Hunter Douglas PowerView® Hub API for Home Automation Integration

Table of Contents

Overview

This document describes the latest PowerView® Hub REST API allowing developers to interact with Hunter Douglas shades with PowerView Motorization.

Background

PowerView Motorization consists of a PowerView Hub plus a collection of motorized shades and other accessories, including remotes, repeaters, secondary hubs, and scene controllers. Shade resources are the central concept in the PowerView API eco-system. Hubs communicate with Shades over a radio interface while the REST API is terminated over an HTTP based interface. Shades are typically organized into Rooms and then controlled either through individual controls or through Scenes. A Scene is a set of one or more Shades located in a single Room that are set to customized positions. So, when a Scene is activated, the Shades in that Scene will move to the positions described in the Scene. Since Scenes may only affect Shades within a single room, there is a Multi-Room Scene resource referenced as a Scene Collection in the API. Scene Collections are simply a set of Scenes that can be activated as one.

In general, most API resources have a number of additional attributes like order, color, icon, etc that are generally used by the PowerView iOS and Android applications to control their display in the application. Resources also tend to contain semi-permanent state information, such as current shade position, firmware revision, etc, that can be periodically queried and updated.

Since Shades are the central resource in the PowerView eco-system, it is useful to understand a few key state relationships between Hubs and Shades. Hubs maintain transient shade state such as position and battery level. Any API call that returns the shade position and/or the battery level is delivering the *last Hub saved value* of these attributes. In general, shades are moved via API calls to Shades, Scenes, and Scene Collections. Since the Hub is always involved in these motion events, it tracks the final shade position and saves it; however, shades can be moved without the Hub's knowledge. An individual can manually move a shade simply by pressing the motion button on the side of the shade. In addition, shades can be moved using a PowerView Motorization handheld remote control device. In both of these cases, the Hub is not told of the shade's new position. Consequently, to ensure an accurate Hub view of shade position, an application may choose to call the *GET /shades/{id}* API with the *refresh* query string set to *true*. This forces the Hub to query that shade for its current position and save it. Similarly, the battery level is transient and is only updated automatically by the Hub once a week. If an application wants to synchronize the battery level state sooner, then the application may choose to call the *GET /shades/{id}* API with the *updateBatteryLevel* query string set to *true*.

Document Conventions

Each individual resource has a separate heading exposing its APIs. Each API call is shown as *{OPERATION} /api/{resource}[?QUERYSTRING]*.

So, for example, the API call to fetch a specific shade's attributes is shown as:

```
GET /api/shades/{id}
```

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer
Query	**refresh** *optional*	Request position status from shade.	boolean
Query	**requestFirmwareRev** *optional*	Request firmware revision status update from shade.	boolean
Query	**updateBatteryLevel** *optional*	Request battery level status update from shade.	boolean

The shade **id** is a required path parameter. To request a position status update as well, the optional **refresh** boolean query parameter would be added (with a value of *true* since it is a boolean).

Given that the PowerView Hub REST API scheme is *HTTP*, the full URL of the *GET /api/shades/{id}* with *refresh* API call would be:

```
GET http://{HubIPAddress}/api/shades/{id}?refresh=true
```

So, given a Hub IP Address of *10.0.0.7* and a known shade id of *2312*, typing into a browser's address bar:

```
http://10.0.0.7/api/shades/2312?refresh=true
```

and pressing Enter, will cause the *GET /api/shades/{id}* API call to be made.

In general, multiple query string options **may not** be used together in a single API call. So, to update the Hub's current view of a shade's position *and* battery level requires two individual API calls.

Most of the service calls described here can be used with both a Generation 1 and Generation 2 PowerView Hub. Any API calls that are only supported by a Generation 2 Hub are identified.

Version information

Version : 2.0.0

URI scheme

BasePath : /api
Schemes : HTTP

Tags

- Firmware Version
- HomeAutomation
- Rooms
- Scene Collections (Multi-Room Scenes)
- Scenes
- Shades
- User Data

Consumes

- `application/json`

Produces

- `application/json`

Resources

Firmware Version

Get firmware version information

```
GET /api/fwversion
```

Responses

HTTP Code	Description	Schema
200	Firmware version returned.	HubFirmware
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/fwversion
```

Example HTTP response

Response 200


```
{
  "firmware" : {
    "mainProcessor" : {
      "build" : 395,
      "name" : "PV Hub2.0",
      "revision" : 2,
      "subRevision" : 0
    },
    "radio" : {
      "build" : 1307,
      "revision" : 2,
      "subRevision" : 0
    }
  }
}
```


HomeAutomation

Retrieve home automation settings and current state.

