

# LivePremier™

# AWJ Protocol Programmer's Guide

For firmware version v6.1 or higher



**ANALOG WAY®**  
*Pioneer in Analog, Leader in Digital*

## Table of contents

1. Presentation .....	3
1.1. Description .....	3
1.2. Syntax .....	3
1.3. Error messages .....	4
1.4. Subscribing to machine state change notifications .....	4
2. System commands .....	6
2.1. Reading the device type .....	6
2.2. Reading the device serial number .....	6
2.3. Reading the device firmware version .....	7
2.4. Restarting the device .....	7
2.5. Shutting down the device (switch off) .....	7
3. Screen/Auxiliary Screen commands .....	8
3.1. Reading a Screen information .....	8
3.2. TAKE: Transitioning a Preview content to the Program .....	9
3.3. Recalling a Screen Preset .....	9
3.4. Recalling a Master Preset .....	10
3.5. Reading a layer information .....	10
3.6. Reading a layer status .....	11
3.7. Reading Preset information .....	12
3.8. Changing the source in a layer .....	13
3.9. Reading a background layer status .....	14
3.10. Changing screen Background source .....	15
3.11. Reading the last loaded preset .....	16
3.12. Reading the last loaded master preset .....	16
3.13. Selecting and activating the keying preset on a layer .....	17
4. Multiviewer commands .....	18
4.1. Reading the Multiviewer output information .....	18
4.2. Recalling a Multiviewer Preset .....	18
4.3. Reading the source of a Multiviewer output widget .....	19
4.4. Reading the status of a Multiviewer output widget .....	19
4.5. Changing Multiviewer widget source .....	20
4.6. Changing Multiviewer VU meter source widget .....	20
5. Audio commands .....	21

5.1.	Audio Inputs commands .....	21
5.1.1.	Reading a specific audio channel information of an input .....	21
5.1.2.	Muting a specific audio channel of an input.....	22
5.2.	Audio Output commands.....	22
5.2.1.	Reading a specific audio channel information of an output.....	22
5.2.2.	Muting a specific audio channel of an output .....	23
5.2.3.	Reading the source of an audio channel of an output.....	23
5.2.4.	Changing the source of an audio channel of an output.....	23
6.	Source commands.....	24
6.1.	Reading input information .....	24
6.2.	Reading still image information .....	26
6.3.	Reading background information .....	27
6.4.	Manual input backup switching.....	28
7.	Using thumbnails.....	31
7.1.	Introduction .....	31
7.2.	Live inputs thumbnails URL.....	31
7.3.	Still Images thumbnails URL (does not work for timers thumbnails) .....	31
7.4.	Outputs thumbnails URL.....	31
7.5.	Multiviewer thumbnails URL.....	31
7.6.	Timers thumbnails URL .....	31
8.	Waking the LivePremier™ device (over LAN) .....	32
8.1.	Description .....	32
8.2.	Wake on LAN and Magic Packet .....	32
8.3.	LivePremier™ device MAC address.....	32
8.4.	Programming example.....	33

## 1. Presentation

### 1.1. Description

The AWJ protocol for LivePremier™ is a powerful way for you to automate your interaction with the LivePremier™ seamless switchers. The AWJ protocol for LivePremier™ is based on TCP/IP communication (port 10606) and uses JSON Patch commands to interact with the device. Up to 5 concurrent TCP clients can be connected to the same device. Before connecting to this port, please check that it has not been disabled by security on the Web RCS and that a firewall is not blocking it.

A LivePremier™ device should be considered as a state machine whose values are stored and organized inside a large JSON object. Changing a value in this JSON object immediately changes the state of the machine. The current state of the machine is always available by reading the JSON object properties. It is possible to use an AWJ command to read or modify one or more properties of the device.

The objective of this document is not to describe the entire LivePremier™ device JSON object model nor to list all the possible commands allowing to read or modify the corresponding values. The objective is however to list the most frequently used commands such preset recall, transition, layer source change, etc. This document refers to LivePremier™ firmware v2.2 or higher.

### 1.2. Syntax

JavaScript Object Notation (JSON) is a common format for the exchange and storage of structured data. JSON Patch is a format for expressing a sequence of operations to apply to a target JSON document.

AWJ protocol read or write commands must be surrounded by { } and must be terminated by the ASCII 0x04 character.

The commands MUST have exactly one "op" member, whose value indicates the operation to perform. Its value MUST be one of "get" (read command) or "replace" (write command).

Additionally, the commands MUST have exactly one "path" member, whose value MUST be a string containing a JSON path value that references the location within the device LivePremier™ JSON object to perform the operation.

The AWJ write commands MUST also have exactly one "value" member, whose value corresponds to the new value to be applied to the property or object defined by the "path" member. For example:

```
{"op": "replace", "path": "/a/b/c", "value": "foo"}\0x04
```

Once an AWJ read command has been received and processed by the device, it will return a JSON string containing the value of the requested property or object. This string is surrounded by { } and is terminated by the ASCII 0x04 character as for the read or write commands.

This answer has exactly one "path" member, whose value is a string containing a JSON path value that references the location within the device JSON object for which the value was requested.

