

HDAnywhere & Grenton

This tutorial presents the integration of HDAnywhere with Grenton using RESTful API and controlling the device using myGrenton application.

The following manual for integration with HDAnywhere device is based on the information provided on: https://cloud.hdanywhere.com/docs/api/hda_api.pdf

The presented configuration was performed on:

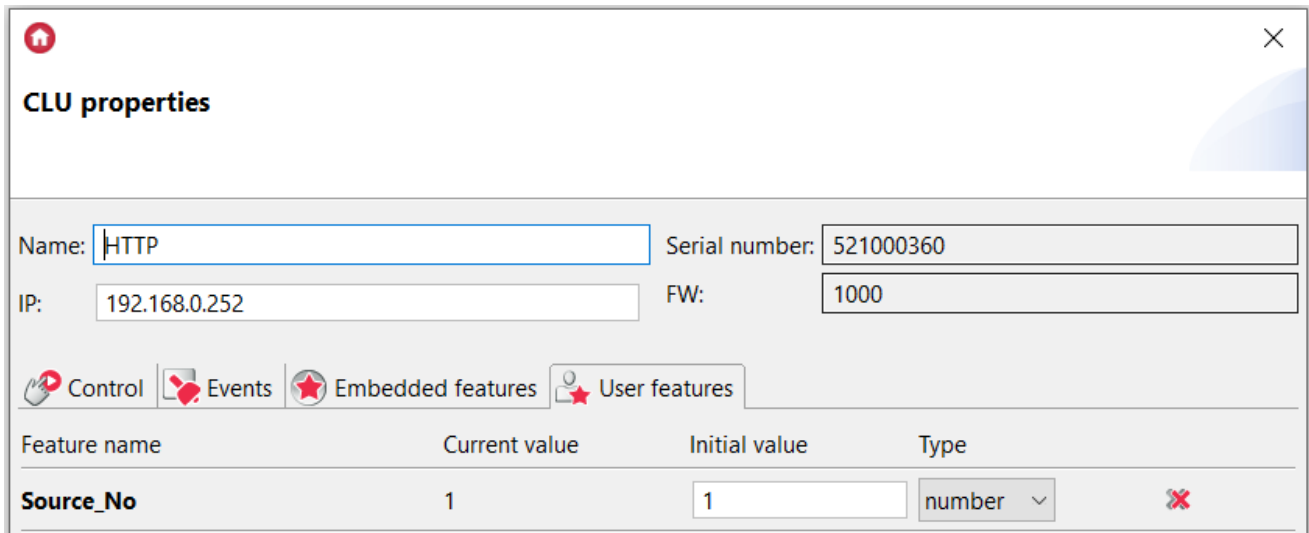
- Object Manager v.1.5.1 (build 214101),
- Gate HTTP 2.0 (FW v1.1.0 (build 2034C)) called `HTTP`,
- HDAnywhere MHUB U (4x3+1) 40.

To integrate Grenton system with HDAnywhere device, please follow the steps described below.

1. Changing `Source` and `Output` of displaying

Preparing

- Create the `User features` on `HTTP` - `Source_No`:



The screenshot shows the 'CLU properties' dialog box for the 'HTTP' feature. The dialog has a title bar with a home icon and a close button. Below the title, there are input fields for 'Name' (HTTP), 'Serial number' (521000360), 'IP' (192.168.0.252), and 'FW' (1000). Below these fields are four tabs: 'Control', 'Events', 'Embedded features', and 'User features'. The 'User features' tab is selected, showing a table with the following columns: 'Feature name', 'Current value', 'Initial value', and 'Type'. The table contains one row for 'Source_No' with a current value of '1', an initial value of '1', and a type of 'number'. There is a red 'X' icon next to the 'Source_No' row.