```
GET /api/homeautomation
```

Description

```
Version 2 Hub Only
```

- Returns enabled state.
- Returns a configuration number that the hub will increment any time one or more shades fail position verification. Failure includes a final position that differs significantly from its target position and a non-responding shade.
- Position verification occurs for all single shade position set and group position set commands and all scene and scenecollection executions.
- Returns time interval used by the hub to delay requesting positions after the hub issues a PowerView® command to modify positions.
- When a position setting command is received by the hub, if enabled, the hub starts a

countdown timer loaded with the value of the timeoutSec option. If the timer is still running due to a previous position setting command then the timer will be restarted.

- When the countdown timer expires the hub will issue a position request to the shades. If a single shade is affected then the request will be addressed specifically to this shade. If more than one shade is affected then the request will be broadcast to all shades in the network. A nominal four seconds is required to complete the transaction. Multiple attempts to retrieve position are made, if necessary.
- Returns active as true while the countdown timer is running or the hub is actively requesting position from the shades.

Responses

HTTP Code	Description	Schema
200	Homeautomation status and configuration is returned.	HomeautoRespObj
400	Bad request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/homeautomation
```

Example HTTP response

Response 200

```
{
  "homeautomation" : {
    "enabled" : true,
    "timeoutSec" : 30,
    "configNum" : 10,
    "active" : false
  }
}
```

Configure home automation position verification options.

```
PUT /api/homeautomation
```

Description

```
Version 2 Hub Only
```

- Three parameters may be set: enabled, timeoutSec and configNum. Each parameter is optional. Any parameter not included in the body of the message is left unchanged in the hub.
- Set enabled to true to configure the hub to verify final shade positions.
- Set configNum. This number defaults to zero from the factory. It is incremented whenever a shade fails to verify to the correct position after a move. The value is saved in non-volatile memory.
- Set timeoutSec in seconds. Valid range is between 15 and 300 seconds. This value should be set to greater than the longest time it takes for any shade in the installation to move from a fully closed to a fully opened position. Additional time should be allowed to account for a nearly discharged battery.

Body parameter

Home automation configuration options.

Name : body
Flags : required
Type : HomeautoReqObj

Responses

HTTP Code	Description	Schema
200	Homeautomation options successfully updated.	HomeautoRespObj
400	Bad request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/homeautomation
```

Request body

```
{
  "homeautomation" : {
    "enabled" : true,
    "timeoutSec" : 30,
    "configNum" : 10
  }
}
```

Example HTTP response

Response 200

```
{
  "homeautomation" : {
    "enabled" : true,
    "timeoutSec" : 30,
    "configNum" : 10,
    "active" : false
  }
}
```

Retrieve position information following last move.

```
GET /api/homeautomation/report
```

Description

```
Version 2 Hub Only
```

- Returns list of position data for shades involved in last move.
- For each shade, the last known position before the move, the target position and the reported position are returned.
- If a shade failed to reach the target position, the state is returned as false. If the move was successful, the state is returned true.
- If the shade failed to respond to repeated position requests, the timedOut parameter is set to true.
- The returned list is valid when the homeautomation GET returns active as false.

Responses

HTTP Code	Description	Schema
200	A report of the last move is returned.	Homeautomation ReportObj
400	Bad request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/homeautomation/report
```

Example HTTP response

Response 200

```
{
  "report" : [ {
    "shadeId" : "<<_shadeid>>",
    "state" : true,
    "timedOut" : false,
    "type" : 8,
    "lastPos" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    },
    "targetPos" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    },
    "reportedPos" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    }
  } ]
}
```


Rooms

Get all rooms

```
GET /api/rooms
```

Description

- Gets a list of all room ids and the corresponding room data.
- The room data is returned in the same order as the room ids.
- If no rooms exist, then empty arrays for room ids and room data are returned.

