SnapAV Connected Device Integration Specification

Base Specification

How to connect a device to the "OvrC Cloud Service"

Connected Systems
Division of SnapAV
1800 Continental Blvd; Charlotte, NC 28273

VERSION CONTROL

Version	Description	Author	Date
0.0	A previous format of this specification was updated to include the OvrC Base features, and provided to vendors who needed it urgently.	AJO KJR	Mid October
0.1	Initial Draft	JET	11/19/2014
0.2	Updated – removed Time Settings and additional APIs (reduced scope)	JET	11/20/2014
0.3	Updated Time settings to include several fields	JET	11/21/2014
0.4	Full review for correctness/completeness. Added help text and usage notes. Added 'Introduction', 'Purpose' and 'Appendix' sections.	AJO	11/21/2014
0.6	Added appendices Added dxSetCloudServerUrl	JET	12/8/2014
1.0	Sign-off received. Published for all vendors.	JET	12/10/2014
1.02	Added "Cloud Connection" visual requirement for device local UI Clarification on firmware version numbering	AJO	12/16/2014
1.03	Fixed a typo (missing "s" on "dsLogStatusUpdates") Added explanation of what 0 means in 'dxSetStatusUpdateFrequency' (no change)	AJO	12/23/2014
1.04	Removed requirement to set time (scope reduced) Changed dxSetTimeSettings 'manualEntry' field name to 'currentTime'	JET	1/8/2015
1.06	Added clarification about case sensitivity (no change) Added clarification about 'dnsServer2' field (no change)	AJO	1/19/2015
1.1	Added 'dsEventNotifications' method (new products only) Added 'uptime' to 'dsLogStatusUpdates' method (new products only)	AJO	1/27/2015
1.21	Fixed dxGet/SetCloudServerUrl 'port' data type to "string" (correction) Updated usage notes for dxEnableRemoteWebUiAccess (no change)	JET AJO	2/2/2015
1.3	Clarified local UI requirements based on device vendor feedback. (no change) Moved all UI requirements to section VI for easy readability (no change)	AJO	2/13/2015
1.4	Removed support for TFTP download method (reduced scope)	AJO	2/19/2015
1.52	Added clarification to dxSetTimeSettings (no change) Added 'webPagePort' field to 'dxGetNetworkSettings' and 'dxSetNetworkSettings' (new products only) Added requirement: "Cloud Service" text should be a hyperlink (small change) Updated screenshot in section VI (no change)	AJO	3/5/2015
1.71	Added enabled/disabled option on webservice configuration page: section VI 'serviceTag' is now a required field Made some Set method parameters 'optional' instead of 'required' Added optional field 'connection' to dxSetCloudServerURL (All of these changes are applicable to new products only)	AJO	4/15/2015
1.72	Added comments in 'Version Control' regarding the implication of some changes	AJO	4/22/2015
1.75	'version' parameter on dx methods changed from 'required' to 'optional' Removed error messages from 'dxUpdateFirmware' response (scope reduced)	AJO	4/29/2015
1.76	Fix typo: "webPagePort" had too many closing quotation marks in JSON	GAK	5/27/2015
1.77	Added clarification that "saveAndRestore" field in dxUpdateFirmware is no longer used (previously, it was not included in this spec at all. It is not included and marked as 'deprecated')	AJO	6/23/2015
1.8	SSL is no longer allowed. TLS is mandatory security.	AJO	6/30/2015
1.81	Added clarification that websocket connection should close and reopen after a dxSetNetworkSettings command	AJO	6/30/2015
1.82	Added clarification to UPnP information	AJO	7/28/2015
1.83	Clarified format of time by ensuring the leading 0 is in the example. Set dxSetTimeSettings 'currentTime' field as optional.	AJO	12/8/2015
1.9	Added fields 'p2pID' and 'p2pPassword' to 'dxGetAbout' method to support Point-to-Point streaming surveillance devices.	AJO	2/28/2016
2.0	Added field 'hasUPS' to 'dxGetAbout' to support WattBox ARM9 devices with a UPS module	GAK	2/21/2016
2.1	Added method 'dxDisableCloud' to instruct the firmware to turn off its OvrC capabilities. Added methods 'dxEnableWebConnect' and 'dxDisableWebConnect' to use SSH tunneling to allow a remote user to view the device's local UI.	GAK	5/12/2016

	Deprecated methods 'dxEnableRemoteWebUiAccess' and 'dxDisableRemoteWebUiAccess'. Support for UPnP is no-longer required.		
2.11	Clarified that surplus fields can be ignored.	GAK	5/16/2016
2.12	Removed methods 'dxGetTimeSettings' and 'dxSetTimeSettings for most devices.	GAK	9/1/2016
2.13	Removed UPnP verbiage and updated contacts	GAK	12/31/2016
2.14	Modified 'dxEnableWebConnect' to respond with folder	GAK	1/9/2017
2.15	Added verbiage to clarify values of 'version' fields	GAK	1/23/2017
2.16	• Added verbiage to clarify operation with devices that have both a wired adapter and a wireless adapter. • Added a "key" field to "dxEnableWebConnect" so the server can send the SSH key as part of the command. • Updated the description of "sshServer" in the "dxEnableWebConnect" method to indicate that the value can be either an IP4 address or a domain name.	GAK	8/2/2017
2.17	Further clarification on which adapter to return information for	GAK	8/9/2017
2.18	Create new versions of dxGetTimeSettings and dxSetTimeSettings. Add dsGetPublicAddress and dsGetCurrentTime.	GAK	3/29/2018
2.19	Update dsGetCurrentTime to add response field 'zoneAbbr'.	GAK	5/31/2018

I. PURPOSE

SnapAV believes that industry growth in Audio/Visual & Networking products will be strongly influenced by how easily products can be installed and used, and the added features provided by internet connectivity. This concept is often referred to as the "Internet of Things" (IoT). SnapAV wishes to embrace this future by providing dealers of our products the ability to remotely setup, configure, control, and monitor all SnapAV internet-connected products. By enabling remote interaction with devices SnapAV and product partners will enable dealers to install and configure products faster, troubleshoot issues remotely, be proactively notified of adverse conditions, and unlock features previously not possible with non-connected devices.

SnapAV has developed a cloud service platform named "OvrC" (pronounced "Over-see"). The OvrC cloud service connects internet-connected products and provides user interfaces (Website, Android app, & iOS app) for dealers to remotely interact with products. This service is currently free for all customers of SnapAV products.

This document defines a specification to connect products to the OvrC cloud service. It describes the basic process, protocol, and methods for an internet-connected product to communicate with the OvrC servers, exchange information, and receive commands. SnapAV also provides a Software Development Kit (SDK), which implements this specification, to help speed your development.

This document defines the base connectivity with OvrC, which includes establishing a secure connection between the product and the cloud service, exchanging vital information about a product, receiving firmware upgrades, resetting the device, and a few other features. **Once a product has implemented the base connectivity, SnapAV will provide a separate device-specific supplement specification** that describes more detailed integration between the product and the cloud service.

If you are reading this, you are probably a SnapAV product partner who also believes in our vision. We look forward to working with you!

- SnapAV Connected Systems Team

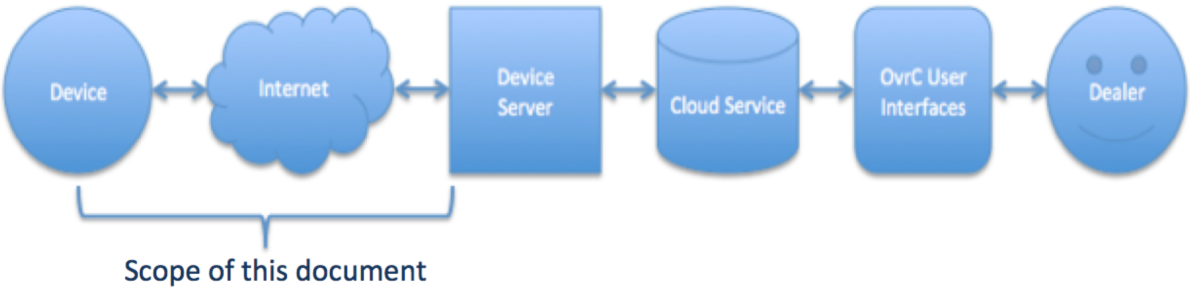
II. INTRODUCTION

FORWARD

The OvrC cloud service provides a Websocket protocol connection to exchange information with internet-connected products. This section describes the detailed Websocket integration, connection process, protocol, and required API methods. Please reference the OvrC Software Development Kit (SDK) for additional help implementing the Websocket protocol in your product.

