific port, then any CPE will automatically be added to the configuration upon being detected on the subscriber's port (be it a new installation or a replacement CPE). The endpoint will automatically be configured in the system with a default name the first time it is detected on the system. The process is as follows:

- Configure a subscriber and assign it to a port
- Service flows as soon as the subscriber installs an endpoint on the assigned G.hn port.

IMPORTANT -NOTE: Changing between the two modes of operation is advised only when configuring a GAM device for the first time since these two modes use different configuration paradigms. *Should you decide to change the mode for a GAM with existing subscriber profiles, it will try to find an 'intelligent' match for the new configuration that is as close as possible with the old configuration. It may be possible for this change to be somewhat transparent, depending on the previous configuration. The following behavior is applied when changing the operating mode:*

2.1.1.1 *Changing from Endpoint-Aware to Port-Aware mode*

For each subscriber that is assigned to a specific endpoint, which in turn is assigned to a specific port (whether the endpoint was present or not on that port), the GAM will update the subscriber profile and will assign the current port serving the customer and will maintain the association with the G.hn endpoint. For each other subscribers (whether they are tied to a specific endpoint not already assigned to a specific port, or not yet assigned to a specific endpoint): the GAM will set the port number to “unassigned” in the subscriber profile. A previously configured endpoint is unassigned from its previous port and from its previous subscribers. It becomes a 'floating' resource, not tied to any specific port or subscriber. This means that an endpoint present on the customer assigned port will start carrying service immediately for this subscriber. In the other cases, where you have a subscriber with no endpoint assigned or if the endpoint is not assigned to a specific port, an operator intervention will be required to assign the subscriber to its port.

2.1.1.2 *Changing from Port-Aware to Endpoint-Aware*

For each subscriber that was assigned to a specific port and with an endpoint already carrying traffic for this subscriber, the endpoint and subscriber profiles will be automatically adjusted by the GAM to preserve the association between them. Subscriber profiles not yet tied to a specific endpoint will need to have an endpoint attached before it can bring up a G.hn link and pass traffic. Endpoints not previously assigned to a specific port are set to auto-port mode and will be set to the port where it is eventually seen when the endpoint is detected. For endpoints not yet assigned to any subscriber), an operator intervention is required to assign the endpoint to a specific subscriber and/or to a specific port.

2.1.2 G.hn Port

As of Version 1.4 of the GAM firmware, the definition of a G.hn port is expanded significantly to support the following new capabilities:

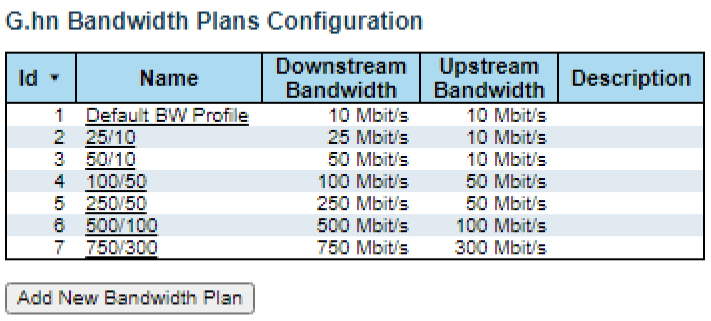
- **G.hn port(s) as link to another GAM:** this is useful to directly connect GAM devices to extend (or replace) a fiber link.
- **Private VLAN (RFC 5517):** using a Private VLAN (RFC 5517) is a convenient way to isolate G.hn ports (and subscribers) from one another.
- **Aggregation:** it is now possible to aggregate two or more G.hn ports to achieve higher bandwidth between GAM devices.

2.1.3 Bandwidth Profiles (Bandwidth Plans)

A bandwidth profile defines the maximum Downstream and Upstream bandwidth assigned to User Profiles. Rate Limiters enforce the bandwidth rules. Rate limiters operate at Layer-2 (Ethernet). For best Speed Test results, the rate limiter should be set 6% to 8% higher to account for the typical IP/TCP overhead of about 6%.

The GAM creates and assigns a default bandwidth plan automatically to any Endpoint not yet attached to a subscriber profile. This default bandwidth plan provides enough bandwidth to support access to the Self-care Captive portal feature (VLAN 4094).

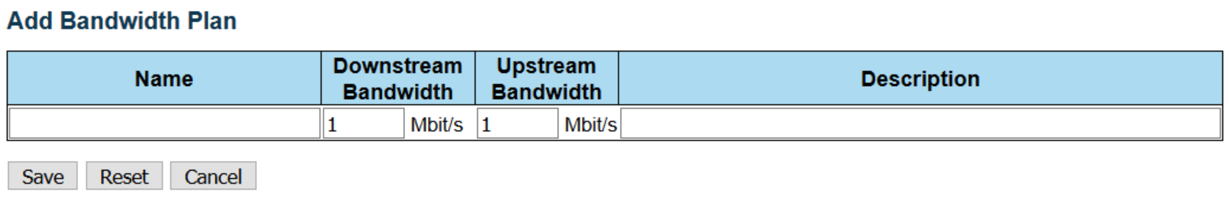
An administrator can create up to 14 additional bandwidth plans.


G.hn Bandwidth Plans Configuration

Id	Name	Downstream Bandwidth	Upstream Bandwidth	Description
1	Default BW Profile	10 Mbit/s	10 Mbit/s	
2	25/10	25 Mbit/s	10 Mbit/s	
3	50/10	50 Mbit/s	10 Mbit/s	
4	100/50	100 Mbit/s	50 Mbit/s	
5	250/50	250 Mbit/s	50 Mbit/s	
6	500/100	500 Mbit/s	100 Mbit/s	
7	750/300	750 Mbit/s	300 Mbit/s	

Figure 5: ***Configuring & Managing Bandwidth Plans***

Click on a Bandwidth to manage it or click on the Add New Bandwidth Plan button to create a new one.


Add Bandwidth Plan

Name	Downstream Bandwidth	Upstream Bandwidth	Description
	1 Mbit/s	1 Mbit/s	

Figure 6: ***Adding a Bandwidth Plan***

Name: This field is mandatory, and the name must be unique among all bandwidth plans. It must contain between 1 and 31 alphanumeric characters.

Bandwidth: Set the Downstream/Upstream bandwidth limits (in Mbps). Acceptable values are:

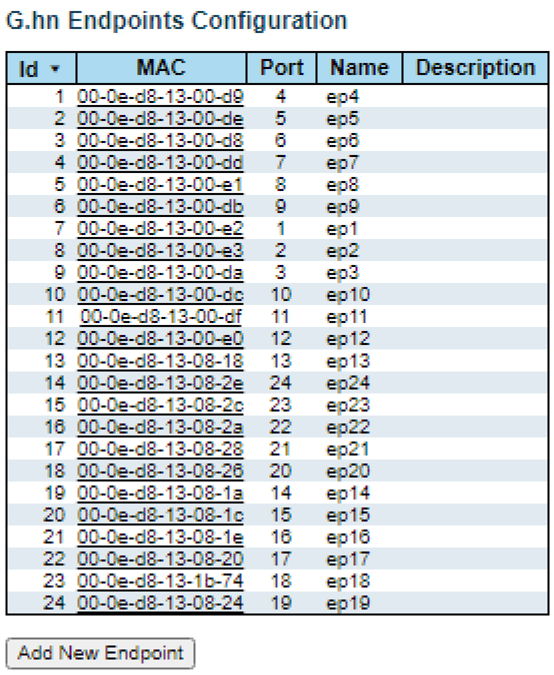
- **MIMO / SISO**: 10 to 1000 Mbps,
- **COAX**: 10 to 800 Mbps. A service over 800Mbps must be set to “unthrottled”.

Description: Use this optional free-form field to add a description of this bandwidth plan. There is no validation for the content of this field, which can hold up to 63 characters.

2.1.4 Endpoint Devices (G.hn Bridge) – Endpoint Aware mode

The GAM device authenticates and authorizes each managed Endpoint device to make sure the overall G.hn infrastructure is secure and delivers the required services to each Subscriber. You can pre-define Endpoints and assign them to a Subscriber profile, or you can select from the list of Endpoint discovered by the GAM.

The following illustrates the list of Endpoint devices for a single Coax Port of the GAM where each port can handle up to 16 subscribers in a Point to Multipoint configuration using standard coax splitters. For copper ports (MIMO or SISO), each port only allows a single Endpoint device.

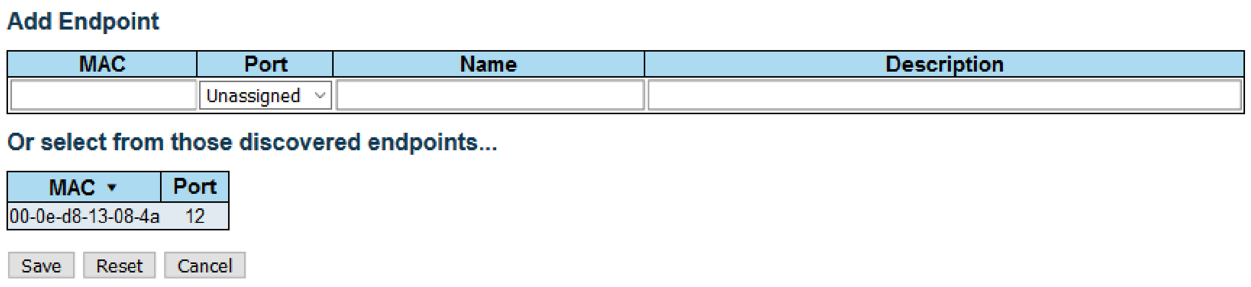


G.hn Endpoints Configuration

Id	MAC	Port	Name	Description
1	00-0e-d8-13-00-d9	4	ep4	
2	00-0e-d8-13-00-de	5	ep5	
3	00-0e-d8-13-00-d8	6	ep6	
4	00-0e-d8-13-00-dd	7	ep7	
5	00-0e-d8-13-00-e1	8	ep8	
6	00-0e-d8-13-00-db	9	ep9	
7	00-0e-d8-13-00-e2	1	ep1	
8	00-0e-d8-13-00-e3	2	ep2	
9	00-0e-d8-13-00-da	3	ep3	
10	00-0e-d8-13-00-dc	10	ep10	
11	00-0e-d8-13-00-df	11	ep11	
12	00-0e-d8-13-00-e0	12	ep12	
13	00-0e-d8-13-08-18	13	ep13	
14	00-0e-d8-13-08-2e	24	ep24	
15	00-0e-d8-13-08-2c	23	ep23	
16	00-0e-d8-13-08-2a	22	ep22	
17	00-0e-d8-13-08-28	21	ep21	
18	00-0e-d8-13-08-26	20	ep20	
19	00-0e-d8-13-08-1a	14	ep14	
20	00-0e-d8-13-08-1c	15	ep15	
21	00-0e-d8-13-08-1e	16	ep16	
22	00-0e-d8-13-08-20	17	ep17	
23	00-0e-d8-13-1b-74	18	ep18	
24	00-0e-d8-13-08-24	19	ep19	

Figure 7: ***Managing Endpoint devices***

Clicking on the Add New Endpoint button takes you to the following screen:



Add Endpoint

MAC	Port	Name	Description
	Unassigned		

Or select from those discovered endpoints...

MAC	Port
00-0e-d8-13-08-4a	12

Figure 8: ***Adding an Endpoint device***

The above page displays a list of Discovered but unconfigured endpoints. These are endpoints detected by the GAM but not yet assigned to a subscriber or to a port. This table is visible only when unconfigured endpoints are detected. Click on the endpoint you wish to configure to select one of the discovered but not yet assigned endpoint.