Responses

HTTP Code	Description	Schema
200	Rooms returned.	RoomsResponse

HTTP Code	Description	Schema
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/rooms
```

Example HTTP response

Response 200

```
{
  "roomData" : [ {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1,
    "type" : 0
  } ],
  "roomIds" : [ 7 ]
}
```

Get a room

```
GET /api/rooms/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Responses

HTTP Code	Description	Schema
200	Room returned.	RoomObject
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/rooms/7
```

Example HTTP response

Response 200

```
{
  "room" : {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1,
    "type" : 0
  }
}
```

Scene Collections (Multi-Room Scenes)

Get all scene collections

```
GET /api/scenecollections
```

Description

- Gets a list of all scene collection ids and the corresponding scene collection data.
- The scene collection data is returned in the same order as the scene collection ids.
- If no scene collections exist, then empty arrays for scene collection ids and scene collection data are returned.

Responses

HTTP Code	Description	Schema
200	Scene collections returned.	SceneCollectionsResponse
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenecollections
```

Example HTTP response

Response 200


```
{
  "sceneCollectionData" : [ {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1
  } ],
  "sceneCollectionIds" : [ 7 ]
}
```


Get a scene collection

```
GET /api/scenecollections/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Responses

HTTP Code	Description	Schema
200	Scene collection returned.	SceneCollectionObject
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenecollections/7
```

Example HTTP response

Response 200

```
{
  "sceneCollection" : {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1
  }
}
```

Activate a scene collection

```
GET /api/scenecollections?sceneCollectionId={id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Responses

HTTP Code	Description	Schema
200	Activated scene ids returned.	ActivatedScenes
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenecollections?sceneCollectionId=7
```

Example HTTP response

Response 200

```
{
  "sceneIds" : [ 7 ]
}
```

Scenes

Get all scenes

```
GET /api/scenes
```

Description

- Gets a list of all scene ids and the corresponding scene data.
- The scene data is returned in the same order as the scene ids.
- If no scenes exist, then empty arrays for scene ids and scene data are returned.

Responses

HTTP Code	Description	Schema
200	Scenes returned.	ScenesResponse
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenes
```

Example HTTP response

Response 200


```
{
  "sceneData" : [ {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "networkNumber" : 12,
    "order" : 1,
    "roomId" : 1385
  } ],
  "sceneIds" : [ 7 ]
}
```

Get a scene

```
GET /api/scenes/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Responses

HTTP Code	Description	Schema
200	Scene returned.	SceneObject
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenes/7
```

Example HTTP response

Response 200

```
{
  "scene" : {
    "colorId" : 18,
    "iconId" : 24,
    "id" : 7,
    "name" : "TmFtZQ==",
    "networkNumber" : 12,
    "order" : 1,
    "roomId" : 1385
  }
}
```


Activate a scene

```
GET /api/scenes?sceneId={id}
```

Description

- Activates a scene.
- Moves all scene contained shades to their pre-programmed position.
- Changes all scene contained repeater colors to their pre-programmed values.
- Returns the ids of all affected shades.

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Responses

HTTP Code	Description	Schema
200	Scene activated.	ActivatedShades
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content

HTTP Code	Description	Schema
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/scenes?sceneId=7
```

Example HTTP response

Response 200

```
{
  "shadeIds" : [ 7 ]
}
```

Shades

Get all shades

```
GET /api/shades
```

Description

- Gets a list of all shade ids and the corresponding shade data.
- The shade data is returned in the same order as the shade ids.
- The results may be filtered by the group and room id query parameters.
- If no shades exist, then empty arrays for shade ids and shade data are returned.

Parameters

Type	Name	Description	Schema
Query	**groupId** *optional*	Filter results to only include those shades in the specified group.	integer
Query	**roomId** *optional*	Filter results to only include those shades in the specified room.	integer

Responses

HTTP Code	Description	Schema
200	Shades returned.	ShadesResponse
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/shades
```

Example HTTP response

Response 200

```
{
  "shadeData" : [ {
    "batteryStatus" : 3,
    "batteryStrength" : 78,
    "firmware" : {
      "build" : 564,
      "revision" : 2,
      "subRevision" : 0,
      "index" : 25
    },
    "groupId" : 37952,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1,
    "positions" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    },
    "roomId" : 1385,
    "secondaryName" : "VG9wIFJhaWwgU2hhZGUgTmFtZQ==",
    "type" : 8
  } ],
  "shadeIds" : [ 7 ]
}
```


Update shade positions for a group

