REST API for Crestron Virtual Control Server-Based Control System

Programming Guide

Crestron Electronics, Inc.

This document was written by the Technical Publications department at Crestron.

Contents

REST API for Crestron Virtual Control

Introduction

This document describes the REST (Representation State Transfer) interface for the Crestron Virtual Control server-based control system, called the Crestron Virtual Control REST API. REST is a set of constraints for architectures that typically communicate over HTTPS.

The Crestron Virtual Control REST API binds to the Crestron Virtual Control server as a new transport interface (much like a direct transport interface or a serial join interface). The REST API layer provides a translation from the Crestron Virtual Control server. Web server requests are stateless, and all stateful information is maintained between the Crestron Virtual Control REST API layer and programs, devices, and other control system objects.

Prerequisites

The guide is written with the following assumptions:

- The programmer has a working knowledge of API programming conventions, including how to format HTTPS requests to transfer data to and from the server.
- The programmer has a working knowledge of Crestron control system infrastructures and protocols.
- The Crestron Virtual Control server has been installed and is running on the network. For more information, refer to the Crestron Virtual Control for Red Hat® OS Installation Guide (Doc. 8912) and the Crestron Virtual Control for Red Hat® OS Deployment Guide (Doc. 8913) at www.crestron.com/manuals.

Authentication

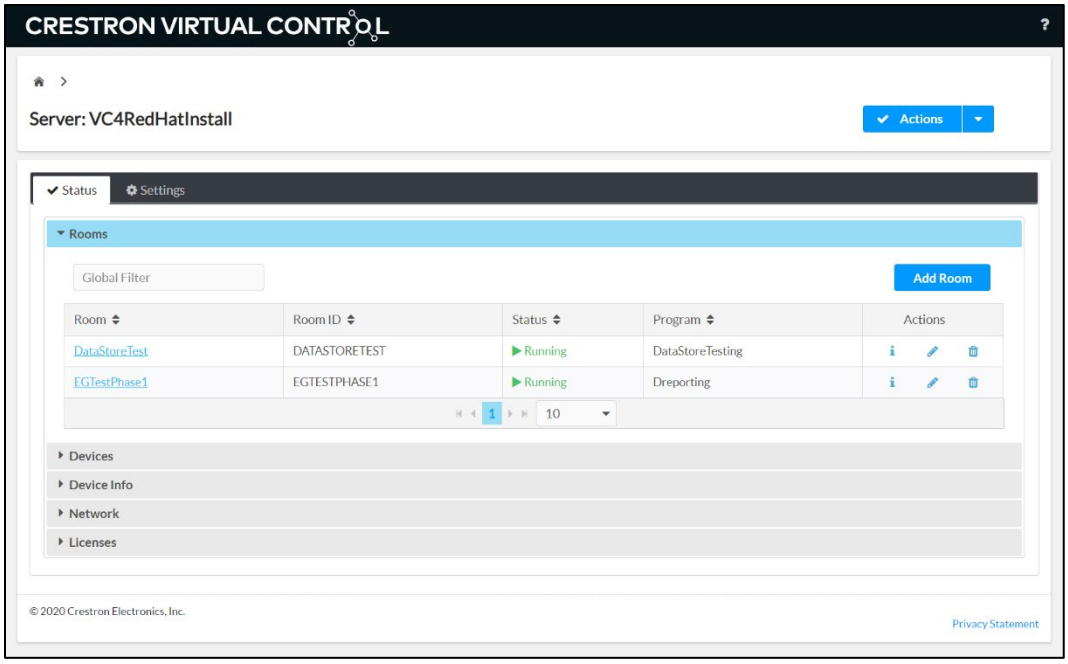
The REST API layer of the Virtual Control server may be accessed by using the "/VirtualControl/config/api" base URI. A token generated by the Crestron Virtual Control server is required for accessing the REST API layer. This token is authenticated by the Crestron Virtual Control server when an HTTPS request is sent.

To generate a token for accessing the Crestron Virtual Control REST API layer:

1. Access the Crestron Virtual Control web user interface by entering "https://[ServerURL]/VirtualControl/config/settings/" into a web browser, where [ServerURL] is the IP address or hostname of the Linux® software platform where the Crestron Virtual Control service is installed.

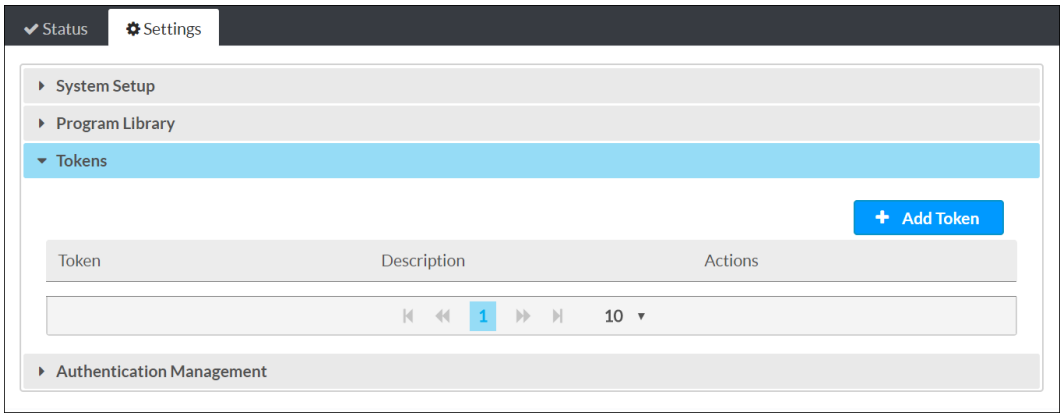
   The **Status** > **Rooms** page displays by default.

   Virtual Control Web User Interface



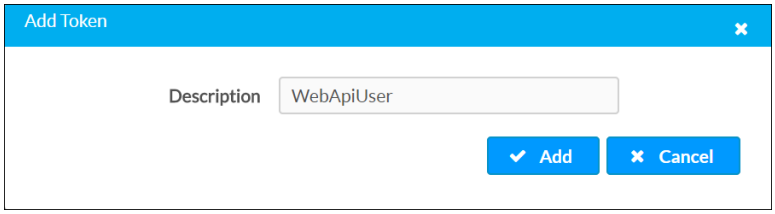
2. Navigate to **Settings** > **Tokens**.

Settings Page - Tokens



3. Click **+ Add Token**. The **Add Token** dialog window displays.

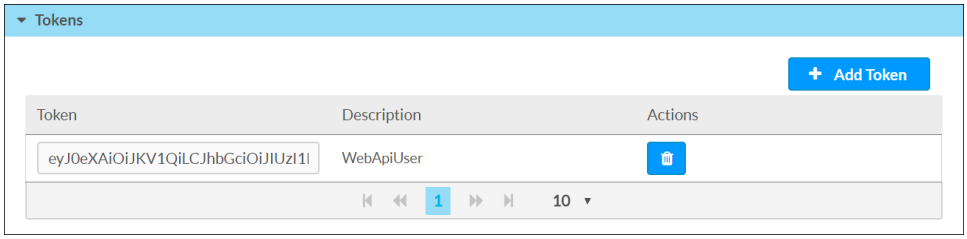
Add Token Window



4. Enter a descriptive name for the token, and then click **Add**.

   A token is generated and added to the interface.