This answer has also exactly one "value" member, whose value corresponds to the value defined by the "path" member. For example:

```
{"path": "/a/b/c", "value": "foo"}\0x04
```

### 1.3. Error messages

If the AWJ command you have sent cannot be processed, the device will return a message describing the reason, for example

```
{"error":{"code":"E12","message":"Unexpected path \\\"DeviceObject/system/@props  
/div\\\""}}\0x04
```

The message contains an error code as well as message describing the error. The most common error codes are:

Code	Description
E09	Unexpected command JSON token
E10	Unexpected keywords. Keywords supported are "op", "path" and "value"
E11	Unexpected operator. Operators supported are "get" and "replace"
E12	Unexpected path
E13	Unexpected value

### 1.4. Subscribing to machine state change notifications

By default, when the current state of the machine changes, the corresponding values are not forwarded to the connected TCP clients. But it is possible to subscribe to some of the machine state change notifications to be automatically notified when the value of one or more machine properties changes (new preset label, new layer source, etc..).

The TCP client's subscription list is empty by default, meaning that this TCP connection won't receive any value/changes from the device. If the TCP client needs to receive some notifications/values from the device, the client must subscribe to the corresponding JSON path.

### Reading subscription filters

```
{"op":"get","path":"Subscriptions"}\0x04
```

The machine returns:

```
{"path":"Subscriptions","value":[]}\0x04
```

### Subscription filters modification

```
{"op":"replace","path":"Subscriptions",  
"value":["DeviceObject/$screenGroup/@items/S1/control/@props",  
"DeviceObject/$screenGroup/@items/S2",  
"DeviceObject/$screenGroup/@items/S1/status/@props/presetStatus"]}\0x04
```

The machine returns:

```
{"path": "Subscriptions", "value": [ "DeviceObject/$screenGroup/@items/S2",  
"DeviceObject/$screenGroup/@items/S1/control/@props",  
"DeviceObject/$screenGroup/@items/S1/status/@props/presetStatus"]}\0x04
```

As soon as a PATH starts with one of the subscriptions, it will be sent to the client. If the PATH does not check any subscriptions, it will be filtered.

### Example:

As soon as transition is requested for Screen 1 with the Web RCS, the "DeviceObject/\$screenGroup/@items/S1/control/@props/xTake" will be transmitted.

**Important:** A GET made directly on a property is never filtered.

With the upgrade of 4.0 version, three main changes on AWJ commands are done:

- The addition of a device selection (due to Link feature). By default, and in case of only one device, the device number of the unique device will be 1.
- Screens and Auxiliaries are now divided in two distinct lists.
- Some elements like "3.11 Reading the last loaded preset" changed location.

## 2. System commands

### 2.1. Reading the device type

#### Poll the type of the device

```
{"op":"get","path":"DeviceObject/system/$device/@items/1/@props/dev"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/system/$device/@items/1/@props/dev","value":"NLC_RS4"}\0x04
```

Possible returned values are:

**NLC\_RSALPHA** for the Aquilon RS Alpha

**NLC\_RS1** for the Aquilon RS1

**NLC\_RS2** for the Aquilon RS2

**NLC\_RS3** for the Aquilon RS3

**NLC\_RS4** for the Aquilon RS4

**NLC\_RS5** for the Aquilon RS5

**NLC\_RS6** for the Aquilon RS6

**NLC\_CMINI** for the Aquilon Cmini

**NLC\_C** for the Aquilon C

**NLC\_CPLUS** for the Aquilon C+

**NLC\_CMAX** for the Aquilon Cmax

### 2.2. Reading the device serial number

#### Poll the serial number of the device (XX9999 for ex)

```
{"op":"get","path":"DeviceObject/system/$device/@items/1/serial/@props/serialNumber"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/system/$device/@items/1/serial/@props/serialNumber",  
"value":"XX9999"}\0x04
```

### 2.3. Reading the device firmware version

Poll the firmware version of the device (2.2.80 for ex)

```
{"op":"get","path":"DeviceObject/system/$device/@items/1/version/@props/updater"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/system/$device/@items/1/version/@props/updater",  
"value":"2.2.80"}\0x04
```

### 2.4. Restarting the device

Perform a soft reboot of the unit

```
{"op":"replace","path":"DeviceObject/system/shutdown/cmd/@props/xRequest",  
"value":"REBOOT"}\0x04
```

The device will not return a string.

### 2.5. Shutting down the device (switch off)

Power the unit down (must be restarted manually)

```
{"op":"replace","path":"DeviceObject/system/shutdown/cmd/@props/xRequest",  
"value":"SHUTDOWN"}\0x04
```

The device will not return a string.

## 3. Screen/Auxiliary Screen commands

### 3.1. Reading a Screen information

#### Poll for the Screen 1 activation status

```
{"op": "get", "path": "DeviceObject/preconfig/resources/new/$screen/@items/S1/control/@props/mode"}\0x04
```

The machine returns:

```
{"path": "DeviceObject/preconfig/resources/new/$screen/@items/S1/control/@props/mode", "value": "FREESTYLE"}\0x04
```

The different values could be: DISABLED, FREESTYLE

#### Poll for the Screen 1 label, which is labeled "Sc1"

```
{"op": "get", "path": "DeviceObject/$screen/@items/S1/control/@props/label"}\0x04
```

The machine returns:

```
{"path": "DeviceObject/$screen/@items/S1/control/@props/label", "value": "Sc1"}\0x04
```

The value is empty if no specific label was registered.