```
PUT /api/shades
```

Description

- Updates the position for each shade in a group.
- To perform a motion operation on a group (down, up, jog, ...), then send a body that only has a motion operation in it.
- Only one of positions or motion may be updated.

Parameters

Type	Name	Description	Schema
Query	**groupId** *required*	Unique id of the associated group.	integer

Body parameter

Name : body
Flags : required

Name	Description	Schema
shade *required*	**Example** : ShadeUpdate	ShadeUpdate

Responses

HTTP Code	Description	Schema
200	Shades moved.	ActivatedShades
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/shades?groupId=37952
```

Request body

```
{
  "shade" : {
    "motion" : "jog",
    "positions" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    }
  }
}
```

Example HTTP response

Response 200

```
{
  "shadeIds" : [ 7 ]
}
```

Get a shade

```
GET /api/shades/{id}
```

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer
Query	**refresh** *optional*	Request position status update from shade.	boolean
Query	**requestFirmwareRev** *optional*	Request firmware revision status update from shade.	boolean
Query	**updateBatteryLevel** *optional*	Request battery level status update from shade.	boolean

Responses

HTTP Code	Description	Schema
200	Shade returned.	ShadeRequestObject
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/shades/7
```

Example HTTP response

Response 200


```
{
  "shade" : {
    "batteryStatus" : 3,
    "batteryStrength" : 78,
    "firmware" : {
      "build" : 564,
      "revision" : 2,
      "subRevision" : 0,
      "index" : 25
    },
    "groupId" : 37952,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1,
    "positions" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    },
    "roomId" : 1385,
    "secondaryName" : "VG9wIFJhaWwgU2hhZGUgTmFtZQ==",
    "type" : 8,
    "timedOut" : true
  }
}
```


Update a shade

```
PUT /api/shades/{id}
```

Description

- Updates an already-existing shade.
- The object returned from the server contains the full representation of the updated shade (all fields, not just the updated ones)

- To perform a motion operation on a shade (down, up, jog, ...), then send a body that only has a motion operation in it.
- Only one of positions or motion may be updated.

Parameters

Type	Name	Description	Schema
Path	**id** *required*	Unique id of the resource.	integer

Body parameter

Name : body
Flags : required

Name	Description	Schema
shade *required*	**Example** : ShadeUpdate	ShadeUpdate

Responses

HTTP Code	Description	Schema
200	Shade updated.	ShadeObject
400	Bad client request.	No Content
404	Resource not found.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/shades/7
```

Request body

```
{
  "shade" : {
    "motion" : "jog",
    "positions" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    }
  }
}
```


Example HTTP response

Response 200


```
{
  "shade" : {
    "batteryStatus" : 3,
    "batteryStrength" : 78,
    "firmware" : {
      "build" : 564,
      "revision" : 2,
      "subRevision" : 0,
      "index" : 25
    },
    "groupId" : 37952,
    "id" : 7,
    "name" : "TmFtZQ==",
    "order" : 1,
    "positions" : {
      "posKind1" : 1,
      "posKind2" : 2,
      "position1" : 20359,
      "position2" : 14060
    },
    "roomId" : 1385,
    "secondaryName" : "VG9wIFJhaWwgU2hhZGUgTmFtZQ==",
    "type" : 8
  }
}
```


User Data

Get user data

```
GET /api/userdata
```

Parameters

Type	Name	Description	Schema
Query	**includeCounts** *optional*	Should counts of database objects, rooms, shades, scenes, etc, be included in the returned user data (This is an expensive operation that should not normally be used).	boolean

Responses

HTTP Code	Description	Schema
200	User data returned.	UserDataObject
400	Bad client request.	No Content
423	Hub is temporarily busy for maintenance (Always returns UserData).	No Content
500	Internal server error.	No Content

