Guide to the SNP/SMM Websocket Interface

Document Notes

This document is provided under NDA to specific SNP users and interfacing partner companies. While effort has been made to ensure that this document is correct, it could contain errors. The contents of the data objects herein and the methods described in this specification are subject to change in the future. Our intent and goal is that future changes are extensions to the object structures and methods described – an in fact our own code has dependencies on the object structures and methods described herein, so breaking changes would also break our own code.

Purpose of this interface:

The SNP/SMM Websocket interface exists to allow rapid (push) notification of clients about changes of status, configuration, and alarm values without needing to poll the SNP's REST API. This interface is used internally by the SNP's user interface, but can also be used independently of the UI, by external systems which seek to track the SNP's configuration, status, and alarm state.

Establishing the Websocket Connection and Security

URL: wss://<address_of_smm>/smm

The interface is secured using WSS (secure websocket protocol) and permissions are established using the same authentication tokens that are used on the REST API. In the HTTPS case, this bearer token (a JWT) is normally put into the header of each REST API request, however for our websocket protocol, it is simply furnished as the contents of the first message.

```
String apiBase = "https://137.237.176.76:9089/api/";
JsonObjectBuilder userAuth = Json.createObjectBuilder();
userAuth.add("username", "admin");
userAuth.add("password", "password");
HTTPResponse res = postJsonObject_rr(apiBase+"auth", userAuth.build());
String jwtAuthToken = res.getResponse();
```

This token is sent as the first message after successfully connecting on the websocket.

```
String wsAddr = "wss://137.237.176.76/smm";
WSClient webSock = null;
webSock = new WSClient(new URI(wsAddr));
webSock.configSecurityLite();
Boolean success = webSock.connectBlocking();
webSock.sendMessage(jwtAuthToken);
```

After connecting and sending the JWT, the websocket is ready for use. "fmmStatus" keep-alive messages are sent periodically by the server.

```
{"msgType":"fmmStatus","event":"keepalive","status":"success","description":"30-03-2021
00:18:43","managed_services":0,"available_services":0}
```

A "permissionsMsg" type message is sent periodically by the server to remind the client what username it is logged in as, and what permission level that user has.

```
{"msgType":"permissionsMsg","currentUser":{"id":7,"username":"jmailhot","activated":true,
"activated_at":"2019-08-30T19:58:43Z","last_login":"2021-03-
```

```
30T00:18:38Z","first_name":"John","last_name":"Mailhot","created_at":"2019-08-
30T19:58:43Z","updated_at":"2019-08-
30T19:58:43Z"},"userGroup":{"id":1,"name":"Administrator","permissions":"{\"administrator
\":1}"}}
```


The server also periodically sends “activeAlarmStatus” messages approximately every two seconds, whether or not you’ve subscribed to them. Details on this message are in a section below.

Once the authentication and bearer token are checked at the beginning, the socket remains secured. If the socket connection is lost for any reason, then when the socket is re-initiated the whole sequence starts over, and a fresh (not timed out) JWT bearer token must be supplied at the startup of the wss.

Getting Started – requesting current configuration and status

The sequence of messages below are a typical return pattern from a single-element SMM (running on the SNP). Note that similar to the SNP’s REST API, the “status” and “config” elements of these JsonObjects often are string values which contain JSON inside, escaped with \ as necessary.

For startup purposes, there is a handy message “syncMe”.

```
Sending {"msgType":"syncMe"}
```

The SMM will respond with

```
{"msgType":"elementStatus","status":"[{ … }, … ] … }
{"msgType":"alarmConfigure","alarms":[{ … }, … ] … }
{"msgType":"logsState","logs":{ … } … }
```

