Powersoft S.p.A.
VAT no./Fiscal Code: IT 04644200489
Business Register of Firenze
Paid-up Share Capital: € 1.141.361,26

POWERSOFT.COM

VIEWS HOST
API

MARCH 2021, REV 1.13

Via Enrico Conti, 5
Scandicci (Fi) 50018 – Italy
powersoft@pec.it
+39 055 73 50 230
+39 055 73 56 235

Drawings and specifications are subject to change.

REVISION HISTORY

Revision	Date	Description	Created by	Verified by
1.11	16/03/21	First issue (rev 1.10 4221- Endpoint)	D. Quarto	
1.12	18/03/21	Aligned with ArmonìaPlus 2.1 terminology	D. Quarto	
1.13	19/03/21	Changed reading layout	D. Quarto	

1. SCOPE

This document describes the API for the Views Host server to control the Powersoft Dynamic Music Distribution system.

2. TERMS, DEFINITIONS AND ABBREVIATIONS

For the purposes of this document, the following terms and definitions apply.

Client: the software running in the equipment

Server: the Views Host software controlling all the devices of the system. It receives commands from the clients and communicate with the devices.

Devices: Powersoft Amplifiers models with DSP+D (Mezzo, Otto/Quattro/Duecanali, X, T series)

3. API

The ViewsHost server API are exposed in different port according to the device used as ViewsHost:

- Powersoft ArmoniaPlusService uses ***port 40469***. (If system is running on ArmoniaPlus)
- Powersoft ViewsHostService uses ***port 80***. (If the system is running on PC ViewsHost)
- Powersoft WM Touch uses ***port 80***. (If the system is running on WM Touch)

When the server is busy it will reply to the request with HTTP status 503 until it is available again.

The result of each API consists of a HTTP status and a JSON file contained in the body. The JSON file consists of the following fields:

- “Code”, containing an error code to be used for debug
- “Message”, containing a descriptive string for debug purposes
- “Result”, the return code related to the endpoint invoked. (This will be present only for HTTP GET that are supposed to reading infos.)

The “Code” field is used to debug the “Status” of the ViewsHost server. Values can be:

- 0 = Status “OK”
- 1 = Status “DOWN”: at least one device is currently offline or not reachable for some reason
- 2 = Status “DIFFERENT CONFIGURATION”: at least one device has been reset

3.1. AVAILABLE ENDPOINTS

3.1.1. ZONES

SOURCES OPERATIONS IN A ZONE

		NOTES
DESCRIPTION	Read the active source in a zone	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/zone/active-source/ZoneID*	
PARAMETERS	• ZoneID: is the ID number of the zone.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": *SourceID*,
	"Message": ""
	}	

		NOTES
DESCRIPTION	Set the active source of a zone	
REQUEST TYPE	HTTP PUT	
ENDPOINT	*/zone/active-source/ZoneID/SourceID*	
PARAMETERS	• ZoneID: is the ID number of the zone. • SourceID: is the ID number of the source.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "*SourceID* is set active for *ZoneID*"
	}	

		NOTES
DESCRIPTION	Disable the active source of a zone	
REQUEST TYPE	HTTP DELETE	
ENDPOINT	*/zone/active-source/ZoneID*	
PARAMETERS	• ZoneID: is the ID number of the zone.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "Is unset source for *ZoneID*"
	}	

ZONE LEVEL

		NOTES
DESCRIPTION	Read the level of a zone	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/zone/gain/ZoneID*	
PARAMETERS	• ZoneID: is the ID number of the zone.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": *ZoneLevel*,
	"Message": ""
	}	

		NOTES
DESCRIPTION	Set the level of a zone	
REQUEST TYPE	HTTP PUT	
ENDPOINT	*/zone/gain/ZoneID/ZoneLevel*	
PARAMETERS	• ZoneID: is the ID number of the zone. • ZoneLevel: is the level to set in the zone	*ZoneLevel is a linear value between 0 and 1*
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "For zone *ZoneID* is set gain to *ZoneLevel*"
	}	

ZONE MUTE

		NOTES
DESCRIPTION	Read mute status of a zone	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/zone/mute/ZoneID*	
PARAMETERS	• ZoneID: is the ID number of the zone.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": false,
	"Message": ""
	}
		false=unmuted
		true=muted

		NOTES
DESCRIPTION	Set mute status for a zone	
REQUEST TYPE	HTTP PUT	
ENDPOINT	*/zone/mute/ZoneID/MuteStatus*	
PARAMETERS	• ZoneID: is the ID number of the zone. • MuteStatus: **0** (*Unmuted*), **1** (*Muted*).	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "zone *ZoneID* is unmuted"
	}
		or “…is muted”

3.1.2. SCENE

CURRENT SCENE PARAMETERS

		NOTES
DESCRIPTION	Read current scene parameters	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/scene*	
PARAMETERS		
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": {
	"IdConfig": "....",
	"Version": 1,
	"scene": {
	"Zones": [...
	{
	...
	},
	],
	"Id": *SceneID*,
	"Version": 0,
	"Name": "Scene Name",
	"Index": 0
	}
	},
	"Message": ""
	}	

Note:

This HTTP request must be performed with the "*Accept*" header.
The accepted value is: "***application/json***" to get the json payload of the current scene.