You will need to enter the following information when adding an endpoint device:

MAC: Set the endpoint MAC address. Enter the MAC address manually or select from the discovered endpoints list.

Port (Endpoint Aware mode only): Set the port that connects to the endpoint. You can set 'Unassigned' to pre-configure a device. However, traffic will not flow until you assign the Endpoint to its specific port. Note: In copper mode (MIMO/SISO), each G.hn port maps to a single Endpoint. In Coax mode, you can assign up to 16 Endpoint devices to a port.

Name: This field is mandatory, and the name must be unique among all endpoints. It must contain between 1 and 31 alphanumeric characters.

Description: Use this optional field to add a free-form description for this Endpoint. There is no validation for the content of this field; it can hold up to 63 characters.

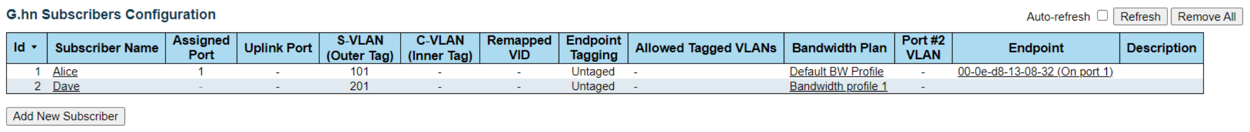
2.1.5 Subscribers

The GAM device relies on Subscriber profiles to control how it delivers High Speed Internet (HSI) services.

As covered earlier, a subscriber is either associated to a specific Endpoint or directly to a port. Each subscriber can be assigned to different VLANs, or all subscribers can be assigned to the same VLAN (data VLAN).

The G.hn protocol will isolate traffic between endpoints when connected to the same port (coax). The GAM will isolate traffic between ports via port isolation or private VLANs (as per RFC 5517). You can then assign an optional Bandwidth Plan to a Subscriber to control how much bandwidth is available at any time.

The following illustrates the list of Subscribers of the GAM along with the Endpoint assigned to each Subscriber.

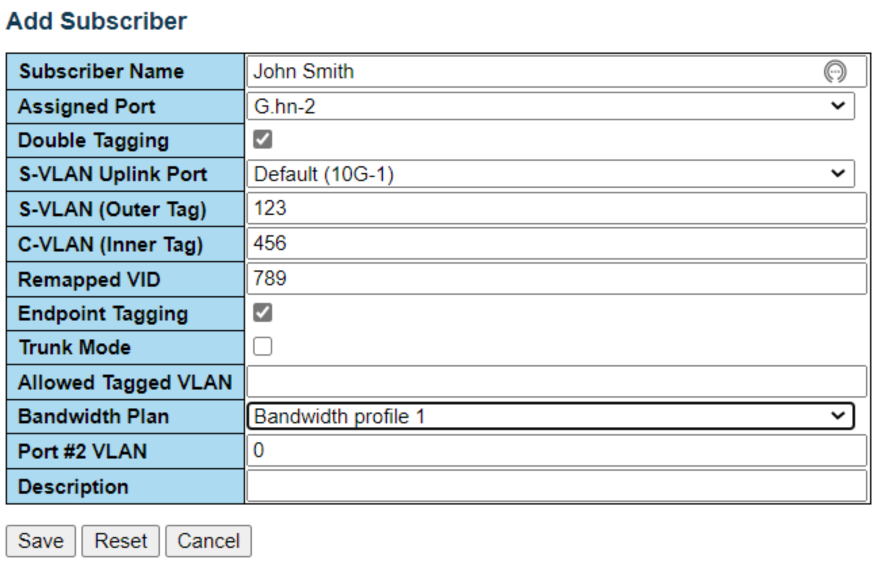


G.hn Subscribers Configuration

Id	Subscriber Name	Assigned Port	Uplink Port	S-VLAN (Outer Tag)	C-VLAN (Inner Tag)	Remapped VID	Endpoint Tagging	Allowed Tagged VLANs	Bandwidth Plan	Port #2 VLAN	Endpoint	Description
1	Alice	1	-	101	-	-	Untaged	-	Default BW Profile	-	00-0e-d8-13-08-32 (On port 1)	
2	Dave	-	-	201	-	-	Untaged	-	Bandwidth profile 1	-		

Figure 9: ***Managing Subscribers***

Click the Add New Subscriber button to get the following screen (the information displayed is adjusted to reflect either Endpoint-aware or Port-aware modes):


Add Subscriber

Subscriber Name	John Smith
Assigned Port	G.hn-2
Double Tagging	✓
S-VLAN Uplink Port	Default (10G-1)
S-VLAN (Outer Tag)	123
C-VLAN (Inner Tag)	456
Remapped VID	789
Endpoint Tagging	✓
Trunk Mode	☐
Allowed Tagged VLAN	
Bandwidth Plan	Bandwidth profile 1
Port #2 VLAN	0
Description	

Figure 10: Adding Subscribers

Subscriber Name: This field is mandatory, and it used as the key whenever accessing the subscriber profile via the JSON-RPC API. This name must be unique among all subscribers. It must contain between 1 and 31 characters.

ASSIGNED PORT (PORT-AWARE Mode Only): Sets the port to which the subscriber is bound to.

DOUBLE TAGGING: enable this setting will allow you to specify the C-VLAN (Inner tag) used with the S-VLAN (Outer tag) to uniquely identify the subscriber traffic.

S-VLAN UPLINK PORT: The uplink (sometimes referred as NNI) port onto which this subscriber traffic is bound. The default value set in the Global parameters can be overridden with another uplink port if required.

S-VLAN (OUTER TAG): This the outer VLAN ID used on the uplink (NNI) port to carry the subscriber traffic. When double tagging is used, the C-VLAN (inner tag) is used as the 2nd VLAN tag.

C-VLAN (INNER Tag): This is the customer (C-TAG) VLAN, this VLAN can be untagged, keep VLAN tag, or remapped to a different VLAN. A G.hn service can use any VLAN between 3 and 4093 except for VLANs 3840 to 3842. The GAM reserves VLANs 1, 2, and 4095 for its internal use. The GAM automatically assigns

VLAN 4094 to non-configured service to force the subscriber to use the Self-Care Captive Portal of the GAM (future capability).

REMAPPED VID: specify the VLAN ID used between the G.hn bridge and the GAM. In other words, the GAM remaps the Ingress user traffic (from the NNI port of the GAM) to this VLAN ID before forwarding to the Endpoint. The reverse action takes place for traffic from the Endpoint to the Internet. The GAM replaces the VLAN ID of the Endpoint with the VLAN ID assigned to the Subscriber on the NNI port by the GAM.

ENDPOINT TAGGING: Enable or disable VLAN tagging at the subscriber endpoint. If disabled, the user traffic is untagged. If enabled, the VLAN tag will be preserved: either the primary VLAN tag or (if set) the remapped VLAN tag.

TRUNK MODE: when enabled, this sets the Ethernet Port #1 of the G.hn Endpoint in VLAN Trunk mode. This is particularly useful when connecting to a Wi-Fi Access Point with Dynamic PSD / VLAN roaming.

ALLOWED TAGGED VLAN: The Allowed tagged VLAN field is a list of VLAN (up to 14) that will be forwarded (untouched). This allows applications such as IPTV to preserve the VLAN and P-BIT information through to the Residential Gateway or set top box.

BANDWIDTH PLAN: Optional Rate limiter associated to the subscriber. The Bandwidth Plan should be created prior of creating the subscriber. If set to "unthrottled", this subscriber will have no bandwidth limit restriction.

PORT #2 VLAN: Some endpoint devices have a 2nd Gigabit Ethernet Port and a VLAN can be assigned for services such as IPTV, VoIP, etc. This applies to devices such as the G1000-M and the G1000-C.

ENDPOINT (Endpoint Aware mode only): The endpoint assigned to this subscriber. We recommend creating the endpoint prior to creating the subscriber. Traffic will only flow upon the association of a subscriber to an endpoint.

DESCRIPTION: The field is optional, free-form description given to this subscriber. There is no validation for the content of this field, apart the number of characters, it can contain up to 63 characters.

2.2 Managing VLANs on the uplink.

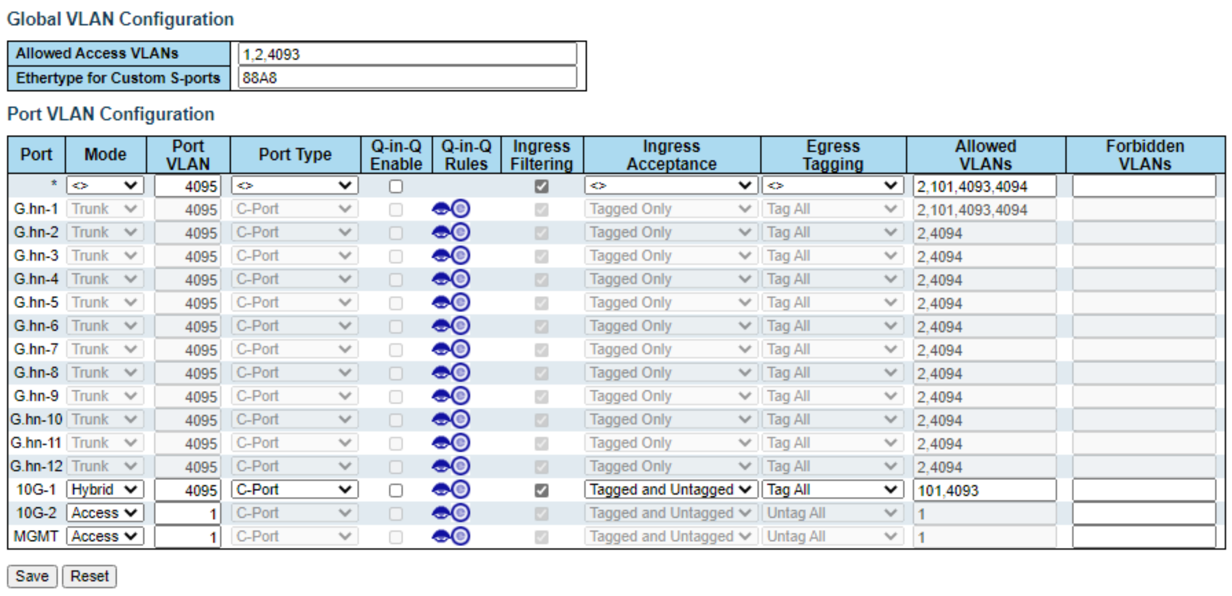
By default, the uplinks of the GAM are set in “trunk” mode and allow all VLANs from 3 to 4094 to go through. In the table below, only the configuration of the 2 uplinks (10G-1 and 10G-2) can be modified. The VLAN table for the G.hn 1 to 12 or 24) is managed by the G.hn subscriber configuration.

A user can decide not to use VLANs on an uplink, the mode must be changed to the following:

- <u>Mode Access</u>: all traffic will be untagged on the egress, and in the ingress all traffic will be tagged (inside de GAM) with the port VLAN.
- <u>Mode Hybrid</u>: in the egress, a specific VLAN (Port VLAN) can be untagged but all other VLAN will stay tagged. In the ingress, the untagged traffic will be tagged with the Port VLAN.

More information below about the different parameters.