Tokens



Once a token has been generated, the token must be appended to any HTTPS request using the "Authorization" header. If the provided token is authorized, the HTTPS request is sent to the server. If the token is not authorized, an error message is returned indicating the credentials are not valid.

Crestron Virtual Control APIs

The following sections describe the various APIs that are available within the Crestron Virtual Control REST API interface. Each section contains the following information:

- The HTTPS methods that may be issued for each API
- A list of parameters available for each HTTPS method and the parameter content type (if applicable)
- A list of responses for each HTTPS method, including the cURL base command, the request URL, and list of response codes

Getting Started

To access the Crestron Virtual Control REST API interface:

1. Obtain a valid authentication token from the Crestron Virtual Control web interface. For more information, refer to "Authentication" on page 2.
2. Open an API development environment program (such as Postman, which may be downloaded at https://www.getpostman.com/) that supports HTTPS requests.
3. Issue an HTTPS request that includes the required HTTPS method, headers, and parameters as outlined in the sections that follow. The authentication token obtained in step 1 must be appended to the request in an "Authorization" header.

   If accessing the REST API layer via cURL, the cURL command must be formatted as follows, with any parameters entered after the authorization token in the appropriate format (multipart/form-data or application/json):

   cURL Base Command

   ```
   curl -X [HTTPSMethod]
   "https://[ServerURL]/VirtualControl/config/api/[APIDirectory]"
   -H "accept: [ResponseContentType]" -H "Authorization: [Token]"
   ```

   The format of the cURL command may also be used to make web calls in other programming languages.

Authentication API

The Authentication API is used to view, create, and remove authentication groups for the Crestron Virtual Control server.

Authentication groups grant different permissions to users based on their assigned group. For example, certain Crestron Virtual Control features may appear as read-only to groups with lower permission settings.

Read All Authentication Objects

This method returns all the authentication objects created for the Crestron Virtual Control server.

NOTE: AddGroup and RemoveGroup are write-only properties and are not returned when a GET method is issued.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/Authentication

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/Authentication" -H
"accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/Authentication"
-H "accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/Authentication
```

Response Codes

Code	Description
200	Successful operation Example Value: `{ "Device": { "Authentication": { "Groups": [ { "Name": "Admins", "AccessLevel": "Administrator" } ] } } }`
404	Not found
500	Internal server error
503	Service unavailable

Read All Authentication Groups

This method returns all the authentication groups created for the Crestron Virtual Control server.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/Authentication/Groups

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/Authentication/Groups"
-H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/Authentication/
Groups" -H "accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/Authentication/Groups
```

Response Codes

Code	Description
200	Successful operation Example Value: `{ "Device": { "Authentication": { "Groups": [ { "Name": "Test", "AccessLevel": "Administrator" } ] } } }`
404	Not found
500	Internal server error
503	Service unavailable

Create Authentication Group

This method creates a new authentication group in the authentication group library.

Syntax

- **HTTPS Method:** POST
- **URI:** /VirtualControl/config/api/Authentication/Groups

Parameters

Parameter Content Type: Application/JSON

POST Authentication/Groups Parameters

Name	Description

`body`	**Required.** JSON body parameter that creates a new authentication group.
	
	Example Value:
	`{`
	`  "Device": {`
	`  "Authentication": {`
	`  "AddGroup": {`
	`  "Name": "Test",`
	`  "AccessLevel": "Administrator"`
	`  }`
	`  }`
	`  }`
	`}`

The following body parameters must be included in the POST request:

POST Authentication/Groups Body Parameters

Name	Description
`Name`	**Required.** String that sets the name of the authentication group.
`AccessLevel`	**Required.** String that sets access level for the authentication group. Valid values are listed below. • `Administrator`: Members are granted full administrative privileges. • `Operator`: Members are granted operating privileges only. • `Connect`: Members are granted read-only privileges only.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X POST
"https://[ServerURL]/VirtualControl/config/api/Authentication/Groups"
-H "accept: application/json" -H: "Authorization: [Token]" -H
"Content-Type: application/json" -d "{ \"[Parameter]\": \"[Value]\ "}"
```

Example cURL Command

```
curl -X POST
"https://123.456.789.000/VirtualControl/config/api/Authentication/Grou
ps" -H "accept: application/json" -H: "Authorization: 1234567890" -H
"Content-Type: application/json" -d "{ \"Device\": {
\"Authentication\": { \"AddGroup\": { \"Name\": \"Test\",
\"AccessLevel\": \"Administrator\" } } }}"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/Authentication/Groups
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	`{`
	`  "Actions": [`
	`  {`
	`  "Operation": "set partial",`
	`  "TargetObject": "Authentication",`
	`  "Results": [`
	`  {`
	`  "StatusInfo": SUCCESS",`
	`  "object": {`
	`  "Name": "Test",`
	`  "AccessLevel": "Administrator"`
	`  }`
	`  "StatusId": 0,`
	`  "path": "Device.Programs.Authentication"`
	`  }`
	`  ],`
	`  "Version": "2.0.1"`
	`  }`
	`  ]`
	`}`

200	Invalid input (returns `"StatusInfo": "DUPLICATE GROUP"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Remove Authentication Group

This method removes an existing authentication group from the authentication group library.

Syntax

- **HTTPS Method:** POST
- **URI:** /VirtualControl/config/api/Authentication/Groups

Parameters

Parameter Content Type: Application/JSON

POST Authentication/Groups Parameters

Name	Description

`body`	**Required.** JSON body parameter that removes an existing authentication group.
	
