



View this document in HTML
crestron.com/docs/8314



Programming Guide

REST API for Crestron Virtual Control Server-Based Control System

Have feedback on this document? Contact docfeedback@crestron.com.

Original Instructions

The U.S. English version of this document is the original instructions.
All other languages are a translation of the original instructions.

Crestron product development software is licensed to Crestron dealers and Crestron Service Providers (CSPs) under a limited nonexclusive, nontransferable Software Development Tools License Agreement. Crestron product operating system software is licensed to Crestron dealers, CSPs, and end-users under a separate End-User License Agreement. Both of these Agreements can be found on the Crestron website at www.crestron.com/legal/software_license_agreement.

The product warranty can be found at www.crestron.com/warranty.

The specific patents that cover Crestron products are listed at www.crestron.com/legal/patents.

Certain Crestron products contain open source software. For specific information, visit www.crestron.com/opensource.

Crestron, the Crestron logo, and Crestron Fusion are either trademarks or registered trademarks of Crestron Electronics, Inc. in the United States and/or other countries. Linux is either a trademark or a registered trademark of Linus Torvalds in the United States and/or other countries. Red Hat is either a trademark or a registered trademark of Red Hat, Inc. in the United States and/or other countries. Other trademarks, registered trademarks, and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. Crestron disclaims any proprietary interest in the marks and names of others. Crestron is not responsible for errors in typography or photography.

©2024 Crestron Electronics, Inc.

Contents

| | |
|--|----------|
| Overview | 1 |
| What's New? | 2 |
| October 29, 2024 | 2 |
| April 15, 2022 | 2 |
| Quick Start | 3 |
| Prerequisites and Assumptions | 4 |
| Authentication | 5 |
| Make API Calls | 7 |
| API Reference | 8 |
| Authentication API | 9 |
| Read All Authentication Objects | 9 |
| Read All Authentication Groups | 10 |
| Create Authentication Group | 11 |
| Remove Authentication Group | 13 |
| DeviceInfo API | 16 |
| Read All DeviceInfo Objects | 16 |
| DeviceMap API | 18 |
| Read All DeviceMaps | 18 |
| DeviceProgramMap API | 20 |
| Create DeviceProgramMap | 20 |
| Read All DeviceProgramMaps | 22 |
| Delete DeviceProgramMap | 24 |
| Read DeviceProgramMap | 26 |
| Ethernet API | 29 |
| Read All Ethernet Objects | 29 |
| IpTableByPID API | 31 |
| Read ProgramInstance | 31 |
| LicenseRegistry API | 33 |
| Read All LicenseRegistry Objects | 33 |
| ProgramInstance API | 35 |
| Add ProgramInstance | 35 |
| Read All ProgramInstances | 38 |
| Modify ProgramInstance | 40 |
| Read ProgramInstance | 43 |
| Delete ProgramInstance | 44 |
| ProgramLibrary API | 47 |
| Add Program | 47 |
| Read All Programs | 49 |
| Modify Program | 50 |

| | |
|-------------------------------------|-----------|
| Read Program | 53 |
| Delete Program | 54 |
| Delete Program File | 56 |
| RestartService API | 59 |
| StopDebugging API | 61 |
| Post Stop Debugging Operation | 61 |
| SystemTable API | 63 |
| Read All SystemTable Objects | 63 |
| Support | 65 |

Overview

The [Crestron Virtual Control server-based control system](#) provides a scalable solution for deploying programs, rooms, and devices across an enterprise. The Crestron® control system infrastructure resides entirely on a remote server, which is installed and configured using supported Linux® operating system platforms.

This documentation describes the REST API for the Crestron Virtual Control server. The REST API allows programmers to modify device settings and view device status using REST API calls.

NOTE: This document is current as of the 4.0004.00093 software release.

The 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 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.

What's New?

Updates to the Crestron Virtual Control REST API and documentation are described below from newest to oldest.

October 29, 2024

- **Documentation updates**
 - Added "RestartRequired" response property to calls within [ProgramInstance API on page 35](#).
-

April 15, 2022

- **Documentation updated for SIMPL on VC-4 Release**
 - Added [StopDebugging API on page 61](#) reference topic.
 - Updated [DeviceInfo API on page 16](#), [ProgramInstance API on page 35](#), [ProgramLibrary API on page 47](#), and [SystemTable API on page 63](#) topics.
 - Converted documentation to latest HTML manual template.

Quick Start

The Quick Start topics describe how to get started with the Crestron Virtual Control REST API, including how to authenticate and make calls with the API.