Configuration>VLAN>Configuration



Global VLAN Configuration

Allowed Access VLANs	1,2,4093
Ethertype for Custom S-ports	88A8

Port VLAN Configuration

Port	Mode	Port VLAN	Port Type	Q-in-Q Enable	Q-in-Q Rules	Ingress Filtering	Ingress Acceptance	Egress Tagging	Allowed VLANs	Forbidden VLANs
*	<>	4095	<>	☐		✓	<>	<>	2,101,4093,4094	
G.hn-1	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,101,4093,4094	
G.hn-2	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-3	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-4	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-5	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-6	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-7	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-8	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-9	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-10	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-11	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
G.hn-12	Trunk	4095	C-Port	☐		✓	Tagged Only	Tag All	2,4094	
10G-1	Hybrid	4095	C-Port	☐		✓	Tagged and Untagged	Tag All	101,4093	
10G-2	Access	1	C-Port	☐		✓	Tagged and Untagged	Untag All	1	
MGMT	Access	1	C-Port	☐		✓	Tagged and Untagged	Untag All	1	

Figure 11: Global VLAN Configuration table

Let us begin with the **Global VLAN Configuration** settings.

Allowed Access VLANs: This field shows the allowed Access VLANs, for the G.hn ports configured as Access ports. Ports in other modes are members of the VLANs specified in the Allowed VLANs field. By default, VLAN 1 is the only one enabled. More VLANs may be created using an enumeration list syntax where the individual elements are separated by commas. You can specify a range of VLANs with a dash separating the lower and upper bound. The following example sets VLANs 1, 10, 11, 12, 13, 200, and 300: 1, 10-13, 200, 300.

Ethertype for Custom S-ports: Specify the Ethertype/TPID (in hexadecimal) used for Custom S-ports. The setting applies to all of the ports with the **Port Type** set to **S-Custom-Port**.

Port VLAN Configuration

The following settings apply on a Port-by-Port basis. Looking at the **Port VLAN Configuration** table, we see:

Port. This is the logical port number of this row.

Mode: The port mode (default is ACCESS) controls the fundamental behavior of the port in question. A port can be in one of three modes as described below. Whenever you select a particular mode, the configurable fields in that row entry may change (grayed out). Grayed out fields will show the value that the port will get when the mode is applied. The available **MODE** settings are:

Access: use Access ports to connect to end stations. Dynamic features like Voice VLAN may add the port to more VLANs behind the scenes. Access ports have the following characteristics:

- Member of exactly one VLAN, the Port VLAN (a.k.a. Access VLAN), which by default is 1
- Accept untagged and C-tagged frames
- Discard all frames not classified to the Access VLAN
- On egress all frames are transmitted untagged

Trunk: Trunk ports carry traffic on multiple VLANs simultaneously, and usually connect to other switches. Trunk ports have the following characteristics:

- By default, a trunk port is member of all VLANs (1-4095)
- The VLANs for a specific port may be limited by the use of Allowed VLANs for that port
- Frames classified to a VLAN outside of the list of allowed VLANs for that port are discarded
- By default, the GAM tags all frames not classified to the Port VLAN (a.k.a. Native VLAN) on Egress. Frames classified to the Port VLAN do not get C-tagged on Egress

- Egress tagging can be changed to tag all frames, in which case only tagged frames are accepted on Ingress

Hybrid: Hybrid ports resemble trunk ports in many ways. They add additional port configuration features beyond those available for **trunk** ports. These additional characteristics (beyond the ones for **trunk** ports above) include:

- VLAN Awareness: each port is configurable to be VLAN tag unaware, C-tag aware, S-tag aware, or S-custom-tag aware
- Ingress filtering is controllable
- Ingress acceptance of frames and configuration of egress tagging is configured independently

Port VLAN: Determines the port's VLAN ID (a.k.a. PVID). Allowed VLANs are in the range of 1 through 4095, with a default of 1. On **ingress**, frames are classified to the Port VLAN whenever the port is configured as: VLAN unaware, the frame is untagged, or VLAN awareness is enabled on the port and the frame is priority tagged (VLAN ID = 0). On **egress**, frames classified to the Port VLAN are untagged when Egress Tagging configuration is set to untag Port VLAN. The Port VLAN is known as "Access VLAN" for ports in Access mode and as “Native VLAN” for ports in Trunk or Hybrid mode.

Port Type: Ports in hybrid mode allow for changing the port type, that is, whether a frame's VLAN tag is used to classify the frame on **ingress** to a particular VLAN, and if so, which TPID it reacts on. Likewise, on **egress**, the Port Type determines the TPID of the VLAN tag, if a VLAN tag is required. Available **Port Types** are:

- **Unaware:** On ingress, all frames (whether carrying a VLAN tag or not) get classified to the Port VLAN, and any VLAN tags present are not removed on egress.
- **C-Port:** On ingress, frames with a VLAN tag with TPID = 0x8100 are classified to the VLAN ID embedded in the tag. If a frame is untagged or priority tagged, the frame is classified to the Port VLAN. If frames are to be tagged on egress, they will be tagged with a C-tag.
- **S-Port:** On egress, whenever frames must be tagged, they will be tagged with an S-tag. On ingress, frames with a VLAN tag and a TPID = 0x88A8 get classified to the VLAN ID embedded in the tag. Priority-tagged frames are classified to the Port VLAN. If the port is configured to accept “Tagged Only” frames (see Ingress Acceptance below), frames without this TPID are dropped.

- **S-Custom-Port:** On egress, if frames must be tagged, they will be tagged with the custom S-tag. On ingress, frames with a VLAN tag and a TPID equal to the Ethertype configured for Custom-S ports are classified to the VLAN ID embedded in the tag. Priority-tagged frames are classified to the Port VLAN. If the port is configured to accept "Tagged Only" frames (see Ingress Acceptance below), frames without this TPID are dropped.

Q-in-Q Enable: This enables 802.1Q double-tag on hybrid ports that have the port type set to C-Port or S-Port. If set, it will set an inner tag of 0x8100, and an outer tag defined by the port type (0x8100 if port type is set to C-Port or 0x88A8 if the port type is set to S-Port).

Q-in-Q Rules: When Q-in-Q is enabled, click the edit button to enter the rule setting page of the designated interface. Summary about the designated interface will be shown by clicking the view button. You can manage or inspect the rules by using the following buttons:

: List the rules associated with the designated interface.

ⓔ: Allows you to define the rules associated with the designated interface

Ingress Filtering: Hybrid ports allow for flexible ingress filtering. Access and Trunk ports always have ingress filtering enabled. If ingress filtering is enabled (checkbox is checked), frames classified to a VLAN that the port is not a member of are discarded. If ingress filtering is disabled, frames classified to a VLAN that the port is not a member of are accepted and forwarded to the switch engine. However, the port will never transmit frames classified to VLANs that it is not a member of.

Ingress Acceptance: Hybrid ports allow for changing the type of frames accepted on ingress. The available settings are:

- **Tagged and Untagged:** Both tagged and untagged frames are accepted. See Port Type for a description of when a frame is considered tagged.
- **Tagged Only:** Only frames tagged with the corresponding Port Type tag are accepted on ingress.
- **Untagged Only:** Only untagged frames are accepted on ingress. See Port Type for a description of when a frame is considered untagged.

Egress Tagging: Ports in Trunk and Hybrid mode may control the tagging of frames on egress. The available settings are:

- **Untag Port VLAN:** Frames classified to the Port VLAN are untagged. Other frames are transmitted with the relevant tag.
- **Tag All:** All frames, whether classified to the Port VLAN or not, are transmitted with a tag.
- **Untag All:** All frames, whether classified to the Port VLAN or not, are transmitted without a tag. This option is only available for ports in Hybrid mode.

Allowed VLANs: Ports in Trunk and Hybrid mode control the VLANs they can become members of. Access ports can only be member of one VLAN, the Access VLAN. The field's syntax is identical to the syntax used in the Enabled VLANs field. By default, a Trunk or Hybrid port will become a member of all VLANs and is therefore set to 3-4094. An empty field means that the port will not become a member of any VLAN.

Forbidden VLANs: A port may be configured to never become member of one or more VLANs. This is particularly useful to prevent dynamic VLAN protocols like MVRP and GVRP from dynamically adding ports to forbidden VLANs. The syntax is identical to the syntax used in the Enabled VLANs field. An empty field means that the port may become a member of all possible VLANs.

2.3 DHCP Snooping / Option-82

The DHCP agent of the GAM supports DHCP Option 82. It forwards and transfers DHCP messages between DHCP clients and the DHCP server on same or different subnets (Relay mode). The GAM stores the incoming interface IP address in the GIADDR field of the DHCP packet. The DHCP server can use the value of GIADDR field to determine the assigned subnet. For such condition, please make sure the switch configuration of VLAN interface IP address and Port VLAN ID (PVID) correctly.

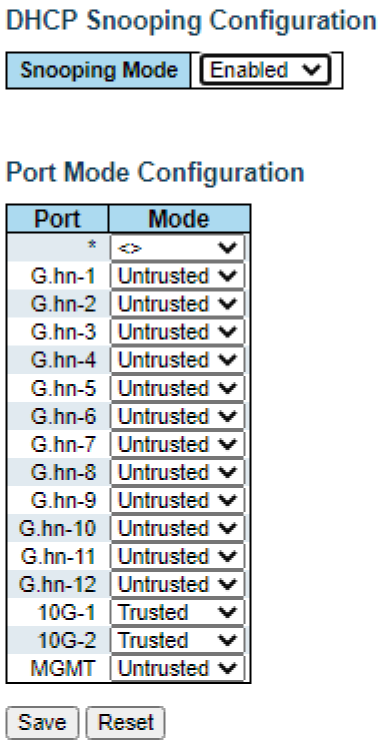
You can configure the **DHCP Snooping** function as follows:

Snooping Mode: Indicates the DHCP snooping mode. Possible modes are:

- **Enabled**: when you enable DHCP snooping, the GAM forwards the DHCP request messages a DHCP Server to trusted ports and only allow reply packets from trusted ports.
- **Disabled**: Disables the DHCP snooping function.

Port Mode Configuration Table

For each port in the table, you can set the mode of a port as **trusted** or **untrusted.**


DHCP Snooping Configuration

Snooping Mode	Enabled

Port Mode Configuration

Port	Mode
*	<>
G.hn-1	Untrusted
G.hn-2	Untrusted
G.hn-3	Untrusted
G.hn-4	Untrusted
G.hn-5	Untrusted
G.hn-6	Untrusted
G.hn-7	Untrusted
G.hn-8	Untrusted
G.hn-9	Untrusted
G.hn-10	Untrusted
G.hn-11	Untrusted
G.hn-12	Untrusted
10G-1	Trusted
10G-2	Trusted
MGMT	Untrusted

Figure 12: Managing DHCP Option 82

You then need to configure the **DHCP Relay** settings as follows:

Relay Mode: select the mode of operation for the DHCP relay function. Possible modes are:

- **Enabled**: DHCP unicast to a specific DHCP server enables the DHCP Relay function where the GAM forwards and transfers DHCP messages between the DHCP clients and the DHCP server on different subnets. DHCP broadcast messages are not flooded on all interfaces for security considerations.
- **Disabled**: Disable the DHCP relay function. DHCP packets will be sent as broadcast packets.

Relay Server: specify the IP address of the DHCP relay server.

Relay Information Mode: when you enable this function, the DHCP Snooping operates as per the Broadband Forum TR-101 specification that dictates how to use Option 82. The Option 82 Circuit ID format is "[Agent Identifier][interface]

[slot]/[port number]". For example, "000ED8-GAM12C-01012973" eth 0/6.0:3 indicates that the DHCP request was issued by Agent 000ED8-GAM12C-01012973 on an Ethernet interface by a GAM at slot 0. It further indicates that the DHCP packet originates from port #6 of the GAM, sent by endpoint #0 to this port, with a VLAN ID of 3. The Option 82 Remote ID value is set to the G.hn subscriber name associated to the G.hn endpoint.