Example HTTP request

Request path

```
/api/userdata
```

Example HTTP response

Response 200

```
{
  "userData" : {
    "autoBackup" : true,
    "color" : {
      "blue" : 155,
      "brightness" : 50,
      "green" : 107,
      "red" : 12
    },
    "dns" : "192.168.1.254",
    "editingEnabled" : true,
```

```
    "enableScheduledEvents" : true,
    "firmware" : {
      "firmware" : {
        "mainProcessor" : {
          "build" : 395,
          "name" : "PV Hub2.0",
          "revision" : 2,
          "subRevision" : 0
        },
        "radio" : {
          "build" : 1307,
          "revision" : 2,
          "subRevision" : 0
        }
      }
    },
    "gateway" : "192.168.1.1",
    "hubName" : "SHViYnk=",
    "ip" : "192.168.1.100",
    "localTimeDataSet" : true,
    "macAddress" : "00:26:74:af:fd:ae",
    "mask" : "255.255.255.0",
    "remoteConnectEnabled" : true,
    "rfID" : "0x695D",
    "rfIDInt" : 26973,
    "rfStatus" : 1,
    "serialNumber" : "927FD402C11CE424",
    "setupComplete" : true,
    "ssid" : "cisco789",
    "staticIp" : false,
    "times" : {
      "currentOffset" : -21600,
      "latitude" : 39.92394425904774,
      "localSunriseInMinutes" : 379,
      "localSunsetInMinutes" : 1187,
      "longitude" : -105.1006371575785,
      "timezone" : "America/Denver"
    }
  }
}
```


Definitions

ActivatedScenes

Name	Description	Schema
sceneIds *required*	List of activated scenes. **Example** : `[ "UniqueId" ]`	< UniqueId > array

ActivatedShades

Name	Description	Schema
shadeIds *required*	List of affected shades. **Example** : `[ "UniqueId" ]`	< UniqueId > array

AutoBackup

When true, backups will be sent periodically to the Hunter Douglas server. Otherwise, no automatic backups will occur.

Type : boolean

BatteryStatus

0 = No Status Available, 1 = Low, 2 = Medium, 3 = High, 4 = Plugged In

Type : enum (0, 1, 2, 3, 4)

BatteryStrength

The current strength of the battery.

Type : integer

BlinkEnabled

Enable repeater blinking.

Type : boolean

Color

Specifies the color of the LEDs.

Name	Description	Schema
blue *required*	The intensity of the blue portion of the LED. **Minimum value** : `0` **Maximum value** : `255` **Example** : `155`	integer
brightness *required*	The brightness of the LED. Range of 0 to 100%. **Minimum value** : `0` **Maximum value** : `100` **Example** : `50`	integer
green *required*	The intensity of the green portion of the LED. **Minimum value** : `0` **Maximum value** : `255` **Example** : `107`	integer
red *required*	The intensity of the red portion of the LED. **Minimum value** : `0` **Maximum value** : `255` **Example** : `12`	integer

ColorId

Id of the resource display color.

Type : integer (int32)

DeviceType

Device type (0 = shade, 1 = repeater). If this field is missing, the scene member is assumed to be a shade.

Type : enum (0, 1)

Dns

The DNS server IP address used by the hub.

Type : string

EditingEnabled

Indicates whether or not the UI should "lock" editing (that is, only allow Scene activation and Shade movement within Rooms).

Type : boolean

EnableScheduledEvents

When true, scheduled events should work as expected; when false, scheduled events should not execute. Note that setting this field to true/false should not impact the "enabled" setting on individual ScheduledEvents. It is a top-level override.

Type : boolean

Firmware

Name	Description	Schema
build *required*	Patch firmware version number. **Minimum value** : `0` **Maximum value** : `65535` **Example** : `564`	integer
revision *required*	Major firmware version number. **Minimum value** : `0` **Maximum value** : `255` **Example** : `2`	integer
subRevision *required*	Minor firmware version number. **Minimum value** : `0` **Maximum value** : `255` **Example** : `0`	integer

Gateway

The gateway IP address used by the hub.

Type : string

GroupId

Unique id of the associated group.

Type : integer

HomeAutomationActive

Returns true when the hub has sent a position set to a shade but the followup position request is not yet complete.

Type : boolean

HomeAutomationConfigNum

Unique identifier that is incremented whenever a shade fails to move to a position correctly.

Type : integer

HomeAutomationEnabled

Position verification is enabled

Type : boolean

HomeAutomationRequestTimeout

Interval between a position set message to shade and a followup position request.