#### Poll for the Auxiliary 1 label, which is labeled "Aux1"

```
{"op": "get", "path": "DeviceObject/$auxiliary/@items/A1/control/@props/label"}\0x04
```

The machine returns:

```
{"path": "DeviceObject/$auxiliary/@items/A1/control/@props/label", "value": "Aux1"}\0x04
```

The value is empty if no specific label was registered.

### 3.2. TAKE: Transitioning a Preview content to the Program

#### Take Screen 1

```
{"op":"replace","path":"DeviceObject/$screenAuxGroup/@items/S1/control/@props/xTake",  
"value":true}\0x04
```

The device will not return a string.

### 3.3. Recalling a Screen Preset

#### Recall preset 33 on the preview of screen 1

```
{"op":"replace","path":"DeviceObject/presetBank/control/load/$slot/@items/33/$screen/  
@items/S1/$preset/@items/PREVIEW/@props/xRequest","value":true}\0x04
```

The device will not return a string.

#### Recall preset 13 on the program of screen 2

```
{"op":"replace","path":"DeviceObject/presetBank/control/load/$slot/@items/13/$screen/  
@items/S2/$preset/@items/PROGRAM/@props/xRequest","value":true}\0x04
```

The device will not return a string.

#### Recall preset 8 on the program of Auxiliary screen 1

```
{"op":"replace","path":"DeviceObject/presetBank/control/load/$slot/@items/8/$auxiliary/  
@items/A1/$preset/@items/PROGRAM/@props/xRequest","value":true}\0x04
```

The device will not return a string.

### 3.4. Recalling a Master Preset

#### Recall master preset 15 to preview

```
{"op":"replace","path":"DeviceObject/masterPresetBank/control/load/$slot/@items/15/$preset/@items/PREVIEW/@props/xRequest","value":true}\0x04
```

The device will not return a string

#### Recall master preset 3 to program

```
{"op":"replace","path":"DeviceObject/ masterPresetBank/control/load/$slot/@items/3/$preset/@items/PROGRAM/@props/xRequest","value":true}\0x04
```

The device will not return a string.

### 3.5. Reading a layer information

#### Poll for the live layer 1 capacity on Screen 2

```
{"op":"get","path":"DeviceObject/preconfig/resources/new/$screen/@items/S2/$layer/@items/1/control/@props/capability"}\0x04
```

The device returns:

```
{"path":"DeviceObject/preconfig/resources/new/$screen/@items/S2/$layer/@items/1/control/@props/capability","value":"4K"}\0x04
```

The different values could be: OFF, DUAL, 4K or 5K

#### Poll for the ability of using a mask on live layer 1 on Screen 2

```
{"op":"get","path":"DeviceObject/preconfig/resources/new/$screen/@items/S2/$layer/@items/1/control/@props/canUseMask"}\0x04
```

The device returns:

```
{"path":"DeviceObject/preconfig/resources/new/$screen/@items/S2/$layer/@items/1/control/@props/canUseMask","value":false}\0x04
```

**Poll for the live layer 1 on Screen 2 frozen state on PGM and PRW (for firmware version v4.4 or higher)**

```
{ "op": "get", "path": "DeviceObject/$screen/@items/S2/$layer/@items/1/control/@props/freeze" }\0x04
```

The machine returns:

```
{ "path": " DeviceObject/$screen/@items/S2/$layer/@items/1/control/@props/freeze", "value": ["DOWN", "UP"] }\0x04
```

The different values could be: [], ["UP"], ["DOWN"] or ["DOWN", "UP"] and depend on the Transition status (3.8 Changing the source in a layer).

		Layer Status			
		[]	["UP"]	["DOWN"]	["DOWN", "UP"]
Transition status	AT DOWN	Not frozen	Frozen on PRW	Frozen on PGM	Frozen on PGM and PRW
	AT UP	Not frozen	Frozen on PGM	Frozen on PRW	Frozen on PGM and PRW

**Poll for the live layer 2 capacity on Auxiliary 1 (on output 13)**

```
{ "op": "get", "path": "DeviceObject/preconfig/resources/new/$output/@items/13/status/$layer/@items/2/@props/auxCapability" }\0x04
```

The device returns:

```
{ "path": " DeviceObject/preconfig/resources/new/$output/@items/13/status/$layer/@items/2/@props/auxCapability value": "DUAL" }\0x04
```

The different values could be: OFF, DUAL or 4K.