	Example Value:
	`{`
	`  "Device": {`
	`  "Authentication": {`
	`  "RemoveGroup": {`
	`  "Name": "Test",`
	`  "AccessLevel": "Administrator"`
	`  }`
	`  }`
	`  }`
	`}`

The following body parameters must be included in the POST request:

POST Authentication/Groups Body Parameters

Name	Description
`Name`	**Required.** String that sets the name of the authentication group to remove.
`AccessLevel`	**Required.** String that sets access level for the authentication group that is being removed. Valid values are listed below. • `Administrator`: Members are granted full administrative privileges. • `Operator`: Members are granted operating privileges only. • `Connect`: Members are granted read-only privileges only.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X POST
"https://[ServerURL]/VirtualControl/config/api/Authentication/Groups"
-H "accept: application/json" -H: "Authorization: [Token]" -H
"Content-Type: application/json" -d "{ \"[Parameter]\": \"[Value]\ "}"
```

Example cURL Command

```
curl -X POST
"https://123.456.789.000/VirtualControl/config/api/Authentication/Grou
ps" -H "accept: application/json" -H: "Authorization: 1234567890" -H
"Content-Type: application/json" -d "{ \"Device\": {
\"Authentication\": { \"RemoveGroup\": { \"Name\": \"Test\",
\"AccessLevel\": \"Administrator\" } } }}"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/Authentication/Groups
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	`{`
	`  "Actions": [`
	`  {`
	`  "Operation": "set partial",`
	`  "TargetObject": "Authentication",`
	`  "Results": [`
	`  {`
	`  "StatusInfo": Deleted",`
	`  "object": "NULL"`
	`  "StatusId": 0,`
	`  "path": "Device.Programs.Authentication"`
	`  }`
	`  ],`
	`  "Version": "2.0.1"`
	`  }`
	`  ]`
	`}`

200	Invalid input (returns `"StatusInfo": "INVALID GROUP"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

ProgramLibrary API

The ProgramLibrary API is used to add, view, modify, and delete programs in the Crestron Virtual Control server's program library.

Add Program

This method adds a new program to the program library.

Syntax

- **HTTPS Method:** POST
- **URI:** /VirtualControl/config/api/ProgramLibrary

Parameters

Parameter Content Type: Multipart/Form-Data

POST ProgramLibrary Parameters

Name	Description
`FriendlyName`	**Required.** String (form data) that sets the name of the program. Length must be between 1 and 64 characters.
`Notes`	Optional. String (form data) that sets notes for the program. Maximum length is 255 characters.
`Tags`	Optional. String (form data) that sets tags for the program. Maximum length is 255 characters.
`AppFile`	**Required.** The program application file (form data). May be uploaded alone or with other files. Valid file extensions: .cpz, .zip.
`MobilityFile`	Optional. A mobile project file for the program (form data). May be uploaded alone or with other files. Valid file extension: .zip.
`WebxPanelFile`	Optional. A web XPanel file for the program (form data). May be uploaded alone or with other files. Valid file extension: .zip.
`ProjectFile`	Optional. A touch screen project file for the program (form data). May be uploaded alone or with other files. Valid file extension: .vtz.
`CwsFile`	Optional. A program CWS configuration file (form data). May be uploaded alone or with other files. Valid file extensions: .zip, .tar, .tgz.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X POST
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary"
-H "accept: application/json" -H "Authorization: [Token]" -H "
Content-Type: multipart/form-data" -F "[Parameter]=[Value]"
```

Example cURL Command

```
curl -X POST
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary"
-H "accept: application/json" -H "Authorization: 1234567890" -H
"Content-Type: multipart/form-data" -F "Friendly Name=Program01" -F
"AppFile=prog01.zip "}
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Actions": [
	{
	"Operation": "set partial",
	"TargetObject": "ProgramLibrary",
	"Results": [
	{
	"StatusInfo": "SUCCESS",
	"object": {
	"ProgramId: "1",
	"FriendlyName": "Prog001",
	"Notes": "Test Program",
	"Tags": "",
	"AppFile": "SIMPSharpProgram1.cpz",
	"MobilityFile": "Mobility.zip",
	"WebxPanelFile": "WebXPanel.zip",
	"ProjectFile": "TE_DIN.vtz",
	"CwsFile": "EXProgram.zip",
	"AppFileTS": "2018-05-03 02:50:36:305441",
	"MobilityFileTS":
	"2018-05-03 02:50:36:305442",
	"WebxPanelFileTS":
	"2018-05-03 02:50:36:305443",
	"ProjectFileTS":
	"2018-05-03 02:50:36:305444",
	"CwsFileTS":"2018-05-03 02:50:36:305445"
	},
	"StatusId": 0,
	"Path":
	"Devices.Programs.ProgramInstanceLibrary"
	}
	],
	"Version": "2.0.1"
	}
	]
	}

200	Invalid input (returns `"StatusInfo": "DUPLICATE ID"`)
400	Bad request error

Code	Description
404	Not found
500	Internal server error
503	Service unavailable

Read All Programs

This method returns all programs in the program library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/ProgramLibrary

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary" -H
"accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary"
-H "accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Device": {
	  "Programs": {
	  "ProgramLibrary": {
	  "1": {
	  "ProgramId: "1",
	  "FriendlyName": "Prog001",
	  "Notes": "Test Program",
	  "Tags": "",
	  "AppFile": "SIMPSharpProgram1.cpz",
	  "MobilityFile": " Mobility.zip ",
	  "WebxPanelFile": "WebXPanel.zip",
	  "ProjectFile": "TE_DIN.vtz",
	  "CwsFile": "EXProgram.zip",
	  "AppFileTS": "2018-05-03 02:50:36:305441",
	  "MobilityFileTS":
	"2018-05-03 02:50:36:305442",
	  "WebxPanelFileTS":
	"2018-05-03 02:50:36:305443",
	  "ProjectFileTS":
	"2018-05-03 02:50:36:305444",
	  "CwsFileTS":"2018-05-03 02:50:36:305445"
	  }
	  }
	  }
	  }
	}

404	Not found
500	Internal server error
503	Service unavailable

Modify Program

This method modifies an existing program in the program library using `ProgramId`.

Syntax

- **HTTPS Method:** PUT
- **URI:** /VirtualControl/config/api/ProgramLibrary

Parameters

Parameter Content Type: Multipart/Form-Data

PUT ProgramLibrary Parameters

Name	Description
`ProgramId`	**Required.** String (path) that sets the ID of the program that will be modified. Length must be between 1 and 32 characters.
`FriendlyName`	Optional. String (form data) that sets the name of the program. Length must be between 1 and 64 characters.
`Notes`	Optional. String (form data) that sets notes for the program. Maximum length is 255 characters.
`AppFile`	Optional. The program application file (form data). May be uploaded alone or with other files. Valid file extensions are .cpz and .zip.
`MobilityFile`	Optional. A mobile project file for the program (form data). May be uploaded alone or with other files. Valid file extension is .zip.
`WebxPanelFile`	Optional. A web XPanel file for the program (form data). May be uploaded alone or with other files. Valid file extension is .zip.
`ProjectFile`	Optional. A touch screen project file for the program (form data). May be uploaded alone or with other files. Valid file extension is .vtz.
`CwsFile`	Optional. A CWS configuration file for the program (form data). May be uploaded alone or with other files. Valid file extensions are .zip, .tar, and .tgz.
`StartNow`	Optional. String (form data) used to start the program when the `AppFile` program file is uploaded. The only valid value is `true`.
`StartLater`	This parameter is not currently being used.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X PUT
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary"
-H "accept: application/json" -H "Authorization: [Token]"
-H "Content-Type: multipart/form-data" -d {"[Parameter]":"[Value]"}
```

Example cURL Command