- **Enabled**: the DHCP agent of the GAM inserts the information for option 82 into DHCP messages toward the DHCP server and removes it on DHCP messages to the DHCP client.
- **Disabled**: Disable the DHCP relay function.

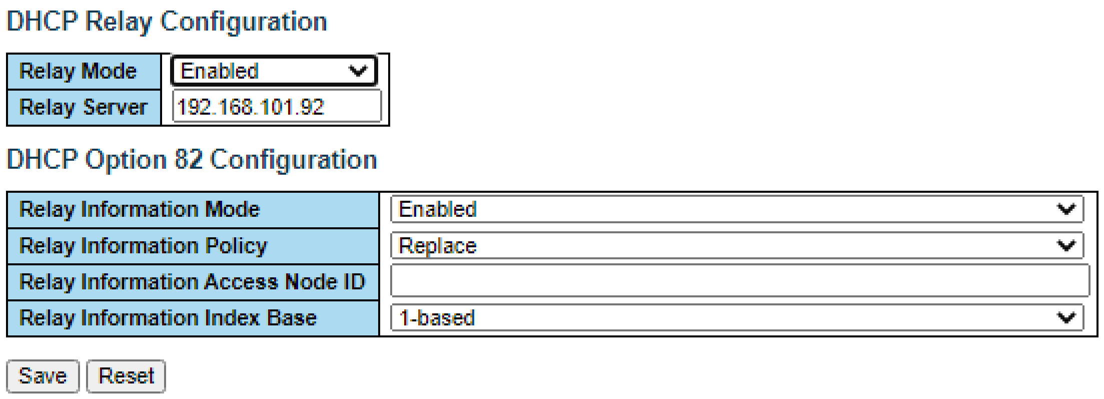
Relay Information Policy: the selected policy dictates the behavior of the DHCP Agent when it receives a DHCP message that already contains relay agent information. Available policies are:

- **Replace**: Replace the original relay information when a DHCP message that already contains it is received.
- **Keep**: Keep the original relay information when a DHCP message that already contains it is received.
- **Drop**: Drop the package when a DHCP message that already contains relay information is received.

Relay Information Access Node ID: specify the Agent Identifier to use when formatting the Circuit ID sub-option of Option 82. When empty, it will use the GAM SNMP System Name. When the SNMP System Name is empty, it will use the GAM serial number.

Relay Information Index Base: specify whether the GAM slot number and the G.hn end-point number in the Circuit ID are zero-based or one-based.

- **0-based**: GAM Slot / G.hn end-point numbers specified in the Circuit ID are zero-based
- **1-based**: GAM Slot / G.hn end-point numbers specified in the Circuit ID are one-based



DHCP Relay Configuration

Relay Mode	Enabled
Relay Server	192.168.101.92

DHCP Option 82 Configuration

Relay Information Mode	Enabled
Relay Information Policy	Replace
Relay Information Access Node ID	
Relay Information Index Base	1-based

Figure 13: Basic DHCP Relay Configuration Settings

2.4 PPPoE Intermediate Agent

The GAM supports the PPPoE Intermediate Agent. It operates as per the Broadband Forum TR-101 specification. The PPPoE Intermediate Agent intercepts all upstream PPPoE discovery stage packets, i.e. the PADI, PADR and upstream PADT packets, but does not modify the source or destination MAC address of these PPPoE discovery packets.

The PPPoE Intermediate Agent operates as per the Broadband Forum TR-101 specification. The Circuit ID format is "[Agent Identifier][interface] [slot]/[port number]". For example, "000ED8-GAM12C-01012973" eth 0/6.0:3 indicates that the PPPoE Discovery request was issued by Agent 000ED8-GAM12C-01012973 on an Ethernet interface by a GAM at slot 0. It further indicates that the PPPoE packet originates from port #6 of the GAM, sent by endpoint #0 to this port, with a VLAN ID of 3. The Remote ID value is set to the G.hn subscriber name associated to the G.hn endpoint.

You can configure the following settings for the PPPoE Intermediate Agent:

Information Mode: select the mode of operation for the PPPoE Intermediate Agent function. Possible modes are:

- **Enabled**: enables the PPPoE Intermediate Agent where the GAM inserts specific information (Agent Information/TR-101) into a PPPoE Discovery message when forwarding to the PPPoE server and removes it from a PPPoE Discovery message when transferring to the PPPoE client.
- **Disabled**: Disable the PPPoE Intermediate Agent function.

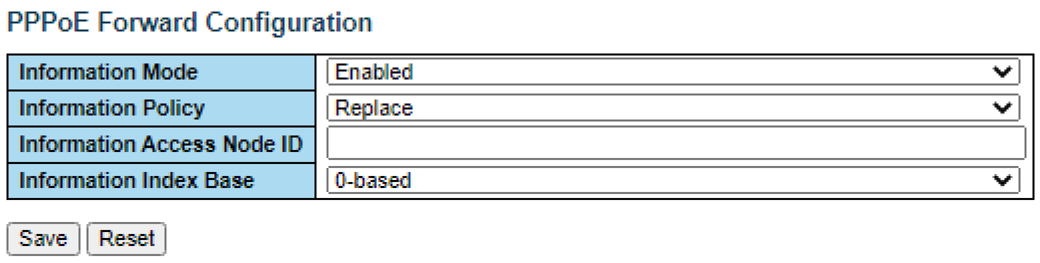
Information Policy: the selected policy dictates the behavior of the PPPoE Intermediate Agent when it receives a PPPoE Discover message that already contains relay agent information. Available policies are:

- **Replace**: Replace the original relay information when a PPPoE Discover message that already contains it is received.
- **Keep**: Keep the original relay information when a PPPoE Discover message that already contains it is received.
- **Drop**: Drop the package when a PPPoE Discover message that already contains relay information is received.

Information Access Node ID: specify the Agent Identifier to use when formatting the Circuit ID sub-option. When empty, it will use the GAM SNMP System Name. When the SNMP System Name is empty, it will use the GAM serial number.

Information Index Base: specify whether the GAM slot number and the G.hn end-point number in the Circuit ID are zero-based or one-based.

- **0-based**: GAM Slot / G.hn end-point numbers specified in the Circuit ID are zero-based
- **1-based**: GAM Slot / G.hn end-point numbers specified in the Circuit ID are one-based


PPPoE Forward Configuration

Information Mode	Enabled
Information Policy	Replace
Information Access Node ID	
Information Index Base	0-based

Figure 14: Basic PPPoE Intermediate Agent Configuration Settings

Chapter 3

Overview of the JSON-RPC API

3.1 Overview of the GAMJSON-RPC API

As covered in the summary section of the GAM WEB GUI, the provisioning of services on a Positron GAM device typically includes the following:

- **GAM Global Settings:** this is provisioned by the following JSON-RPC API Methods:
  - *ghnAgent.config.global.get:* retrieve the GAM Global settings
  - *ghnAgent.config.global.set:* modify the GAM Global settings
- **Bandwidth Profiles:** this is provisioned by the following JSON-RPC API Methods where the Bandwidth Profile Name is used as the key:
  - *ghnAgent.config.bwProfileByName.add:* add a new Bandwidth Profile
  - *ghnAgent.config.bwProfileByName.get:* retrieve existing Bandwidth Profile(s)
  - *ghnAgent.config.bwProfileByName.set:* modify an existing Bandwidth Profile
  - *ghnAgent.config.bwProfileByName.del:* remove an existing Bandwidth Profile
- **Subscriber (User) Profiles:** this is provisioned by the following JSON-RPC API Methods:
  - *ghnAgent.config.userByName.add:* create a new Subscriber Profile
  - *ghnAgent.config.userByName.get:* retrieve existing Subscriber Profile(s)
  - *ghnAgent.config.userByName.set:* modify an existing Subscriber Profile
  - *ghnAgent.config.userByName.del:* remove an existing Subscriber Profile
- **Endpoint Devices (applicable when operating in Endpoint Aware mode):** this is provisioned by the following JSON-RPC API Methods:
  - *ghnAgent.config.endpointByName.add:* add a new Endpoint entry and assign it to a specific GAM port
  - *ghnAgent.config.endpointByName.get:* retrieve existing Endpoint Devices defined for the GAM
  - *ghnAgent.config.endpointByName.set:* modify an existing Endpoint Device
  - *ghnAgent.config.endpointByName.del:* remove an existing Endpoint Device

- **Ports:** this is provisioned by the following JSON-RPC API Methods:
    - *ghnAgent.config.port.get: retrieve information about port(s) of the GAM*
    - *ghnAgent.config.port.set: modify an existing Port entry of the GAM*

NOTE: upon completion, the JSON-RPC Methods return a JSON object in the form of: ***{"id":"?","error":?,"result":?}.*** The ***id*** will always reflect the value provided in the JSON-RPC Method; ***error*** will be present if the method could not be completed successfully. If successful, ***result*** may be present depending on the action assigned to the Method.

3.1.1 Retrieving the JSON Contract from the GAM

You can retrieve the JSON contract (***inventory-r21156.json)*** from the GAM by issuing the following JSON RPC Method:

http://${USERNAME}:${PASSWORD}@${SERVER}:${PORT}/json_rpc -s -d '{"method":"jsonRpc.status.introspection.generic.inventory.get", "params":[""],"id":"1"}'

3.2 Committing provisioning changes to the GAM

It is important to commit any provisioning changes to the GAM startup configuration to make sure the new settings are still in force following an eventual restart of the GAM device. This is achieved with the following JSON-RPC Method:

> ***'icfg.control.copy.set' '[{"Copy": true, "SourceConfigType": "runningConfig",*** **"SourceConfigFile": "runningConfig", "DestinationConfigType": "startupConfig", "DestinationConfigFile": "startupConfig", "Merge": false }]'**

3.3 Managing GAM Global Settings

The global GAM settings and they are defined by the following JSON object: vtss_appl_ghn_agent_global_config_t.