### 3.6. Reading a layer status

**Poll for the live layer 1 source on Screen 2 program (with TBAR at DOWN)**

```
{ "op": "get", "path": "DeviceObject/$screen/@items/S2/$preset/@items/A/$layer/@items/1/source/@props/inputNum" }\0x04
```

The device returns:

```
{"path":"DeviceObject/$screen/@items/S2/$preset/@items/A/$layer/@items/1/source/@props/inputNum","value":"LIVE_4"}\0x04
```

The different values could be: LIVE\_X, STILL\_X, SCREEN\_X, NATIVE\_X, COLOR or “NONE” if no source is set.

### Poll for the live layer 2 source on Auxiliary 1 program (with TBAR at DOWN)

```
{"op":"get","path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/$layer/@items/2/source/@props/inputNum"}\0x04
```

The device returns:

```
{"path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/$layer/@items/2/source/@props/inputNum","value":"LIVE_4"}\0x04
```

The different values could be: LIVE\_X, STILL\_X, SCREEN\_X, NATIVE\_X, COLOR or “NONE” if no source is set.

Program and preview commands are explained on section 3.8

## 3.7. Reading Preset information

### Poll for the name of Master Preset 3, which is labeled “Preset3”

```
{"op":"get","path":"DeviceObject/masterPresetBank/$bank/@items/3/control/@props/label"}\0x04
```

The device returns:

```
{"path":"DeviceObject/masterPresetBank/$bank/@items/3/control/@props/label","value":"Preset3"}\0x04
```

### Poll for the name of Screen Preset 12, which is labeled “ScreenPre12”

```
{"op":"get","path":"DeviceObject/presetBank/$bank/@items/12/control/@props/label"}\0x04
```

The device returns:

```
{"path":"DeviceObject/presetBank/$bank/@items/12/control/@props/label","value":
"ScreenPre12"}\0x04
```

### 3.8. Changing the source in a layer

The change of layer parameters is a bit more complex as the corresponding command set is indexed by preset **A/B**.

Preset A corresponds to the parameter set when the virtual TBAR is at the bottom (Down) and preset B corresponds to the parameter set when the virtual TBAR is at the top (Up).

It is therefore necessary to determine where the TBAR (Up or Down) is located before changing any layer parameter(s). For Screen 1, the following command must be sent to the device:

```
{"op":"get","path":"DeviceObject/$screenAuxGroup/@items/S1/status/@props/transition"}\0x04
```

If the device returns:

```
{"path":"DeviceObject/$screenAuxGroup/@items/S1/status/@props/transition",
"value":"AT_DOWN"}\0x04
```

This means that the TBAR of screen 1 is at the bottom. If you want to modify any layer parameters on the Program, you must therefore change DOWN parameters (or UP to change layer parameters on the Preview)  
If the device returns:

```
{"path":" DeviceObject/$screenAuxGroup/@items/S1/status/@props/transition","value":
"AT_UP"}\0x04
```

This means that the TBAR of screen 1 is at the top.

If you want to modify any layer parameters on the Program, you must therefore change UP parameters (or DOWN to change layer parameters on the Preview):

Transition status	To work on PGM	To work on PRW
AT DOWN	A	B
AT UP	B	A

**Load Live 3 source on layer 2 of the screen 1 program (with TBAR at DOWN)**

```
{"op":"replace","path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/2/source/@props/inputNum","value":"LIVE_3"}\0x04
```

The device will not return a string.

**Load Live 5 source on layer 1 of the screen 2 preview (with TBAR at DOWN)**

```
{"op":"replace","path":"DeviceObject/$screen/@items/S2/$preset/@items/B/$layer/@items/1/source/@props/inputNum","value":"LIVE_5"}\0x04
```

The device will not return a string.

**Load Live 8 source on layer 2 of the Auxiliary Screen 1 program (with TBAR at DOWN)**

```
{"op":"replace","path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/$layer/@items/2/source/@props/inputNum","value":"LIVE_8"}\0x04
```

The device will not return a string.

**Important:** A global update is required to consider all the changes on the layers, mainly on Foreground Layer

```
{"op":"replace","path":"DeviceObject/$screenAuxGroup/control/@props/xUpdate","value":true}\0x04
```

The device will not return a string.

### **3.9. Reading a background layer status**

**Poll for the background layer source on Screen 2 program (with TBAR at DOWN)**

```
{"op":"get","path":"DeviceObject/$screen/@items/S2/$preset/@items/A/$layer/@items/NATIVE/source/@props/inputNum"}\0x04
```

The device returns:

```
{"path":"DeviceObject/$screen/@items/S2/$preset/@items/A/$layer/@items/NATIVE/source/@props/inputNum","value":"NATIVE_1"}\0x04
```

The different values could be: LIVE\_X, STILL\_X, SCREEN\_X, NATIVE\_X, COLOR or “NONE” if no source is set.

### Poll for the background layer source on Auxiliary 1 program (with TBAR at DOWN)

```
{"op":"get","path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/$layer/@items/NATIVE/source/@props/inputNum"}\0x04
```

The device returns:

```
{"path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/$layer/@items/NATIVE/source/@props/inputNum","value":"NONE"}\0x04
```

The different values could be: LIVE\_X, STILL\_X, SCREEN\_X, NATIVE\_X, COLOR or “NONE” if no source is set.