Type : integer

HomeautoReq

Name	Description	Schema
configNum *optional*	**Example** : HomeAutomationConfigNum	HomeAutomationConfigNum
enabled *optional*	**Example** : HomeAutomationEnabled	HomeAutomationEnabled
timeoutSec *optional*	**Example** : HomeAutomationRequestTimeout	HomeAutomationRequestTimeout

HomeautoReqObj

Name	Description	Schema
homeautomation *required*	**Example** : HomeautoReq	HomeautoReq

HomeautoResp

Name	Description	Schema
active *required*	**Example** : HomeAutomationActive	HomeAutomationActive
configNum *required*	**Example** : HomeAutomationConfigNum	HomeAutomationConfigNum
enabled *required*	**Example** : HomeAutomationEnabled	HomeAutomationEnabled
timeoutSec *required*	**Example** : HomeAutomationRequestTimeout	HomeAutomationRequestTimeout

HomeautoRespObj

Name	Description	Schema
homeautomation *required*	**Example** : HomeautoResp	HomeautoResp

HomeautomationReport

Name	Description	Schema
lastPos *required*	**Example** : ShadePosition	ShadePosition
reportedPos *required*	**Example** : ShadePosition	ShadePosition
shadeId *required*	**Example** : [_shadeid]	ShadeId
state *required*	Returns true if the shade is at target position. **Example** : `true`	boolean
targetPos *required*	**Example** : ShadePosition	ShadePosition
timedOut *required*	Returns true if the shade failed to respond to a position request. **Example** : `false`	boolean

Name	Description	Schema
type *required*	**Example** : ShadeType	ShadeType

HomeautomationReportObj

Name	Description	Schema
report *required*	**Example** : `[ "HomeautomationReport" ]`	< HomeautomationReport > array

HubFirmware

There are multiple processors in the hub (the primary processor and the Nordic chip for RF processing); the "mainProcessor" sub-key is intended to indicate that the firmware information being returned is for the main processor on the hub, not the Nordic.

Name	Description	Schema
firmware *required*	**Example** : `{ "mainProcessor" : { "build" : 395, "name" : "PV Hub2.0", "revision" : 2, "subRevision" : 0 }, "radio" : { "build" : 1307, "revision" : 2, "subRevision" : 0 } }`	firmware

firmware

Name	Description	Schema
mainProcessor *required*	**Example** : `"object"`	object
radio *required*	**Example** : Firmware	Firmware

HubName

Base64-encoded hub name.

Type : string

IconId

Id of the resource display icon.

Type : integer (int32)

IpAddress

Type : string

Latitude

Latitude.

Type : number (float)

LocalSunriseTimeInMinutes

Number of minutes into the day that sunrise occurs (based on lat/long).

Type : integer

LocalSunsetTimeInMinutes

Number of minutes into the day that sunset occurs (based on lat/long).

Type : integer

Longitude

Longitude.

Type : number (float)

Mask

The network mask used by the hub.

Type : string

Name

Base64 encoded name.

Type : string (byte)

NetworkNumber

Unique scene index value loaded into a shade.

Type : integer

Offset

Number of seconds before or after UTC.

Type : integer (int32)

Order

Display order of the resource.

Type : integer (int32)

PositionKind1

The type of position - 0 = None, 1 = Primary Rail, 2 = Secondary Rail, 3 = Vane Tilt, 4 = Error.

Type : enum (0, 1, 2, 3, 4)

PositionKind2

The type of position - 0 = None, 1 = Primary Rail, 2 = Secondary Rail, 3 = Vane Tilt, 4 = Error.

Type : enum (0, 1, 2, 3, 4)

PositionValue1

The value, with 0 = closed and 65535 = open.

Type : integer

PositionValue2

The value, with 0 = closed and 65535 = open.

Type : integer

RepeaterId

The repeater id associated to this member.

Type : integer

RfID

The ID of the RF network on which the hub talks to shades, represented in hexadecimal. 0x1111 or 0xFFFF both mean "no network set."

Type : string

RfIDInt

The integer value of the RF network ID on which the hub talks to shades. Values of 4369 or 65535 indicate that no network has been set.

Type : integer (int32)

RfStatus

0 means the hub is not busy; 1 means the hub is busy (discovering shades, joining a network, etc).

Type : enum (0, 1)

Room

Name	Description	Schema
colorId *required*	**Example** : ColorId	ColorId
iconId *required*	**Example** : IconId	IconId
id *required*	**Example** : UniqueId	UniqueId
name *required*	**Example** : Name	Name
order *required*	**Example** : Order	Order
type *required*	**Example** : RoomType	RoomType

RoomId

Unique id of the associated room.

Type : integer

RoomObject

Name	Description	Schema
room *required*	**Example** : Room	Room

RoomType

Room type (0 Regular Room, 1 Repeater Room, 2 Default Room).