INFORMATION FLOW

The below diagram shows the information flow between a Device (end point) and Dealer (user). This document defines the specification for the communication between the Device and Device Server. All communication between the Device Server and Dealer is handled by SnapAV.



SOFTWARE DEVELOPMENT KIT

SnapAV makes available a Software Development Kit (SDK) to device firmware developers. This SDK implements the Base Specification defined below, and will be helpful to firmware developers, allowing them to quickly connect an internet-connected device to the OvrC cloud service. The SDK is available to compile on many common platforms. **SnapAV strongly recommends using the SDK** – it will reduce development time and ensures proper usage of the Methods described in this specification.

TERMS

These are some commonly used terms in this document:

Term	Description
Device	An internet-connected hardware product provided by a manufacturer partner of SnapAV. A few examples of devices include: Access Points, Routers, Cameras, and Power Strips.
Product	Used interchangeably with “Device”
Dealer	A customer of SnapAV who purchases products from SnapAV and resells or integrates these products into homes and businesses. Dealers are the “Users” of the OvrC platform.
Customer	Used interchangeably with “Dealer” and “Integrator”
Integrator	Used interchangeably with “Dealer” and “Customer”
OvrC	The cloud service platform developed by SnapAV. OvrC allows devices to connect with it, exchange information, and provide that information to a User Interface,

	which is used by SnapAV customers.
Cloud Service	A generic term for the OvrC platform servers
User Interface	The ability to interact with devices remotely is provided through OvrC via a website, iOS app, and Android app. Collectively, these are the OvrC User Interfaces.
WebSocket	The communication protocol between a device and the OvrC cloud service
API	Application Programming Interface. This refers to the communication language (protocol) between a device and the cloud service.
Method	A standard set of communication instructions for exchanging specific data between a device and the cloud service (or vice versa). Each method relates to a specific and discreet set of information (such as network settings, time settings, or port settings).
Device Server	The OvrC cloud service module which communicates directly with OvrC enabled devices

NAMING CONVENTIONS

The following naming conventions are used throughout this document:

Name	Description
dx_____	"Destination device X" is used to prefix method names in which the communication originates within the OvrC cloud service, and the destination is a device
ds_____	"Destination server" is used to prefix method names in which the communication originates within the device, and the destination is the OvrC cloud service.
ans_____ wbs____ xxx_____	Methods which are device-specific are not contained in this document. They can be found in a separate device-specific supplemental specification. These methods are assigned a prefix which indicates the device to which they relate. For example, methods named "ans____" relate to Araknis Networks Switches.
Get / Set	Generally, methods include "Get" or "Set" in the name to indicate if they are requesting ("Getting") information from a device, or sending ("Setting") information to a device. Methods which request a specific action be performed may not include "Get" or "Set" – for example, 'dxResetDevice' is used to reboot a device.

III. PROTOCOL

INTRODUCTION TO WEBSOCKET PROTOCOL

The WebSocket protocol provides full-duplex communications between two programs over a single TCP connection. It was standardized by the IETF in 2011 and has been incorporated into every major web browser.

WebSocket protocol is ideal for OvrC usage because it eliminates the need for a DDNS service and the security risk of opening ports in firewalls. It also leverages standard port and security protocols, and thus will be allowed by default by most home network configurations.

When a client (that is, a device) opens a connection with a server (such as the OvrC cloud service), both the client and the server are free to send messages whenever they need to. Polling is never required and commands can be instantly delivered.

Additional information about Websockets can be found at http://www.w3.org/TR/websockets/

CONVERSATION WITH THE OVRC CLOUD SERVICE

A device will talk with an OvrC cloud service module we call Device Server. User requests will also be directed to Device Server, which will then pass each request on to the desired device via its WebSocket connection. The device will respond to Device Server, which will then forward the response back to the User Interface.

This means that each device will need to create a WebSocket connection - the Device Server will be its WebSocket conversation partner.

The URL of Device Server must be embedded in the device firmware, though it should also be changeable. We must be able to change the URL to point to a development server for testing or to a production server for live use. (The server URL must be configurable. See the "Hidden Configuration Page" section below.)

The URL for the production server is **wss://cloud.ovrc.com** and it uses **port 443** by default. There is also a firmware test server that the device can connect to during firmware development. SnapAV will provide the URL and Port of this server to the firmware developer and create a test harness the developer can use to test the device's APIs.

DO NOT USE wss://cloud.ovrc.com FOR TESTING PURPOSES, OR CONNECTING DEVICES WITH UNAPPROVED FIRWARE

DEVICE LIFE CYCLE

This section describes the big picture of device communications with Device Server starting from when it is first powered on, completes its self-tests, and is ready for network communication.

WEBSOCKET CONNECTION

To establish a secure connection with the server, the device should be able to use TLS (HTTPS). It should use TLS only if the server URL begins with "wss://". If the URL begins with "ws://" it should use an unsecured (HTTP) connection."

When the device powers on, or if the device loses its web socket connection, then the device should attempt to make a new connection with Device Server. If the attempt fails, try again in 10 seconds. The retry time will be configurable.

See the section "Making a Connection" for details on how this is done. As noted above, the device always initiates the connection.

The server will send a 'Ping' to the device approximately every 20 seconds. The device should respond to this ping promptly with a 'Pong.' If the server does not receive a 'Pong' response within 10 seconds of sending the 'Ping,' (2 times in a row) the server will close the web socket connection.

The device can consider the presence of the server's 'Ping' as an indication that the server is still talking with the device. If a 'Ping ' does not arrive within 40 seconds, the device should close the web socket connection and then attempt repeatedly to reconnect until successful.

For more information on pings via the WebSocket connection, see "WebSocket Frames and Pings" below.

COMMUNICATION SECURITY

The device shall communicate to the cloud server via TLS. Please note that SSL is no longer a secure method, and TLS must be used (reference: https://www.rfc-editor.org/rfc/rfc7568.txt)

RECONNECTION FREQUENCY

In the event that the device loses a connection with the Cloud, the device will use "back-off algorithm" to attempt to connect with the Cloud. This algorithm proceeds as follows:

- For the first 10 minutes, the device will attempt to make a connection every 10 seconds.
- If after 10 minutes of trying the device cannot connect with the Cloud, it will then select a random value between 60 and 120 seconds and attempt to connect at that frequency indefinitely.

INCOMING REQUEST

As long as the web socket connection is open, the device shall listen for incoming requests. If a request arrives, execute it and send a response.

See the section "API to Issue Commands" for details. Remember that Device Server initiates these conversations.

The device must be able to queue commands if it cannot execute them as fast as it receives them. The queue must be able to hold a minimum of ten commands. If the command queue is full, the device can ignore additional commands until there is room in the command queue to hold them. Commands should be removed from the queue and executed in the order in which they were received.

PERIODIC UPDATES

As long as the web socket connection is open, the device shall send a status update to Device Server every 60 seconds. (The update time will be configurable. See the "Hidden Configuration Page" and 'dxSetStatusUpdateFrequency' sections below.)

See the section "API to Send Status Updates" for details. The device initiates this conversation.

EVENT NOTIFICATIONS

When the device writes an event into its event log, it should also send a notification of the event to Device Server.

See the section "API to Send Event Notifications" for details. The device initiates this conversation.

MAKING A CONNECTION

The device connects to the Device Server using WebSocket version 13, the "ovrc-protocol", a base64-encoded key value, and the device's MAC address as the Origin. The MAC address must always be formatted using uppercase hexadecimal characters with each group separated by a colon. An example of a correctly-formatted MAC address is 00:03:CE:46:84:4D.