The “elementStatus” message’s status contains an array, with an entry for each SNP managed by that SMM, including the IP address of that SNP. This IP address will be needed later to qualify requests for information about that SNP in subsequent requests. Note if the SMM you are connecting to appears to be running on the SNP itself (is on the same IP address as the SNP for example), as this detail (smm-hosted-on-snp) is important in the syntax of some subsequent commands.


```
{"msgType":"elementStatus","status":"[{\"fme_ip_address\":\"137.237.176.76\",\"fme_type\"
:\"Selenio Network
Processor\",\"state\":\"connected\",\"cfg_cnt\":0,\"version\":\"1.7.0.78\",\"interfaces\"
:\"[]\",\"properties\":\"{\\\"network\\\":[],\\\"software\\\":{\\\"installed\\\":\\\"Aug
19 10:30:30 EDT 2017\\\",\\\"product\\\":{\\\"name\\\":\\\"Selenio Network
Processor\\\",\\\"version\\\":\\\"1.7.0.78\\\"}},\\\"capabilities\\\":3,\\\"element_type\
\\":\\\"Selenio Network
Processor\\\"}\",\"alarm_threshold\":10,\"manager_ip\":\"137.237.176.76\",\"managed_svc_c
nt\":0,\"interface_version\":\"\",\"fme_name\":\"\",\"preset_timestamp\":{\"seconds\":0,\
"nanos\":0,\"nano\":0,\"epochSecond\":0},\"discovery_timestamp\":\"2021-03-
05T05:06:28Z\",\"compatible\":\"YES\",\"compatible_msg\":\"--- Selenio Network Processor
: 1.7.0.78\"}]"}
```


Note also, specifically in the elementStatus message response, that not only is the status element a stringified JSON value, but in fact the properties element inside each of the entries in that array is a stringified object INSIDE the stringified object – note the double-escaping above.

The “alarmConfigure” message summarizes the configuration of the alarms within the SMM. Note that this can be a ***large*** array, since there is one object for each alarm type at each location within each SNP under the control of this SMM. New alarm types are added in each release.


```
{"msgType":"alarmConfigure","alarms":
[ { "fme_ip_address":"137.237.176.76",
```

```
    "identifier":"IP Receiver Video Out Of Range Packets",
    "description":"IP Receiver Video Out Of Range Packets",
    "id":72,"priority":4,"enable":true,"raiseDelay":0,"clearDelay":0,
    "count":0,"location":"[\"ProcessorC\",\"Section2\",\"Program3\"]"},
  { "fme_ip_address":"137.237.176.76",
    "identifier":"IP Receiver Video Out Of Range Packets",
    "description":"IP Receiver Video Out Of Range Packets",
    "id":72,"priority":4,"enable":true,"raiseDelay":0,"clearDelay":0,
    "count":0,"location":"[\"ProcessorC\",\"Section2\",\"Program2\"]"},

// and so on there are a very large number of alarms for each SNP, a few examples…

  { "fme_ip_address":"137.237.176.76",
    "identifier":"IP Receiver Ancillary Missing Packets",
    "description":"IP Receiver Ancillary Missing Packets",
    "id":71,"priority":4,"enable":true,"raiseDelay":0,"clearDelay":0,
    "count":0,"location":"[\"ProcessorC\",\"Section2\",\"Program4\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Audio Missing Packets",
    "description":"IP Receiver Audio Missing Packets","id":70,"priority":4,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorD\",\"Section2\",\"Program1\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Video Missing Packets",
    "description":"IP Receiver Video Missing Packets","id":69,"priority":4,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorD\",\"Section3\",\"Program1\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Ancillary Switch Error",
    "description":"IP Receiver Ancillary Switch Error","id":67,"priority":4,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorC\",\"Section1\",\"Program2\"]"},

  { "fme_ip_address":"137.237.176.76",
    "identifier":"IP Receiver Ancillary Seamless Protection Missing",
    "description":"IP Receiver Ancillary Seamless Protection Missing",
    "id":66,"priority":4,"enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorD\",\"Section1\",\"Program4\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Ancillary Input Missing",
    "description":"IP Receiver Ancillary Input    Missing","id":65,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorB\",\"Section1\",\"Program3\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"QSFP B Low Input Power Alarm",
    "description":"QSFP B Low Input Power Alarm","id":64,"priority":8,
    "enable":false,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"System\",\"QSFP\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"QSFP A Temperature Alarm",
    "description":"QSFP A Temperature Alarm","id":61,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"System\",\"Temperature\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"Power Supply Fan Failure",
    "description":"Power Supply Fan Failure","id":56,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"System\",\"Power Supply\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Audio Input Missing",
    "description":"IP Receiver Audio Input Missing","id":38,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorB\",\"Section2\",\"Program4\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"IP Receiver Video Input Missing",
    "description":"IP Receiver Video Input Missing","id":35,"priority":8,
```

```
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorD\",\"Section3\",\"Program1\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"Video Input Lost",
    "description":"Video Input Lost","id":0,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorB\",\"Section2\",\"Program3\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"Video Input Black",
    "description":"Video Input Black","id":83,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"ProcessorD\",\"Section3\",\"Program1\"]"},

  { "fme_ip_address":"137.237.176.76","identifier":"PTP Grandmaster UUID Changed",
    "description":"PTP Grandmaster UUID Changed","id":87,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"System\",\"Reference\"]"},

  { "fme_ip_address":"137.237.176.76",
    "identifier":"Quad 3G Conversion Feature Not Licensed",
    "description":"SNP-SK-QUAD-3GCONV Not Licensed","id":88,"priority":8,
    "enable":true,"raiseDelay":0,"clearDelay":0,"count":0,
    "location":"[\"System\",\"License\"]"},

  { "fme_ip_address":"137.237.176.76",
    "identifier":"IP WAN FCS Error Threshold Exceeded",
    "description":"IP WAN FCS errors have exceeded the set threshold",
    "id":96,"priority":5,"enable":true,"raiseDelay":0,"clearDelay":0,
    "count":0,"location":"[\"System\",\"IP\"]"}
]}
```