Type : enum (0, 1, 2)

RoomsResponse

Name	Description	Schema
roomData *required*	Room data for included rooms. **Example** : `[ "Room" ]`	< Room > array
roomIds *required*	Unique ids of all rooms. **Example** : `[ "UniqueId" ]`	< UniqueId > array

Scene

Name	Description	Schema
colorId *required*	**Example** : ColorId	ColorId
iconId *required*	**Example** : IconId	IconId
id *required*	**Example** : UniqueId	UniqueId
name *required*	**Example** : Name	Name
networkNumber *required*	**Example** : NetworkNumber	NetworkNumber
order *required*	**Example** : Order	Order

Name	Description	Schema
roomId *required*	**Example** : RoomId	RoomId

SceneCollection

Name	Description	Schema
colorId *required*	**Example** : ColorId	ColorId
iconId *required*	**Example** : IconId	IconId
id *required*	**Example** : UniqueId	UniqueId
name *required*	**Example** : Name	Name
order *required*	**Example** : Order	Order

SceneCollectionId

The id of the Scene Collection to which this member belongs.

Type : integer

SceneCollectionObject

Name	Description	Schema
sceneCollection *required*	**Example** : SceneCollection	SceneCollection

SceneCollectionsResponse

Name	Description	Schema
sceneCollectionData *required*	Scene collection data for included scene collections. **Example** : `[ "SceneCollection" ]`	< SceneCollection > array
sceneCollectionIds *required*	Unique ids of all scene collections. **Example** : `[ "UniqueId" ]`	< UniqueId > array

SceneObject

Name	Description	Schema
scene *required*	**Example** : Scene	Scene

ScenesResponse

Name	Description	Schema
sceneData *required*	Scene data for included scenes. **Example** : `[ "Scene" ]`	< Scene > array
sceneIds *required*	Unique ids of all scenes. **Example** : `[ "UniqueId" ]`	< UniqueId > array

SerialNumber

The unique id / serial number of the hub or device.

Type : string

SetupComplete

Indicates whether the initial setup of a hub has been completed.

Type : boolean

Shade

Name	Description	Schema
batteryStatus *required*	**Example** : BatteryStatus	BatteryStatus

Name	Description	Schema
batteryStrength *required*	**Example** : BatteryStrength	BatteryStrength
firmware *optional*	**Example** : ShadeFirmware	ShadeFirmware
groupId *optional*	**Example** : GroupId	GroupId
id *required*	**Example** : UniqueId	UniqueId
name *optional*	**Example** : Name	Name
order *optional*	**Example** : Order	Order
positions *optional*	**Example** : ShadePosition	ShadePosition
roomId *optional*	**Example** : RoomId	RoomId
secondaryName *optional*	**Example** : ShadeSecondaryName	ShadeSecondaryName
type *required*	**Example** : ShadeType	ShadeType

ShadeData

Name	Description	Schema
id *required*	**Example** : UniqueId	UniqueId
type *required*	**Example** : ShadeType	ShadeType

ShadeFirmware

Polymorphism : Composition

Name	Description	Schema
build *required*	Patch firmware version number. **Minimum value** : `0` **Maximum value** : `65535` **Example** : `564`	integer
index *required*	The index number. **Example** : `25`	integer
revision *required*	Major firmware version number. **Minimum value** : `0` **Maximum value** : `255` **Example** : `2`	integer
subRevision *required*	Minor firmware version number. **Minimum value** : `0` **Maximum value** : `255` **Example** : `0`	integer

ShadeObject

Name	Description	Schema
shade *required*	**Example** : Shade	Shade

ShadePosition

Specifies the position of the shade. Top-down shades are in the same coordinate space as bottom-up shades. Shade position values for top-down shades would be reversed for bottom-up shades. For example, since 65535 is the open value for a bottom-up shade, it is the closed value for a top-down shade. The top-down/bottom-up shade is different in that instead of the top and bottom rail operating in one coordinate space like the top-down and the bottom-up, it operates in two where the top (middle) rail closed value is 0 and the bottom (primary) rail closed position is also 0 and fully open for both is 65535.