When operating in Port Aware mode, you need to set the global settings as follows:

- **SubscriberModel:** shall be set to "**portAware**"
- **DefaultNNIIfIndex:** shall be set to "**10G 1/1**" to select the first port of the GAM as the default uplink port

```
{
	"type-name":"vtss_appl_ghn_agent_global_config_t",
	"class":"Struct",
	"description":"",
	"encoding-type":"Object",
	"elements":[
		{
		"name":"SubscriberModel",
		"type":"vtss_appl_ghn_agent_subscriber_model_t",
		"description":"The subscriber model.  In endpoint-aware, each
		subscriber is assigned an endpoint.  In port-aware, each
		subscriber is assigned a port.  The Port-Aware model is only
		offered on MIMO/SISO systems (GAM-xx-M)."
		},
		{
		"name":"DefaultNNIIfIndex",
		"type":"vtss_ifindex_t",
		"description":"The default UPLINK (NNI) port to use for double-
		tagging when the subscriber UPLINK (NNI) port is set to the value
		'default'."
		}
	]
}
```

3.3.1 ghnAgent.config.global.get

Use this method to retrieve the GAM Global settings to make sure they are set as per your preferred operating mode.

```
{
	"method-name":"ghnAgent.config.global.get",
	"params":[],
	"result":[
		{
		"name":"config",
		"type":"vtss_appl_ghn_agent_global_config_t"
		}
	]
}
```

3.3.2 ghnAgent.config.global.set

Use this method to retrieve the GAM Global settings to make sure they are set as per your preferred operating mode.

```
{
	"method-name":"ghnAgent.config.global.set",
	"params":[
		{
		"name":"config",
		"type":"vtss_appl_ghn_agent_global_config_t"
		}
```

```
      ],
      "result":[]
}
```

3.4 Managing GAM Bandwidth Profiles

A bandwidth profile is defined by the following attributes:

- **Id:** ignore this parameter, NOT used when using the ByName JSON-RPC Methods
- **Uid:** ignore this parameter, NOT used when using the ByName JSON-RPC Methods
- **Name**: Unique name of this Bandwidth Profile. **This is the key to identify a Bandwidth Profile in the JSON-RPC Methods acting on Bandwidth Profiles**
- **Description**: Free-form description string for the Bandwidth Profile
- **DsBw**: Downstream Bandwidth (in Mbps) assigned to this Bandwidth profile
- **UsBw**: Upstream Bandwidth (in Mbps) assigned to this Bandwidth profile

The JSON Object encoding for a GAM Bandwidth Profile is as follows:

```
{
      "type-name":"vtss_appl_ghn_agent_bw_profile_config_t",
      "class":"Struct",
      "description":"",
      "encoding-type":"Object",
      "elements":[
            {
            "name":"Id",
            "type":"uint32_t",
            "description":"The bandwidth profile ID.  NOT used when
            using the “ByName” Bandwidth profile methods"
            },
            {
            "name":"Uid",
            "type":"uint32_t",
            "description":"A number uniquely identifying this endpoint.
            Incremented for each newly added endpoint.  Not persistent
            across reboot.  This value is read-only.  NOT used when
            using the “ByName” Bandwidth profile methods."
            },
            {
            "name":"Name",
            "type":"string",
            "description":"unique name – used as the key for all other
            JSON-RPC Methods involving a Bandwidth profile"
            },
            {
            "name":"Description",
            "type":"string",
                "description":"User-configurable description string"
                },
                {
                "name":"DsBw",
                "type":"uint32_t",
                "description":"Bandwidth profile downstream bandwidth."
                },
                {
                "name":"UsBw",
                "type":"uint32_t",
                "description":"Bandwidth profile upstream bandwidth."
                }
        ]
}
```

3.4.1 ghnAgent.config.bwProfileByName.add

Use this method to create a NEW Bandwidth Profile with a unique Bandwidth Profile name since this is the key used to identify the profile in the other JSON-RPC Methods that make use or refer to a Bandwidth profile.

```
{
        "method-name":"ghnAgent.config.bwProfileByName.add",
        "group-name":"ghnAgent.config.bwProfile",
        "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
        "params":[
                {
                "name":"name",
                "type":"string"
                },
                {
                "name":"config",
                "type":"vtss_appl_ghn_agent_bw_profile_config_t"
                }
        ],
        "result":[]
}
```

3.4.2 ghnAgent.config.bwProfileByName.get

Use this method to retrieve **all** the existing Bandwidth Profile(s) that are returned as an array of Bandwidth Profiles. You can retrieve a specific Bandwidth Profile if you specify the unique Bandwidth Profile name as an input to the Method:

```
{
        "method-name":"ghnAgent.config.bwProfileByName.get",
	"group-name":"ghnAgent.config.bwProfileByName",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
	"params":[
		{
		"name":"name",
		"type":"string"
		}
	],
	"result":[
		{
		"name":"config",
		"type":"vtss_appl_ghn_agent_bw_profile_config_t"
		}
	]
}
```

You can retrieve ALL the Bandwidth Profiles if you do not specify a unique Bandwidth Profile name as an input to the Method:

```
{
	"method-name":"ghnAgent.config.bwProfileByName.get",
	"description":"This is an overload of 
	ghnAgent.config.bwProfileByName.get without any input parameters. It is 
	used to implement a get-all functionallity.",
	"group-name":"ghnAgent.config.bwProfileByName",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
	"params":[],
	"result":[
		{
			"name":"res",
			"type":{
				"class": "Array",
				"encoding-type": "Array",
				"type":{
					"class": "Struct",
					"encoding-type": "Object",
					"elements":[
						{
						"name":"key",
						"semantic-name":"name",
						"type":"string"
						},
						{
						"name":"val",
						"semantic-name":"config",
						"type":"vtss_appl_ghn_agent_bw_profile_co
						nfig_t"
						}
```

```
                                        ]
                                }
                        }
                }
        ]
}
```

3.4.3 ghnAgent.config.bwProfileByName.set

Use this method to modify an existing Bandwidth Profile identified by its unique Bandwidth Profile name. The value of **DsBw** and **UsBw** will be applied to the specified Bandwidth Profile.

```
{
        "method-name":"ghnAgent.config.bwProfileByName.set",
        "group-name":"ghnAgent.config.bwProfile",
        "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
                "params":[
                        {
                        "name":"name",
                        "type":"string"
                        },
                        {
                        "name":"config",
                        "type":"vtss_appl_ghn_agent_bw_profile_config_t"
                        }
                ],
                "result":[]
},
```

3.4.4 ghnAgent.config.bwProfileByName.del

Use this method to remove (delete) an existing Bandwidth Profile identified by its unique Bandwidth Profile name.

```
{
        "method-name":"ghnAgent.config.bwProfileByName.del",
        "group-name":"ghnAgent.config.bwProfile",
        "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
        "params":[
                {
                "name":"name",
                "type":"string"
                }
        ],
        "result":[]
},
```

3.5 Managing GAM Subscriber Profiles

A Subscriber Profile is defined by the following JSON object:
vtss_appl_ghn_agent_user_config_t

When operating in ByName mode, the key to a Subscriber Profile is the "**name**" element in the following JSON object.

```
{
            "type-name":"vtss_appl_ghn_agent_user_config_t",
            "class":"Struct",
            "description":"",
            "encoding-type":"Object",
            "elements":[
                  {
                  "name":"Id",
                  "type":"uint32_t",
                  "description":"The subscriber ID.  NOT used when operating
                  in ByName mode."
                  },
                  {
                  "name":"Uid",
                  "type":"uint32_t",
                  "description":"A number uniquely identifying this
                  subscriber.  Incremented for each newly added subscriber.
                  Not persistent across reboot.  This value is read-only. NOT
                  used when operating in ByName mode."
                  },
                  {
                  "name":"Name",
                  "type":"string",
                  "description":"Unique Name of the Subscriber Profile, used
                  as a key whenever referencing a Subscriber Profile in any
                  of the other ByName JSON-RPC Methods."
                  },
                  {
                  "name":"Description",
                  "type":"string",
                  "description":"User-configurable description string"
                  },
                  {
                  "name":"EndpointId",
                  "type":"uint32_t",
                  "description":"G.hn endpoint ID assigned to this
                  subscriber, 0 to configure subscriber with no assigned
                  endpoint. Writable only when configuring subscriber by id."
                  },
                  {
                  "name":"EndpointName",
                  "type":"string",
                  "description":"G.hn endpoint assigned to this subscriber.
                  Leave unspecified when operating in PortAware mode.
                  Writable only when configuring subscriber by name."
                  },
                  {
                  "name":"EndpointMacAddress",
                  "type":"mesa_mac_t",
                  "description":"Endpoint MAC address (valid if an endpoint
                  is assigned)."
                  },
                  {
                  "name":"BwProfileId",
                  "type":"uint32_t",
                  "description":"G.hn bandwidth profile ID assigned to this
                  subscriber, 0 for unthrottled. Writable only when
                  configuring subscriber by id.  Use the BwProfileName object
                  when operating in ByName mode."
                  },
                  {
                  "name":"BwProfileName",
                  "type":"string",
                  "description":"G.hn bandwidth profile assigned to this
                  subscriber.  Writable only when configuring subscriber by
                  name."
                  },
                  {
                  "name":"BwProfileUid",
                  "type":"uint32_t",
                  "description":"Id uniquely identifying the assigned
                  bandwidth profile."
                  },
                  {
                  "name":"VlanId",
                  "type":"uint16_t",
                  "description":"Vlan ID assigned to this subscriber."
                  },
                  {
                  "name":"VlanIsTagged",
                  "type":"vtss_bool_t",
                  "description":"Keep VLAN tag at endpoint facility port."
                  },
                  {
                  "name":"RemappedVlanId",
                  "type":"uint16_t",
                  "description":"Remapped Vlan ID.  0 for no remapping,
                  otherwize, must be a value between 3 and 4093."
                  },
                  {
                  "name":"Port2VlanId",
                  "type":"uint16_t",
                  "description":"The VLAN ID attributed to the second
                  Ethernet port, for services such as IPTV, VoIP, etc.  This
                  is applicable only for endpoints which is equipped with two
                  Ethernet ports, such as the G1000-M/C, and not applicable
                  for endpoints equipped with a single Ethernet port such as
                  the G1001-M/C.  That correspond to the VLAN used for the
                  second Ethernet port."
                  },
                  {
                  "name":"AllowedTaggedVlans",
                  "type":"vtss_vid_list_t",
                  "description":"Extra allowed tagged VLANs.  Valid VLANS are
                  VID 3~4093."
                  },
                  {
                  "name":"TrunkMode",
                  "type":"vtss_bool_t",
                  "description":"Allow CPE trunk mode.  Not available for all
                  CPE."
                  },
                  {
                  "name":"PortIfIndex",
                  "type":"vtss_ifindex_t",
                  "description":"G.hn subscriber assigned port (applicable
                  only when the system is in port-aware mode)."
                  },
                  {
                  "name":"DoubleTags",
                  "type":"vtss_bool_t",
                  "description":"Use double-tagging."
                  },
                  {
                  "name":"NNIIfIndex",
                  "type":"vtss_ifindex_t",
                  "description":"Uplink port to use for double-tagging
                  traffic.  Set to None to use default Uplink (NNI) defined
                  in the global configuration."
                  },
                  {
                  "name":"OuterTagVlanId",
                  "type":"uint16_t",
                  "description":"Outer tag used in double tagging."
```

```
                    }
          ]
},
```

3.5.1 ghnAgent.config.userByName.add

Use this method to create a new Subscriber Profile for the GAM.

```
{
          "method-name":"ghnAgent.config.userByName.add",
          "group-name":"ghnAgent.config.userByName",
          "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
          "params":[
                    {
                    "name":"name",
                    "type":"string"
                    },
                    {
                    "name":"config",
                    "type":"vtss_appl_ghn_agent_user_config_t"
                    }
          ],
          "result":[]
}
```

3.5.2 ghnAgent.config.userByName.get

Use this method to retrieve **all** the existing Subscriber Profile(s) that are returned as an array of Subscriber Profiles. You can retrieve a specific Subscriber Profile if you specify the unique Subscriber Profile name as an input to the Method:

```
{
          "method-name":"ghnAgent.config.userByName.get",
          "group-name":"ghnAgent.config.userByName",
          "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
          "params":[
                    {
                    "name":"name",
                    "type":"string"
                    }
          ],
          "result":[
                    {
                    "name":"config",
                    "type":"vtss_appl_ghn_agent_user_config_t"
                    }
          ]
}
```

You can retrieve ALL the Bandwidth Profiles if you do not specify a unique Bandwidth Profile name as an input to the Method. The method will return an array of ***vtss_appl_ghn_agent_user_config_t*** JSON objects.

```
{
      "method-name":"ghnAgent.config.userByName.get",
      "description":"This is an overload of ghnAgent.config.userByName.get
      without any input parameters. It is used to implement a get-all
      functionallity.",
      "group-name":"ghnAgent.config.userByName",
      "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
      "params":[],
      "result":[
            {
            "name":"res",
            "type":{
                  "class": "Array",
                  "encoding-type": "Array",
                  "type":{
                        "class": "Struct",
                        "encoding-type": "Object",
                        "elements":[
                              {
                              "name":"key",
                              "semantic-name":"name",
                              "type":"string"
                              },
                              {
                              "name":"val",
                              "semantic-name":"config",
                              "type":"vtss_appl_ghn_agent_user_config_t"
                              }
                        ]
                        }
                  }
            }
      ]
}
```

3.5.3 ghnAgent.config.userByName.set

Use this method to modify an existing Subscriber Profile by specifying its name:

```
{
      "method-name":"ghnAgent.config.userByName.set",
      "group-name":"ghnAgent.config.userByName",
      "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
      "params":[
            {
            "name":"name",
            "type":"string"
            },
            {
            "name":"config",
            "type":"vtss_appl_ghn_agent_user_config_t"
            }
      ],
      "result":[]
}
```

3.5.4 ghnAgent.config.userByName.del

Use this method to remove (delete) an existing Subscriber Profile by specifying its name.

```
{
        "method-name":"ghnAgent.config.userByName.del",
        "group-name":"ghnAgent.config.userByName",
        "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
        "params":[
                {
                "name":"name",
                "type":"string"
                }
        ],
        "result":[]
}
```

3.6 Managing GAM Endpoint Devices

When operating in Endpoint Aware mode (default operating mode for the GAM unless the Operating Mode is set to Port Aware), you can use the following JSON-RPC Methods to create, query, modify and delete Endpoint (G.hn to Ethernet) devices.