This “alarmConfigure” message is only sent in response to the “syncMe” message, or if there is a change to the alarm configuration.

Active Alarm Status Messages

Where the alarmConfigure message listed all the types of alarms and their location properties (and raise and clear thresholds), the “activeAlarmStatus” message returns an array of the actually currently asserted alarms – hopefully a much smaller array. The IDs here relate to the IDs in the alarmConfigure message. Includes timestamps of when the alarm was raised. These activeAlarmStatus messages are sent periodically without subscription.


```
{"msgType":"activeAlarmStatus","activeAlarms":[
  {"id":0,"reason":"No SDI video input detected. Check BNC signals and
connections.","location":"ProcessorB Section2
Program1","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
03T02:27:45Z","object_id":"B-UHD-2"},
  {"id":84,"reason":"Video input frozen detected.","location":"ProcessorA Section1
Program3","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:43Z","object_id":"A-HD-3"},
  {"id":53,"reason":"Front panel fan 3 failure detected.","location":"System
Fan","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:44Z","object_id":"system"},
  {"id":21,"reason":"No input detected for embedded audio group
1.","location":"ProcessorD Section2
Program4","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:44Z","object_id":"D-HD-8"},
  {"id":39,"reason":"Either primary or secondary audio seamless protection IP input is
missing on channels 1, 2.","location":"ProcessorA Section2
Program1","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:45Z","object_id":"ipAudRx"},
```

```
  {"id":66,"reason":"Either primary or secondary ancillary seamless protection IP input
is missing on stream 1.","location":"ProcessorA Section2
Program1","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:45Z","object_id":"ipAncRx"},
  {"id":84,"reason":"Video input frozen detected.","location":"ProcessorC Section2
Program2","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
30T00:18:16Z","object_id":"C-HD-6"},
  {"id":71,"reason":"Missing ancillary IP packets detected on stream
1.","location":"ProcessorC Section1
Program1","fme_ip_address":"137.237.176.76","time_stamp":"2021-03-
05T05:07:49Z","object_id":"ipAncRx"}
]}
```


The “logsState” has to do with the log consolidation process on the SMM.


```
{"msgType":"logsState","logs":{"system":{"isCollecting":false,"isAvailable":false,"collec
tedOn":"N/A","collectedOnMsec":0}}}
```


Note that the following messages:

- fmmStatus
- permissionMessage
- logState
- activeAlarmStatus

are all sent periodically without need for subscription.