If a device has multiple MAC addresses, as routers and some cameras do, the firmware developer must select one to be the identifying address for the device to the Cloud. For example, if a router has two WAN MAC addresses and one LAN MAC address, the LAN address may be chosen as the MAC address that identifies the device to the Cloud. A second

example would be a camera that has both a wired LAN adapter and a wireless LAN adapter. In this case, the firmware should always use the MAC address of the wired LAN adapter as its identifying address to OvrC.

If a device has both a wired LAN adapter and a wireless LAN adapter that it can switch between, then it should report the network information of the currently-operating adapter to OvrC in the "dxGetAbout" and "dxGetNetworkSettings" methods with the exception of the MAC address, which is always fixed. So, for example, if a device is set to use its wired adapter for configuration and then is switched to wireless for operation, it should report the IP address etc. of the wired adapter to OvrC until it is switched to wireless. Then it should report the IP address etc. of the wireless adapter. The "dxSetNetworkSettings" method will also set the parameters for the currently-operating adapter.

A base64-encoded key value is required, though at the present time, it can be generated from a 16-byte buffer of random values. The following JavaScript code creates such a buffer:

```
var nonce = new Buffer(16);
for (var i=0; i < 16; i++) {
    nonce[i] = Math.round(Math.random()*0xFF);
}
```

This buffer would then be base64-encoded as the key value.

The handshake to establish a connection begins with the device sending what appears to be an HTTP GET to the server's URL and port. The content of the header is as follows:

```
GET <path> HTTP/1.1
Host:  <host>
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: <base64-encoded value>
Sec-WebSocket-Protocol: <protocol name>
Sec-WebSocket-Version: 13
Origin: <client info>
```

For example, for a device with MAC address 00:03:CE:46:84:4D to connect to the firmware test server using TLS and a key value of "x3JJHMbDL1EzLkh9GBhXDw==" the HTTP header would look this way:

```
GET / HTTP/1.1
Host: firmware-test.connectedsystems.org:443
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: ovrc-protocol
Sec-WebSocket-Version: 13
Origin: 00:03:CE:46:84:4D
```

Device Server responds with the following header info:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: ovrc-protocol
X-Server-Id: DEVICE_SERVER
```

The way Device Server generates the accept value is as follows:

- It concatenates "258EAFA5-E914-47DA-95CA-C5AB0DC85B11" to the base64-encoded key sent by the device.
- The string is then hashed with SHA-1 and base64 encoded.
- It returns this string as the value of Sec-WebSocket-Accept.

The device generates the same accept value and compares it with the one returned from Device Server. If the two accept values don't match, the device closes the connection. Otherwise, the device begins listening for commands and sending its periodic updates to the server.

See Appendix for a diagram showing this connection sequence.

WEBSOCKET FRAMES AND PINGS

A WebSocket message is sent via frames. Each frame has an operation code (opcode) and other flags to help the recipient decode the payload it carries.

See the following document for details regarding the WebSocket specification: https://tools.ietf.org/html/rfc6455. Also see the following source code for the WebSocket libraries that are used by OvrC: https://github.com/Worlize/WebSocket-Node and https://github.com/einaros/ws.

The OvrC Device Server pings a device every 20 seconds to see if it is still connected. A ping frame is described in the above document and source code. The device should promptly respond with a pong frame, as described in this document.

Each data frame the device receives contains header information. The header is not part of the data itself, but it is useful to describe the data. For example, a header code indicates whether the data is text or binary, and one or more bytes give the length of the data. A header flag indicates whether the message is fragmented and will be transmitted via multiple frames. Please refer to these resources to identify and be able to act on the frame header information.

JSON

Each web socket message in OvrC is comprised of a JSON string that is formatted specifically for OvrC cloud communications (hence, "ovrc-protocol"). The content of the JSON string indicates what action the device needs to take and it provides all of the parameters required for the requested action. The JSON string also includes fields that confirm the identity of the device (its MAC address) and a unique id to aid in pairing up asynchronous requests and responses.

JSON is an object comprised of key value pairs. Please reference www.json.org for more information regarding JSON formatting and data type syntax. JSON objects should not contain any formatting characters, such as carriage returns or tabs.

For example, a JSON object will be sent as such:

```
{"jsonrpc":"2.0","id":"f3363ed5-2585-2bea-9420-0c5eeaeb337d|f3363ed5-2585-2bea-9420-
0c5eeaeb337d|1","result":{"deviceId":"DD:DD:DD:DD:DD:DD","macAddress":"DD:DD:DD:DD:DD:DD"
,"firmware":"1.0.19.3","model":"WPS-300-BUL-
IP","serviceTag":"ST731212619","lanAddress":"192.68.1.110"}}
```

***Please note that all parameters included in the JSON message are case sensitive. For example: serviceTag and ServiceTag are not the same thing.

API TO ISSUE COMMANDS

To request that the device take an action, return the current status, etc., the Device Server will send a JSON string with the following format:

```
{
  "jsonrpc": "2.0",
  "method": "<the name of the method to execute",
  "id": "<an identifying string for this request>",
  "params":
  {
    "deviceId": "<the device's MAC address>",
```

```
    "version": <the API version number>,
    <additional parameters required by the method>
  }
}
```

The `id` is a string that the device echoes back to the server in its response. This helps the server to pair up the response it receives with the request it sent. The maximum length of the `id` field is eighty (80) characters. (UI developers should know that the Device Server will add 37 of these characters.)

The presence of the `deviceId` in the params is intended for the device to confirm that this command is intended for this device.

If the `version` field is absent, default to 0 (zero). Firmware updates must support backward compatibility for previous API versions.

If the device successfully executes the method, it will respond with a JSON string like the following:

```
{
  "jsonrpc": "2.0",
  "id": "<echo the id from the request>",
  "result":
  {
    "deviceId": "<the device's MAC address>",
    <additional data depending on the method>
  }
}
```

If there is a problem with the request and the device cannot execute the method, it will respond with a JSON string like the following:

```
{
  "jsonrpc": "2.0",
  "id": "<echo the id from the request>",
  "error":
  {
    "deviceId": "<the device's MAC address>",
    "code": <an error code as defined below>,
    "message": "<a short error message>",
    "data": "<any extra information about the error>"
  }
}
```

If the device receives a packet that has more fields than necessary to perform the requested method, then the device should simply ignore the surplus fields. That is, having more fields than necessary is not an error condition.

ERROR CODES AND MESSAGES

An error code for a response follows the below structure:

Error Messages		
Response { "jsonrpc": "2.0", "id": *id* "error": { "deviceId": *deviceId,* "code": *code,* "message": *message,*	**Field Information**	
	Field Name	**Description**
	jsonrpc	This is the version of JSON RPC.
	id	This field contains a string that the device echoes back to the server in its response.
	deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this

"data": *data* } }		device.
	code	This field has the error code that is listed below.
	message	This is a short error message. See below for content.
	data	This field is used for any extra information about the error. (optional)
Error Codes		
Code	**Message**	**Meaning**
-32700	Parse error	Invalid JSON was received
-32600	Invalid Request	The JSON 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
10001	Action not taken	Cannot execute request because the device is not in an appropriate state
10002	Maximum exceeded	Too many items in a list
10003	Invalid value	A field does not have a correct or acceptable value.
10100	Not a UPnP router	The router does not support UPnP, hence it will not open a port for remote access.
10101	Router refused request	The router supports UPnP, but it could not open or close the port as requested.
10102	Protocol not supported	The device does not support the protocol requested.
10120	Download failed	The device was instructed to download a file (such as a firmware update) and the download failed.
10121	Download cancelled	The device cancelled the download, perhaps because a timeout expired.
10122	Upload failed	The device was instructed to upload a file, but the upload failed.
10123	Upload cancelled	The device cancelled the upload, perhaps because a timeout expired.
Additional API-specific error codes are located in the Standard Methods section.		

If the device is asked to take an unnecessary action, such as to turn on an output when the output is already on, the device should return a success result.

The below sections provide details about the standard methods required to be supported by all OvrC-enabled devices. Collectively, this set of standard methods comprises what is called the “Base Specification”. These method names begin with “ds” or “dx”. All OvrC-enabled devices must support these methods.

IV: STANDARD DEVICE-SENT METHODS (DS)

The following device-sent methods originate from the device. They are sent to the cloud service. In other words, these methods are sent by the device unsolicited.

dsLogStatusUpdates
version 0

Devices must report periodic status updates to the Cloud Server. The default delay between status updates is 60 seconds, but can be configured to send a status update to the server at a set interval (see ‘dxSetStatusUpdateFrequency’ method). The data in a status update should be as real-time as possible and be time-stamped.

The device-specific content of this method shall be defined in a separate device-specific supplemental specification.

Request (from Device, to Server)	Response (from Server, to Device)
{ “jsonrpc”: “2.0”, “method”: “dsLogStatusUpdates”, “params”: { “deviceId”: *deviceId* “version”: *version* “updates”: [{ “dateTime”: *dateTime,* “uptime”: *uptime* *<other parameters defined in the device-specific supplemental specification>* }, { “dateTime”: “<date & time stamp in ISO format>”, “uptime”: *uptime* *<other parameters defined in the device-specific supplemental specification>* }] } }	There will be no response from the server for automatic status updates.

Usage Notes

This method shall be sent by the device to the Device Server on a regular basis.

Note that this API allows a device to send more than one update in the same message. This allows the device to queue status updates when there is no connection to the server and to send them when the connection is re-established. The device will queue a maximum of 10 items. In the case that more than 10 status updates are queued, the oldest item shall be removed from the queue to make room for the newest item.

Additional parameters may be defined at a later date and included in the status update.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**

jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	Value must be 0.	Required
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
dateTime	This field returns date and time of the status.	ISO Date		Required
uptime	This field contains the uptime as reported by the device. Uptime is the length of time the device has been running since last reboot	string	Value should be returned as: “X Day(s) XHr XMin”	Required

dsEventNotifications
version 1

The device should send notifications to the server when those events are written to its system/event log. Please ask SnapAV if there is any question as to what events to send. Generally, all system log events should be reported.

Request (from Device, to Server)

```
{
  “jsonrpc”: “2.0”,
  “method”: “dsEventNotifications”,
  “params”:
  {
    “deviceId”: deviceId
    “version”: version
    “events” :
    [
      {
        “number” : number
        “dateTime” : dateTime
        “type” : type
        “event” : event
      },
      {
        “number” : number
        “dateTime” : dateTime
        “type” : type
        “event” : event
      }
    ]
  }
}
```

Response (from Device, to Server)

There will be no response from the server.

Usage Notes

Note that this API allows a device to send more than one notification in the same message. This will allow the device to

queue up notifications when there is no connection to the server and to send them when the connection is re-established. The device will queue a maximum of 10 items on a FIFO basis.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version *This field was previously not included.	This field contains the API version number.	number	Version must be 1.	Required
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
dateTime	This field indicates the date and time of when the event occurred.	ISO date/time		Required
number	This field contains the event log ID. This ID is assigned by the device, usually in sequential ascending order.	number		Required
type	This field indicates the type of the log event.	string	“Debug” “Info” “Warning” “Error” “Critical”	Required
event	This field contains the description of the event. Generally, this is the text that is written in the device’s system log.	string		Required

dsGetPublicAddress
version 0

The device should send this method to the server when it wants the server to return the device’s public IP (WAN) address and port number.

Request (from Device, to Server)	Response (from Server, to Device)
{ “jsonrpc”: “2.0”, “method”: “dsGetPublicAddress”, “id”: *id*, “params”: { “deviceId”: *deviceId,*	{ “jsonrpc”: “2.0”, “id”: *id* “result”: { “deviceId”: *deviceId,*

“version”: *version* } }	“ipAddress”: *ipAddress*, “port”: *port* } }
Usage Notes	

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the server echoes back to the hub in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number		Required
deviceId	This field contains the MAC address of the device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
ipAddress	This field indicates the hub’s public IP address.	number	Value must be in this format: X.X.X.X	Required
port	This field indicates the hub’s public port number.	number		Required

dsGetCurrentTime
version 0

The device should send this method to the server when it wants the server to return the device’s current time. The current time can be returned either in UTC or local time, whichever the device prefers.

Request (from Device, to Server)	**Response (from Server, to Device)**

{
“jsonrpc”: “2.0”,
“method”: “dsGetCurrentTime”,
“id”: *id*,
“params”:
{
“deviceId”: *deviceId,*
“version”: *version*,
“timezone”: *timezone*
}
}
	{
	“jsonrpc”: “2.0”,
	“id”: *id*
	“result”:
	{
	“deviceId”: *deviceId,*
	“dateTime”: *dateTime*,
	“zoneAbbr”: zoneAbbr
	}
	}

Usage Notes	

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required

id	This field contains a string that the server echoes back to the hub in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number		Required
deviceId	This field contains the MAC address of the device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
timezone	This optional field indicates whether the device wants the response to be UTC or the device's local time. The field should be omitted if the device wants to receive the time in UTC.	number	If present, this value needs to be the IANA timezone designation. Example values: "America/New_York" "America/Chicago" "America/Denver" "America/Pheonix" "America/Los_Angeles" "America/Anchorage" "Pacific/Honolulu"	Optional
dateTime	This field contains the date and time.	string	Value is in ISO format. For example: "2018-05-31T11:47:59-04:00"	Required
zoneAbbr	This optional response is the abbreviation of the time zone.	string	If present, the value will be the standard abbreviation, such as: "EST", "EDT", "CST", "CDT", "MST", "MDT", "PST", "PDT", "IST"	Optional

V. STANDARD DEVICE-RECEIVED METHODS (DX)

The following device-received methods originate from the cloud server. They are sent to the device. Most methods include a “get” or “set” in the name to indicate whether the server is requesting information (get) or sending updated information to the device (set).

dxGetAbout
version 0

The dxGetAbout method instructs the device to return information about itself.

Request (from Server, to Device)	Response (from Device, to Server)
{ “jsonrpc”: “2.0”, “method”: “dxGetAbout”, “id”: *id* “params”: { “deviceId”: *deviceId* “version”: *version* } }	{ “jsonrpc”: “2.0”, “id”: *id* “result”: { “deviceId”: *deviceId* “macAddress”: *macAddress* “firmware”: *firmware* “platform”: *platform* “model”: *model* “serialNum”: *serialNum* “serviceTag”: *serviceTag* “lanAddress”: *lanAddress* “p2pID”: *p2pID* “p2pPassword” : *p2pPassword* “hasUPS”: *hasUPS* } }

Usage Notes

This method shall be sent by the Device Server to a device soon after a connection has been established with a device.

If a device has both a wired LAN adapter and a wireless LAN adapter that it can switch between, then it should report the network information of the currently-operating adapter to OvrC in the response with the exception of the MAC address, which is always fixed.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is	string	Value must include colons between octets. All characters shall be uppercase. Example:	Required

	intended for this device.		12:34:56:78:90:AB	
macAddress	This field returns the device MAC address.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
firmware	This field returns the firmware version currently installed on the device.	string		Required
platform	This field returns information regarding the device platform. This field can be used to distinguish between devices that have the same model number but which require different firmware, based on the chipset.	string		Optional
model	This field returns the model number of the device.	string		Required
serialNum	This field returns the device serial number.	string		Required
serviceTag	This field returns the service tag of the device.	string	This is the SnapAV service tag number (must begin with “ST”)	Required
lanAddress	This field returns the IP address of the device on the local network.	string		Required
p2pID	The P2P Identification number. Used to identify Point-to-Point streaming devices.	string	Used only by Point-to-Point streaming devices.	Optional
p2pPassword	The P2P password.	string	Used only by Point-to-Point streaming devices.	Conditional (required only if ‘p2pID’ is present)
hasUPS	This field indicates whether a WattBox has a UPS attached.	boolean	Value should be one of the following: true (UPS attached) false (UPS not attached)	Required by WattBox with a UPS attached

dxGetNetworkSettings
version 1

The dxGetNetworkSettings method instructs the device to return its network settings. These are the current settings for the device.

Request (from Server, to Device)	Response (from Device, to Server)
{ “jsonrpc”: “2.0”, “method”: “dxGetNetworkSettings”, “id”: “*id* “params”: { “deviceId”: *deviceId* “version”: *version*	{ “jsonrpc”: “2.0”, “id”: *id* “result”: { “deviceId”: *deviceId* “deviceName”: *deviceName* “deviceIpAddress”: *deviceIpAddress*

} }	"deviceSubnetMask": *deviceSubnetMask* "deviceDefaultGateway": *deviceDefaultGateway* "dhcpEnabled: *dhcpEnabled* "dnsServer1": *dnsServer1* "dnsServer2": *dnsServer2* "webPagePort": *webPagePort* } }

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

If a device has both a wired LAN adapter and a wireless LAN adapter that it can switch between, then it should report the network information of the currently-operating adapter to OvrC in the response with the exception of the MAC address, which is always fixed.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
deviceName	This field returns the name of the device. This value should be the same as the name the device broadcasts on the local network.	string		Required
deviceIpAddress	This field returns the local IP address of the device.	string		Required
deviceSubnetMask	This field returns the Subnet Mask used on the device.	string		Required
deviceDefaultGateway	This field returns the Default Gateway address used on the device.	string		Required
dhcpEnabled	This field will be True if DHCP is enabled on the device. The field will be False if DHCP is disabled on the device. When dhcpEnabled is	Boolean	Value must be one of the following: true false	Required

	true, the device will obtain its IP address from a DHCP server. When it is false, the device will use the provided static IP address.			
dnsServer1	This field returns the address of the DNS server.	string		Required
dnsServer2	This field returns the address of the secondary DNS server.	string	If the device does not support a secondary DNS server, it may ignore this parameter.	Optional
webPagePort	This field returns the port number that the device is broadcasting it's local Web GUI (example: 80)	number	0 - 65535	Required

dxSetNetworkSettings
version 1

The dxSetNetworkSettings method instructs the device to set its network settings.

Request (from Server, to Device)