This section provides the following information:

- [Prerequisites and Assumptions](#)
- [Authentication](#)
- [Make API Calls](#)

Prerequisites and Assumptions

This document assumes the following knowledge prerequisites prior to working with the Crestron Virtual Control REST API:

- 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 Server Software Product Manual](#).

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

The **Status > Rooms** page is displayed by default.

Crestron Virtual Control Web Configuration Interface







The screenshot shows the Crestron Virtual Control web configuration interface. The header displays the Crestron logo and the text "CRESTRON VIRTUAL CONTROL". Below the header, there is a navigation bar with "Server: VC-4" and an "Actions" button. The main content area is titled "Rooms" and contains a table with the following data:

| Room | Room ID | Status | Program | Actions | Debugging |
|--------------------------------|-----------------|-----------|--------------|---|-----------|
| CCUIVswitch | CCUIVSWITCHRM1 | ▶ Running | CCUIsttBuild | i ✎ 🗑 | |
| JNREG624 | JNREG | ▶ Running | JN624REG | i ✎ 🗑 | |
| KVV624Merged | KVV624MERGEDRM1 | ▶ Running | KVV624Merged | i ✎ 🗑 | |
| KVVCCUIHDMD400 | CCUIHDMD400RM1 | ▶ Running | CCUIsttBuild | i ✎ 🗑 | |
| Room001 | ROOM001 | ▶ Running | Testbug | i ✎ 🗑 | |

The interface also includes a sidebar with navigation options: Devices, Device Info, Network, Licenses, Crestron Fusion HTML5, BACnet, and BACnet Advanced. The footer contains the text "© 2022 Crestron Electronics, Inc." and a "Privacy Statement" link.

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

Settings Tab - Tokens

| ▼ Tokens | | | | Add Token |
|------------------------|-------------|------------|---|-----------|
| Token | Description | Status | Actions | |
| eyJ0eXAiOiJKV1QiLCJhbC | SIMPL | Read-Write |   | |
| eyJ0eXAiOiJKV1QiLCJhbC | Test | Read-Write |   | |
| eyJ0eXAiOiJKV1QiLCJhbC | ZRC | Read-Only |   | |

« < 1 > » 10 ▾

3. Select **Add Token**. The **Add Token** dialog box is displayed.

Add Token Dialog Box

+ Add Token ×

Description

Read-Only

Add Cancel

4. Enter a description for the token in the **Description** text field.
5. (Optional) Fill the **Read-Only** check box to give the token read-only permissions. The token will have read/write permissions if this check box is not filled.
6. Select **Add**. A new token is generated and added to the **Tokens** table.

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.

Make API Calls

To make API calls to the Crestron Virtual Control server:

1. Obtain a valid authentication token from the Crestron Virtual Control web configuration interface. For more information, refer to [Authentication on page 5](#).
2. Open an API development environment program (such as [Postman](#)) 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.

API Reference

The REST API interface allows configuration and device status changes to be passed to the Crestron Virtual Control server. The REST API calls defined in this section are used to get and set data for different device functions and are current as of the version 4.0004.00093 software release.

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

This section provides the following information:

- [Authentication API](#)
- [DeviceInfo API](#)
- [DeviceMap API](#)
- [DeviceProgramMap API](#)
- [Ethernet API](#)
- [IpTableByPID API](#)
- [LicenseRegistry API](#)
- [ProgramInstance API](#)
- [ProgramLibrary API](#)
- [RestartService API](#)
- [StopDebugging API](#)
- [SystemTable API](#)

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.

NOTE: Authentication groups and users must be created on the Linux platform before they can be configured for the Crestron Virtual Control server. For more information, refer to the [Crestron Virtual Control Server Software Product Manual](#).

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
- **Base 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 |
| <div style="border: 1px solid #ccc; padding: 10px;"><p><u>Example Response</u></p><pre>{ "Device": { "Authentication": { "Groups": [{ "Name": "Admins", "AccessLevel": "Administrator" }] } } }</pre></div> | |
| 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
- **Base 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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Response</u></p><pre>{ "Device": { "Authentication": { "Groups": [{ "Name": "Test", "AccessLevel": "Administrator" }] } } }</pre></div> | |
| 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
- **Base URI:** /VirtualControl/config/api/Authentication/Groups

Parameters

POST Authentication/Groups Parameters

| Name | Description |
|------|---|
| body | Required. JSON body parameter that creates a new authentication group. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "Authentication": { "AddGroup": { "Name": "Test", "AccessLevel": "Administrator" } } } }</pre></div> |

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 add. |
| AccessLevel | Required. String that sets the access level for the authentication group. Valid values are listed below. <ul style="list-style-type: none">• 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/Groups" -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
- **Base URI:** /VirtualControl/config/api/Authentication/Groups