```
curl -X PUT
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary"
-H "accept: application/json" -H "Authorization: 1234567890" -H
"Content-Type: multipart/form-data" -d
{"ProgramId":"ProgramId1","FriendlyName":"Program011",
"Notes":"For Testing"}
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Actions": [
	{
	"Operation": "set partial",
	"TargetObject": "ProgramLibrary",
	"Results": [
	{
	"StatusInfo": "SUCCESS",
	"object": {
	"ProgramId: "1",
	"FriendlyName": "Prog001",
	"Notes": "Test Program",
	"Tags": "",
	"AppFile": "SIMPSharpProgram1.cpz",
	"MobilityFile": "Mobility.zip",
	"WebxPanelFile": "WebXPanel.zip",
	"ProjectFile": "TE_DIN.vtz",
	"CwsFile": "EXProgram.zip",
	"AppFileTS": "2018-05-03 02:50:36:305441",
	"MobilityFileTS":
	"2018-05-03 02:50:36:305442",
	"WebxPanelFileTS":
	"2018-05-03 02:50:36:305443",
	"ProjectFileTS":
	"2018-05-03 02:50:36:305444",
	"CwsFileTS":"2018-05-03 02:50:36:305445"
	},
	"StatusId": 0,
	"Path": "Devices.Programs.ProgramLibrary"
	}
	],
	"Version": "2.0.1"
	}
	]
	}

200	Invalid input (returns `"StatusInfo": "INVALID ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Read Program

This method returns information about a specific program in the program library using `ProgramId`.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/ProgramLibrary/{ProgramId}

Parameters

GET ProgramLibrary/{ProgramId} Parameters

Name	Description
`ProgramId`	**Required.** String (path) that sets the ID of the program to return.

Responses

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]" -H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary/
ProgramId1" -H "accept: application/json" -H "Authorization:
1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	<pre>{
	  "Device": {
	  "Programs": {
	  "ProgramLibrary": {
	  "1": {
	  "ProgramId: "1",
	  "FriendlyName": "Prog001",
	  "Notes": "Test Program",
	  "Tags": "",
	  "AppFile": "SIMPSharpProgram1.cpz",
	  "MobilityFile": " Mobility.zip ",
	  "WebxPanelFile": "WebXPanel.zip",
	  "ProjectFile": "TE_DIN.vtz",
	  "CwsFile": "EXProgram.zip",
	  "AppFileTS": "2018-05-03 02:50:36:305441",
	  "MobilityFileTS":
	"2018-05-03 02:50:36:305442",
	  "WebxPanelFileTS":
	"2018-05-03 02:50:36:305443",
	  "ProjectFileTS":
	"2018-05-03 02:50:36:305444",
	  "CwsFileTS":"2018-05-03 02:50:36:305445"
	  }
	  }
	  }
	  }
	}</pre>

200	Invalid input (returns `"StatusInfo": "INVALID ID"`)
404	Not found
500	Internal server error
503	Service unavailable

Delete Program

This method deletes a program from the program library using `ProgramId`.

Syntax

- **HTTPS Method:** DELETE
- **URI:** /VirtualControl/config/api/ProgramLibrary/{ProgramId}

Parameters

DELETE ProgramLibrary/{ProgramId} Parameters

Name	Description
ProgramId	**Required.** String (path) that sets the ID of the program to delete.

Responses

cURL Base Command

```
curl -X DELETE
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]" -H "accept: application/json" -H "Authorization: [Token]"
```

cURL Base Command

```
curl -X DELETE
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary/
ProgramId1" -H "accept: application/json" -H "Authorization:
1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Actions": [
	{
	"Operation": "set partial",
	"TargetObject": "ProgramLibrary",
	"Results": [
	{
	"StatusInfo": Deleted",
	"object": "NULL"
	"StatusId": 0,
	"path": "Device.Programs.ProgramLibrary"
	}
	],
	"Version": "2.0.1"
	}
	]
	}

200	Invalid input (returns "StatusInfo": "INVALID ID")
400	Bad request error

Code	Description
404	Not found
500	Internal server error
503	Service unavailable

NOTE: When a program instance (room) is created, it will move from the initializing to running state in sequence. If a DELETE method is issued for a program running in the room during this time, an UNHANDLED ERROR is returned. Waiting a few seconds between the POST and DELETE requests avoids this error.

Delete Program File

This method deletes a file from a specific program in the program library without deleting the entire program.

Syntax

- **HTTPS Method:** DELETE
- **URI:** /VirtualControl/config/api/ProgramLibrary/{ProgramId}/{FileType}

Parameters

DELETE ProgramLibrary/{ProgramId}/{FileType} Parameters

Name	Description
`ProgramId`	**Required.** String (path) that sets the ID of the program that contains the file that will be deleted.
`FileType`	**Required.** String (path) that sets the type of the program file to be deleted (MobilityFile, WebxPanelFile, ProjectFile, CwsFile).

Responses

cURL Base Command

```
curl -X DELETE
"https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]/[FileType]" -H "accept: application/json" -H
"Authorization: [Token]"
```

Example cURL Command

```
curl -X DELETE
"https://123.456.789.000/VirtualControl/config/api/ProgramLibrary/
ProgramId1/ProjectFile" -H "accept: application/json" "Authorization:
1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramLibrary/
[ProgramId]/[FileType]
```

Response Codes

Code	Description
200	Successful operation Example Value: `{ "Actions": [ { "Operation": "set partial", "TargetObject": "ProgramLibrary", "Results": [ { "StatusInfo": Deleted", "object": "NULL" "StatusId": 0, "path": "Device.Programs.ProgramLibrary" } ], "Version": "2.0.1" } ] }`
200	Invalid input (returns `"StatusInfo": "FILE NOT FOUND ERROR"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

ProgramInstance API

The ProgramInstance API is used to add, view, modify, and delete program instances in the Crestron Virtual Control server's program instance library.

Program instances allow the same program to be run in multiple rooms across the Crestron Virtual Control server, but each instance may be customized to fit the specific properties of the room (such as location and time zone).

NOTE: The API name for this functionality is "ProgramInstance," while the web user interface name is "Room." The ProgramInstance API corresponds with the **Status** > **Rooms** configuration settings in the user interface.

Add ProgramInstance

This method adds a new program instance into the program instance library.

Syntax

- **HTTPS Method:** POST
- **URI:** /VirtualControl/config/api/ProgramInstance

Parameters

Parameter Content Type: Multipart/Form-Data

POST ProgramInstance Parameters

Name	Description
`Name`	**Required.** String (form data) that sets the name of the program instance. Maximum length is 255 characters.
`ProgramInstanceId`	**Required.** String (form data) that sets the program instance ID, which provides a path for the program instance. Length must be between 1 and 32 characters.
`ProgramLibraryId`	**Required.** String (form data) that sets the program library ID for the program instance. Length must be between 1 and 32 characters.
`Notes`	Optional. String (form data) that sets notes for the program instance. Maximum length is 255 characters.
`Level`	Optional. String (form data) that sets the access level of the program instance. Maximum length is 255 characters.
`Location`	Optional. String (form data) that sets the location where the program instance will be run. Maximum length is 255 characters.
`TimeZone`	Optional. String (form data) that sets the time zone where the program instance will be run. Maximum length is 255 characters.
`Latitude`	Optional. String (form data) that sets the latitude where the program instance will be run. Maximum length is 255 characters.
`Longitude`	Optional. String (form data) that sets the longitude where the program instance will be run. Maximum length is 255 characters.

Name	Description
AddressSets Location	Optional. Boolean (form data) that determines whether an address is used to set the location properties automatically.
UserFile	Optional. A custom user file (form data) for the program instance. Files may be uploaded individually or zipped.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X POST
"https://[ServerURL]/VirtualControl/config/api/ProgramInstance" -H
"accept: application/json" -H "Authorization: [Token]" -H
"Content-Type: multipart/form-data" -F "[Parameter]=[Value]"
```

