External Control Protocol (JSON) Reference Manual

020-000965-04

Phoenix

CHRISTIE®

External Control Protocol (JSON) Reference

020-000965-04

Phoenix

NOTICES

COPYRIGHT AND TRADEMARKS

GENERAL

Every effort has been made to ensure accuracy, however in some cases changes in the products or availability could occur which may not be reflected in this document. Christie reserves the right to make changes to specifications at any time without notice. Performance specifications are typical, but may vary depending on conditions beyond Christie's control such as maintenance of the product in proper working conditions. Performance specifications are based on information available at the time of printing. Christie makes no warranty of any kind with regard to this material, including, but not limited to, implied warranties of fitness for a particular purpose. Christie will not be liable for errors contained herein or for incidental or consequential damages in connection with the performance or use of this material. Canadian manufacturing facility is ISO 9001 and 14001 certified.

WARRANTY

Products are warranted under Christie's standard limited warranty, the complete details of which are available by contacting your Christie dealer or Christie. In addition to the other limitations that may be specified in Christie's standard limited warranty and, to the extent relevant or applicable to your product, the warranty does not cover:

a) Problems or damage occurring during shipment, in either direction.
b) Projector lamps (See Christie's separate lamp program policy).
c) Problems or damage caused by use of a projector lamp beyond the recommended lamp life, or use of a lamp other than a Christie lamp supplied by Christie or an authorized distributor of Christie lamps.
d) Problems or damage caused by combination of a product with non-Christie equipment, such as distribution systems, cameras, DVD players, etc., or use of a product with any non-Christie interface device.
e) Problems or damage caused by the use of any lamp, replacement part or component purchased or obtained from an unauthorized distributor of Christie lamps, replacement parts or components including, without limitation, any distributor offering Christie lamps, replacement parts or components through the internet (confirmation of authorized distributors may be obtained from Christie).
f) Problems or damage caused by misuse, improper power source, accident, fire, flood, lightning, earthquake or other natural disaster.
g) Problems or damage caused by improper installation/alignment, or by equipment modification, if by other than Christie service personnel or a Christie authorized repair service provider.
h) Problems or damage caused by use of a product on a motion platform or other movable device where such product has not been designed, modified or approved by Christie for such use.
i) Problems or damage caused by use of a projector in the presence of an oil-based fog machine or laser-based lighting that is unrelated to the projector.
j) For LCD projectors, the warranty period specified in the warranty applies only where the LCD projector is in "normal use" which means the LCD projector is not used more than 8 hours a day, 5 days a week.
k) Except where the product is designed for outdoor use, problems or damage caused by use of the product outdoors unless such product is protected from precipitation or other adverse weather or environmental conditions and the ambient temperature is within the recommended ambient temperature set forth in the specifications for such product.
l) Image retention on LCD flat panels.
m) Defects caused by normal wear and tear or otherwise due to normal aging of a product.

The warranty does not apply to any product where the serial number has been removed or obliterated. The warranty also does not apply to any product sold by a reseller to an end user outside of the country where the reseller is located unless (i) Christie has an office in the country where the end user is located or (ii) the required international warranty fee has been paid.
The warranty does not obligate Christie to provide any on site warranty service at the product site location.

PREVENTATIVE MAINTENANCE

Preventative maintenance is an important part of the continued and proper operation of your product. Please see the Maintenance section for specific maintenance items as they relate to your product. Failure to perform maintenance as required, and in accordance with the maintenance schedule specified by Christie, will void the warranty.

REGULATORY (if applicable)

The product has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the product is operated in a commercial environment. The product generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of the product in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at the user's own expense.
CAN ICES-3 (A) / NMB-3 (A)

이 기기는 업무용(A급)으로 전자파적합등록을 한 기기이오니 판매자 또는 사용자는 이점을 주의하시기 바라며, 가정 외의 지역에서 사용하는 것을 목적으로 합니다.

ENVIRONMENTAL

The product is designed and manufactured with high-quality materials and components that can be recycled and reused. This symbol means that electrical and electronic equipment, at their end-of-life, should be disposed of separately from regular waste. Please dispose of the product appropriately and according to local regulations. In the European Union, there are separate collection systems for used electrical and electronic products. Please help us to conserve the environment we live in!

Addendum

Read all instructions before using or servicing this product.