Feature name	Current value	Initial value	Type
Source_No	1	1	number

- Create the `Source_No_Choice` script on `HTTP`:

```
if (HTTP->Source_No==1) then
HTTP->Source_No = 2

elseif (HTTP->Source_No==2) then
HTTP->Source_No = 3

elseif (HTTP->Source_No==3) then
HTTP->Source_No = 4

elseif (HTTP->Source_No==4) then
HTTP->Source_No = 1

end
```

- Create the `User features` on `HTTP` - `Output_No`:

Output_No

a

a

string



- Create the `Output_No_Choice` script on `HTTP`:

```
if (HTTP->Output_No=="a") then
HTTP->Output_No = "b"

elseif (HTTP->Output_No=="b") then
HTTP->Output_No = "c"

elseif (HTTP->Output_No=="c") then
HTTP->Output_No = "d"

elseif (HTTP->Output_No=="d") then
HTTP->Output_No = "a"

end
```

- Create the `HttpRequest` virtual object on `HTTP - Source_Output_Choice_Req`:

The screenshot shows the 'Object properties' dialog box for a virtual object named 'Source_Output_Choice_Req' of type 'HttpRequest'. The dialog has a 'Name' field containing 'Source_Output_Choice_Req' and a 'Type' dropdown set to 'HttpRequest'. Below this is an 'Id' field with the value 'CLU521000360->HTT2105'. There are three tabs: 'Control' (selected), 'Events', and 'Embedded features'. A table lists various properties with their current and initial values, units, and ranges. At the bottom, there is an 'Auto refresh' checkbox (checked) and a 'Refresh' button. 'OK' and 'Cancel' buttons are at the bottom right.

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	192.168.0.247	string	
Path	/api/control/switch/a/1	/api/control/switch/a/1	string	
QueryStringParams	-	\z	string	
Method	GET	GET	string	
Timeout	5	5	s	[1-255]
RequestType	1	Text	-	0,1,2,3,4,5
ResponseType	2	JSON	-	0,1,2,3,4,5
RequestHeaders	-	\z	string	
RequestBody	-	\z	string	
ResponseBody	-	\z	string	
StatusCode	0		-	

Where:

- `Host` : 192.168.0.247(IP address of your device)
- `Path` : /api/control/switch/a/1
- `QueryStringParams` : \z

- Create the `SetPathSourceOutput_Req` script which allow you to set the path depending on your source/output choice:

```
path_var="/api/control/switch/" .. "" .. getVar("Output_No").. "/" .. HTTP->Source_No
HTTP->Source_Output_Choice_Req->SetPath(path_var)
HTTP->Source_Output_Choice_Req->SendRequest()
```

- After uploading the configuration and run the above script, the `StatusCode` of `Source_Output_Choice_Req` object should indicate `200`.



Object properties

Name: Type:

Id:

Control Events Embedded features

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	<input type="text" value="192.168.0.247"/>	string	
Path	/api/control/switch/a/1	<input type="text" value="/api/control/switch/a/1"/>	string	
QueryStringParams	-	<input type="text" value="\z"/>	string	
Method	GET	<input type="text" value="GET"/>	string	
Timeout	5	<input type="text" value="5"/>	s	[1-255]
RequestType	1	<input type="text" value="Text"/>	-	0,1,2,3,4,5
ResponseType	2	<input type="text" value="JSON"/>	-	0,1,2,3,4,5
RequestHeaders	-	<input type="text" value="\z"/>	string	
RequestBody	-	<input type="text" value="\z"/>	string	
ResponseBody	-	<input type="text" value="\z"/>	string	
StatusCode	200		-	

Auto refresh

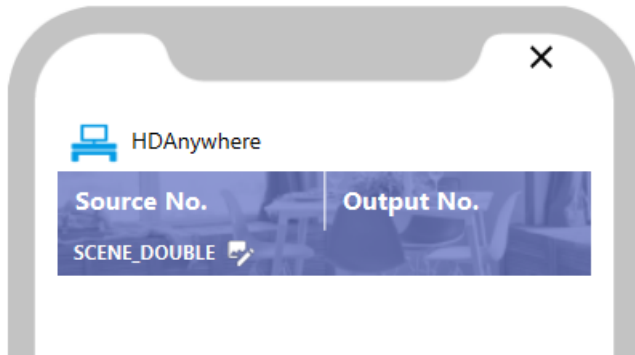
Refresh

OK

Cancel

Control via myGrenton

- Create a new myGrenton interface.
- Add `SCENE_DOUBLE` widget:

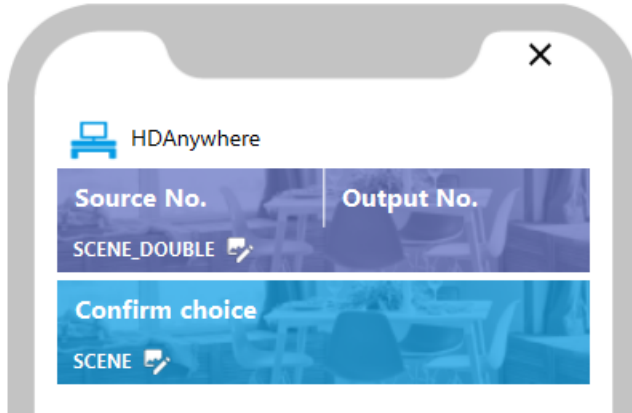


Properties

Name	Value
Type	SCENE_DOUBLE
Background image	dining_room (indigo)
▼ Button 1	
Label	Source No.
Action click	HTTP->Source_No_Choice()
▼ Button 2	
Label	Output No.
Action click	HTTP->Output_No_Choice()

Close

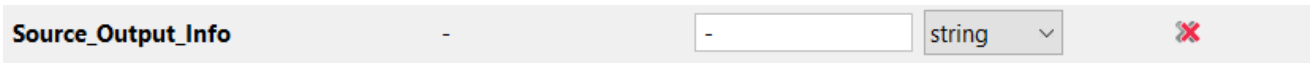
- Add **SCENE** widget:



Properties	
Name	Value
Type	SCENE
Background image*	dining_room (blue)
▼ Monostable button*	
Label*	Confirm choice
Action click*	HTTP->SetPathSourceOutput_Req()

In order to watch number of source and output which is choosing, create a widget to displaying current values of these features.

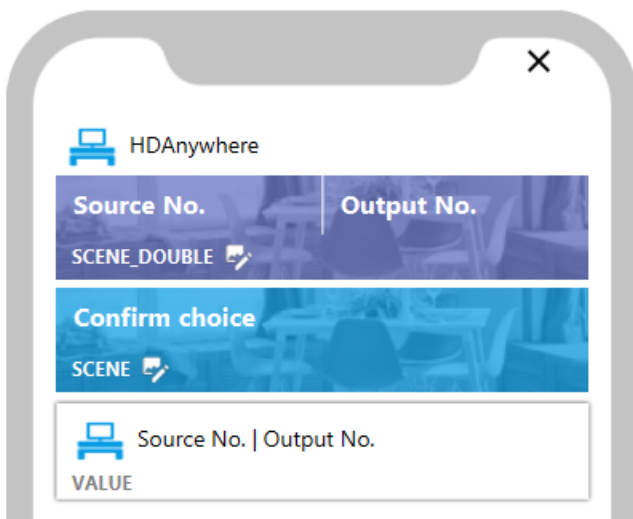
- Create the **User features** on **HTTP - Source_Output_Info**:



- Create the **Source_Output_No_Text** script which merges two variables to one:

```
HTTP->Source_Output_Info=HTTP->Source_No .. " | " .. HTTP->Output_No
```

- Go back to myGrenton interface and add the **VALUE** widget:



Name	Value
Type	VALUE
▼ Value	
Label	Source No. Output No.
Icon	tv
Unit	UNKNOWN
Min	0.0
Max	1.0
State	HTTP->Source_Output_Info

To refresh the value of `Source_Output_Info` feature each time with changing the choice, go back to `Source_No_Choice` and `Output_No_Choice` scripts and modify them by adding calling the `Source_Output_No_Text` at the end:

```
--Source_No_Choice--

if (HTTP->Source_No==1) then
HTTP->Source_No = 2

elseif (HTTP->Source_No==2) then
HTTP->Source_No = 3

elseif (HTTP->Source_No==3) then
HTTP->Source_No = 4

elseif (HTTP->Source_No==4) then
HTTP->Source_No = 1

end

HTTP->Source_Output_No_Text()
```

```
--Output_No_Choice--

if (HTTP->Output_No=="a") then
HTTP->Output_No = "b"

elseif (HTTP->Output_No=="b") then
HTTP->Output_No = "c"

elseif (HTTP->Output_No=="c") then
HTTP->Output_No = "d"

elseif (HTTP->Output_No=="d") then
HTTP->Output_No = "a"

end

HTTP->Source_Output_No_Text()
```