Parameters

POST Authentication/Groups Parameters

| Name | Description |
|------|--|
| body | Required. JSON body parameter that removes an existing authentication group. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "Authentication": { "RemoveGroup": { "Name": "Test", "AccessLevel": "Administrator" } } } }</pre></div> |

The following body parameters must be included in the POST request

POST Authentication/Groups Body Parameters

| Name | Description |
|-------------|---|
| Name | Required. String that provides the name of the authentication group to remove. |
| AccessLevel | Required. String that provides the access level of the removed authentication group. Valid values are listed below. <ul style="list-style-type: none">• 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/Groups" -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 |

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
- **Base 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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "DeviceInfo": { "DeviceKey": "5365473269d5257756327770050e8c7", "DeviceId": "000c29d3c7a2000c29d3c7a2000c29d", "ApplicationVersion": "1.8001.0124", "Manufacturer": "Crestron", "ID": "1", "MacAddress": "11:22:33:44:55:66", "Model1": "VC-4", "Category": "Control System", "BuildDate": "Feb 20 2022", "Version": "2.7100.00016", "Name": "redhat", "PythonVersion": "3.8.8\n", "MonoVersion": "6.12.0.107" } } }</pre></div> | |
| 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 environment.

Read All DeviceMaps

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

Syntax

- **HTTPS Method:** GET
- **Base 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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "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 } } } } }</pre></div> | |
| 404 | Not found |
| 500 | Internal server error |
| 503 | Service unavailable |

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
- **Base URI:** /VirtualControl/config/api/DeviceProgramMap

Parameters

POST DeviceProgramMap Parameters

| Name | Description |
|------|---|
| body | Required. JSON body parameter that creates a new device-to-program map entry. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p><u>Example Value</u></p><pre>{ "ProgramInstanceId": "1", "MacAddress": "11:22:33:44:55:66", "DeviceIpId": "9", "ProgramIpId": "10" }</pre></div> |

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

POST DeviceProgramMap Body Parameters

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

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

Response Codes

| Code | Description |
|------|---|
| 200 | Successful operation |
| | <pre>Example Response { "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
- **Base 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 |
| | <div style="border: 1px solid #ccc; padding: 10px;"><p><u>Example Response</u></p><pre>{ "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", } } } } }</pre></div> |
| 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
- **Base URI:** /VirtualControl/config/api/DeviceProgramMap

Parameters

DELETE DeviceProgramMap Parameters

| Name | Description |
|------|---|
| body | Required. JSON body parameter that deletes a new device-to-program map entry. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "ProgramInstanceId": "1", "MacAddress": "11:22:33:44:55:66", "DeviceIpId": "9", "ProgramIpId": "10" }</pre></div> |

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

DELETE DeviceProgramMap Body Parameters

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

Syntax

- **HTTPS Method:**GET
- **Base 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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "Programs": { "DeviceProgramMapLibrary": { "DeviceProgramMapId1" { "UniqueId": "DeviceProgramMapId1", "MacAddress": "11:22:33:44:55:66", "DeviceIpId": "dev03", "ProgramInstanceId": "p003", "ProgramIpId": "p02", "Status": "ONLINE" } } } } }</pre></div> | |
| 200 | Invalid input (returns "StatusInfo": "INVALID ID") |
| 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
- **Base 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 |

IpTableByPID API

The IpTableByPID API is used to view the Crestron Virtual Control IP Table for different programs instances by using **ProgramInstanceId**.

Read ProgramInstance

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

Syntax

- **HTTPS Method:** GET
- **Base 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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "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", "SupportAssociation": "" } } } } }</pre></div> | |
| 200 | Invalid input (returns "StatusInfo": "INVALID ID") |
| 404 | Not found |
| 500 | Internal server error |
| 503 | Service unavailable |

LicenseRegistry API

The LicenseRegistry API 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
- **Base 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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "LicenseRegistry": { "Licenses": [{ "Name": "AirMedia", "VendorId": "1", "ModuleId": "1", "Info": "", "Expiration": "123", "Key": "12345", "Count": "1" }] } } }</pre></div> | |
| 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).