本文档的翻译副本在本文档背面的 CD 上提供。该 CD 中还可能包含其他产品文档。使用或维修本产品之前请务必阅读所有说明。

文件背面的光碟提供了本文件的翻譯副本。這張光碟可能另外包含其他產品文件。請先閱讀所有指示再使用或送修本產品。

Le CD au dos de ce document contient des traductions de celui-ci dans différentes langues. Ce CD peut également contenir de la documentation supplémentaire sur le produit. Lisez toutes les instructions avant d'utiliser ou d'entretenir ce produit.

Übersetzte Versionen dieses Dokuments werden auf der CD auf dem Vorsatzblatt dieses Dokuments bereitgestellt. Die CD kann auch zusätzliche Produktdokumentation enthalten. Bitte lesen Sie diese Anweisungen vor der Verwendung dieses Produkts oder vor der Ausführung von Wartungsarbeiten am Produkt.

Le copie tradotte di questo documento sono fornite sul CD, sul retro di questo documento. Il CD potrebbe anche contenere altra documentazione sul prodotto. Si prega di leggere tutte le istruzioni prima di utilizzare questo prodotto o sottoporlo a manutenzione.

このドキュメントの翻訳版がこのドキュメントの裏面のCDで提供されています。CDには追加の製品マニュアルも収められています。この製品を使用したり、機能させたりする前に、すべての指示をお読みください。

이 문서의 번역된 사본이 이 문서 후면의 CD에서 제공됩니다. 이 CD에는 추가 제품 설명서가 포함되어 있을 수 있습니다. 이 제품을 사용하거나 수리하기 전에 모든 지침을 확인하십시오.

Copias traduzidas deste documento são fornecida no CD contido na parte de trás deste documento. O CD pode conter documentação adicional do produto. Leia todas as instruções antes de usar ou prestar serviço com este produto.

Перевод данного документа представлен на компакт-диске на оборотной стороне документа. Компакт-диск может также содержать дополнительную документацию по продукту. Перед использованием или обслуживанием продукта ознакомьтесь со всеми инструкциями.

Las copias traducidas de este documento se proporcionan en el CD que se encuentra en la parte trasera. En el CD también puede encontrar documentación adicional del producto. Lea todas las instrucciones antes de utilizar o realizar el mantenimiento de este producto.

Перекладені екземпляри цього документа містяться на компакт-диску, який додано до цього документа. На компакт-диску може також бути додаткова документація до виробу. Перш ніж користуватися виробом або його обслуговувати, прочитайте всі інструкції.

Content

Phoenix API

This document describes the JSON interface control protocol commands for remote access to the Phoenix system. This protocol is based on the JSON-RPC 2.0 specification.

Reference and Related Documents

The following documents may be useful references:

- Phoenix System Quick Start Guide (020-101184)
- Phoenix System External Control Protocol Reference Manual (020-101425)
- Phoenix Release Notes (020-101254)

Standards and conventions

1. RFC 4627, "The application/json Media Type for JavaScript Object Notation (JSON)", http://www.ietf.org/rfc/rfc4627
2. JSON-RPC 2.0 Specification, http://www.jsonrpc.org/specification
3. M Series Serial Commands Technical Reference 020-100224-*nn*

RFC 4627 defines the underlying data encoding format, while the JSON-RPC 2.0 specification outlines the message structure, message semantics, and request, response protocol.
The Phoenix protocol supports all features of JSON-RPC 2.0 including notifications and batch messages.
This document should not be read without reference to the above to specifications. In particular, the nuances of the JSON-RPC semantics and of JSON encoding are not described in this document.

Parameters

For any message that specifies no parameters, the `params` member may either be omitted (as per JSON-RPC 2.0 spec), or transmitted as an empty array [] or object {}.

Batch messages

Batch messages are supported as described by the JSON-RPC 2.0 specification with the caveat that they will be executed in order. That is, the second method in a batch request may rely on the first. Note: there may be some commands whose response does not necessarily indicate completion.

Languages

When the language for a Phoenix controller has been set, all labels/names are localized using UTF-8 encoding.

Reply and acknowledgement

Reply and acknowledgement follow the JSON-RPC 2.0 standard.
Notifications have no `id` value and are used by the Phoenix controller for notifications.
Requests must have an `id` value, this value may be any JSON primitive value, though it should normally be a unique number within the range of a signed int32. JSON requests may be used for either a request/query or for a command, where an acknowledgement is desired.
JSON-RPC 2.0 requests are nominally asynchronous, though a client may employ synchronous communication by not sending a subsequent request until previous replies have been received.