Name	Description	Schema
posKind1 *required*	**Example** : PositionKind1	PositionKind1

Name	Description	Schema
posKind2 *optional*	**Example** : PositionKind2	PositionKind2
position1 *required*	**Example** : PositionValue1	PositionValue1
position2 *optional*	**Example** : PositionValue2	PositionValue2

ShadeRequest

Polymorphism : Composition

Name	Description	Schema
batteryStatus *required*	**Example** : BatteryStatus	BatteryStatus
batteryStrength *required*	**Example** : BatteryStrength	BatteryStrength
firmware *optional*	**Example** : ShadeFirmware	ShadeFirmware
groupId *optional*	**Example** : GroupId	GroupId
id *required*	**Example** : UniqueId	UniqueId
name *optional*	**Example** : Name	Name
order *optional*	**Example** : Order	Order
positions *optional*	**Example** : ShadePosition	ShadePosition
roomId *optional*	**Example** : RoomId	RoomId

Name	Description	Schema
secondaryName *optional*	**Example** : ShadeSecondaryName	ShadeSecondaryName
timedOut *required*	Did the shade not respond to the request. **Example** : `true`	boolean
type *required*	**Example** : ShadeType	ShadeType

ShadeRequestObject

Name	Description	Schema
shade *required*	**Example** : ShadeRequest	ShadeRequest

ShadeSecondaryName

The secondary name of the shade base64 encoded. Used by the Apple Home application as the secondary service name to control shades with blackout blinds or a top rail movement.

Type : string (byte)

ShadeType

The shade type.

Type : integer

ShadeUpdate

Name	Description	Schema
motion *optional*	The motion operation to perform on a shade or group. **Example** : `"jog"`	enum (down, heart, jog, leftTilt, rightTilt, stop, up)
positions *optional*	**Example** : ShadePosition	ShadePosition

ShadesResponse

Name	Description	Schema
shadeData *required*	Shade data for included shades. **Example** : `[ "Shade" ]`	< Shade > array
shadeIds *required*	Unique ids of all shades. **Example** : `[ "UniqueId" ]`	< UniqueId > array

StaticIp

True, if a static IP is assigned to the hub. False, if DHCP was used for IP address assignment

Type : boolean

SunriseToday

UTC time of sunrise for the current lat/long, today.

Type : string

SunsetToday

UTC time of sunset for the current lat/long, today.

Type : string

Times

Name	Description	Schema
currentOffset *required*	**Example** : Offset	Offset
latitude *optional*	**Example** : Latitude	Latitude
localSunriseInMinutes *optional*	**Example** : LocalSunriseTimeInMinutes	LocalSunriseTimeInMinutes
localSunsetInMinutes *optional*	**Example** : LocalSunsetTimeInMinutes	LocalSunsetTimeInMinutes

Name	Description	Schema
longitude *optional*	**Example** : Longitude	Longitude
timezone *required*	**Example** : Timezone	Timezone

Timezone

Timezone name.

Type : string

UTC

Current UTC time on the hub.

Type : string

UniqueId

Unique resource identifier.

Type : integer

UserData

User data associated with this hub.

Name	Description	Schema
autoBackup *required*	**Example** : AutoBackup	AutoBackup
color *required*	Specifies the color of the repeater LEDs. **Example** : Color	Color
dns *required*	**Example** : Dns	Dns
editingEnabled *required*	**Example** : EditingEnabled	EditingEnabled

Name	Description	Schema
enableScheduledEvents *required*	**Example** : EnableScheduledEvents	EnableScheduledEvents
firmware *required*	**Example** : HubFirmware	HubFirmware
gateway *required*	**Example** : Gateway	Gateway
hubName *required*	**Example** : HubName	HubName
ip *required*	**Example** : IpAddress	IpAddress
localTimeDataSet *required*	Whether or not time has been set by the app or remote connect. **Example** : `true`	boolean
macAddress *required*	The MAC address of the hub. **Example** : `"00:26:74:af:fd:ae"`	string
mask *required*	**Example** : Mask	Mask
remoteConnectEnabled *required*	Whether or not the hub is currently registered with Remote Connect. **Example** : `true`	boolean
rfID *required*	**Example** : RfID	RfID
rfIDInt *required*	**Example** : RfIDInt	RfIDInt
rfStatus *required*	**Example** : RfStatus	RfStatus
serialNumber *required*	**Example** : SerialNumber	SerialNumber

Name	Description	Schema
setupComplete *required*	**Example** : SetupComplete	SetupComplete
ssid *required*	The service set identifier - the unique identifier for the wireless network **Example** : `"cisco789"`	string
staticIp *required*	**Example** : StaticIp	StaticIp
times *required*	**Example** : Times	Times

UserDataObject

Name	Description	Schema
userData *required*	**Example** : UserData	UserData