Example cURL Command

```
curl -X POST
"https://123.456.789.000/VirtualControl/config/api/ProgramInstance" -H
"accept: application/json" -H "Authorization: 1234567890" -H
"Content-Type: multipart/form-data" -F "Name=ProgInstance1" -F
"ProgramInstanceId=PI1"-F "ProgramLibraryId=Program01"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramInstance
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	<pre>{
	  "Actions": [
	    {
	      "Operation": "set partial",
	      "TargetObject": "ProgramLibrary",
	      "Results": [
	        {
	          "StatusInfo": "SUCCESS",
	          "object": {
	            "id": 2,
	            "Notes": "",
	            "Level": "",
	            "ProgramInstanceId" = "22",
	            "Configuration Link: "",
	            "ProgramLibraryId": "2",
	            "Latitude": "41.0",
	            "Status": "Running",
	            "WorkingDirectory: "User",
	            "XpanelUrl": "",
	            "Longitude": "-73.93333",
	            "Name": "Room01",
	            "AddressSetsLocation": true,
	            "Time Zone": "America/New_York",
	            "Location": "NJC"
	          },
	          "StatusId": 0,
	          "Path":
	"Devices.Programs.ProgramInstanceLibrary"
	        }
	      ],
	      "Version": "2.0.1"
	    }
	  ]
	}</pre>

200	Invalid input (returns `"StatusInfo": "DUPLICATE ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Read All ProgramInstances

This method returns all program instances in the program instance library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/ProgramInstance

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/ProgramInstance" -H
"accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/ProgramInstance" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramInstance
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Device": {
	  "Programs": {
	  "ProgramInstance": {
	  "ProgramInstanceId1": {
	  "id": 2,
	  "Notes": "",
	  "Level": "",
	  "Configuration Link: "",
	  "ProgramLibraryId": "33",
	  "Latitude": "41.0",
	  "Status": "Running",
	  "XpanelUrl": "",
	  "WorkingDirectory: "User",
	  "Longitude": "-73.93333",
	  "ProgramInstanceId": "21",
	  "Name": "progInstance RoomName2",
	  "AddressSetsLocation": false,
	  "Time Zone": "America/New_York",
	  "Location": "NJC"
	  },
	  "ProgramInstanceId2": {
	  "id": 1,
	  "Notes": "",
	  "Level": "",
	  "Configuration Link: "",
	  "ProgramLibraryId": "34",
	  "Latitude": "41.0",
	  "Status": "Running",
	  "XpanelUrl": "",
	  "WorkingDirectory: "User",
	  "Longitude": "-33.93333",
	  "ProgramInstanceId": "20",
	  "Name": "progInstance RoomName1",
	  "AddressSetsLocation": true,
	  "Time Zone": "America/New_York",
	  "Location": "NJC",
	  },
	  }
	  }
	  }
	}

404	Not found
500	Internal server error
503	Service unavailable

Modify ProgramInstance

This method modifies an existing program instance in the program instance library using `ProgramInstanceId`.

Syntax

- **HTTPS Method:** PUT
- **URI:** /VirtualControl/config/api/ProgramInstance

Parameters

Parameter Content Type: Multipart/Form-Data

PUT ProgramInstance Parameters

Name	Description
`ProgramInstanceId`	**Required.** String (path) that sets the ID of the program instance that will be modified.
`Name`	Optional. String (form data) that modifies the name of the program instance. Maximum length is 255 characters.
`Notes`	Optional. String (form data) that modifies notes for the program instance. Maximum length is 255 characters.
`Level`	Optional. String (form data) that modifies the access level of the program instance. Maximum length is 255 characters.
`Location`	Optional. String (form data) that sets the location where the program instance will be run. Maximum length is 255 characters.
`TimeZone`	Optional. String (form data) that sets the time zone where the program instance will be run. Maximum length is 255 characters.
`Latitude`	Optional. String (form data) that sets the latitude where the program instance will be run. Maximum length is 255 characters.
`Longitude`	Optional. String (form data) that sets the longitude where the program instance will be run. Maximum length is 255 characters.
`Start`	Optional. String (form data) used to start the specific program. The only valid value is `true`.
`Stop`	Optional. String (form data) used to stop the specific program. The only valid value is `true`.
`AddressSets Location`	Optional. Boolean property (form data) that determines whether an address is used to set the location properties automatically.
`UserFile`	Optional. A custom user file (form data) for the program instance. Files may be uploaded individually or zipped.

NOTE: `Start` and `Stop` are mutually exclusive parameters.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X PUT
"https://[ServerURL]/VirtualControl/config/api/ProgramInstance"
-H "accept: application/json" -H "Authorization: [Token]" -H
"Content-Type: multipart/form-data" -d {"[Parameter]": "[Value]"}
```