Asynchronous notifications (Events)

The Phoenix protocol is designed to use asynchronous messaging where appropriate. Clients can register and unregister for various notification events, which are messages sent asynchronously from the server, typically upon a state change in the server.

Each namespace contains `addListener` and `removeListener` methods, and often it is possible to register for multiple (related) methods in a single call. Unless noted, the default behavior of the server is to immediately send the initial state in the form of an asynchronous notification message. This behavior can be avoided by including the parameter `sendNow: false` in the call to `addListener`.

If `addListener` has been called more times than its corresponding `removeListener` method, notifications will continue. This implies that the server will count registration instances for each client.

Notifications may be sent by the Phoenix controller individually or in batches. Note, batched notification messages shall not be mixed with other replies or notifications from other namespaces.

The Phoenix protocol provides a Notification Event to inform third-party systems of certain system actions. To receive the Notification events, subscribe to the event "notifications".

A Notification Event is sent when one of the following actions occur in the Phoenix system:

Action	Description
LossOfNode	A Phoenix Node is no longer available to the controller. The EntityID contains the IP address that was lost.
EncodingHasStopped	Encoding of a source by a Phoenix node has stopped. The EntityID identifies the source.
NotEnoughSourcesForTranscode	There are not enough resources on the Phoenix system to encode the request.

Each event will send information about what entity the action occurred on. The structure of the data for each event is:

```
{ “NotificationType” : “”,
“EntityID” : “”,
“EntityName” : “”,
“Details” : “” }
```

For the following events, the parameters are:

Event	Parameter
LossOfNode	Address, hostname, macaddress
EncodingHasStopped	SourceID, SourceName, Status
NotEnoughSourcesForTranscodes	SourceID, SourceName, empty

Examples

Most methods described in this document include examples, using the following notation to illustrate a particular client request and server response:

Request

```
{"id": 1, "jsonrpc": "2.0", "method": "abc", "params": {}}
```

Response

```
{"id": 1, "jsonrpc": "2.0", "result": 123}
```

Generic Event Message

The following is the format for a generic event message:

{"id": 1, "jsonrpc": "2.0", "method": "abc", "params": {}}

Event Types

Events are sent with the “method” property stating the type of event. The following are allowed:

- presetslistevent
- wallslistevent
- sourceslistevent
- treatmentslistevent
- layoutrecallevent
- notificationevent

Event Actions

Each event coming from the server includes the event type. The action for events differ based on the event. The action for presetslistevent, wallslistevent, sourceslistevent, and treatmentslistevent are:

Action	Description
Reset	An action that resends the entire event list.
Update	An action that is sent when an item was added or an existing item was modified.
Remove	An action that is sent when an item was deleted.

Example Event Parameters

The event type that comes in the method property has a params property that contains data relevant the corresponding event. The following are examples:

presetslistevent
"params": {
layoutid: 1,
layoutname: "layout_name",
action: reset
}
wallslistevent
"params": {
pixelspaceid: 1,
pixelspacename: "pixelspace_name",
action: update
}
sourceslistevent
"params": {
sourceid: 1,
sourcename: "source_name",
action: remove

}
treatmentslistevent
"params": {
treatmentid: 1,
treatmentname: "treatment_name",
action: update
}
notificationevent
{"id": 1, "jsonrpc": "2.0", "method": "notificationevent", "params": {
notificationtype: LossOfNode,
entityid: "Node IP Address",
entityname: "Node Hostname",
details: "Node MacAddress"
}}
notificationevent
{"id": 1, "jsonrpc": "2.0", "method": "notificationevent", "params": {
notificationtype: EncodingHasStopped,
entityid: "Source ID",
entityname: "Source Name",
details: "Source Status"
}}
notificationevent
{"id": 1, "jsonrpc": "2.0", "method": "notificationevent", "params": {
notificationtype: NotEnoughSourcesForTranscode ,
entityid: "Source ID",
entityname: "Source Name",
details: empty
}}

Examples

Most methods described in this document include examples, using the following notation:
To illustrate a particular client request and server response:

Request

```
{"id": 1, "jsonrpc": "2.0", "method": "abc", "params": {}}
```

Response

```
{"id": 1, "jsonrpc": "2.0", "result": 123}
```

Error messages

Protocol-level errors must comply with the JSON-RPC 2.0 standard error codes and formats.
Generic errors such as invalid (or missing) parameters, etc. are not explicitly specified in this document.
However, runtime errors are occasionally spelled out, where the specific message may be of importance.

code	message	meaning
-32700	Parse error	Invalid JSON was received by the server. An error occurred on the server while parsing the JSON text.
-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.