```
{
  "jsonrpc": "2.0",
  "method": "dxSetNetworkSettings",
  "id": id
  "params":
  {
    "deviceId": deviceId
    "version": version
    "deviceName": deviceName
    "deviceIpAddress": deviceIpAddress
    "deviceSubnetMask": deviceSubnetMask
    "deviceDefaultGateway": deviceDefaultGateway
    "dhcpEnabled: dhcpEnabled
    "dnsServer1": dnsServer1
    "dnsServer2": dnsServer2
    "webPagePort": webPagePort
  }
}
```

Response (from Device, to Server)

```
{
  "jsonrpc": "2.0",
  "id": id
  "result":
  {
    "deviceId": deviceId
  }
}
```

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.
For Optional fields, if a value is not sent by the server, the device should assume no change to that value from the current value.

Upon receiving the command, the device should update it's Network/IP settings, close the web socket connection, reboot the network stack (if required), and re-connect the web socket connection to the cloud server.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the	string	Value must be "2.0".	Required

	version of JSON RPC.			
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
deviceName	This field returns the name of the device. This value should be the same as the name the device broadcasts on the local network.	string		Optional
deviceIpAddress	This field returns the local IP address of the device.	string		Optional
deviceSubnetMask	This field returns the Subnet Mask used on the device.	string		Optional
deviceDefaultGateway	This field returns the Default Gateway address used on the device.	string		Optional
dhcpEnabled	This field will be True if DHCP is enabled on the device. The field will be False if DHCP is disabled on the device. When dhcpEnabled is true, the device will obtain its IP address from a DHCP server. When it is false, the device will use the provided static IP address.	Boolean	Value must be one of the following: true false	Optional
dnsServer1	This field returns the address of the DNS server.	string		Optional
dnsServer2	This field returns the address of the secondary DNS server.	string	If the device does not support a secondary DNS server, it may ignore this parameter. If the device does support a secondary DNS server, but this field is not sent, or send	Optional

			blank, then the device should ignore it and continue to use the last-known value.	
webPagePort	This field returns the port number that the device is broadcasting it's local Web GUI (example: 80)	number	0 - 65535	Optional

dxResetDevice
version 0

The dxResetDevice method instructs the device to perform a soft reset. This is NOT a factory reset – it is a reboot of the device.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxResetDevice",
"id":
"params":
{
"deviceId": *deviceId*
"version": *version*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.
IMPORTANT: The device must first send a response before executing the command and rebooting.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required

dxEnableWebConnect
version 0

The dxEnableWebConnect method instructs the device to cooperate with OvrC to provide an SSH tunnel so a remote

user can access the device's web server.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxEnableWebConnect",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
"tunnelPort": *tunnelPort*
"sshServer": *sshServer*,
"devicePort": *devicePort*,
"key": *key*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*,
	"folder": *folder*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

OvrC will initiate an SSH tunnel prior to sending this method to the device. Included in the device parameters is the IP address of the tunnel server and the port the device should use when establishing the SSH connection.

SnapAV will provide a private key file for the firmware to use to make an SSH connection to the tunnel server.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
sshServer	This field contains the IP address or domain name of the tunnel server in the OvrC cloud that the device should use to establish the SSH tunnel.	string	Value will be in a typical IP4 format or will be a domain name.	Required
tunnelPort	This field contains the port the tunnel server is listening on for a connection from the device.	number		Required
devicePort	This field contains the port on the device that the server wants to access.	number	If present, the value will be like: • 80 • 443	Optional
key	This field contains the private key to use for SSH tunneling.	string	If present, the value will be in RSA format.	Optional

folder	This field contains a folder that should be added to the browser's URL.	string	If present, the value can be something like: • "/" • "/config"	Optional

Usage Notes

If the device returns a value in the "folder" field, the server will build a URL for the browser that includes the value of this field.

For example, say the device returns a value of "/config" in the "folder" field. The server will construct a URL for the browser in the following format:

<protocol>://<tunnel server domain>:<port for UI side of tunnel><folder>

A typical URL for the browser might look like the following:

http://webconnect.ovrc.com:6100/config?id=4WmId8w7kOv6Kqip

dxDisableWebConnect
version 0

The dxDisableWebConnect method instructs the device to close the SSH tunnel that a remote program uses to access the device's web server.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxDisableWebConnect",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
"tunnelPort": *tunnelPort*
"sshServer": *sshServer*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

The device should close the SSH tunnel for the given server IP and tunnel port.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present,	Optional