## 3.10. Changing screen Background source

**Important:** Preview and Program are indexed to the TBAR, such that the current position of the TBAR will need to be known to correctly route to Preview or Program. See “Changing the source in a layer” for more details.

### Load Background Set 2 to Screen 1 program (with TBAR at DOWN)

```
{"op":"replace","path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/NATIVE/source/@props/inputNum","value":"NATIVE_2"}\0x04
```

The device will not return a string.

### Load Background Set 3 to Screen 2 preview (with TBAR at DOWN)

```
{"op":"replace","path":"DeviceObject/$screen/@items/S2/$preset/@items/B/$layer/@items/NATIVE/source/@props/inputNum","value":"NATIVE_3"}\0x04
```

The device will not return a string.

**Load Background Set 2 to Auxiliary 1 program (with TBAR at DOWN)**

```
{"op":"replace","path":"DeviceObject/$auxiliary/@items/A1/$preset/@items/A/
$layer/@items/NATIVE/source/@props/inputNum","value":"NATIVE_2"}\0x04
```

The device will not return a string.

**3.11. Reading the last loaded preset**

**Important:** Preview and Program are indexed to the TBAR, such that the current position of the TBAR will need to be known to correctly read the Preview or Program status. See “Changing the source in a layer” for more details.

**Poll for the last recalled preset to program on screen 2 (last recalled was preset 3, with TBAR at DOWN)**

```
{"op":"get","path":"DeviceObject/presetBank/status/presetId/$screen/@items/S2/$preset/@items/
A/@props/id"}\0x04
```

The device returns:

```
{"path":"DeviceObject/presetBank/status/presetId/$screen/@items/S2/$preset/@items/A/@props/id",
"value":3}\0x04
```

**Poll for the last recalled preset to preview on Aux 1 (last recalled was preset 4, with TBAR at DOWN)**

```
{"op":"get","path":"DeviceObject/presetBank/status/presetId/$auxiliary/@items/A1/$preset/@items/
A/@props/id"}\0x04
```

The device returns:

```
{"path":"DeviceObject/presetBank/status/presetId/$screen/@items/A1/$preset/@items/A/@props/id",
"value":4}\0x04
```

**3.12. Reading the last loaded master preset****Poll for the last recalled master preset to program (last recalled was master preset 3)**

```
{"op":"get","path":"DeviceObject/masterPresetBank/status/lastUsed/$presetMode/@items/
PROGRAM/@props/memoryId"}\0x04
```

The device returns:

```
{"path":"DeviceObject/masterPresetBank/status/lastUsed/$presetMode/@items/  
PROGRAM/@props/memoryId","value":3}\0x04
```

**Poll for the last recalled master preset to preview (last recalled was master preset 2)**

```
{"op":"get","path":"DeviceObject/masterPresetBank/status/lastUsed/$presetMode/@items/  
PREVIEW/@props/memoryId"}\0x04
```

The device returns:

```
{"path":"DeviceObject/masterPresetBank/status/lastUsed/$presetMode/@items/  
PREVIEW/@props/memoryId","value":2}\0x04
```

### **3.13. Selecting and activating the keying preset on a layer**

**Select the keying preset 2 on Layer 3 of the Screen 1**

```
{"op":"replace","path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/3/  
keying/@props/source","value":"SLOT_2"}\0x04
```

The device returns:

```
{"path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/3/keying/@props/  
source","value":"SLOT_2"}\0x04
```

**Activate of the keying on Layer 2 of the Screen 1**

```
{"op":"replace","path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/2/  
keying/@props/enable","value":true}\0x04
```

The device returns:

```
{"path":"DeviceObject/$screen/@items/S1/$preset/@items/A/$layer/@items/2/keying/  
@props/enable","value":true}\0x04
```

## 4. Multiviewer commands

### 4.1. Reading the Multiviewer output information

#### Poll for the Multiviewer 2 source status

```
{"op":"get","path":"DeviceObject/$monitoring/@items/2/status/@props/isEnabled"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$monitoring/@items/2/status/@props/isEnabled","value":true}\0x04
```

#### Poll for the Multiviewer 2 label, which is labeled “MVW2”

```
{"op":"get","path":"DeviceObject/$monitoring/@items/2/control/@props/label"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$monitoring/@items/2/control/@props/label","value":"MVW2"}\0x04
```

The value is empty if no specific label was registered.

### 4.2. Recalling a Multiviewer Preset

#### Recall multiviewer preset 15 on Multiviewer 1

```
{"op":"replace","path":"DeviceObject/monitoringBank/control/load/$slot/@items/15/$output/@items/1/@props/xRequest","value":true}\0x04
```

The device will not return a string.