These methods operate on the ***vtss_appl_ghn_agent_endpoint_config_t*** JSON object:

```
{
        "type-name":"vtss_appl_ghn_agent_endpoint_config_t",
        "class":"Struct",
        "description":"",
        "encoding-type":"Object",
        "elements":[
                {
                "name":"Id",
                "type":"uint32_t",
                "description":"The endpoint ID."
                },
                {
                "name":"MacAddress",
                "type":"mesa_mac_t",
                "description":"The endpoint MAC address."
                },
                {
                "name":"Name",
                "type":"string",
                "description":"unique User-configurable name, used as the key by
                JSON-RPC Methods operating on this JSON object."
                },
                {
                "name":"Description",
                "type":"string",
                "description":"User-configurable description string"
                },
                {
                "name":"PortIfIndex",
                "type":"vtss_ifindex_t",
                "description":"G.hn port index assigned to this endpoint, 0 to
                leave unassigned from all ports."
                },
                {
                "name":"AutoPort",
                "type":"vtss_bool_t",
                "description":"Indicates if the endpoint gets assigned to the
                first port where it is discovered.  This is effective only if the
                port is left unassigned"
                },
                {
                "name":"AllowMgmtVlan",
                "type":"vtss_bool_t",
                "description":"Allow access to management VLAN from subscriber
                endpoint."
                }
            ]
}
```

3.6.1 **ghnAgent.config.endpoint*ByName*.add**

Use this JSON-RPC Method to define a NEW Endpoint device and assign it to the port where it will be connected. You can allow the GAM to accept the discovery of an Endpoint device on any port by setting the **AutoPort** parameter to YES.

```
{
	"method-name":"ghnAgent.config.endpointByName.add",
	"group-name":"ghnAgent.config.endpointByName",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
	"params":[
		{
		"name":"name",
		"type":"string"
		},
		{
		"name":"config",
		"type":"vtss_appl_ghn_agent_endpoint_config_t"
		}
	],
	"result":[]
}
```

3.6.2 **ghnAgent.config.endpoint*ByName*.get**

Use this JSON-RPC Method to either retrieve information about a specific Endpoint or for ALL the Endpoints known to the GAM.

```
{
	"method-name":"ghnAgent.config.endpointByName.get",
	"group-name":"ghnAgent.config.endpointByName",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
	"params":[
		{
		"name":"name",
		"type":"string"
		}
		],
	"result":[
		{
		"name":"config",
		"type":"vtss_appl_ghn_agent_endpoint_config_t"
		}
	]
}
```

You can retrieve ALL the Bandwidth Profiles if you do not specify a unique Bandwidth Profile name as an input to the Method. The method will return an array of **vtss_appl_ghn_agent_endpoint_config_t** JSON objects.

```
{
      "method-name":"ghnAgent.config.endpointByName.get",
      "description":"This is an overload of
      ghnAgent.config.endpointByName.get without any input parameters. It is
      used to implement a get-all functionallity.",
      "group-name":"ghnAgent.config.endpointByName",
      "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
      "params":[],
      "result":[
                  {
                  "name":"res",
                  "type":{
                        "class": "Array",
                        "encoding-type": "Array",
                        "type":{
                        "class": "Struct",
                        "encoding-type": "Object",
                        "elements":[
                              {
                              "name":"key",
                              "semantic-name":"name",
                              "type":"string"
                              },
                              {
                              "name":"val",
                              "semantic-name":"config",
                              "type":"vtss_appl_ghn_agent_endpoint_config_t"
                              }
                        ]
                        }
                  }
            }
      ]
}
```

3.6.3 **ghnAgent.config.endpoint*ByName*.set**

Use this method to modify an existing Endpoint by specifying its name:

```
{
      "method-name":"ghnAgent.config.endpointByName.set",
      "group-name":"ghnAgent.config.endpointByName",
      "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
      "params":[
            {
            "name":"name",
            "type":"string"
            },
            {
            "name":"config",
            "type":"vtss_appl_ghn_agent_endpoint_config_t"
            }
      ],
      "result":[]
}
```

3.6.4 ghnAgent.config.endpoint*ByName*.del

Use this method to remove (delete) an existing Endpoint by specifying its name.

```
{
      "method-name":"ghnAgent.config.endpointByName.del",
      "group-name":"ghnAgent.config.endpointByName",
      "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
      "params":[
            {
            "name":"name",
            "type":"string"
            }
      ],
      "result":[]
}
```

3.6.5 Retrieving detected but quarantined Endpoint devices

In addition to the previous JSON-RPC Methods operating on Endpoint devices, it may be useful to query the GAM to retrieve information about any Endpoint devices detected but not assigned to a specific subscriber. Such Endpoint devices are automatically put in a Quarantine state until they are explicitly assigned to a port.

The ***ghnAgent.status.endpointByMac.brief.get*** will return an array of JSON objects of type ***vtss_appl_ghn_agent_endpoint_brief_status_t***:

```
{
      "type-name":"vtss_appl_ghn_agent_endpoint_brief_status_t",
      "class":"Struct",
      "description":"",
      "encoding-type":"Object",
      "elements":[
            {
```

```
"name":"MacAddress",
"type":"mesa_mac_t",
"description":"G.hn endpoint MAC address."
},
{
"name":"ConfEndpointId",
"type":"uint32_t",
"description":"If this endpoint is configured, this is the
configured endpoint ID.  That value is mainly useful when
enumerating endpoints by MAC address.  In this case, the key is
the MAC address, so this give a reference to the configured
endpoint ID.  In case of endpoints which are not configured in
the system, this value will be 0.  That would indicate a newly
discovered or rogue endpoint."
},
{
"name":"ConfEndpointName",
"type":"string",
"description":"If this endpoint is configured, this is the
configured endpoint name.  In case of endpoints which are not
confiured in the system, this value will be blank.  That would
indicate a newly discovered or rogue endpoint."
},
{
"name":"ConfPortIfIndex",
"type":"vtss_ifindex_t",
"description":"G.hn port index on which this endpoint is
assigned. 0 means not assigned to any port."
},
{
"name":"ConfAutoPort",
"type":"vtss_bool_t",
"description":"Indicates if the endpoint gets assigned to the
first port where it is discovered."
},
{
"name":"DetectedPortIfIndex",
"type":"vtss_ifindex_t",
"description":"G.hn port index on which this endpoint is
detected. 0 means not detected on any port."
},
{
```

```
"name":"ConfUserId",
"type":"uint32_t",
"description":"If this endpoint is configured, this is the
subscriber ID that this endpoint is associated with. 0 means not
assigned to any subscriber."
},
{
"name":"ConfUserName",
"type":"string",
"description":"If this endpoint is configured, this is the
subscriber name that this endpoint is associated with."
},
{
"name":"ConfUserUid",
"type":"uint32_t",
"description":"Id uniquely identifying the subscriber onto which
this endpoint is assigned. Not persistent across reboot"
},
{
"name":"ConfBwProfileId",
"type":"uint32_t",
"description":"G.hn bandwidth profile instance number on which
this endpoint is assigned. That is, the bandwidth profile
instance number which is assigned to the subscriber to which this
endpoint is currently assigned to. If this endpoint is not
assigned to any subscriber, then usually the default bandwidth
profile will be assigned to this endpoint. 0 means no bandwidth
profile assigned (i.e. unthrottled)."
},
{
"name":"ConfBwProfileName",
"type":"string",
"description":"Name of the G.hn bandwidth profile on which this
endpoint is assigned. That is, the bandwidth profile which is
assigned to the subscriber to which this endpoint is currently
assigned to. If this endpoint is not assigned to any subscriber,
then usually the default bandwidth profile will be assigned to
this endpoint."
},
{
"name":"ConfBwProfileUid",
"type":"uint32_t",
        "description":"Id uniquely identifying the bandwidth profile. Not
        persistent across reboot"
        },
        {
        "name":"State",
        "type":"vtss_appl_ghn_agent_endpoint_state_t",
        "description":"Indicates the endpoint state."
        },
        {
        "name":"ModelType",
        "type":"vtss_appl_ghn_agent_cpe_model_type_t",
        "description":"Endpoint model type.  Available only when endpoint
        is detected."
        },
        {
        "name":"ModelString",
        "type":"string",
        "description":"Endpoint model string.  Available only when
        endpoint is detected."
        },
        {
        "name":"FwMismatch",
        "type":"vtss_bool_t",
        "description":"Indicates if endpoint is in firmware
        mismatch.  Available only when endpoint is detected."
        }
    ]
}
```

Use the ***ghnAgent.status.endpointByMac.brief.get*** method below to get the list of all Endpoint device(s) known by the GAM with their detailed information, including their **state** to determine whether they are authorized on the GAM or not. You can specify a MAC address if you want to retrieve information about a specific Endpoint. Otherwise, leave the **params** field empty to retrieve all the Endpoint devices known to the GAM.

```
{
    "method-name":"ghnAgent.status.endpointByMac.brief.get",
    "group-name":"ghnAgent.status.endpointByMac.brief",
    "web-privilege":{"id":"GhnAgent", "type":"STATUS_RO"},
    "params":[
        {
        "name":"mac",
        "type":"mesa_mac_t"
        }
],
    "result":[
        {
        "name":"status",
        "type":"vtss_appl_ghn_agent_endpoint_brief_status_t"
        }
    ]
}
```

3.7 PORT METHODS

Use the following JSON-RPC Methods to create, query, modify and delete Port(s) settings for the GAM.