Enquiries for information – getting all configuration or all status

Clients can request specific classes of information on the websocket. For example sending:

```
"{"msgType":"getAllObjects", "elementIp":"137.237.176.76"}");
```

This request (client to server) message returns a compendium of the configuration objects available from that element. As you might imagine, ***this is a large response***. Please do not request the whole structure of all objects except in rare/startup cases, as it is processor-intensive on the SNP to generate this much output volume. Note that the elementIp field can be set to the IP of the SNP, or for the smm-hosted-on-SNP case, it can (but is not required to) be set to 127.0.0.1 (the localhost IP). A typical allObjects response contains about 1440 Kbytes of characters describing 142 configuration objects, for one SNP.

The response contains an array of strings, where each string inside is a json representation of one of the configuration objects inside the SNP. These configuration strings are exactly the same configuration objects that are returned when reading the SNP REST API, please consult the SNP REST API specification for details on the format and contents of these objects.


```
{"msgType":"allObjects","elementIp":"137.237.176.76","configurations":[
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-1\", … }”,
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-3\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-4\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-5\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-6\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-7\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-8\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"A-HD-9\", … }",
"{\"type\":\"procAudio\",\"name\":\"A-procAudio\",… }”,
```

```
"{\"type\":\"procChannelUHD\",\"name\":\"A-UHD-1\", … },
"{\"type\":\"procChannelUHD\",\"name\":\"A-UHD-2\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-1\",… }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-2\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-3\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-4\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-5\", … }",
"{\"type\":\"procChannelHD\",\"name\":\"B-HD-6\", … }"
…
]}
```

A similar command "getAllStatuses" retrieves a (quite large) array of status objects. The typical response is about 580 Kbytes describing 94 objects. Again these objects are exactly as described in the REST API specification. Please DO NOT request all statuses except in rare or startup situations, as fulfilling this request is extremely resource intensive on the target device.

```
Sending {"msgType":"getAllStatuses", "elementIp":"137.237.176.76"}
Got: {"msgType":"allStatuses","elementIp":"137.237.176.76","statuses":[
"{\"name\":\"C-HD-7\",\"type\":\"procChannelHD\", … }",
"{\"name\":\"C-HD-8\",\"type\":\"procChannelHD\", … }",
"{\"name\":\"C-HD-9\",\"type\":\"procChannelHD\", … }",
"{\"name\":\"C-HD-1\",\"type\":\"procChannelHD\", … }",
…
"{\"name\":\"BNC-5\",\"type\":\"videoIn\", … }",
"{\"name\":\"A-procAudio\",\"type\":\"procAudio\", … }",
"{\"name\":\"B-procAudio\",\"type\":\"procAudio\", … }",
"{\"name\":\"SNP IP Video Rx Status\",\"type\":\"ipVidRx\", … }",
"{\"name\":\"system\",\"type\":\"system\", … }",
"{\"name\":\"SNP IP Audio Tx Status\",\"type\":\"ipAudTx\", … }"
…
]}
```