### 4.3. Reading the source of a Multiviewer output widget

The possible sources of a widget are:

Sources	Names
No Source	NONE
Program Screen ; Preview Screen	PROGRAM_Sx ; PREVIEW_Sx
Auxiliary Screen	A_x
Live Input	IN_x
Timer	TIMER_x
Still Image	STILL_x

*The Widget value is indexed from 0 to 63.  
So, if you want to target the widget 6, you shall indicate 5 as value.*

#### Poll for the source of the widget 6 on Multiviewer 2

```
{"op": "get", "path": "DeviceObject/$monitoring/@items/2/layout/$widget/@items/5/control/@props/source"}\0x04
```

The machine returns:

```
{"path": "DeviceObject/$monitoring/@items/2/layout/$widget/@items/5/control/@props/source", "value": "IN_4"}\0x04
```

### 4.4. Reading the status of a Multiviewer output widget

#### Poll for the status of the widget 6 on Multiviewer 2

```
{"op": "get", "path": "DeviceObject/$monitoring/@items/2/layout/$widget/@items/5/status/@props/isEnabled"}\0x04
```

Same as section 4.3, please note that you need to downgrade the widget number by 1 on the command (here we want the info on widget 6, so the request is on number 5)

The machine returns:

```
{"path": "DeviceObject/$monitoring/@items/2/layout/$widget/@items/5/status/@props/isEnabled", "value": true}\0x04
```

#### 4.5. Changing Multiviewer widget source

Load live source 3 to multiviewer widget 13 on Multiviewer 1

```
{"op":"replace","path":"DeviceObject/$monitoring/@items/1/layout/$widget/@items/12/control/@props/source","value":"IN_3"}\0x04
```

The device will not return a string.

#### 4.6. Changing Multiviewer VU meter source widget

Enable the VU meter on multiviewer source widget 4 on Multiviewer 1

```
{"op":"replace","path":"DeviceObject/$monitoring/@items/1/layout/control/vuMeters/@props/widget","value":"4"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$monitoring/@items/1/layout/control/vuMeters/@props/widget",  
"value":"4"}\0x04
```

The different values could be: "None" or "a number".

## 5. Audio commands

The audio routing is done at channel level and is static. It means that it is required to configure each audio channel individually for each output or Dante Output.

### 5.1. Audio Inputs commands

The possible sources values are

Sources	Channel Names
No source	NONE
Live input channels	INPUT_xx_CHANNEL_x
Dante input channels	DANTE_xx_CHANNEL_x

*Since version 4.0, audio inputs and outputs are indexed as Physical inputs or outputs while video inputs are indexed as logical inputs. This means that the system considers all entry cards to have 8 entries even if there are only 4 inputs on the card. So, to access the first input of the second input card, it will always be input 9 without dependence on the first input card.*

#### 5.1.1. Reading a specific audio channel information of an input

##### Poll for the audio level detection of the channel 7 of input 5

```
{"op":"get","path":"DeviceObject/audio/control/$device/@items/1/$rx/@items/INPUT_5_CHANNEL_7/status/@props/isLevelDetected"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/audio/control/$device/@items/1/$rx/@items/INPUT_5_CHANNEL_7/status/@props/isLevelDetected","value":false}\0x04
```

##### Poll for the audio mute information of the channel 7 of input 5

```
{"op":"get","path":"DeviceObject/audio/control/$device/@items/1/$rx/@items/INPUT_5_CHANNEL_7/control/@props/mute"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/audio/control/$device/@items/1/$rx/@items/INPUT_5_CHANNEL_7/control/@props/mute","value":true}\0x04
```

### 5.1.2. Muting a specific audio channel of an input

#### Mute of the channel 7 of input 5

```
{ "op": "replace", "path": "DeviceObject/audio/control/$device/@items/1/$rx/@items/INPUT_5_CHANNEL_7/control/@props/mute", "value": true } \0x04
```

The device will not return a string.

## 5.2. Audio Output commands

The possible destination values are

Destinations	Channel Names
Screen/Aux Output	OUTPUT_x
Dante output group	DANTE_x (from 1 to
Multiviewer	MVW_x

*Since version 4.0, audio inputs and outputs are indexed as Physical inputs or outputs while video inputs are indexed as logical inputs. This means that the system considers all entry cards to have 8 entries even if there are only 4 inputs on the card. So, to access the first input of the second input card, it will always be input 9 without dependence on the first input card.*

### 5.2.1. Reading a specific audio channel information of an output

#### Poll for the audio level detection of the channel 3 of output 4

```
{ "op": "get", "path": "DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/status/@props/isLevelDetected" } \0x04
```

The machine returns:

```
{ "path": "DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/status/@props/isLevelDetected", "value": false } \0x04
```

#### Poll for the audio mute information of the channel 3 of output 4

```
{ "op": "get", "path": "DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/control/@props/mute" } \0x04
```

The machine returns:

```
{"path":"DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/control/@props/mute","value":true}\0x04
```

### 5.2.2. Muting a specific audio channel of an output

#### Mute of the channel 3 of output 4

```
{"op":"replace","path":"DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/control/@props/mute","value":true}\0x04
```

The device will not return a string.

### 5.2.3. Reading the source of an audio channel of an output

#### Poll for the audio source of the channel 3 of output 4

```
{"op":"get","path":"DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/control/@props/source"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_4/$channel/@items/3/control/@props/source","value":"DANTE_1_CHANNEL_8"}\0x04
```

### 5.2.4. Changing the source of an audio channel of an output

#### Route audio channel from Dante Group 4 Channel 2 to Output 8 Channel 1

```
{"op":"replace","path":"DeviceObject/audio/control/$device/@items/1/$tx/@items/OUTPUT_8/$channel/@items/1/control/@props/source","value":"DANTE_4_CHANNEL_2"}\0x04
```

The device will not return a string.