2. Turning ON/OFF the HDAnywhere device

Preparing

- Create the `HttpRequest` virtual object on `HTTP - Power_On_Req` :

Object properties

Name: Type:

Id:

Control Events Embedded features

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	<input type="text" value="192.168.0.247"/>	string	
Path	/api/power/1	<input type="text" value="/api/power/1"/>	string	
QueryStringParams	-	<input type="text" value="\z"/>	string	
Method	GET	<input type="text" value="GET"/>	string	
Timeout	5	<input type="text" value="5"/>	s	[1-255]
RequestType	2	<input type="text" value="JSON"/>	-	0,1,2,3,4,5
ResponseType	2	<input type="text" value="JSON"/>	-	0,1,2,3,4,5
RequestHeaders	-	<input type="text" value="\z"/>	string	
RequestBody	-	<input type="text" value="\z"/>	string	
ResponseBody	-	<input type="text" value="\z"/>	string	
StatusCode	0		-	

Auto refresh

Where:

- `Host` : 192.168.0.247 (`IP address` of your device)
- `Path` : /api/power/1
- `QueryStringParams` : \z

- Create the `HttpRequest` virtual object on `HTTP` - `Power_Off_Req`;

Object properties
✕

Name: Type:

Id:

Control
 Events
 Embedded features

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	<input type="text" value="http://192.168.0.247"/>	string	
Path	/api/power/0	<input type="text" value="/api/power/0"/>	string	
QueryStringParams	-	<input type="text" value="\z"/>	string	
Method	GET	<input type="text" value="GET"/>	string	
Timeout	5	<input type="text" value="5"/>	s	[1-255]
RequestType	1	<input type="text" value="JSON"/>	-	0,1,2,3,4,5
ResponseType	1	<input type="text" value="JSON"/>	-	0,1,2,3,4,5
RequestHeaders	-	<input type="text" value="\z"/>	string	
RequestBody	-	<input type="text" value="\z"/>	string	
ResponseBody	-	<input type="text" value="\z"/>	string	
StatusCode	0		-	

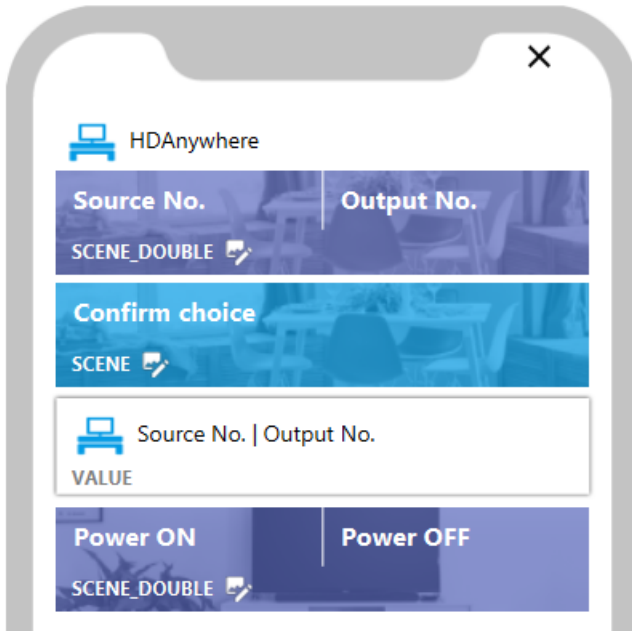
Auto refresh

Where:

- `Host` : 192.168.0.247 (IP address of your device)
- `Path` : /api/power/0
- `QueryStringParams` : \z

Control via myGrenton

- Add `SCENE_DOUBLE` widget:



Properties

Name	Value
Type	SCENE_DOUBLE
Background image	movie_watching (indigo)
▼ Button 1	
Label	Power ON
Action click	HTTP->Power_On_Req->SendRequest()
▼ Button 2	
Label	Power OFF
Action click	HTTP->Power_Off_Req->SendRequest()

Close

3. Sending HEX codes

Preparing

In order to send HEX codes, there is a need to gain these codes before configure this functionality in Object Manager. The HEX codes you can get for example here: <http://files.remotecentral.com/pronto/14-1/index.html>

In this example, the HEX code in charge of changing channel down on Sony TV is used.