Transport and Framing

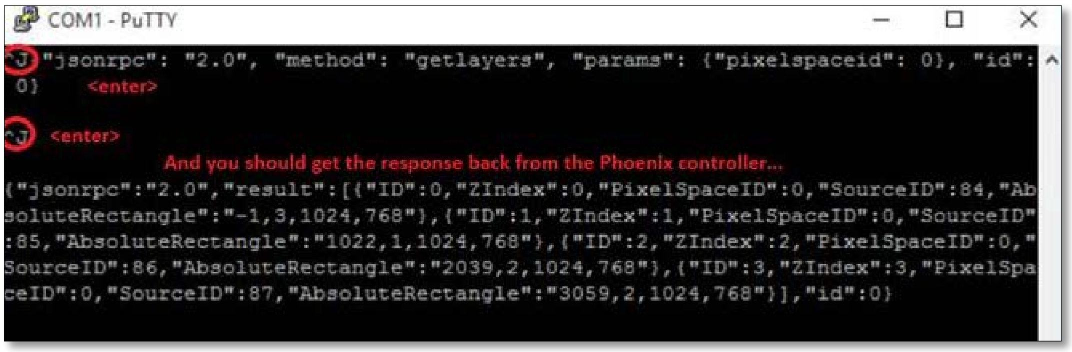
There are several transport methods allowed for this protocol. Message framing varies by transport protocol. All commands should respond within 15 seconds (round trip) unless otherwise specified.

TCP

The baseline communication method for JSON-RPC will be TCP. Each session will consist of one TCP session.
Port 23456 is the TCP/IP port used for JSON communications.
Messages must be framed using the CRLF characters. That is, messages must be separated by the {0x0D, 0x0A} characters. Any data before the two null characters that is not valid JSON is an invalid message and will generate an error messages.

When testing with PuTTY, follow every JSON command with Ctrl+J <enter>.

See the following example:


COM1 - PuTTY
<enter>
<enter>
And you should get the response back from the Phoenix controller...

```
^J "jsonrpc": "2.0", "method": "getlayers", "params": {"pixelspaceid": 0}, "id":
 0}

^J

{"jsonrpc":"2.0","result":[{"ID":0,"ZIndex":0,"PixelSpaceID":0,"SourceID":84,"Ab
soluteRectangle":"-1,3,1024,768"},{"ID":1,"ZIndex":1,"PixelSpaceID":0,"SourceID"
:85,"AbsoluteRectangle":"1022,1,1024,768"},{"ID":2,"ZIndex":2,"PixelSpaceID":0,"
SourceID":86,"AbsoluteRectangle":"2039,2,1024,768"},{"ID":3,"ZIndex":3,"PixelSpa
ceID":0,"SourceID":87,"AbsoluteRectangle":"3059,2,1024,768"}],"id":0}
```

Asynchronous notifications may be supported by the server. Therefore, if a client sends overlapping requests, the server may return the replies out-of-order. The client must use the JSON-RPC `id` value to distinguish replies. Server notifications are inserted into the connection that registered for them, therefore a client waiting for a reply may receive other messages before the appropriate reply. A well-behaved client should not open more than 2 TCP connections.
Flow control is handled by TCP. When the TCP stream is broken, any pending registrations/notifications are cleared.

Serial Port (COM1)

JSON RPC can be used over a serial port link via the COM1 port on the controller. The communication parameters must be set as follows:

Baud Rate	9600
Parity	None
StopBits	1
DataBits	8
Handshake	None

Other than the hardware link, all messaging is identical to that of TCP.
The **RS232** or **Network & RS232** option must be selected in the **API Connection** page in the Web Manager **Global Settings** page to enable this method of communication.

HTTP/HTTPS AJAX

Not supported.

WebSocket - RFC 6455

Not supported.

UDP

Not supported.

Methods

The methods are categorized by types: listener, layer, walls, layouts, and treatments.

Listener methods

The PHOENIX protocol is designed to use asynchronous messaging (where appropriate).

addlistener

Adds a listener for the type specified in the request. If **sendnow** is set to true, it sends back all the objects for the type specified.