Add ProgramInstance

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

Syntax

- **HTTPS Method:** POST
- **Base 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 (street address) 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 |
|---------------------|---|
| AddressSetsLocation | Optional. Boolean (form data) that determines whether the address provided for Location 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 |
| | <div style="border: 1px solid black; padding: 10px;"><p><u>Example Response</u></p><pre>{ "Actions": [{ "Operation": "set partial", "TargetObject": "ProgramLibrary", "Results": [{ "StatusInfo": "SUCCESS", "object": { "id": 2, "Notes": "", "Level": "", "ProgramInstanceId": "22", "ProgramLibraryId": "2", "UserFile": "", "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></div> |
| 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
- **Base 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
```

| Code | Description |
|------|---|
| 200 | Successful operation |
| | <p data-bbox="315 254 532 279"><u>Example Response</u></p> <pre data-bbox="332 331 1031 1780"> { "Device": { "Programs": { "ProgramInstanceLibrary": { "ProgramInstanceId1": { "id": 2, "Notes": "", "Level": "", "ProgramLibraryId": "33", "Latitude": "41.0", "Status": "Running", "XpanelUrl": "", "WorkingDirectory": "User", "Longitude": "-73.93333", "ProgramInstanceId": "21", "UserFile": "", "Name": "progInstance RoomName2", "AddressSetsLocation": false, "Time Zone": "America/New_York", "Location": "NJC", "DebuggingEnabled": false, "SupportsProgrammaticIntegration": false, "Runtime": "", "LastStarted": "2024-08-13T13:33:35Z" "RestartRequired": true }, "ProgramInstanceId2": { "id": 1, "Notes": "", "Level": "", "ProgramLibraryId": "34", "Latitude": "41.0", "Status": "Running", "XpanelUrl": "", "WorkingDirectory": "User", "Longitude": "-33.93333", "ProgramInstanceId": "20", "UserFile": "", "Name": "progInstance RoomName1", "AddressSetsLocation": true, "Time Zone": "America/New_York", "Location": "NJC", "DebuggingEnabled": true, "SupportsProgrammaticIntegration": false, "Runtime": "", "LastStarted": "2024-08-13T13:33:35Z" "RestartRequired": false } } } } } </pre> |

| Code | Description |
|------|-----------------------|
| 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
- **Base URI:** /VirtualControl/config/api/ProgramInstance

Parameters

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 sets the access level of the program instance. Maximum length is 255 characters. |
| Location | Optional. String (form data) that sets the location (street address) 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 <code>true</code> . |
| Stop | Optional. String (form data) used to stop the specific program. The only valid value is <code>true</code> . |
| AddressSetsLocation | Optional. Boolean (form data) that determines whether the address provided for <code>Location</code> is used to set the location properties automatically. |

| Name | Description |
|------------------|--|
| UserFile | Optional. A custom user file (form data) for the program instance. Files may be uploaded individually or zipped. |
| DebuggingEnabled | Optional. Boolean property (form data) that determines whether the debugging option for the program is turned on or off (SIMPL programs only). |
| Restart | Optional. String (form data) used to restart the specific program. The only valid value is <code>true</code> . |

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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "TargetObject": "ProgramInstanceLibrary", "Results": [{ "StatusInfo": "SUCCESS", "object": { "id": 2, "Notes": "", "Level": "", "ProgramInstanceId": "22", "UserFile": "", "ProgramLibraryId": "2", "Latitude": "41.0", "Status": "Running", "WorkingDirectory": "User", "XpanelUrl": "", "Longitude": "-73.93333", "Name": "Room01", "AddressSetsLocation": true, "Time Zone": "America/New_York", "Location": "NJC", "DebuggingEnabled": true, "SupportsProgrammaticIntegration": false, "Runtime": "", "LastStarted": "2024-08-13T13:33:35Z" "RestartRequired": false }, "StatusId": 0, "Path": "Devices.Programs.ProgramInstanceLibrary" }], "Version": "2.0.1" }] }</pre></div> | |
| 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
- **Base 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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "Programs": { "ProgramInstance": { "ProgramInstanceId": { "id": 2, "Notes": "", "Level": "", "ProgramLibraryId": "33", "UserFile": "", "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", "DebuggingEnabled": true, "SupportsProgrammaticIntegration": false, "Runtime": "", "LastStarted": "2024-08-13T13:33:35Z" "RestartRequired": true }, }, } } }</pre></div> | |
| 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
- **Base 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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "TargetObject": "ProgramLibrary", "Results": [{ "StatusInfo": Deleted, "object": "NULL" "StatusId": 0, "path": "Device.Programs.ProgramInstanceLibrary" }], "Version": "2.0.1" }] }</pre></div> | |
| 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.

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
- **Base 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 "FriendlyName=Program01" -F "AppFile=prog01.zip"
```

Request URL

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

Response Codes

| Code | Description |
|------|----------------------|
| 200 | Successful operation |