- Create the `HttpRequest` virtual object on `HTTP` - `IR_HEX_CH_Down_Req`:

The screenshot shows the 'Object properties' dialog box for a virtual object named 'IR_HEX_Req'. The object is of type 'HttpRequest' and has the ID 'CLU521000360->HTT6315'. The dialog is divided into three tabs: 'Control', 'Events', and 'Embedded features'. The 'Embedded features' tab is active, displaying a table of properties for the 'HttpRequest' object.

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	192.168.0.247	string	
Path	/api/command/irpass/2	/api/command/irpass/2	string	
QueryStringParams	-	\z	string	
Method	POST	POST	string	
Timeout	5	5	s	[1-255]
RequestType	2	JSON	-	0,1,2,3,4,5
ResponseType	2	JSON	-	0,1,2,3,4,5
RequestHeaders	-	\z	string	
RequestBody	-	\z	string	
ResponseBody	-	\z	string	
StatusCode	0		-	

At the bottom of the dialog, there is a checkbox for 'Auto refresh' which is checked, and a 'Refresh' button. The 'OK' and 'Cancel' buttons are also visible at the bottom right.

Where:

- `Host` : 192.168.0.247 (IP address of your device)
- `Path` : /api/command/irpass/2 (2 is the IR port ID, enter your one)
- `QueryStringParams` : \z

- Create a `IR_HEX_CH_Down` script:

```

local body = { irdata = "0000 0067 0000 000d 0061 0018 0030 0018 0018 0018 0030 0018
0018 0018 0030 0018 0018 0018 0018 0018 0030 0018 0018 0018 0018 0018 0018 0018
0403"
}
HTTP->IR_HEX_CH_Down_Req->SetRequestBody(body)
HTTP->IR_HEX_CH_Down_Req->SendRequest()

```

- Send the configuration to `HTTP`.
- After uploading the configuration and run the above script, the `StatusCode` of `Source_Output_Choice_Req` object should indicate `200`.

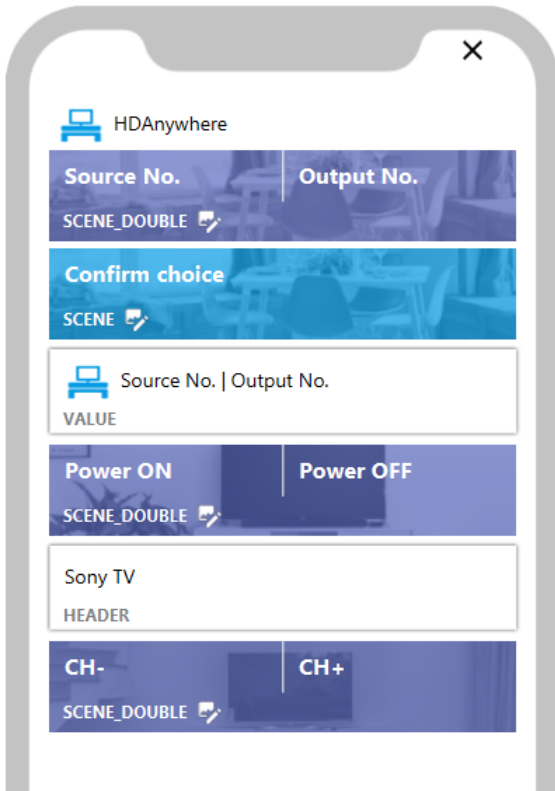
The screenshot shows the 'Object properties' dialog box for an object named `IR_HEX_CH_Down_Req`. The object is of type `HttpRequest` and has the ID `CLU521000360->HTT6315`. The dialog is divided into three tabs: **Control**, **Events**, and **Embedded features**. The **Embedded features** tab is active, displaying a table of configuration parameters.