## 6. Source commands

### 6.1. Reading input information

#### Poll for the input 8 enabled state

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/status/@props/isEnabled"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/status/@props/isEnabled","value":false}\0x04
```

#### Poll for the input 8 capacity

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/status/@props/capability"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/status/@props/capability","value":"DUAL"}\0x04
```

The different values could be: OFF, DUAL, 4K or 5K.

#### Poll for the input 8 label

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/control/@props/label"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/control/@props/label","value":"IN8"}\0x04
```

The value is empty if no specific label was registered.

#### Poll for the input 8 available state

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/status/@props/isAvailable"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/status/@props/isAvailable","value":true}\0x04
```

#### Poll for the input 8 frozen state

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/control/@props/freeze"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/control/@props/freeze","value":false}\0x04
```

#### Poll for the input 8 device number (on device 1)

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/mapping/@props/device"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/mapping/@props/device","value":"1"}\0x04
```

#### Poll for the input 8 share status

```
{"op":"get","path":"DeviceObject/$input/@items/IN_8/status/@props/scope"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_8/status/@props/scope","value":"GLOBAL"}\0x04
```

The different values could be: GLOBAL, LOCAL

## 6.2. Reading still image information

### Poll for the still image 12 (of the image list) used state

```
{"op":"get","path":"DeviceObject/$still/@items/12/status/@props/isUsed"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/@items/12/status/@props/isUsed","value":true}\0x04
```

### Poll for the source image (in image library) of still image 12 (of the image list)

```
{"op":"get","path":"DeviceObject/$still/@items/12/control/@props/source"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/@items/12/control/@props/source","value":1}\0x04
```

### Poll for the still image 12 (of the image list) capacity which is image 1 in the image library

```
{"op":"get","path":"DeviceObject/$still/library/$bank/@items/1/status/@props/capability"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/library/$bank/@items/1/status/@props/capability","value":"DUAL"}\0x04
```

The different values could be: OFF, DUAL, 4K or 5K.

### Poll for the still image 12 (of the image list) label

```
{"op":"get","path":"DeviceObject/$still/@items/12/control/@props/label"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/@items/12/control/@props/label","value":"Pic12"}\0x04
```

The value is empty if no specific label was registered.

#### Poll for the still image 12 available state

```
{"op":"get","path":"DeviceObject/$still/@items/12/status/@props/isAvailable"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/@items/12/status/@props/isAvailable","value":true}\0x04
```

#### Poll for the sill image 12 device number (on device 1)

```
{"op":"get","path":"DeviceObject/$still/@items/12/mapping/@props/device"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$still/@items/12/mapping/@props/device","value":"1"}\0x04
```

### 6.3. Reading background information

#### Poll for the source of the Background 7 of the Screen 1 background set (linked to output 3)

```
{"op":"get","path":"DeviceObject/preconfig/backgrounds/$screen/@items/S1/$backgroundSet/@items/7/$output/@items/3/control/@props/content"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/preconfig/backgrounds/$screen/@items/S1/$backgroundSet/@items/7/$output/@items/3/control/@props/content","value":"STILL_12"}\0x04
```

“NONE” is returned if no background is set.

#### Poll for the validity of the Background 7 of the Screen 1 background set content (linked to output 3)

```
{"op":"get","path":"DeviceObject/preconfig/backgrounds/$screen/@items/S1/$backgroundSet/@items/7/$output/@items/3/status/@props/isContentValid"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/preconfig/backgrounds/$screen/@items/S1/$backgroundSet/@items/7/$output/@items/3/status/@props/isContentValid","value":true}\0x04
```

#### 6.4. Reading the input backup of an input backup set

Pool for the input backup 1 source of input 9

```
{"op":"get","path":"DeviceObject/$input/@items/IN_9/backup/$slot/@items/1/control/@props/source"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_9/backup/$slot/@items/1/control/@props/source",  
"value":"LIVE_7"}\0x04
```

The different values could be: NONE, LIVE\_X or STILL\_X.

#### 6.5. Triggering input backup manually

Manually trigger the backup 1 of the input 9

```
{"op":"replace","path":"DeviceObject/$input/@items/IN_9/backup/control/@props/xSelectSlot",  
"value":"1"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_9/backup/control/@props/xSelectSlot","value":"1"}\0x04
```

The different values could be: NONE, 1 or 2.

## 6.6. Changing input backup mode

### Change input 9 to automatic backup

```
{"op":"replace","path":"DeviceObject/$input/@items/IN_9/backup/control/@props/enableAutoSelect","value":true}\0x04
```

The machine returns:

```
{"path":"DeviceObject/$input/@items/IN_9/backup/control/@props/enableAutoSelect",  
"value":true}\0x04
```

## 6.7. Reading the background set backup of a background set backup set

### Pool for the background set backup 1 source of background set 3 screen 2

```
{"op":"get","path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/  
@items/3/backup/$slot/@items/1/control/@props/source"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/@items/3/  
backup/$slot/@items/1/control/@props/source","value":"NATIVE_4"}\0x04
```

The different values could be: NONE or NATIVE\_X.