Example cURL Command

```
curl -X PUT
"https://123.456.789.000/VirtualControl/config/api/ProgramInstance" -H
"accept: application/json" -H "Authorization: 1234567890" -H
"Content-Type: multipart/form-data" -d
{"ProgramInstanceId":"21","Name":"ProgInstance1","Start":"true"}
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramInstance
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	<pre>{
	  "Actions": [
	    {
	      "Operation": "set partial",
	      "TargetObject": "ProgramLibrary",
	      "Results": [
	        {
	          "StatusInfo": "SUCCESS",
	          "object": {
	            "id": 2,
	            "Notes": "",
	            "Level": "",
	            "ProgramInstanceId" = "22",
	            "Configuration Link: "",
	            "ProgramLibraryId": "2",
	            "Latitude": "41.0",
	            "Status": "Running",
	            "WorkingDirectory: "User",
	            "XpanelUrl": "",
	            "Longitude": "-73.93333",
	            "Name": "Room01",
	            "AddressSetsLocation": true,
	            "Time Zone": "America/New_York",
	            "Location": "NJC"
	          },
	          "StatusId": 0,
	          "Path":
	"Devices.Programs.ProgramInstanceLibrary"
	        }
	      ],
	      "Version": "2.0.1"
	    }
	  ]
	}</pre>

200	Invalid input (returns `"StatusInfo": "INVALID ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Read ProgramInstance

This method returns information about a specific program instance in the program instance library using `ProgramInstanceId`.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/ProgramInstance/{ProgramInstanceId}

Parameters

GET ProgramInstance/{ProgramInstanceId} Parameters

Name	Description
`ProgramInstanceId`	**Required.** String (path) that sets the ID of the program instance that will be returned.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/ProgramInstance/
[ProgramInstanceId]" -H "accept: application/json" -H "Authorization:
[Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/ProgramInstance/PI1
" -H "accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramInstance/
[ProgramInstanceId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	<pre>{
	  "Device": {
	  "Programs": {
	  "ProgramInstance": {
	  "ProgramInstanceId1": {
	  "id": 2,
	  "Notes": "",
	  "Level": "",
	  "Configuration Link: "",
	  "ProgramLibraryId": "33",
	  "Latitude": "41.0",
	  "Status": "Running",
	  "XpanelUrl": "",
	  "WorkingDirectory: "User",
	  "Longitude": "-73.93333",
	  "ProgramInstanceId": "21",
	  "Name": "progInstance RoomName2",
	  "AddressSetsLocation": false,
	  "Time Zone": "America/New_York",
	  "Location": "NJC"
	  },
	  }
	  }
	  }
	  }
	}</pre>

200	Invalid input (returns `"StatusInfo": "INVALID ID"`)
404	Not found
500	Internal server error
503	Service unavailable

Delete ProgramInstance

This method deletes a program instance from the program instance library using `ProgramInstanceId`.

Syntax

- **HTTPS Method:** DELETE
- **URI:** /VirtualControl/config/api/ProgramInstance/{ProgramInstanceId}

Parameters

DELETE ProgramInstance/{ProgramInstanceId} Parameters

Name	Description
ProgramInstanceId	**Required.** String (path) that sets the ID of the program instance to delete.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X DELETE
"https://[ServerURL]/VirtualControl/config/api/ProgramInstance/
[ProgramInstanceId]" -H "accept: application/json" -H "Authorization:
[Token]"
```

Example cURL Command

```
curl -X DELETE
"https://123.456.789.000/VirtualControl/config/api/ProgramInstance/PI1
" -H "accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/ProgramInstance/
[ProgramInstanceId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Actions": [
	{
	"Operation": "set partial",
	"TargetObject": "ProgramLibrary",
	"Results": [
	{
	"StatusInfo": Deleted",
	"object": "NULL"
	"StatusId": 0,
	"path":
	"Device.Programs.ProgramInstanceLibrary"
	}
	],
	"Version": "2.0.1"
	}
	]
	}

Code	Description
200	Invalid input (returns `"StatusInfo": "INVALID ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

NOTE: When a program instance (room) is created, it will move from the initializing to running state in sequence. If a DELETE method is issued for the room during this time, an INVALID DELETE OPERATION ERROR is returned. Waiting a few seconds between the POST and DELETE requests avoids this error.

DeviceProgramMap API

The DeviceProgramMap API is used to create, view, and delete device-to-program maps in the Crestron Virtual Control server's DeviceProgramMap library.

Device-to-program mapping allows devices connected to the Crestron Virtual Control server to be mapped to specific programs across the platform.

Create DeviceProgramMap

This method creates a new device-to-program map in the device-to-program map library.

Syntax

- **HTTPS Method:** POST
- **URI:** /VirtualControl/config/api/DeviceProgramMap

Parameters

Parameter Content Type: Application/JSON

POST DeviceProgramMap Parameters

Name	Description

`body`	**Required.** JSON body parameter that creates a new device-to-program map entry.
	
	Example Value:
	`{`
	`  "ProgramInstanceId": "1",`
	`  "MacAddress": "11:22:33:44:55:66",`
	`  "DeviceIpId": "9",`
	`  "ProgramIpId": "10"`
	`}`

The following body parameters may be included in the POST request:

POST DeviceProgramMap Body Parameters

Name	Description
`ProgramInstanceId`	String that sets the ID of the program instance that will be mapped to the device. Length must be between 1–64 characters.
`MacAddress`	String that sets the MAC (Media Access Control) address of the mapped device.
`DeviceIpId`	String that sets the IP ID of the mapped device on the network. Value must be between 2 and 65535.
`ProgramIpId`	String that sets the IP ID of the mapped program instance on the network. Value must be between 2 and 65535.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X POST
"https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap" -H
"accept: application/json" -H: "Authorization: [Token]" -H
"Content-Type: application/json" -d "{ \"[Parameter]\": \"[Value]\ "}"
```

Example cURL Command

```
curl -X POST
"https://123.456.789.000/VirtualControl/config/api/DeviceProgramMap" -
H "accept: application/json" -H: "Authorization: 1234567890" -H
"Content-Type: application/json" -d "{ \"ProgramInstanceId\": \"1\",
\"MacAddress\": \"11:22:33:44:55:66\", \"DeviceIpId\": \"9\",
\"ProgramIpId\": \"10\"}"
```