These methods operate on the ***vtss_appl_ghn_agent_port_config_t*** JSON object:

```
{
        "type-name":"vtss_appl_ghn_agent_port_config_t",
        "class":"Struct",
        "description":"",
        "encoding-type":"Object",
        "elements":[
                {
                "name":"IfIndex",
                "type":"vtss_ifindex_t",
                "description":"Logical interface number."
                },
                {
                "name":"Name",
                "type":"string",
                "description":"User-configurable name"
                },
                {
                "name":"MacAddress",
                "type":"mesa_mac_t",
                "description":"G.hn local port MAC address."
                },
                {
                "name":"PortMode",
                "type":"vtss_appl_ghn_agent_port_mode_t",
                "description":"G.hn port mode."
                },
                {
                "name":"PortRole",
                "type":"vtss_appl_ghn_agent_node_mode_t",
                "description":"G.hn port role."
                },
                {
```

```
        "name":"Rfc5517Mode",
        "type":"vtss_appl_rfc5517_mode_t",
        "description":"G.hn port RFC5517 mode."
        },
        {
        "name":"Rfc5517Community",
        "type":"uint8_t",
        "description":"G.hn port RFC5517 community number.  This
        parameter is effective only when the RFC 5517 mode is set to
        'community'"
        },
        {
        "name":"AllowedVlans",
        "type":"vtss_vid_list_t",
        "description":"Allowed VLANs, valid only if RFC 5517 mode is set
        to a promiscuous mode (uplink or stacking)."
        }
    ]
}
```

3.7.1 ghnAgent.config.port.get

Use this JSON-RPC Method to retrieve information about a specific port of the GAM by specifying the port:

```
{
    "method-name":"ghnAgent.config.port.get",
    "group-name":"ghnAgent.config.port",
    "web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
    "params":[
        {
        "name":"ifindex",
        "type":"vtss_ifindex_t"
        }
    ],
    "result":[
        {
        "name":"config",
        "type":"vtss_appl_ghn_agent_port_config_t"
        }
    ]
}
```

If you omit the ***params*** object, the method will return an array of JSON object of type ***vtss_appl_ghn_agent_port_config_t***

```
{
	"method-name":"ghnAgent.config.port.get",
	"description":"This is an overload of ghnAgent.config.port.get without
	any input parameters. It is used to implement a get-all
	functionallity.",
	"group-name":"ghnAgent.config.port",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RO"},
	"params":[],
	"result":[
			{
			"name":"res",
			"type":{
			"class": "Array",
			"encoding-type": "Array",
			"type":{
			"class": "Struct",
			"encoding-type": "Object",
			"elements":[
					{
					"name":"key",
					"semantic-name":"ifindex",
					"type":"vtss_ifindex_t"
					},
					{
					"name":"val",
					"semantic-name":"config",
					"type":"vtss_appl_ghn_agent_port_config_t"
					}
				]
			}
		}
	}
	]
}
```

3.7.2 ghnAgent.config.port.set

Use this method to modify an existing Port by specifying its port index:

```
{
	"method-name":"ghnAgent.config.port.set",
	"group-name":"ghnAgent.config.port",
	"web-privilege":{"id":"GhnAgent", "type":"CONFIG_RW"},
	"params":[
		{
        "name":"ifindex",
        "type":"vtss_ifindex_t"
        },
        {
        "name":"config",
        "type":"vtss_appl_ghn_agent_port_config_t"
        }
    ],
    "result":[]
}
```

3.8 JSON-RPC API ERROR MESSAGES

When a JSON-RPC method call encounters an error, the Response Object MUST contain the error member with a value that is an Object with the following members:

- **Code:** A Number that indicates the error type that occurred. This MUST be an integer.
- **Message:** A String providing a short description of the error.
  The message SHOULD be limited to a concise single sentence.
- **Data:** A Primitive or Structured value that contains additional information about the error. This may be omitted. The value of this member is defined by the JSPN-RPC Server (e.g. detailed error information, nested errors etc.).

For instance, the following error defined in the JSON-RPC contract file:

```
        {
        "vtss-error-code":-9437162,
        “vtss-module-code”:-22,
        “vtss-message”:"Invalid G.hn bandwidth profile name"
        },
```

Would be reported as follows:

```
{
    "id": "1",
    "result": null,
    "error": {
        "code": -32603,
        "message": "Invalid G.hn bandwidth profile name",
        "data": {
            "vtss-error-code": -9437162,
            "vtss-failing-function-ptr": 67234856,
            "vtss-failing-module": 143,
            "vtss-module-code": -22
        }
    }
}
```

NOTE: The error codes from and including -32768 to -32000 are reserved for pre-defined errors. Any code within this range, but not defined explicitly below is reserved for future use. The error codes are nearly the same as those suggested for XML-RPC at the following url: http://xmlrpc-epi.sourceforge.net/specs/rfc.fault_codes.php

code	message	meaning
-32700	Parse error	Invalid JSON-RPC method was received by the server. An error occurred on the server while parsing the JSON-RPC text.
-32600	Invalid Request	The JSON-RPC sent is not a valid Request object.
-32601	Method not found	The method does not exist / is not available.
-32602	Invalid params	Invalid method parameter(s).
-32603	Internal error	Internal JSON-RPC error.
-32000 to -32099	Server error	Reserved for implementation-defined server-errors.

The JSON-RPC Method will return Error Messages to report any issues that prevented the JSON-RPC Method from completing successfully. The currently used Error Messages are:

```
{
	"vtss-module-name":"GhnAgent",
	"vtss-module-id":143,
	"vtss-error-codes":[
		{
		"vtss-error-code":-9437183,
		“vtss-module-code”:-1,
		“vtss-message”:"G.hn generic error"
		},
		{
		"vtss-error-code":-9437182,
		“vtss-module-code”:-2,
		“vtss-message”:"G.hn internal error"
		},
		{
		"vtss-error-code":-9437181,
		“vtss-module-code”:-3,
		“vtss-message”:"G.hn function not implemented"
		},
		{
```

```
"vtss-error-code":-9437180,
“vtss-module-code”:-4,
“vtss-message”:"Invalid G.hn port or endpoint"
},
{
"vtss-error-code":-9437179,
“vtss-module-code”:-5,
“vtss-message”:"Invalid G.hn port"
},
{
"vtss-error-code":-9437178,
“vtss-module-code”:-6,
“vtss-message”:"Invalid G.hn subscriber"
},
{
"vtss-error-code":-9437177,
“vtss-module-code”:-7,
“vtss-message”:"Non-existing G.hn subscriber"
},
{
"vtss-error-code":-9437176,
“vtss-module-code”:-8,
“vtss-message”:"Invalid G.hn endpoint"
},
{
"vtss-error-code":-9437175,
“vtss-module-code”:-9,
“vtss-message”:"Non-existing G.hn endpoint (no such endpoint
configured)"
},
{
"vtss-error-code":-9437174,
“vtss-module-code”:-10,
“vtss-message”:"No user for endpoint"
},
{
"vtss-error-code":-9437173,
“vtss-module-code”:-11,
“vtss-message”:"Endpoint not found on specified port"
},
{
"vtss-error-code":-9437172,
“vtss-module-code”:-12,
“vtss-message”:"Endpoint not found on any ports"
},
{
"vtss-error-code":-9437171,
“vtss-module-code”:-13,
“vtss-message”:"G.hn endpoint is already assigned to another
subscriber"
},
{
"vtss-error-code":-9437170,
“vtss-module-code”:-14,
“vtss-message”:"G.hn port is already assigned to another
subscriber"
},
{
"vtss-error-code":-9437169,
“vtss-module-code”:-15,
“vtss-message”:"Invalid G.hn subscriber name"
},
{
"vtss-error-code":-9437168,
“vtss-module-code”:-16,
“vtss-message”:"Non-unique G.hn subscriber name"
},
{
"vtss-error-code":-9437167,
“vtss-module-code”:-17,
“vtss-message”:"Non-unique G.hn endpoint name"
},
{
"vtss-error-code":-9437166,
“vtss-module-code”:-18,
“vtss-message”:"Non-unique G.hn bandwidth profile name"
},
{
"vtss-error-code":-9437165,
“vtss-module-code”:-19,
“vtss-message”:"Non-unique G.hn port name"
},
{
"vtss-error-code":-9437164,
“vtss-module-code”:-20,
“vtss-message”:"Invalid G.hn bandwidth profile"
},
{
"vtss-error-code":-9437163,
“vtss-module-code”:-21,
“vtss-message”:"Non-existing G.hn bandwidth profile"
},
{
"vtss-error-code":-9437162,
“vtss-module-code”:-22,
“vtss-message”:"Invalid G.hn bandwidth profile name"
```

```
},
{
"vtss-error-code":-9437161,
“vtss-module-code”:-23,
“vtss-message”:"Invalid G.hn bandwidth profile downstream
bandwidth"
},
{
"vtss-error-code":-9437160,
“vtss-module-code”:-24,
“vtss-message”:"Invalid G.hn bandwidth profile upstream
bandwidth"
},
{
"vtss-error-code":-9437159,
“vtss-module-code”:-25,
“vtss-message”:"Invalid G.hn data VLAN id"
},
{
"vtss-error-code":-9437158,
“vtss-module-code”:-26,
“vtss-message”:"Invalid G.hn remapped data VLAN id"
},
{
"vtss-error-code":-9437157,
“vtss-module-code”:-27,
“vtss-message”:"Invalid G.hn port #2 VLAN id"
},
{
"vtss-error-code":-9437156,
“vtss-module-code”:-28,
“vtss-message”:"G.hn VLAN cannot be the same as port #2 VLAN id"
},
{
"vtss-error-code":-9437155,
“vtss-module-code”:-29,
“vtss-message”:"Too many allowed tagged VLAN"
},
{
"vtss-error-code":-9437154,
“vtss-module-code”:-30,
“vtss-message”:"Invalid G.hn endpoint name"
},
{
"vtss-error-code":-9437153,
“vtss-module-code”:-31,
“vtss-message”:"Invalid G.hn endpoint MAC address"
},
{
"vtss-error-code":-9437152,
“vtss-module-code”:-32,
“vtss-message”:"Non-unique G.hn endpoint MAC address"
},
{
"vtss-error-code":-9437151,
“vtss-module-code”:-33,
“vtss-message”:"Maximum number of G.hn endpoint reached for the
system"
},
{
"vtss-error-code":-9437150,
“vtss-module-code”:-34,
“vtss-message”:"Maximum number of G.hn banbdwidth profile reached
for the system"
},
{
"vtss-error-code":-9437149,
“vtss-module-code”:-35,
“vtss-message”:"Maximum number of G.hn subscriber reached for the
system"
},
{
"vtss-error-code":-9437148,
“vtss-module-code”:-36,
“vtss-message”:"Maximum number of G.hn endpoint reached for that
port"
},
{
"vtss-error-code":-9437147,
“vtss-module-code”:-37,
“vtss-message”:"Invalid G.hn port mode"
},
{
"vtss-error-code":-9437146,
“vtss-module-code”:-38,
“vtss-message”:"Invalid G.hn port role"
},
{
"vtss-error-code":-9437145,
“vtss-module-code”:-39,
“vtss-message”:"Invalid G.hn port RFC 5517 mode"
},
{
"vtss-error-code":-9437144,
“vtss-module-code”:-40,
“vtss-message”:"Invalid G.hn port RFC 5517 community number"
```

```
},
{
"vtss-error-code":-9437143,
“vtss-module-code”:-41,
“vtss-message”:"Invalid NNI port"
},
{
"vtss-error-code":-9437142,
“vtss-module-code”:-42,
“vtss-message”:"Invalid outer tag VID"
},
{
"vtss-error-code":-9437141,
“vtss-module-code”:-43,
“vtss-message”:"Specified NNI port is not uplink"
},
{
"vtss-error-code":-9437140,
“vtss-module-code”:-44,
“vtss-message”:"G.hn bandwidth profile is not supported on this
platform yet"
},
{
"vtss-error-code":-9437139,
“vtss-module-code”:-45,
“vtss-message”:"G.hn endpoint is already configured"
},
{
"vtss-error-code":-9437138,
“vtss-module-code”:-46,
“vtss-message”:"G.hn bandwidth profile is already configured"
},
{
"vtss-error-code":-9437137,
“vtss-module-code”:-47,
“vtss-message”:"G.hn subscriber is already configured"
},
{
"vtss-error-code":-9437136,
“vtss-module-code”:-48,
“vtss-message”:"Invalid G.hn VectorBoost parameter"
},
{
"vtss-error-code":-9437135,
“vtss-module-code”:-49,
“vtss-message”:"Invalid G.hn powermask notch Id"
},
{
        "vtss-error-code":-9437134,
        “vtss-module-code”:-50,
        “vtss-message”:"Invalid G.hn powermask notch parameter"
        },
        {
        "vtss-error-code":-9437133,
        “vtss-module-code”:-51,
        “vtss-message”:"Invalid measurement averaging value"
        },
        {
        "vtss-error-code":-9437132,
        “vtss-module-code”:-52,
        “vtss-message”:"Device not found"
        },
        {
        "vtss-error-code":-9437131,
        “vtss-module-code”:-53,
        “vtss-message”:"Error querying G.hn device.  May be missing or
        busy."
        },
        {
        "vtss-error-code":-9437130,
        “vtss-module-code”:-54,
        “vtss-message”:"Error device not in valid state"
        },
        {
        "vtss-error-code":-9437129,
        “vtss-module-code”:-55,
        “vtss-message”:"Error device model requiered manual upgrade"
        },
        {
        "vtss-error-code":-9437128,
        “vtss-module-code”:-56,
        “vtss-message”:"Auto-endpoint subscriber already assigned to
        another port"
        },
        {
        "vtss-error-code":-9437127,
        “vtss-module-code”:-57,
        “vtss-message”:"Invalid G.hn subscriber model"
        }
    ]
}
```