ACTIVE SCENE

		NOTES
DESCRIPTION	Read active scene ID number	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/scene/active*	
PARAMETERS		
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": *SceneID,*
	"Message": ""
	}	

		NOTES
DESCRIPTION	Set the active scene for the system	
REQUEST TYPE	HTTP PUT	
ENDPOINT	*/scene/active/SceneID*	
PARAMETERS	- SceneID: is the ID number of the scene.	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "Current scene ID is *SceneID*"
	}	

SCENES LIST

<table>
<tr><th></th><th></th><th>NOTES</th></tr>
<tr><td>DESCRIPTION</td><td>Get a list of the scenes and their parameters</td><td></td></tr>
<tr><td>REQUEST TYPE</td><td>HTTP GET</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>ENDPOINT</td><td><i>/scene/list</i></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>PARAMETERS</td><td></td><td></td></tr>
<tr><td>RESPONSE</td><td>HTTP Status 200</td><td></td></tr>
<tr><td>RESPONSE PAYLOAD (JSON)</td><td><pre>
{
  "Code": 0,
  "Result": {
    "IdConfig": "6a0b0aa3186d49499df355b0d06dbac4",
    "Version": 1,
    "scene": {
      "Zones": [...
        {
         ...
        }
      ],
      "Id": SceneID,
      "Version": 0,
      "Name": "Scene Name",
      "Index": 1
    }
  },
  "Message": ""
}
</pre></td><td></td></tr>
</table>

3.1.3. PROJECT INFO

		NOTES
DESCRIPTION	Read project info	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/project/info*	
PARAMETERS		
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": {
	"Name": "Project Name",
	"Contacts": [
	{
	"Name": "Contact Name",
	"Email": "contact@emailaddress.com",
	"Phone": "Contact mobile phone",
	"Company": "Contact Company"
	}
	],
	"Company": "Company",
	"Location": "Company Location"
	},
	"Message": ""
	}	

3.1.4. SYSTEM ON / OFF

		NOTES
DESCRIPTION	Read the power state of the entire system	
REQUEST TYPE	HTTP GET	
ENDPOINT	*/power*	
PARAMETERS		
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Result": true,
	"Message": ""
	}
		true = system ON
		false = system OFF

		NOTES
DESCRIPTION	Set the power state for the entire system	
REQUEST TYPE	HTTP PUT	
ENDPOINT	*/power/PowerState*	
PARAMETERS	- PowerState: **0** (System is OFF), **1** (System is ON)	
RESPONSE	HTTP Status 200	

RESPONSE PAYLOAD (JSON)	{
	"Code": 0,
	"Message": "Power is set to true"
	}
		true = system ON
		false = system OFF

3.2. WEBSOCKET SERVER

The server exposes in a dedicated port a websocket, that can be used by clients for getting unsolicited notifications or for special operations.

Websocket port is a different port according to the device used as ViewsHost:

- Powersoft ArmoniaPlusService uses ***port 40470***. (If system is running on ArmoniaPlus)
- Powersoft ViewsHostService uses ***port 80***. (If the system is running on PC ViewsHost)
- Powersoft WM Touch uses ***port 80***. (If the system is running on WM Touch)

The connection must be opened by giving a unique ID to identify the client in the querystring URL used for the connection, having "***clientId***" as the name of the field.
Moreover, the communication with the websocket can be done using "***protocols***", meaning messages exchanged are strings with the following format:

<protocol_identifier>/<payload> *(i.e. PINGPONG/__ping__)*

Supported protocols are:

- Generic status update, with protocol_identifier "STATEUPDATE", in which the payload is a JSON contains a status change as described in the following paragraph (when doing a connection with this protocol each client will receive a full JSON of the current system state)

- Online/offline server check, with protocol_identifier "PINGPONG", in which the payload is sent by the client is the string "***__ping__***", and the server must reply with the string "__pong__"

- Full system status request, with protocol_identifier "STATEREQUEST"; the payload sent by the client must be the string "***request***" and the server will reply with the JSON containing the full current state (the same payload received by a client when connecting with the "STATEUPDATE" protocol)

Note:

A client, in the query string used to connect to the websocket, must specify the protocols to subscribe; if a client is not subscribed to a specific protocol it will not receive messages related to that protocol.

Example of a possible URL:

ws://<server_ip>:<websocket_port>/?clientId=<guid_client>&protocols=StateUpdate&protocols=PingPong

<u>The connection to the websocket will not be accepted if a valid “ClientID” is not specified and at least one protocol is specified.</u>

Using the “StateUpdate” protocol the client will receive a JSON object defined below (only changed status will be available; in case of “Scene” and “Power” if nothing change there will be a “null”, while for “Zones”, “Sources”, “Speakers” and “SceneKnobs” an empty array will be returned):

```
{
        “Scene”: {
                “Current”: {
                        “Id”: 1,
                        “IsModified”:  false
                },
                “Scenes”: [...
                ]
        },
        “Power”: {
                “Power”: true
        },
        “Zones”: [...

        ]
}
```