			its value must be 0.	
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
sshServer	This field contains the IP address of the tunnel server in the OvrC cloud that the device should use to establish the SSH tunnel.	string	Value will be in a typical IP4 format.	Required
tunnelPort	This field contains the port the tunnel server is listening on for a connection from the device.	number		Required

dxUpdateFirmware
version 0

The dxUpdateFirmware method instructs the device to download and install a firmware file.

Request (from Server, to Device)	Response (from Device, to Server)

{
“jsonrpc”: “2.0”,
“method”: “dxUpdateFirmware”,
“id”: *id*
“params”:
{
“deviceId”: *deviceId*
“version”: *version*
“url”: *url*
}
}
	{
	“jsonrpc”: “2.0”,
	“id”: *id*
	“result”:
	{
	“deviceId”: *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

The server will send the device the URL of the firmware file in the ‘url’ parameter. The device shall download the firmware using one of these methods: HTTPS (Preferred method) or HTTP. The URL in the ‘url’ parameter will always begin with “https://”. If the device cannot download the file via HTTPS, it may instead download this file via HTTP.

The device must send an immediate response (ACK) to this method before initiating the download. The device should not wait for the download to complete before sending the response.

IF DOWNLOAD AND UPGRADE IS SUCCESSFUL: Device should disconnect and reconnect to the Cloud Service. The Cloud Service will recognize that the new firmware version is higher than the previous firmware version and deem the upgrade was successful.

IF DOWNLOAD AND/OR UPGRADE IS UNSUCESSFUL: Device should disconnect and reconnect to the Cloud Service. The Cloud Service will recognize that the new firmware version is the same as the previous firmware version and deem the upgrade failed.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required

method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
url	This field contains the URL of the firmware file.	string	The supplied URL will always begin with "https"	Required
saveAndRestore	This is an old/deprecated parameter which is no longer used. It should be ignored.	boolean	No longer used	Optional

dxGetStatusUpdateFrequency
version 0

The dxGetStatusUpdateFrequency method instructs the device to return the frequency at which it sends "dsLogStatusUpdates" messages. This API allows someone via the cloud UI to do what a user can do via the hidden web page.

Request (from Server, to Device)	**Response (from Device, to Server)**

{
"jsonrpc": "2.0",
"method": "dxGetStatusUpdateFrequency",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	"frequency": *frequency*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present,	Optional

			its value must be 0.	
frequency	This field is the interval between device updates to the server.	number	Value must be returned in seconds.	Required

dxSetStatusUpdateFrequency
version 0

The dxSetStatusUpdateFrequency method instructs the device to set the frequency at which it sends "dsLogStatusUpdates" messages. This API allows someone via the cloud UI to do what a user can do via the hidden web page discussed below.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxSetStatusUpdateFrequency",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
"frequency": *frequency*
}
}
	{
	"jsonrpc": "2.0",
	"id":, *id*
	"result":
	{
	"deviceId": *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis. It would generally be sent soon after initiating a connection with the Device Server, if needed.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
frequency	This field is the interval between device updates to the server.	number	Value must be supplied in seconds. A value of 0 means that status updates should be paused indefinitely.	Required

dxGetTimeSettings
version 1

This method is not required for most devices. Please reference your supplemental spec to see if it is required for your device. (If we have not provided you a supplemental spec, then do not implement this method.)

The dxGetTimeSettings method instructs the device to return its system time configuration.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxGetTimeSettings",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	"timeZone": *timeZone*
	"currentTime": "*currentTime*
	"daylightSavings": *daylightSavings*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
timeZone	This field indicates the time zone as the difference from Greenwich Mean Time.	string	Value must be returned in the following format: "±H:MM" There shall be no leading zero on hours.	Required
currentTime *This field was previously named manualEntry in an older version of this spec	This field indicates the current time of the device.	ISO date	Example: 2015-10-21T05:41:36-04:00	Required
daylightSavings	This field indicates whether or not the device should automatically adjust the time during daylight savings.	string	Value must be one of the following*: "enabled"	Required

			“disabled” *These values differ from an older version of this spec	

dxGetTimeSettings
version 2

This method is not required for most devices. Please reference your supplemental spec to see if it is required for your device. (If we have not provided you a supplemental spec, then do not implement this method.)

The dxGetTimeSettings method instructs the device to return its system time configuration.

Request (from Server, to Device)	**Response (from Device, to Server)**

{
“jsonrpc”: “2.0”,
“method”: “dxGetTimeSettings”,
“id”: *id*
“params”:
{
“deviceId”: *deviceId,*
“version”: *version*
}
}
	{
	“jsonrpc”: “2.0”,
	“id”: *id*
	“result”:
	{
	“deviceId”: *deviceId,*
	“name”: *name*,
	“notes”: *notes,*
	“offset”: *offset*,