Request URL

```
http://[ServerURL]/VirtualControl/config/api/DeviceProgramMap
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	<pre>{
	  "Actions": [
	    {
	      "Operation": "set partial",
	      "TargetObject": "DeviceProgramMapLibrary",
	      "Results": [
	        {
	          "StatusInfo": "SUCCESS",
	          "object": {
	            "UniqueId: "DeviceProgramMapId1",
	            "MacAddress": "11:22:33:44:55:66",
	            "DeviceIpId": "9",
	            "ProgramInstanceId": "p003",
	            "ProgramIpId": "10",
	            "Status": "ONLINE"
	          },
	          "StatusId": 0,
	          "Path":
	"Devices.Programs.DeviceProgramMapLibrary"
	        }
	      ],
	      "Version": "2.0.1"
	    }
	  ]
	}</pre>

200	Invalid input (returns `"StatusInfo": "DUPLICATE UNIQUE ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Read All DeviceProgramMaps

This method returns all device-to-program maps in the device-to-program map library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/DeviceProgramMap

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap" -H
"accept: application/json" -H: "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/DeviceProgramMap" -
H "accept: application/json" -H: "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Device": {
	"Programs": {
	"DeviceProgramMapLibrary": {
	"DeviceProgramMapId1" {
	"UniqueId: "DeviceProgramMapId1",
	"MacAddress": "11:22:33:44:55:66",
	"DeviceIpId": "9",
	"ProgramInstanceId": "p003",
	"ProgramIpId": "10",
	"Status": "ONLINE",
	},
	"DeviceProgramMapId2" {
	"UniqueId: "DeviceProgramMapId2",
	"MacAddress": "aa:bb:cc:dd:ee:ff",
	"DeviceIpId": "11",
	"ProgramInstanceId": "p001",
	"ProgramIpId": "12",
	"Status": "ONLINE",
	}
	}
	}
	}
	}

404	Not found
500	Internal server error
503	Service unavailable

Delete DeviceProgramMap

This method deletes a device-to-program map from the device-to-program map library.

Syntax

- **HTTPS Method**: DELETE
- **URI:** /VirtualControl/config/api/DeviceProgramMap

Parameters

Parameter Content Type: Application/JSON

DELETE DeviceProgramMap Parameters

Name	Description

`body`	**Required.** JSON body parameter that deletes a device-to-program map entry.
	
	Example Value:
	`{`
	`  "ProgramInstanceId": "1",`
	`  "MacAddress": "11:22:33:44:55:66",`
	`  "DeviceIpId" "9",`
	`  "ProgramIpId": "10"`
	`}`

The following body parameters may be included in the DELETE request:

DELETE DeviceProgramMap Body Parameters

Name	Description
`ProgramInstanceId`	String that sets the program instance ID of the program instance to delete. Length must be between 1–64 characters.
`MacAddress`	String that sets the MAC (Media Access Control) address of the mapped device to delete.
`DeviceIpId`	String that sets the IP ID of the mapped device on the network to delete. Value must be between 2 and 65535.
`ProgramIpId`	String that sets the IP ID of the mapped program instance on the network to delete. Value must be between 2 and 65535.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X DELETE
"https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap" -H
"accept: application/json" -H: "Authorization: [Token]" -H "Content-
Type: application/json" -d "{ \"[Parameter]\": \"[Value]\ "}"
```

Example cURL Command

```
curl -X DELETE
"https://123.456.789.000/VirtualControl/config/api/DeviceProgramMap" -
H "accept: application/json" -H: "Authorization: 1234567890" -H
"Content-Type: application/json" -d "{ \"ProgramInstanceId\": \"1\",
\"MacAddress\": \"11:22:33:44:55:66\", \"DeviceIpId\": \"9\",
\"ProgramIpId\": \"10\"}"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap
```

Response Codes

Code	Description
200	Successful operation Example Value: `{ "Actions": [ { "Operation": "set partial", "TargetObject": "DeviceProgramMapLibrary", "Results": [ { "StatusInfo": Deleted", "object": "NULL" "StatusId": 0, "path": "Device.Programs.DeviceProgramMapLibrary" } ], "Version": "2.0.1" } ] }`
200	Invalid input (Returns `"StatusInfo": "INVALID ID"`)
400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Read DeviceProgramMap

This method returns information about a specific device-to-program map in the device-to-program map library using `ProgramInstanceId`.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/DeviceProgramMap/{DeviceProgramMapId}

Parameters

GET DeviceProgramMap/{DeviceProgramMapId} Parameters

Name	Description
`DeviceProgramMapId`	**Required.** String (path) that sets the ID of the device-to-program map that will be returned. Maximum length is 255 characters.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap/
[DeviceProgramMapId]" -H "accept: application/json" -H:
"Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/DeviceProgramMap/
DeviceProgramMapId1" -H "accept: application/json" -H: "Authorization:
1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/DeviceProgramMap/
[DeviceProgramMapId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Device": {
	"Programs": {
	"DeviceProgramMapLibrary": {
	"DeviceProgramMapId1" {
	"UniqueId: "DeviceProgramMapId1",
	"MacAddress": "11:22:33:44:55:66",
	"DeviceIpId": "dev03",
	"ProgramInstanceId": "p003",
	"ProgramIpId": "p02",
	"Status": "ONLINE"
	}
	}
	}
	}
	}

200	Invalid input (Returns `"StatusInfo": "INVALID ID"`)
404	Not found
500	Internal server error
503	Service unavailable

DeviceMap API

The DeviceMap API is used to view device maps in the Crestron Virtual Control server's device map library. Device maps allow access to device resources across the entire Crestron Virtual Control server.

Read All DeviceMaps

This method returns all device maps in the device map library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/DeviceMap

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET "https://[ServerURL]/VirtualControl/config/api/DeviceMap"
-H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/DeviceMap" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/DeviceMap
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Device": {
	"Programs": {
	"DeviceMapLibrary": {
	"1: {
	"SupportAssociation: "false",
	"UniqueId": "1"
	"ProgramInstanceId": "PGM1"
	"DeviceId": "DMID1"
	"MacAddress": "11:22:33:44:55:66",
	"Make": "Crestron",
	"Model": "VC-4",
	"HostName": "ubuntu",
	"Name": "DevMap",
	"Description": "new device map object"
	"ProgramIpId": "2",
	"Status": online
	}
	}
	}
	}
	}

404	Not found
500	Internal server error
503	Service unavailable

SystemTable API

The SystemTable API is used to view the Crestron Virtual Control server's system table and to modify objects within the system table.

Read All SystemTable Objects

This method returns all system table information stored in the system table library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/SystemTable

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/SystemTable" -H
"accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/SystemTable" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/SystemTable
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Device": {
	  "Programs": {
	  "SystemTable": {
	  "HydrogenUrl":
	"https://fc.crestron.com.io/api/Device/Create",
	  "NumProgramsRegistered": 0,
	  "ID": 1,
	  "OCSPState": 1,
	  "OCSPTimeoutSeconds": 30,
	  "CloudUrl": "",
	  "NumProgramsRunning": 0,
	  "HydrogenConnectedState": 0,
	  "Valid": 0,
	  "NumProgramsLicensed": 500,
	  "SecureGatewayMode": 3
	  }
	  }
	  }
	}

404	Not found
500	Internal server error
503	Service unavailable

Modify SystemTable Object

This method modifies a specific system table object in the system table library.

Syntax

- **HTTPS Method:** PUT
- **URI:** /VirtualControl/config/api/SystemTable

Parameters

Parameter Content Type: Application/JSON

PUT SystemTable Parameters

Name	Description

`body`	**Required.** JSON body parameter that modifies one or more SystemTable objects.
	