## 6.8. Triggering background set backup manually

### Manually trigger the backup 2 of the background set 1 of Screen 2

```
{"op":"replace","path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/  
@items/1/backup/control/@props/xSelectSlot","value":"2"}\0x04
```

The machine returns:

```
{"path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/@items/1/  
backup/control/@props/xSelectSlot","value":"2"}\0x04
```

The different values could be: NONE, 1 or 2.

## 6.9. Changing background set backup mode

Change background set 3 of Screen 2 to automatic backup

```
{"op":"replace","path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/@items/3/backup/control/@props/enableAutoSelect","value":true}\0x04
```

The machine returns:

```
{"path":"DeviceObject/preconfig/backgrounds/$screen/@items/S2/$backgroundSet/@items/3/backup/control/@props/enableAutoSelect","value":true}\0x04
```

## 7. Using thumbnails

### 7.1. Introduction

Thumbnails of live inputs, still images, outputs and multiviewer outputs are available. These thumbnails are regularly refreshed (except still images thumbnails which are refreshed only on change). Snapshot request rate must not be more than 1 per second.

Picture size is 256 pixels (width) by up to 256 pixels (height).

Black borders are automatically added, depending on aspect ratio. Picture type is PNG.

### 7.2. Live inputs thumbnails URL

<http://<ipaddress>/api/device/snapshots/inputs/1>

up to

<http://<ipaddress>/api/device/snapshots/inputs/256>

### 7.3. Still Images thumbnails URL (does not work for timers thumbnails)

<http://<ipaddress>/api/device/snapshots/images/1>

up to

<http://<ipaddress>/api/device/snapshots/images/192>

The number is depending on machine type and configuration

### 7.4. Outputs thumbnails URL

<http://<ipaddress>/api/device/snapshots/outputs/1>

up to

<http://<ipaddress>/api/device/snapshots/outputs/96>

The number is depending on machine type and configuration

### 7.5. Multiviewer thumbnails URL

<http://<ipaddress>/api/device/snapshots/multiviewers/1>

up to

<http://<ipaddress>/api/device/snapshots/multiviewers/8>

### 7.6. Timers thumbnails URL

<http://<ipaddress>/api/device/snapshots/timers/1>

up to

<http://<ipaddress>/api/device/snapshots/timers/4>

## 8. Waking the LivePremier™ device (over LAN)

### 8.1. Description

When the device has been shut down with the Wake on LAN feature enabled ('sleep' state), the only way to wake this device is to send a broadcast message over the network ('magic packet').

### 8.2. Wake on LAN and Magic Packet

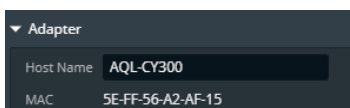
Wake on LAN (or WOL) is the ability to send a signal over a local area network (LAN) to wake up a device. When the device is powered off with the Wake-On-LAN feature enabled, no operating system is running and there is no IP address assigned. Luckily, the MAC (Media Access Control) address being hardcoded in the network adaptor remains usable to identify a device in all states.

Using this MAC address, another system on the network may send to the sleeping device a wake-up signal. The wake-up signal is a specific data frame, called 'magic packet', containing the MAC address of the remote network card. The magic packet is sent to all devices on the network (UDP broadcast) but is caught only by the device owning the matching MAC Address.

In case of linked devices, all devices (leader and follower(s)) must be woken up within the same time frame.

### 8.3. LivePremier™ device MAC address

You can get the LivePremier™ device MAC address using the Web RCS: Click the Information button located in the top right corner then select the Network option:



You can also retrieve the MAC address from the front panel menu: CONTROL -> Network.

## 8.4. Programming example

This .NET C# example sends a 'magic packet' for MAC address 00:25:90:3D:11:4A.

```
void SendWOLPacket(byte[] macAddress)
{
    // WOL 'magic' packet is sent over UDP.
    using (UdpClient client = new UdpClient())
    {
        // Send to: 255.255.255.0:9 over UDP (port number 9: Discard)
        client.Connect(IPAddress.Broadcast, 9);

        // Two parts to a 'magic' packet:
        // First is 0xFFFFFFFFFFFF,
        // Second is 16 * MACAddress.
        byte[] packet = new byte[17 * 6];

        // Set to: 0xFFFFFFFFFFFF.
        for (int i = 0; i < 6; i++)
            packet[i] = 0xFF;

        // Set to: 16 * MACAddress
        for (int i = 1; i <= 16; i++)
        {
            for (int j = 0; j < 6; j++)
                packet[i * 6 + j] = macAddress[j];
        }
        // Send WOL 'magic' packet.
        client.Send(packet, packet.Length);
    }
}

byte[] macaddress = new byte[] {0x00, 0x25, 0x90, 0x3D, 0x11, 0x4A};
WakeOnLan(macaddress);
```



ANALOG WAY®

March 2026  
Document version 6.1