Example Response

```
{
  "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"
    }
  ]
}
```

| Code | Description |
|------|--|
| 200 | Invalid input (returns "StatusInfo": "DUPLICATE ID") |
| 400 | Bad request error |
| 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
- **Base 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 |
| | <div style="border: 1px solid #ccc; padding: 10px;"><p><u>Example Response</u></p><pre>{ "Device": { "Programs": { "ProgramLibrary": { "1": { "ProgramId": "1", "FriendlyName": "Prog001", "Notes": "Test Program", "Tags": "", "AppFile": "Oscillator.zip", "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" "ProgramType": "SIMPL Windows", "ProgramName": "Oscilator.zip", "CompileDateTime": "8/20/2021 5:06 PM", "CresDBVersion": "205.05.004.00", "DeviceDBVersion": "200.75.001.00", "IncludeDatVersion": "2.17.035" } } } } }</pre></div> |
| 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
- **Base URI:** /VirtualControl/config/api/ProgramLibrary

Parameters

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: .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. |
| StartNow | Optional. String (form data) used to start the program when the AppFile program file is uploaded. The only valid value is <code>true</code> . |
| StartLater | This parameter is not currently in use. |

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
- **Base 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

Response Content Type: Application/JSON

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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Device": { "Programs": { "ProgramLibrary": { "1": { "ProgramId": "1", "FriendlyName": "Prog001", "Notes": "Test Program", "Tags": "", "AppFile": "Oscillator.zip", "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" "ProgramType": "SIMPL Windows", "ProgramName": "Oscillator.zip", "CompileDateTime": "8/20/2021 5:06 PM", "CresDBVersion": "205.05.004.00", "DeviceDBVersion": "200.75.001.00", "IncludeDatVersion": "2.17.035" } } } } }</pre></div> | |
| 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
- **Base 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

Response Content Type: Application/JSON

cURL Base Command

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

Example cURL 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 |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "TargetObject": "ProgramLibrary", "Results": [{ "StatusInfo": Deleted", "object": "NULL" "StatusId": 0, "path": "Device.Programs.ProgramLibrary" }], "Version": "2.0.1" }] }</pre></div> | |
| 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 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
- **Base 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

Response Content Type: Application/JSON

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 |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "TargetObject": "ProgramLibrary", "Results": [{ "StatusInfo": Deleted", "object": "NULL", "StatusId": 0, "path": "Device.Programs.ProgramLibrary" }], "Version": "2.0.1" }] }</pre></div> | |
| 200 | Invalid input (returns "StatusInfo": "FILE NOT FOUND ERROR") |
| 400 | Bad request error |
| 404 | Not found |
| 500 | Internal server error |
| 503 | Service unavailable |

RestartService API

This is used to restart the full Crestron Virtual Control service on the server.

Syntax

- **HTTPS Method:** POST
- **Base URI:** /VirtualControl/config/api/RestartService

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

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

Example cURL Command

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

Request URL

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

Response Codes

| Code | Description |
|---|-----------------------|
| 200 | Successful operation |
| <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "Results": [{ "path": "Device.Programs.SystemTable", "object": "Restarting VirtualControl Service", "StatusInfo": "SUCCESS", "StatusId": 0 }], "TargetObject": "SystemTable", "Version": "2.0.1" }] }</pre></div> | |
| 404 | Not found |
| 500 | Internal server error |
| 503 | Service unavailable |

StopDebugging API

The StopDebugging API is used to turn off debugging for all SIMPL programs that are running on the Crestron Virtual Control server.

Post Stop Debugging Operation

This method sets the stop debugging operation for the Crestron Virtual Control server.

Syntax

- **HTTPS Method:** POST
- **Base URI:** /VirtualControl/config/api/StopDebugging

Parameters

None

Responses

Response Content Type: Application/JSON

cURL Base Command

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

Example cURL Command

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

Request URL

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

Response Codes

| Code | Description |
|--|-----------------------|
| 200 | Successful operation |
| <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"><p><u>Example Value</u></p><pre>{ "Actions": [{ "Operation": "set partial", "Results": [{ "path": "Device.Programs.ProgramInstanceLibrary", "object": "Debugging Rooms Stopped", "StatusInfo": "SUCCESS", "StatusId": 0 }], "TargetObject": "ProgramInstanceLibrary", "Version": "2.0.1" }] }</pre></div> | |
| 404 | Not found |
| 500 | Internal server error |
| 503 | Service unavailable |

SystemTable API

This method is used to view the Crestron Virtual Control server's system table.

Read All SystemTable Objects

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

Syntax

- **HTTPS Method:** GET
- **Base 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 |

Support

For technical support when using the Crestron Virtual Control REST API, contact Crestron True Blue support via phone, email, or chat as described at www.crestron.com/Support.