Request parameters

listenertype	The type of item to send a notification for when an item is added, modified, or deleted. Valid types are: sources, walls, presets, treatments, and layoutrecall.
sendnow	Send out the list of items as soon as the command is issued. Valid inputs: false or true.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "sources", "sendnow": false }, "id": 1}

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "walls", "sendnow": false }, "id": 1}

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "presets", "sendnow": false }, "id": 1}

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "treatments", "sendnow": false }, "id": 1}

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "layoutrecall", "sendnow": false }, "id": 1}

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "notifications", "sendnow": false }, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

removelistener

This method unregisters from notifications for a particular code(s). This method requires a single named parameter `code`, which may be an array of multiple codes.

Parameters

listenertype	The type of item to send a notification for when an item is added, modified, or deleted. Valid types are: sources, walls, presets, treatments, and layoutrecall.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "sources" }, "id": 1}

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "walls" }, "id": 1}

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "presets" }, "id": 1}

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "treatments" }, "id": 1}

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "layoutrecall" }, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

removealllisteners

The top level `removealllisteners` method completely unregisters all notifications currently registered to the client. It is recommended that this is called just before a client exits.

Request parameters

None.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "removealllisteners", "params": {}, "id": 1}
response:
{"jsonrpc":"2.0","result":{},"id":1}

Layer methods

addlayer

Adds a new layer to a wall, specifying the associated source, size, and position for the new layer.

Request parameters

pixelspaceid	Wall ID (number) to create the new layer on.
sourceid	Source ID (number) for the new layer.
width	Width, in pixels, for the new layer.
xposition	X Position, in pixels, relative to the left edge of the wall.
yposition	Y Position, in pixels, relative to the top edge of the wall.

Response parameters

layerid	Wall ID (number) to create the new layer on.

Responses

-32000,"Too Many Layers On Output"
-32001, "Failed to Start Transcode"

Also, see *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "addlayer", "params": {"pixelspaceid": 1, "sourceid": 1, "xposition": 100, "yposition": 100, "width": 200}, "id": 1}

response:

{"jsonrpc":"2.0","result":{ layerid },"id":1}

movelayer

Moves the position of one or more layers to either an absolute pixel location on a wall, or offsets the current layer position by a horizontal and/or vertical offset.

Request parameters

horizontaloffset	Horizontal Offset in Pixels (Not used but **required** if move direction is V).
layerid	Layer ID(s) to move.

movedirection	Move Directions (h = Horizontal, v = Vertical, b = Both)
movetype	Move Type (0 = Absolute Position, 1 = Offset Position)
verticaloffset	Vertical Offset in Pixels (Only **required** if move direction is H; otherwise, it is not used.)

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "movelayer", "params": {"layerid": 1, "movetype": 0|1, "movedirection": "h"|"v"|"b", "horizontaloffset": 100, "verticaloffset": 100}, "id": 1}
response:
{"jsonrpc":"2.0","result":{},"id":1}

removelayer

Removes one or more layer(s) from one or more walls.

Request parameters

layerid	Layer ID(s) to remove.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "removelayer", "params": {"layerid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

resetaspectlayer

Resets the aspect ratio of a layer.

Parameters

layerid	Layer ID(s) to reset aspect ratio.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "resetaspectlayer", "params": {"layerid": 1}, "id": 1}
response:
{"jsonrpc":"2.0","result":{},"id":1}

getlayer

Gets information on a single layer on a wall.

Request parameters

layerid	Layer ID to get its information.
pixelspaceid	Wall ID where the layer is located.

Response parameters

absoluteleft	Absolute left, in pixels, of the layer.
absolutetop	Absolute top, in pixels, of the layer.
height	Height, in pixels, for the layer.
layerid	Layer ID to get its information.
pixelspaceid	Wall ID where the layer is located.
sourceid	Source ID (number).
width	Width, in pixels, for the layer.
zindex	z-index position.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "getlayer", "params": {"pixelspaceid": 1, "layerid": 1}, "id": 1}
response:
{"jsonrpc":"2.0","result":{"pixelspaceid": 1, "layerid": 1, "sourceid": 1, "zindex": 10, "absolutetop": 100, "absoluteleft": 100, "width": 200, "height": 400},"id":1}

getlayers

Gets a list of all layers on a wall.