Feature name	Current value	Initial value	Unit	Range
Host	http://192.168.0.247:80	192.168.0.247	string	
Path	/api/command/irpass/2	/api/command/irpass/2	string	
QueryStringParams	-	\z	string	
Method	POST	POST	string	
Timeout	5	5	s	[1-255]
RequestType	2	JSON	-	0,1,2,3,4,5
ResponseType	2	JSON	-	0,1,2,3,4,5
RequestHeaders	-	\z	string	
RequestBody	-	\z	string	
ResponseBody	-	\z	string	
StatusCode	200	-	-	

At the bottom of the dialog, there is a checkbox for **Auto refresh** (checked) and a **Refresh** button. The **OK** and **Cancel** buttons are located at the bottom right.

Control via myGrenton

- Add `SCENE_DOUBLE` widget:

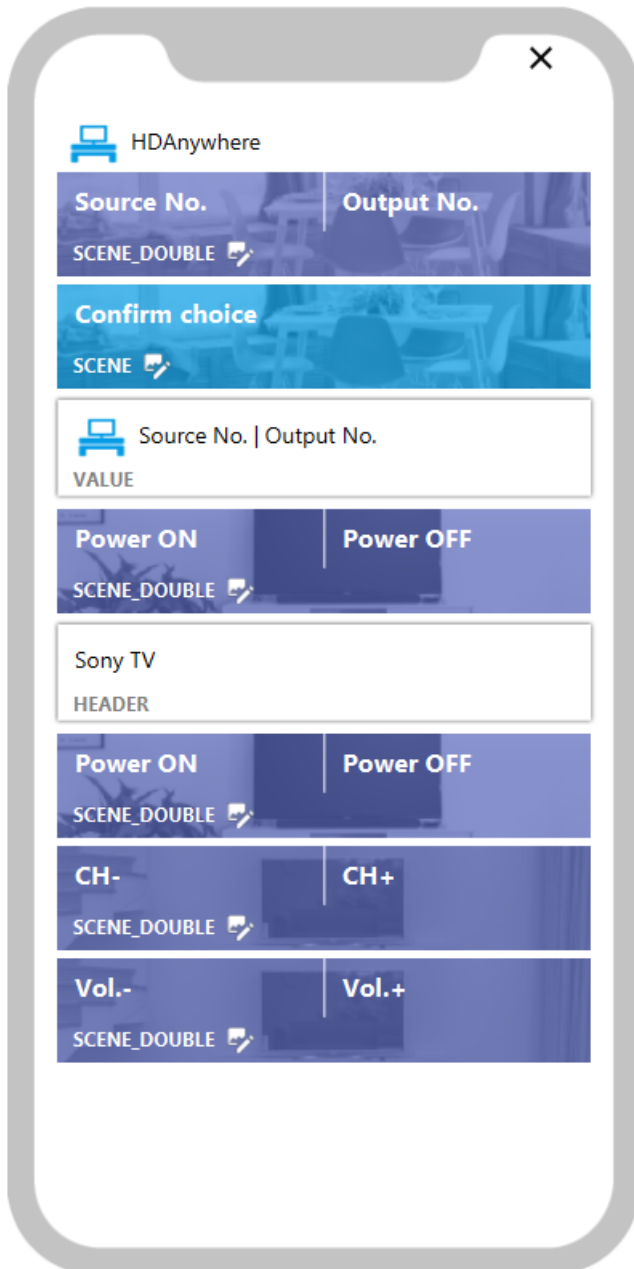


Properties

Name	Value
Type	SCENE_DOUBLE
Background image	tv_watching_2 (indigo)
▼ Button 1	
Label	CH-
Action click	HTTP->IR_HEX_CH_Down()
▼ Button 2	
Label	CH+
Action click	HTTP->IR_HEX_CH_Up()

Close

Similarly, you can add more actions based on HEX codes:



After preparing your interface, send it to your device.