	“currentTime”: *currentTime*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
name	This field indicates the name of the time zone per the IANA Time Zone Database.	string	Example values: “America/New_York” “America/Chicago” “America/Denver” “America/Pheonix” “America/Los_Angeles” “America/Anchorage”	Required

			“Pacific/Honolulu”	
notes	This field contains an alternative display string for the UI.	string	Examples: “Eastern Time (US & Canada)” “Central Time (US & Canada)”	Required
offset	This field contains the offset of the time zone from UTC. This value indicates how far ahead or behind UTC is from the local time. The value is in minutes.	number	Examples: 300 (for Eastern Time) 360 (for Central Time)	Required
currentTime	This field indicates the current time of the device.	ISO date	Example: 2015-10-21T05:41:36-04:00	Required

dxSetTimeSettings
version 1

This method is not required for most devices. Please reference your supplemental spec to see if it is required for your device. (If we have not provided you a supplemental spec, then do not implement this method.)

The dxSetTimeSettings method instructs the device to set its system time configuration.

Request (from Server, to Device)	Response (from Device, to Server)

{{
“jsonrpc”: “2.0”,
“method”: “dxSetTimeSettings”,
“id”:,*id*
“params”:
{
“deviceId”: *deviceId*
“version”: *version*
“timeZone”: *timeZone*
“currentTime”: *currentTime*
“daylightSavings”: *daylightSavings*
}
}
	{
	“jsonrpc”: “2.0”,
	“id”: *id*
	“result”:
	{
	“deviceId”: *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional

deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
timeZone	This field indicates the time zone as the difference from Greenwich Mean Time.	string	Value must be returned in the following format: "±H:MM" There shall be no leading zero on hours.	Optional
currentTime *This field was previously named manualEntry in an older version of this spec	This field indicates the current time of the device. In most cases, it is currently not used to set the current time on the device – the device should use NTP to get current time.	ISO date	Example: 2015-10-21T05:41:36-04:00	Optional
daylightSavings	This field indicates whether or not the device should adjust the time during daylight savings.	string	Value must be one of the following*: "enabled" "disabled" *These values differ from an older version of this spec	Optional

dxSetTimeSettings
version 2

This method is not required for most devices. Please reference your supplemental spec to see if it is required for your device. (If we have not provided you a supplemental spec, then do not implement this method.)

The dxSetTimeSettings method instructs the device to set its system time configuration.

Request (from Server, to Device)	Response (from Device, to Server)

{
"jsonrpc": "2.0",
"method": "dxSetTimeSettings",
"id":,*id*
"params":
{
"deviceId": *deviceId*
"version": *version*
"name": *name*,
"notes": *notes,*
"offset": *offset*,
"currentTime": *currentTime*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information				
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version	string	Value must be "2.0".	Required

	of JSON RPC.			
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 1. If this field is present, its value must be 1.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
name	This field indicates the name of the time zone per the IANA Time Zone Database.	string	Example values: “America/New_York” “America/Chicago” “America/Denver” “America/Pheonix” “America/Los_Angeles” “America/Anchorage” “Pacific/Honolulu”	Required
notes	This field contains an alternative display string for the UI.	string	Examples: “Eastern Time (US & Canada)” “Central Time (US & Canada)”	Required
offset	This field contains the offset of the time zone from UTC. This value indicates how far ahead or behind UTC is from the local time. The value is in minutes.	number	Examples: 300 (for Eastern Time) 360 (for Central Time)	Required
currentTime	This field indicates the current time of the device. In most cases, it is currently not used to set the current time on the device – the device should use NTP to get current time or call the dsGetCurrentTime method.	ISO date	Example: 2015-10-21T05:41:36-04:00	Optional

dxSetCloudServerUrl
version 0

The dxSetCloudServerUrl method instructs the device to set the URL of the cloud server connection.

Request (from Server, to Device)	Response (from Device, to Server)

{{
“jsonrpc”: “2.0”,
“method”: “dxSetCloudServerUrl”,
“id”: *id*
“params”:
{
	{
	“jsonrpc”: “2.0”,
	“id”:, *id*
	“result”:
	{
	“deviceId”: *deviceId*

“deviceId”: *deviceId* “version”: *version* “url”: *url* “port”: *port* “connection”: *connection* } }	} }

Usage Notes

This method shall be sent by the Device Server to a device on an ad-hoc basis.

Field Information

Field Name	Description	Format	Constraints	Condition
jsonrpc	This field contains the version of JSON RPC.	string	Value must be “2.0”.	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required
url	This field indicates the URL of the cloud server.	string	Value must begin with the protocol, which is one of the following values: wss;// ws:// (See Hidden Configuration Page section above for more information)	Required
port	This field is the port number to connect to the cloud server.	string		Required
connection	Specifies if the device should communicate to the cloud server. Setting this to “disabled” will cause the device to stop communication to the cloud server until a Factory Reset is performed, or the user manually changes it to “enabled” on the device’s ‘webservice.htm’ page.	Boolean	“enabled” “disabled”	Optional

dxDisableCloud
version 0