Getting specific configuration or status objects

The "**`getObjects`**" command exists for retrieving individual config objects. Please note that inside the json objects we return, we use "`object_ID`", but in the command you send, it must be camel-cased as "`objectId`" (sorry for the inconsistency). Also note that even if the smm is hosted on the SNP, in the `getObjects` command the `elementIp` must be the actual IP, it cannot be 127.0.0.1. (and note the casing of `elementIp` in this command). Despite the plural (getObjects) the command only allows you to ask for one status object, though the syntax of the returned response does permit the SNP to return multiple configuration objects. This same return message format is used when the SNP returns asynchronous status updates, which are caused by any other control path making a configuration change, including changes to IP multicast address of receivers caused by routing.

```
Sending {"msgType":"getObjects", "elementIp":"137.237.176.76",
"objectType":"procChannelHD", "objectId":"C-HD-5"}

got message: {"msgType":"objectsState","configurations":[
"{\"type\":\"procChannelHD\",\"name\":\"C-HD-5\",\"object_ID\":\"C-HD-5\",
\"active\":true,\"FrameSync\":{ … },
\"VideoTsg\":{\"Enable\":false,\"Select\":\"Horizontal Sweep Y-only\"},
\"ColorCorrector2\":{ … }, \"AudioProcessing\":{ … },
\"AudDmxVbitOvrParams\":[ … ], \"AudMuxGeneralParams\":{ … },
\"AncDataProcessing\":{ … },\"fme_ip\":\"137.237.176.76\"}"]}
```

And similarly for retrieving individual status objects on demand there is the **`getStatuses`** command, which again requests only one object, but the format of the response is used for all status objects communicated from the SNP to the client (except for the "allStatuses" response above). For the return

object, despite the singular name "`statusState`" it actually contains an array that can contain multiple objects. When requesting status objects, if the SMM is hosted on the SNP, then the `elementIp` can be either the actual IP, or 127.0.0.1 – but in the response object the "`fme_ip`" field will always have the actual IP.

```
Sending {"msgType":"getStatuses", "elementIp":"137.237.176.76",
"objectType":"procChannelHD", "objectId":"C-HD-5"}

got message {"msgType":"statusState","statuses":[
"{\"VideoTsg\":{},\"FrameSync\":{ … },\"name\":\"C-HD-5\",
\"ColorCorrector2\":{ … },\"active\":true,
\"AncDataProcessing\":{},\"type\":\"procChannelHD\",\"object_ID\":\"C-HD-5\",
\"AudioProcessing\":{ … },\"fme_ip\":\"137.237.176.76\"}"]}
```

Subscriptions for streaming status updates

Clients can also subscribe for a stream of updates of a particular object or objects. Note the very slightly different capitalization of "elementIP" in the array below, compared to the commands above. Sorry for the inconsistency. Note also that in the case of these status subscriptions, if the SMM is running on the SNP then you MUST refer to "elementIP" as 127.0.0.1 – failing to do so will give you an immediate response but will not deliver any subscribed objects. Note again the case on elementIP differs from other commands. Also note that the frequency parameter is an integer number of milliseconds but MUST be divisible by 1000. This does not mean you will get a message every so many milliseconds, rather it means that the FMM will wait at least 1000 milliseconds since the last update of that object before sending another update, marshaling any changes into the eventual update. If the status does not change, then no updates are sent.

```
Subscribing--
{"msgType":"statusListSubscribe","frequency":1000,"objectIds":[
{"elementIP":"127.0.0.1","objectType":"procChannelHD","objectId":"A-HD-5"},
{"elementIP":"127.0.0.1","objectType":"procChannelHD","objectId":"A-HD-6"}]}

Received later--
{"msgType":"statusState","statuses":[
"{\"type\":\"procChannelHD\",\"object_ID\":\"A-HD-5\", … \"fme_ip\":\"137.237.176.76\"}",
"{\"type\":\"procChannelHD\",\"object_ID\":\"A-HD-6\", … \"fme_ip\":\"137.237.176.76\"}"
]}
{"msgType":"statusState","statuses":[
"{\"type\":\"procChannelHD\",\"object_ID\":\"A-HD-5\", … \"fme_ip\":\"137.237.176.76\"}"
]}
{"msgType":"statusState","statuses":[
"{\"type\":\"procChannelHD\",\"object_ID\":\"A-HD-5\" … ,\"fme_ip\":\"137.237.176.76\"}"
]}
{"msgType":"statusState","statuses":[
"{\"type\":\"procChannelHD\",\"object_ID\":\"A-HD-6\", … \"fme_ip\":\"137.237.176.76\"}"
]}
```

In addition to the statusListSubscribe command, there is a similar statusListUnsubscribe command:

```
{"msgType":"statusListUnsubscribe","objectIds":[
{"elementIP":"127.0.0.1","objectType":"procChannelHD","objectId":"A-HD-8"}
]}
```

As the name implies, this command removes the subscription to the specific object. Many UI clients use this feature in order to only subscribe to live status updates of the specific object in the user's focus.