Request parameters

pixelspaceid	Wall ID where the layers are located.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "getlayers", "params": {"pixelspaceid": 1}, "id": 1}
response:
{"jsonrpc":"2.0","result":{["pixelspaceid": 1, "layerid": 1, "sourceid": 1, "zindex": 10, "absolutetop": 100, "absoluteleft": 100, "width": 200, "height": 400}],"id":1}

getlayersbyrange

Gets a list of layers within the specified range.

Request parameters

endindex	The number of the last layer to retrieve.
startindex	The number of the first layer to retrieve.

Response parameters

absoluteleft	Absolute left, in pixels, of the layer.
absolutetop	Absolute top, in pixels, of the layer.
height	Height, in pixels, for the layer.
layerid	Layer ID to get its information.
pixelspaceid	Wall ID where the layer is located.
sourceid	Source ID (number).
width	Width, in pixels, for the layer.
zindex	z-index position for the layer.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "getlayersbyrange", "params": {"startindex": 0, "endindex": 100}, "id": 1}
response:
{"jsonrpc":"2.0","result":{"pixelspaceid": 1, "layerid": 1, "sourceid": 1, "zindex": 10, "absolutetop": 100, "absoluteleft": 100, "width": 200, "height": 400},"id":1}

getlayerscount

Gets the total number of layers currently on the wall.

Request parameters

None.

Response parameters

count	The total number of layers configured.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getlayerscount", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{count},"id":1}

Wall methods

getwalls

Returns a brief listing of walls defined in the system.

Request parameters

None.

Response parameters

name	Wall Name.
pixelspaceid	Wall ID.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "getwalls", "params": {}, "id": 1}
response:
{"jsonrpc":"2.0","result":[{"pixelspaceid": 1, "name": "wall1"},{"pixelspaceid": 2, "name": "wall2"}],"id":1}

getwall

Returns detailed configuration information about a specified wall ID.

Request parameters

pixelspaceid	Wall ID (retrieved using **getwalls**).

Response parameters

audiosourceid	Audio Source ID, or -1 for no audio source.
backgroundcolorblue	Background Color – Blue (0-255).
backgroundcolorgreen	Background Color – Green (0-255).
backgroundcolorred	Background Color – Red (0-255).
backgroundimagehorizontalalignment	Background Image Horizontal Alignment (L=Left, C=Center, R=Right).
backgroundimagename	Background Image Name (text).
backgroundimagestretch	Background Stretch (N=None, F=Fill, U=Uniform, UF=Uniform Fill, T=Tile).
backgroundimageverticalalignment	Background Image Vertical Alignment (T=Top, M=Middle, B=Bottom)
height	Height in pixels.
name	Wall Name (text).
pixelspaceid	Wall ID.
width	Width in pixels

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getwall", "params": {"pixelspaceid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{"pixelspaceid":0,"name":"wall1","width":1920.0,"height":1080.0,"backgroundcolorred":0,"backgroundcolorblue":159,"backgroundcolorgreen":83,"backgroundimagename":"","backgroundimagehorizontalalignment":"L","backgroundimageverticalalignment":"T","backgroundimagestretch":"F","audiosourceid":-1},"id":1}

getwallsbyrange

Gets a list of walls within the specified range.

Request parameters

endindex	The number of the last wall to retrieve.
startindex	The number of the first wall to retrieve.

Response parameters

name	Wall name.
pixelspaceid	Wall ID.

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getwallsbyrange", "params": {"startindex": 0, "endindex": 100}, "id": 1}

response:

{"jsonrpc":"2.0","result":[{"pixelspaceid": 1, "name": "wall1"},{"pixelspaceid": 2, "name": "wall2"}],"id":1}

getwallscount

Gets the total number of walls configured.

Request parameters

None.

Response parameters

count	The total number of walls configured.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getwallscount", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{count},"id":1}

turnmonitoron

Turns on the specified monitor.

Request parameters

pixelspaceid	Wall ID (retrieved using **getwalls**).

Response parameters

None.

Responses

None.

Example

request:

{"jsonrpc":"2.0","method":"turnmonitorson","params":{"pixelspaceid":0},"id":528875}

response:

{"jsonrpc":"2.0","result":[ ],"id":1}

turnmonitoroff

Turns off the specified monitor.

Request parameters

pixelspaceid	Wall ID (retrieved using **getwalls**).

Response parameters

Responses

None.

Example

request:

{"jsonrpc":"2.0","method":"turnmonitorsoff","params":{"pixelspaceid":0},"id":254254}

response:

{"jsonrpc":"2.0","result":[ ],"id":1}

Layout methods

getlayouts

Returns a brief listing of layouts (presets) defined in the system.