The dxDisableCloud method instructs the device to turn off its OvrC capabilies. This method is sent from the cloud if a particular device should not provide OvrC connectivity. For example, the cloud may determine that the device does not belong to a list of supported devices, and therefore it shouldn't be connected to the cloud.

Request (from Server, to Device)	**Response (from Device, to Server)**

{
"jsonrpc": "2.0",
"method": "dxDisableCloud",
"id": *id*
"params":
{
"deviceId": *deviceId*
"version": *version*
}
}
	{
	"jsonrpc": "2.0",
	"id": *id*
	"result":
	{
	"deviceId": *deviceId*
	}
	}

Usage Notes

This method shall be sent by the Device Server to a device soon after a connection has been established with a device.

If a device receives this command, it should permanently close its WebSocket connection to OvrC and hide any OvrC-related verbiage, buttons, etc. from its local user interface. The device should remember that it is no-longer OvrC-enabled, even when it reboots.

Field Information

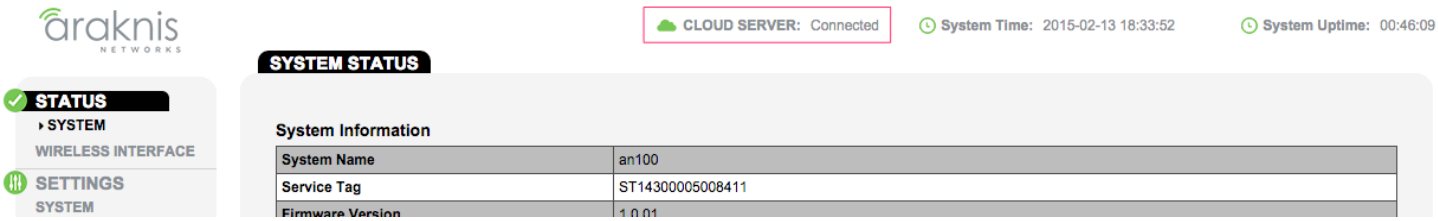
Field Name	**Description**	**Format**	**Constraints**	**Condition**
jsonrpc	This field contains the version of JSON RPC.	string	Value must be "2.0".	Required
method	This field lists the API used in this request.	string		Required
id	This field contains a string that the device echoes back to the server in its response.	string	Value has maximum length of 80 characters.	Required
version	This field contains the API version number.	number	If this field is absent, then assume a value of 0. If this field is present, its value must be 0.	Optional
deviceId	This field contains the MAC address of the device. This is intended for the device to confirm that this command is intended for this device.	string	Value must include colons between octets. All characters shall be uppercase. Example: 12:34:56:78:90:AB	Required

VI: DEVICE LOCAL UI REQUIREMENTS

CLOUD CONNECTION STATUS

The device shall have a web user interface which is assessable to users within the network. Within this interface shall be shown the “Cloud Service” status, which shall be shown as either “Connected” or “Not Connected” (“Not Connected” text color should be red), based on whether the device has a connection established with the OvrC Cloud Server.

Here is an example of how this appears on some products:



Note: error in the above screenshot. “CLOUD SERVER” should say “CLOUD SERVICE”

The words “CLOUD SERVICE” should be a hyperlink, that when clicked by the user links to this URL: www.OvrC.com.

WEBSERVICE CONFIGURATION PAGE

The device’s web user interface will have a page for developer/testing use. This page will only be assessable if a developer/tester types in the exact URL: **<device_ip_address>/webservice.htm** . This page shall allow the developer/tester to change the following:

- Cloud Service – Enabled / Disabled
  - This shall default to Enabled (and will reset to Enabled upon a factory reset)
  - If disabled, the device will not attempt to communicate to the cloud service

- Cloud Service URL
  - This shall default to: **wss://cloud.ovrc.com**.
  - Do NOT use wss://cloud.ovrc.com for development/testing purposes. For development/testing purposes, the device should be pointed to wss://firmware-test.connectedsystems.org.

- Cloud Service Port
  - The port number shall default to **443**.
  - Do NOT use port 443 for development/testing purposes. For development/testing purposes, the device should be pointed to the port number provided by SnapAV. (contact SnapAV for this port number)
  - For development purposes, a non-secure connection, ws:// may be used in place of wss://. Please ask SnapAV for the port which supports non-secure connections.

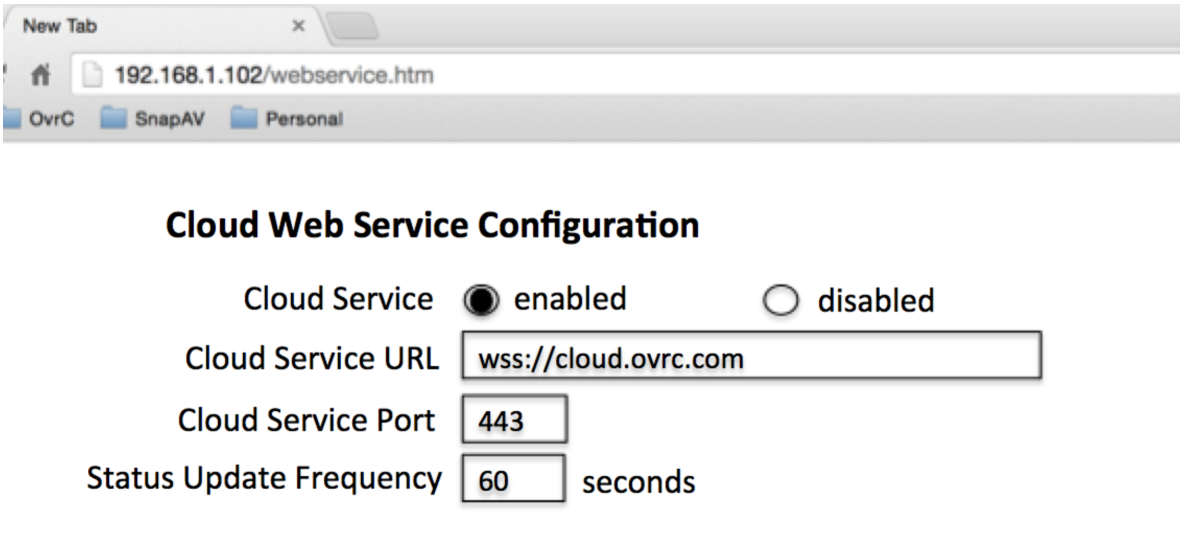
- Status Update Frequency
  - This specifies how often ‘dsLogStatusUpdates’ methods are sent by the device
  - The default shall be **60 seconds**.

Requirements of the Webservice Configuration page:

- Only assessable to user by typing in the correct URL
- Require admin authentication (the same authentication to view the device’s local UI) to view it
- There should be no navigation link to it from any of the existing web pages on the device local user interface.
- The URL should not be advertised in the user manual or other documentation – it is for developer/tester use only

- The name of the hidden page is “webservice.htm”. The format of the URL: **<device_ip_address>/webservice.htm**

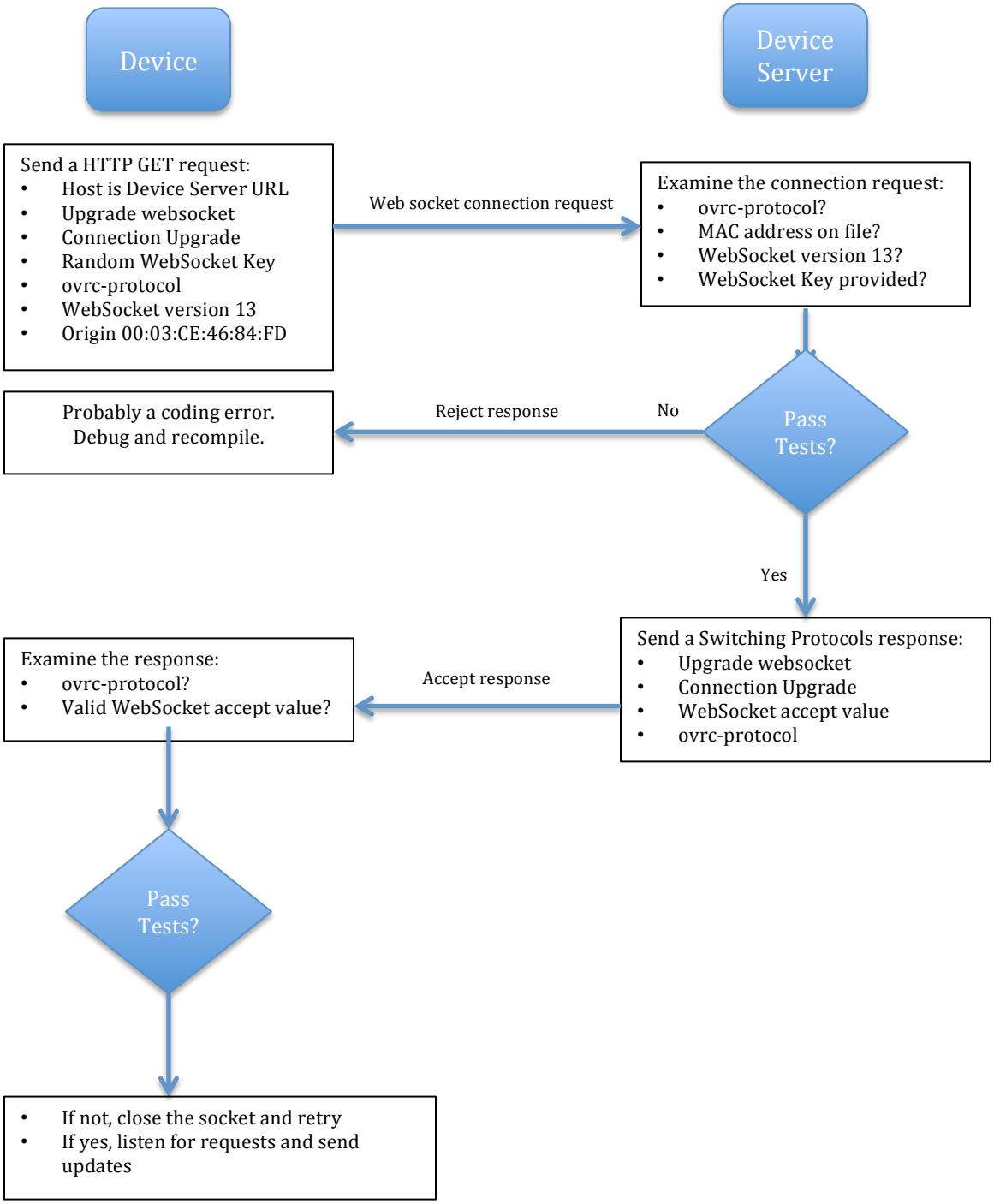
EXAMPLE WEBSERVICE CONFIGURATION PAGE:



VII. APPENDIX

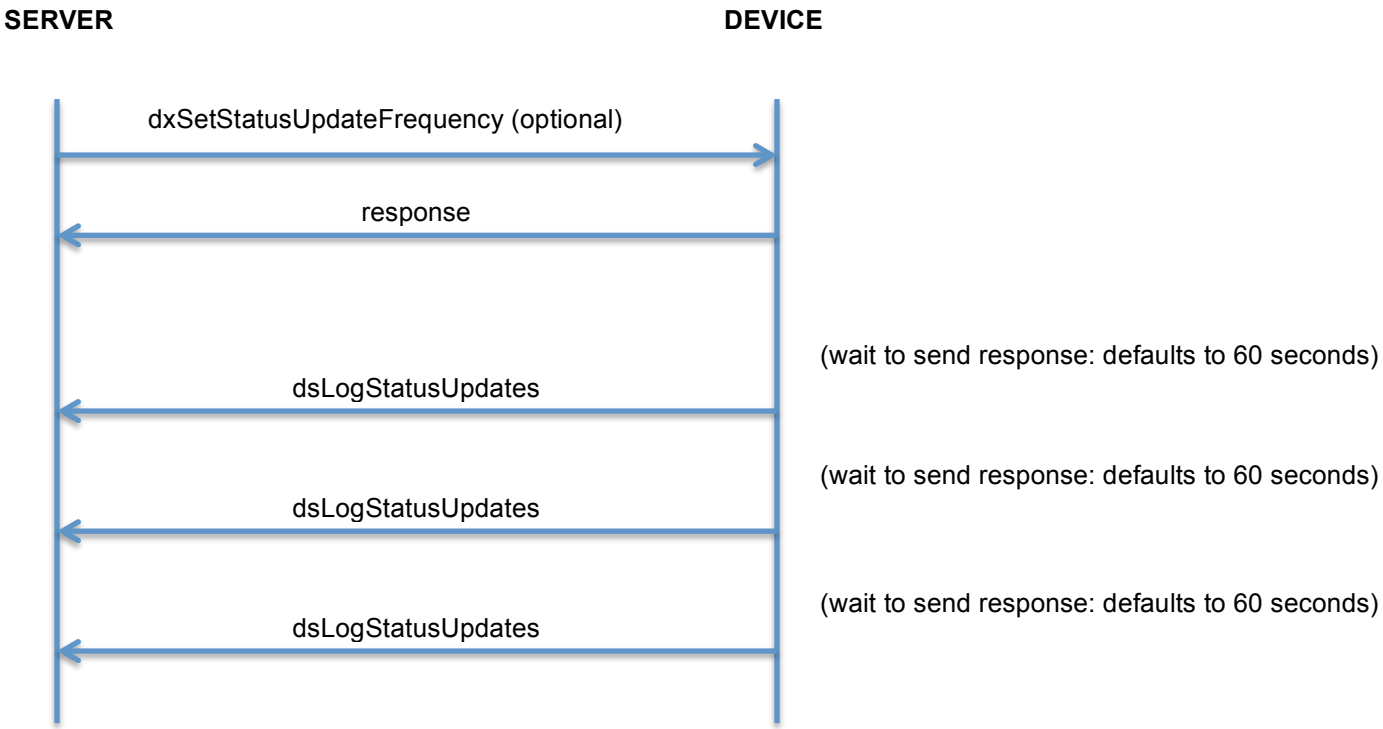
Appendix A: Websocket Connection Process Flow

Diagram of a device with MAC address 00:03:CE:46:84:FD Connecting to Device Server



Appendix B: Communication Process Flows

dsLogStatusUpdate Process Flow



Additional process flows will be added at a later date.

Appendix C: Firmware Naming and Download

In order to support firmware downloads on a device, please note the following information:

A. Firmware files must be named in the following format:

   ModelNumber(XX-XXX-XXXXXX)_Version(A.B.C.D)_Date(YYYYMMDD).xxx

   Example: AN-300-SW-8-POE_0.2.0.0_20141113.dat

B. Once a firmware file has been delivered to SnapAV, it will be hosted at the following locations:

   HTTPS:
   https://s3.amazonaws.com/snapav-firmware/CATEGORY/MODEL/FIRMWARE_FILE

   Example: https://s3.amazonaws.com/snapav-firmware/Network/AN300-Switch/AN-300-SW-8-POE_0.2.0.0_20141113.dat

Appendix D: SnapAV Contact Information

For any questions regarding this document, please reach out to the following team members at SnapAV:

Kenny Kim, Product Manager
Email: Kenny.Kim@snapav.com

Gregg Kellum, Server Architect
Email: Gregg.Kellum@snapav.com

Jessica Temple, Business Analyst
Email: Jessica.Temple@snapav.com