Subscriptions for streaming configuration updates

There are no commands for subscribing or un-subscribing to configuration updates. Instead, if the websocket is open and authenticated, then configuration updates will simply stream towards the client as they happen. If the SNP configuration is updated by any means (even a parametric tweak of a live parameter value) an updated version of the encompassing configuration object will be sent to the client. The `objectsState` message format is used, the same way as if the object had been requested.

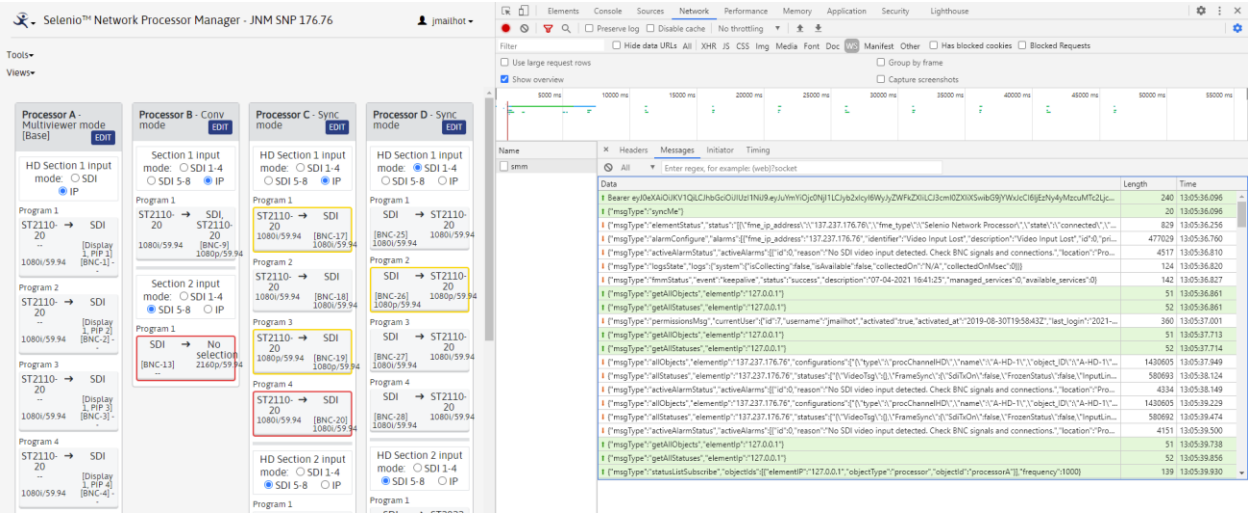
```
{"msgType":"objectsState","configurations":[
"{\"type\":\"procChannelHD\",\"object_ID\":\"C-HD-3\" … \"fme_ip\":\"137.237.176.76\"}"
]}
```

Updating Configuration Objects through this interface

The commands for updating configuration objects through the websocket are not documented in this document. Please use the REST API with its partial update facility to make any configuration changes.

Tips for debugging

Since the SNP's web UI also uses this websocket interface, its easy to see examples of the establishment and startup sequence using the google chrome developer tools. You may want to refresh the page after starting the developer tools in order to capture the websocket initiation.



If you are sending commands and its generally working, but you get no response to some specific command you are sending, check the FlexMediaManager.log for errors. This log is available wherever the SMM is running. If the SMM is running on the SNP, you can follow this log by logging into the SNP using SSH, and then

```
tail -f /mnt/logfs/FlexMediaManager.log
```

if the FMM cannot parse a command it gets over the websocket, generally the exception details will show up here.

Regarding object naming, Numbering, and Indexing

This WSS API follows the SNP REST API from an object naming, structure, and numbering perspective.