Request parameters

None.

Response parameters

layoutid	Layout ID.
layoutname	Layout Name.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getlayouts", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":[{"layoutid": 1, "layoutname": "layout1"},{"layoutid": 2, "layoutname": "layout2"}],"id":1}

layoutrecall

Recalls the specified layout ID on the preset's associated wall(s).

After using this command, use the **getlayers** command to get the current layers on the wall.

Response parameters

presetid	Preset ID to recall.

Response parameters

layoutid	ID of the layout that was recalled.
layoutname	Name of the layout that was recalled.
pixelspaceid	Wall ID associated with the layout.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "layoutrecall", "params": {"presetid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

layoutrecallforpixelspacecommand

Recalls the specified layout ID on the preset's specified wall(s).

Request parameters

pixelspaceid	Wall ID (retrieved using **getwalls**).
presetid	Preset ID to recall.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

```
{"jsonrpc":"2.0","method":"recalllayoutforpixelspace","params":{"presetId":0,"pixelspaceid":0},"id":473884}
```

response:

```
{"jsonrpc":"2.0","result":[],“id":1}
```

Treatment methods

gettreatments

Returns a brief listing of treatments defined in the system.

Request parameters

None.

Response parameters

treatmentid	Treatment ID.
treatmentname	Treatment Name.

Responses

See *Error messages* (page *12*).

Example

request:

```
{"jsonrpc": "2.0", "method": "gettreatments", "params": {}, "id": 1}
```

response:

```
{"jsonrpc":"2.0","result":[{"treatmentid": 1, "treatmentname": "treatment1"},{"treatmentid": 2,
"treatmentname": "treatment2"}],"id":1}
```

gettreatmentsbyrange

Returns a list of treatments within the specified range.

Request parameters

endindex	The number of the first treatment to retrieve.
startindex	The number of the last treatment to retrieve.

Response parameters

treatmentid	Treatment ID.
treatmentname	Treatment Name.

Responses

See *Error messages* (page *12*).

Example

request:
{"jsonrpc": "2.0", "method": "gettreatmentsbyrange", "params": {"startindex": 0, "endindex": 100}, "id": 1}
response:
{"jsonrpc":"2.0","result":[{"treatmentid": 1, "treatmentname": "treatment1"},{"treatmentid": 2, "treatmentname": "treatment2"}],"id":1}

gettreatmentscount

Gets the total number of treatments currently configured.

Request parameters

None.

Response parameters

count	The total number of treatments configured.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "gettreatmentscount", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{count},"id":1}

treatmentapply

Applies a treatment to a layer.

Request parameters

layoutid	Layout ID.
treatmentid	Treatment ID.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "treatmentapply", "params": {"treatmentid": 1, "layoutid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

Source methods

getsources

Returns a brief listing of sources defined in the system.

Parameters

isaudioavailable	Indicates (true or false) if the source is the audio source for the wall.
sourceid	Source ID for getting information.
sourcename	Source Name.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getsources", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":[{"sourceid": 1, "sourcename": "source1", "isaudioavailable": true },{"sourceid": 2, "sourcename": "source2", "isaudioavailable": false }],"id":1}

getsource

Returns detailed information about a source. The first part of the response is always the same but depending on the type, the remaining responses will change. See the *Response parameters* details for each source type include:

Request parameters

sourceid	Source ID to get information.
id	Type ID of the Source: 0 - RTSP Stream 1 - Still Image 2 - VNC 3 - Remote Desktop 4 - Application 5 - Phoenix DVI Input

Response parameters

RTSP Stream sources

aspectratio	Aspect ratio of the source.
audioavailable	Indicates if the source has audio associated with it.
defaulttreatmentid	ID of the default treatment for the source.
forceunicast	Indicates if only unicast transmission is enforced and multicasting is disabled.
hactive	Horizontal active pixel count.
ipaddress	IP address of the source.
keepalive	Timing for the keep alive message.
port	Port number associated with the IP address of the source.
sourceid	Source ID to get information.
sourcename	Source Name.
sourcetype	Type of source.
titletext	User-defined title of the source.
vactive	Vertical active pixel count.
videofilename	Video file name for the source.

Still Image sources

aspectratio	Aspect ratio of the source.
defaulttreatmentid	ID of the default treatment for the source.
filename	File name of the source.
hactive	Horizontal active pixel count.
sourceid	Source ID to get information.
sourcename	Source Name.
sourcetype	Type of source.
titletext	User-defined title of the source.
vactive	Vertical active pixel count.