	Example Value:
	`{`
	`  "OCSPState": "1",`
	`  "OCSPTimeoutSeconds": "33",`
	`  "CloudUrl": "[CloudUrl]",`
	`  "SecureGatewayMode": "3"`
	`}`

The following body parameters may be included in the PUT request:

PUT SystemTable Body Parameters

Name	Description
`OCSPState`	String that sets the state of the OCSP (Online Certificate Status Protocol) client for validating digital certificates. Valid values are 1–3. • Enter `1` to turn OCSP off. • Enter `2` to set the OCSP client behavior to staple only. In this state, the Crestron Virtual Control server appends a time-stamped, self-signed response to a certificate sent by the web browser client for self-validation. • Enter `3` to set OCSP client behavior to remote. In this state, the web browser client sends remote certificates that are validated by the Crestron Virtual Control server.
`OCSPTimeoutSeconds`	String that sets the duration (in seconds) that OCSP times out a certificate validation request.
`CloudUrl`	String that sets the URL for connecting the Crestron Virtual Control server to Crestron Fusion® software.
`SecureGatewayMode`	String that sets the secure gateway mode for the Crestron Virtual Control server. Valid values are 1–3. • Enter `1` to set the gateway mode to secure only. • Enter `2` to set the gateway mode to both secure and not secure. • Enter `3` to set the gateway mode to secure when using remote subnets and not secure when using local subnets.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X PUT
"https://[ServerURL]/VirtualControl/config/api/SystemTable" -H
"accept: application/json" -H "Authorization: [Token]" -H
"Content-Type: application/json" -d "{ \"[Parameter]\": \"[Value]\"}"
```

Example cURL Command

```
curl -X PUT
"https://123.456.789.000/VirtualControl/config/api/SystemTable" -H
"accept: application/json" -H "Authorization: 1234567890" -H
"Content-Type: application/json" -d "{ \"OCSPState\": \"1\"}"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/SystemTable
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Actions": [
	  {
	  "Operation": "set partial",
	  "TargetObject": "SystemTable",
	  "Results": "[
	  {
	  "StatusInfo": "SUCCESS",
	  "object": {
	  "OCSPState": 1,
	  "OCSPTimeoutSeconds": 33,
	  "CloudUrl": "",
	  "SecureGatewayMode": 3
	  },
	  "StatusId": 0,
	  "Path": "Device.Programs.SystemTable"
	  }
	  ],
	  "TargetObject": "SystemTable",
	  "Version": "2.0.1"
	  }
	  ]
	}

400	Bad request error
404	Not found
500	Internal server error
503	Service unavailable

Ethernet API

The Ethernet API is used to view the Ethernet status and settings for the Crestron Virtual Control server.

Read All Ethernet Objects

This method returns all Ethernet information stored in the Ethernet library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/Ethernet

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET "https://[ServerURL]/VirtualControl/config/api/Ethernet" -
H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/Ethernet" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/Ethernet
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Device": {
	  "Ethernet": {
	  "Adapters": [
	  {
	  "LinkStatus": "OK",
	  "IPv4": {
	  "Addresses": [
	  {
	  "SubnetMask": "255.255.255.0",
	  "Address": "192.255.255.135",
	  }
	  ],
	  "Dnsservers": [
	  "192.255.255.2"
	  ],
	  "DefaultGateway": "192.255.255.2",
	  "IsDhcpEnabled": "ON"
	  },
	  "Name": "ens33",
	  "MacAddress": "11.22.33.44.55.66"
	  }
	  ],
	  "HostName": "ubuntu",
	  "DomainName": "localdomain"
	  }
	  }
	}

404	Not found
500	Internal server error
503	Service unavailable

DeviceInfo API

The DeviceInfo API is used to view information for the Crestron Virtual Control server.

Read All DeviceInfo Objects

This method returns all information stored for devices in the device info library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/DeviceInfo

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET "https://[ServerURL]/VirtualControl/config/api/DeviceInfo"
-H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/DeviceInfo" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/DeviceInfo
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Device": {
	"DeviceInfo:" {
	"DeviceKey": "5365473269d52577563277770050e8c7",
	"DeviceId": "000c29d3c7a2000c29d3c7a2000c29d",
	"ApplicationVersion": "2.000.3703.23036",
	"Manufacturer": "Crestron",
	"ID": "1",
	"MacAddress": "11:22:33:44:55:66",
	"Model1": "VC-4",
	"Category": "Control System",
	"BuildDate": "Feb 20 2018",
	"Version": "2.000.3703.23036",
	"Name": "ubuntu"
	}
	}
	}

404	Not found
500	Internal server error
503	Service unavailable

LicenseRegistry API

This method is used to view license information for the Crestron Virtual Control server.

Read All LicenseRegistry Objects

This method returns all licenses stored for devices in the license registry library.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/LicenseRegistry

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/LicenseRegistry"
-H "accept: application/json" -H "Authorization: [Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/LicenseRegistry" -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/LicenseRegistry
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	  "Device": {
	  "LicenseRegistry": {
	  "Licenses": [
	  {
	  "Name": "AirMedia",
	  "VendorId": "1",
	  "ModuleId": "1",
	  "Info": "",
	  "Expiration": "123",
	  "Key": "12345",
	  "Count": "1"
	  }
	  ],
	  }
	  }
	}

404	Not found
500	Internal server error
503	Service unavailable

IpTableByPID API

This method is used to view the Crestron Virtual Control IP Table for different programs instances by using `ProgramInstanceId`.

Read IpTableByPID Object

This method returns a specific IP table for the supplied `ProgramInstanceId`.

Syntax

- **HTTPS Method:** GET
- **URI:** /VirtualControl/config/api/IpTableByPID/{ProgramInstanceID}

Parameters

GET IpTableByPID/{ProgramInstanceId} Parameters

Name	Description
`ProgramInstanceID`	**Required.** String (path) that sets the ID of the program instance that will have its IP table returned. Maximum length is 255 characters.

Responses

Response Content Type: Application/JSON

cURL Base Command

```
curl -X GET
"https://[ServerURL]/VirtualControl/config/api/IpTableByPID/
[ProgramInstanceId]" -H "accept: application/json" -H "Authorization:
[Token]"
```

Example cURL Command

```
curl -X GET
"https://123.456.789.000/VirtualControl/config/api/IpTableByPID/P01 -H
"accept: application/json" -H "Authorization: 1234567890"
```

Request URL

```
https://[ServerURL]/VirtualControl/config/api/IpTableByPID/
[ProgramInstanceId]
```

Response Codes

Code	Description

200	Successful operation
	
	Example Value:
	{
	"Device": {
	"Programs": {
	"IpTableByPID": {
	"IPTableByPIDGetId": {
	"UniqueId": "1",
	"ProgramInstanceId": "P01",
	"ProgramIpId": "12",
	"Model": "VC-4",
	"Description: "Iptable",
	"remote_ip": "192.255.255.4",
	"Status": "Online",
	"device_type": "1",
	"MacAddress": "11:22:33:44:55:66",
	"DeviceId": "1",
	"Hostname": "VC4",
	"SupportAssocation": ""
	}
	}
	}
	}
	}

404	Not found
500	Internal server error
503	Service unavailable

This page is intentionally left blank.

Crestron Electronics, Inc.
15 Volvo Drive, Rockleigh, NJ 07647
Tel: 888.CRESTRON
Fax: 201.767.7576
www.crestron.com

Programming Guide – DOC. 8314A
(2051308)
11/30/20
Specifications subject to
change without notice.