Chapter 4

Sample JSON Script – PORT AWARE mode

4.1 JSON-RPC API Example: Port-Aware

Here is a simple example that illustrates how to create and update bandwidth profile settings as well as subscriber profiles when the GAM is set to operate in the **PORT Aware** mode. Other examples are available from Positron Access.

The following example uses a simple bash script along with CURL to issue a sequence of JSON-RPC method commands.

```
#!/bin/bash

# example: SERVER=192.168.1.1 PORT=80 USERNAME=admin PASSWORD="" ./example-
port-aware-double-tag-by-name-1.4-21270.sh

USERNAME=${USERNAME:-admin}
PASSWORD=${PASSWORD:-""}
SERVER=${SERVER:-192.168.100.65}
PORT=${PORT:-80}

# Remarks:
# -------
#
# - All API types/methods are specified in inventory-r21156.json - this file
can be retrieved from a running firmware using the following command:
#   curl --verbose http://${USERNAME}:${PASSWORD}@${SERVER}:${PORT}/json_rpc
-s -d '{"method":"jsonRpc.status.introspection.generic.inventory.get",
"params":[""],"id":"1"}'
#
# - All commands will return an  JSON object the form:
{"id":"?","error":?,"result":?}. The id will always have a value, result may
be null if error is not, error
#   may be null if the result was not, and both error/result can be null,
i.e. in the case of a configuration command with a 'set' semantic.
#
# - For accessing the specification of a specific method, i.e.
ghnAgent.config.userByName.set, one should look for "method-
name":"ghnAgent.config.userByName.set" in the JSON contract specification
file.
#
# - The GAM does not really care about the id provided in a command and will
return the once it received in its response.
#   It is highly recommended than a different id would be used for every
request in order to allow proper testing/debugging.
#

# Check requirements
# curl
```

```
command -v curl >/dev/null 2>&1
if [[ $? -ne 0 ]]; then
    echo "curl is not installed.  Aborting."
    echo "Common ways to install curl:"
    echo "- Ubuntu : sudo apt install curl"
    echo "- Fedora : sudo yum install curl"
    echo "- CentOS : sudo dnf install curl"
    exit 1
fi

# json_reformat
command -v json_reformat >/dev/null 2>&1
if [[ $? -ne 0 ]]; then
    echo "json_reformat is not installed.  Aborting."
    echo "Common ways to install json_reformat:"
    echo "- Ubuntu : sudo apt install yajl-tools"
    echo "- Fedora : sudo yum install yajl"
    echo "- CentOS : sudo dnf install yajl"
    exit 1
fi

do_command()
{
    prompt=$1
    method=$2
    param=$3

    json_command="{\"method\":\"$method\", \"id\":"$((ID++))",
\"params\":"$param", \"jsonrpc\":\"2.0\"}"

    if [[ -n "$1" ]]; then
        echo "---------- $1 - press Enter to continue or CTRL+C for
terminating. ----------"
        read line

        printf "SEND:\n"
        printf "%s\n" "$json_command" | json_reformat
        printf "\n"
        printf "RECEIVE:\n"
        curl http://${USERNAME}:${PASSWORD}@${SERVER}:${PORT}/json_rpc -s -d
"$json_command" | json_reformat
        printf "\n"
    else
        curl http://${USERNAME}:${PASSWORD}@${SERVER}:${PORT}/json_rpc -s -d
"$json_command" >/dev/null 2>&1
    fi
}
```

```
clear

ID=1

#---

printf "=======================\n"
printf "SWITCHING TO PORT-AWARE\n"
printf "=======================\n"
printf "\n"

do_command 'Retrieve global configs'     \
           'ghnAgent.config.global.get' \
           '[]'

do_command 'Set subscriber model to port-aware' \
           'ghnAgent.config.global.set'         \
           '[{"SubscriberModel":"portAware"}]'

do_command 'Retrieve global configs'     \
           'ghnAgent.config.global.get' \
           '[]'

#---

printf "============================\n"
printf "BANDWIDTH PLAN CONFIGURATION\n"
printf "============================\n"
printf "\n"

do_command 'Retrieve all configured bw profile.' \
           'ghnAgent.config.bwProfileByName.get'       \
           '[]'

do_command 'Edit default bw profile'       \
           'ghnAgent.config.bwProfileByName.set' \
           '["Default BW Profile",{"DsBw":50,"UsBw":5}]'

do_command 'Retrieve all configured bw profile.' \
           'ghnAgent.config.bwProfileByName.get'       \
           '[]'

do_command 'Add bw profile'                \
           'ghnAgent.config.bwProfileByName.add' \
           '["BW profile 1",{"Description":"Example bw profile
#1","DsBw":500,"UsBw":50}]'

do_command 'Retrieve all configured bw profile.' \
           'ghnAgent.config.bwProfileByName.get'       \
           '[]'

#---

printf "========================\n"
printf "SUBSCRIBER CONFIGURATION\n"
printf "========================\n"
printf "\n"

do_command 'Retrieve all configured subscribers.' \
           'ghnAgent.config.userByName.get'             \
           '[]'

do_command 'Add subscriber #1, on port #1 and with default bw profile, VLAN
ID 101' \
           'ghnAgent.config.userByName.add'
\
           '["Subscriber 1",{"Description":"Example subscriber
#1","PortIfIndex":"G.hn 1/1","BwProfileId":1,"VlanId":101}]'

do_command 'Retrieve all configured subscribers.' \
           'ghnAgent.config.userByName.get'             \
           '[]'

do_command 'Add subscriber #2, on port #2, with the bw profile we added
above, Outer VLAN ID 2800, VLAN ID 1111, REMAP 201 and extra allowed vlan
301, 302 and 310.' \
           'ghnAgent.config.userByName.add'
\
           '["Subscriber 2",{"Description":"Example subscriber
#2","PortIfIndex":"G.hn
1/2","BwProfileId":2,"DoubleTags":true,"OuterTagVlanId":2800,"VlanId":1111,"V
lanIsTagged": true,"RemappedVlanId":
201,"AllowedTaggedVlans":[301,302,310]}]'

do_command 'Retrieve all configured subscribers.' \
           'ghnAgent.config.userByName.get'             \
           '[]'

do_command 'Add subscriber #3, on port #3, with the default bw profile, Outer
VLAN ID 30 (to uplink port 10G 1/2), VLAN ID 301.' \
           'ghnAgent.config.userByName.add'
\
           '["Subscriber 3",{"Description":"Example subscriber
#3","PortIfIndex":"G.hn
1/3","BwProfileId":1,"DoubleTags":true,"NNIIfIndex":"10G
1/2","OuterTagVlanId":30,"VlanId":301}]'
```

```
do_command 'Retrieve all configured subscribers.' \
           'ghnAgent.config.userByName.get'             \
           '[]'

#---

printf "====================\n"
printf "GLOBAL CONFIGURATION\n"
printf "====================\n"
printf "\n"

do_command 'Retrieve global configuration.' \
           'ghnAgent.config.global.get'     \
           '[]'

do_command 'Change default uplink port to 10G 1/2.' \
           'ghnAgent.config.global.set'             \
           '[{"DefaultNNIIfIndex":"10G 1/2"}]'

do_command 'Retrieve global configuration.' \
           'ghnAgent.config.global.get'     \
           '[]'

#---

printf "==================\n"
printf "SAVE CONFIGURATION\n"
printf "==================\n"
printf "\n"

# Important: this command will make your changes persistent.
do_command 'Persist configuration changes (copy running-config startup-
config).' \
           'icfg.control.copy.set'
\
           '[{ "Copy": true, "SourceConfigType": "runningConfig",
"SourceConfigFile": "runningConfig", "DestinationConfigType":
"startupConfig", "DestinationConfigFile": "startupConfig", "Merge": false }]'

#---

printf "====\n"
printf "DONE\n"
printf "====\n"
printf "\n"

printf "PRESS ENTER TO CLEANUP...\n"
```

```
read line
printf "Cleaning up...\n"

ID=1000

do_command '' 'ghnAgent.config.global.set' '[{"DefaultNNIIfIndex":"10G
1/1"}]'
do_command '' 'ghnAgent.config.userByName.del' '["Subscriber 1"]'
do_command '' 'ghnAgent.config.userByName.del' '["Subscriber 2"]'
do_command '' 'ghnAgent.config.userByName.del' '["Subscriber 3"]'
do_command '' 'ghnAgent.config.bwProfileByName.set' '["Default BW
Profile",{"DsBw":10,"UsBw":10}]'
do_command '' 'ghnAgent.config.bwProfileByName.del' '["BW profile 1"]'
do_command '' 'ghnAgent.config.global.set'
'[{"SubscriberModel":"endpointAware"}]'
# In case some endpoints got auto-assigned...
do_command '' 'ghnAgent.config.endpoint.del' '[1]'
do_command '' 'ghnAgent.config.endpoint.del' '[2]'
do_command '' 'ghnAgent.config.endpoint.del' '[3]'
do_command '' 'icfg.control.copy.set' '[{ "Copy": true, "SourceConfigType":
"runningConfig", "SourceConfigFile": "runningConfig",
"DestinationConfigType": "startupConfig", "DestinationConfigFile":
"startupConfig", "Merge": false }]'
```

Chapter 5

Warranty and Customer Service

Positron Access Solutions will replace or repair this product within the warranty period if it does not meet its published specifications or fails while in service. Warranty information can be found in your Positron Access customer web portal: http://www.positronaccess.com/Portal.php

Positron Access Solutions Sales Pricing/Availability and Technical Support

US and Canada: 1-888-577-5254

International: +1-514-345-2220

customerservice@positronaccess.com

Repair and Return Address

Contact Customer Service prior to returning equipment to Positron.

Telephone US and Canada: 1-888-577-5254 option 6

International: +1-514-345-2220 option 6