VNC sources

aspectratio	Aspect ratio of the source.
defaulttreatmentid	ID of the default treatment for the source.
displayindex	Number related to the display attached to the VNC source.
encodebitrate	Bitrate for the source.
hactive	Horizontal active pixel count.
ipaddress	IP address of the source.
ismanned	Indicates if the source is monitored by a person.
port	Port number associated with the IP address of the source.
reachbackavailable	Indicates if a user is able to interact with the source.
sourceid	Source ID to get information.
sourcename	Source Name.
sourcetype	Type of source.
titletext	User-defined title of the source.
vactive	Vertical active pixel count.

Remote Desktop sources

aspectratio	Aspect ratio of the source.
defaulttreatmentid	ID of the default treatment for the source.
encodebitrate	Bitrate for the source.
hactive	Horizontal active pixel count.
ipaddress	IP address of the source.
ismanned	Indicates if the source is monitored by a person.
port	Port number associated with the IP address of the source.
reachbackavailable	Indicates if a user is able to interact with the source.
sourceid	Source ID to get information.
sourcename	Source Name.
sourcetype	Type of source.
titletext	User-defined title of the source.
username	User name to log into the remote desktop.
vactive	Vertical active pixel count.

Application sources

aspectratio	Aspect ratio of the source.
command	Web address for the Website or a command to run a specified application/program stored on a virtual machine.
defaulttreatmentid	ID of the default treatment for the source.
hactive	Horizontal active pixel count.
sourceid	Source ID to get information.
sourcename	Source Name.
sourcetype	Source type.
titletext	User-defined title of the source.
vactive	Vertical active pixel count.

Phoenix DVI Input

aspectratio	Aspect ratio of the source.
defaulttreatmentid	ID of the default treatment for the source.
hactive	Horizontal active pixel count.
sourceid	Source ID to get information.
sourcename	Source Name.
titletext	User-defined title of the source.
vactive	Vertical active pixel count.
sourcetype	Source type.
encodermacaddress	MAC address of the source that is encoding.
inputchannel	Input channel (A or B) for the source.
encodebitrate	Bit rate of the source.
audioavailable	Indicates if the source has audio associated with it.
reachbackavailable	Indicates if a user is able to interact with the source.
ismanned	Indicates if the source is monitored by a person.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getsource", "params": {"sourceid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

getsourcesbyrange

Returns a list of sources within the specified range.

Response parameters

endindex	The number of the first source to retrieve.
startindex	The number of the last source to retrieve.

Response parameters

sourceid	Source ID for getting information.
sourcename	Source Name.
isaudioavailable	Indicates (true or false) if the source is the audio source for the wall.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getsourcesbyrange", "params": {"startindex": 0, "endindex": 100}, "id": 1}

response:

{"jsonrpc":"2.0","result":[{"sourceid": 1, "sourcename": "source1", "isaudioavailable": true },{"sourceid": 2, "sourcename": "source2", "isaudioavailable": false }],"id":1}

getsourcescount

Gets the total number of sources currently configured.

Request parameters

None.

Response parameters

count	The total number of treatments configured.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "getsourcescount", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{count},"id":1}

applysource

Applies a source to a layer.

Response parameters

layerid	Layer ID to get its information.
sourceid	Source ID (number).

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "applysource", "params": {"layerid": 1, "sourceid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

resetsource

Resets all decoders that are currently using the source.

Request parameters

sourceid	Source ID (number) to reset.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "resetsource", "params": {"sourceid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

setaudiosource

Sets the audio source for a wall.

Request parameters

pixelspaceid	Wall ID.
sourceid	Source ID (number).

Response parameters

None.

Responses

-32002,"Source Does not support Audio"

Also, see *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "setaudiosource", "params": {"pixelspaceid": 1, "sourceid": 1}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

Controller methods

restart

Restarts all nodes that are currently connected to the controller.

Request parameters

None.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "restart", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

shutdown

Shutsdown (powers off) all nodes that are currently connected to the controller.

Request parameters

None.

Response parameters

None.

Responses

See *Error messages* (page *12*).

Example

request:

{"jsonrpc": "2.0", "method": "shutdown", "params": {}, "id": 1}

response:

{"jsonrpc":"2.0","result":{},"id":1}

Index