iuTestTarget2

The uiTestTarget2 was created to provide a target virtual LON device to be used for custom web page examples and data access development testing for challenging union based types. The device run as an internal LON device on the SmartServer IoT and implements a network variable interface that behaves much like an actual device would behave. It implements a handful of common types that are commonly found in building controls system. Many of the network variables are complex types which could challenge the Custom web UI developer.

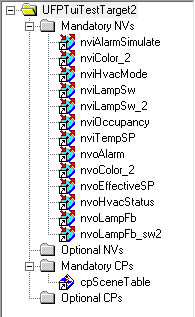


Figure 1 UFPTuiTestTarget2 Profile.

Table 1 describes the profile variables, and the behavior implemented by the node.js application which implements the simulation of the device.

Table 1- UFPTuiTestTarget2 Profile datapoints.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Index | Type | Notes |
| nviAlarmSimulate | 0 | SNVT\_alarm\_2 | Update is reflected on the output nvoAlarm. |
| nviColor\_2 | 1 | SNVT\_color\_2 | Update reflects on nvoColor\_2 |
| nviHvacMode | 2 | SNVT\_hvac\_mode | Valid inputs HVAC\_AUTO (0), HVAC\_HEAT (1), HVAC\_COOL (3). HVAC\_OFF (6). AUTO acts like HVAC\_COOL. The mode will control which temperature setpoint is played. Occupied, Standby, and Unoccupied setpoints for heating and cooling are defined by nviTempSP. When heating, the heat\_output\_primary will saw tooth between 25-90% with a 1-2 percent step. |
| nviLampSw | 3 | SNVT\_switch | Value is reflected on nvoLampFb without change. |
| nviLampSw\_2 | 4 | SNVT\_switch\_2 | This will implement a scene lookup with the state is SW\_RECALL\_SCENE (8). A scene value of 0xFF will turn the load OFF. The result of the scene lookup is reflected in the setting.value field on nvoLampFb\_sw2. SW\_SET\_LEVEL (5) will cause the level to be reported. SW\_SET\_OFF (0) will go to 0%, and SE\_SET\_ON (1) will go to 100%. |
| nviOccupancy | 5 | SNVT\_occupancy | Implements a response to OC\_OCCUPIED (0) and OC\_UNOCCUPIED (1). The nvoEffective Setpoint will change between occupied (OC\_OCCUPIED) and standby (OC\_UNOCCUPIED) levels. Remember that the nviHvacMode has bearing on the value played. A value of OC\_UNOCCUPIED will also set the nvoLampFB, and nvoLampFb\_sw2 levels to 0, and restore to the last nvi values when it is OC\_OCCUPIED. |
| nviTempSP | 6 | SNVT\_temp\_setpt | The point is a structure containing 6 setpoints for occupied, standby and unoccupied levels for heating and cooling. At startup, they are initialized to:  {  occupied\_cool: 24.0,  standby\_cool: 26.0,  unoccupied\_cool: 30.0,  occupied\_heat: 20.0,  standby\_heat: 17.8,  unoccupied\_heat: 15.0  } |
| nvoAlarm | 7 | SNVT\_alarm\_2 | Reflects nviAlarmSimulate |
| nvoColor\_2 | 8 | SNVT\_color\_2 | Reflects nviColor\_2 |
| nvoEffectiveSP | 9 | SNVT\_temp\_p | Reports the setpoint in play when considering the values of nviHvacMode, and nviOccupancy. |
| nvoHvacStatus | 10 | SNVT\_hvac\_status | Reports the mode, and simulates action on the fields heat\_output\_primary, and cool\_output depending on the mode. Both outputs go to 0 when the mode is HVAC\_OFF. |
| nvoLampFb | 11 | SNVT\_switch | Reflects the value of nviLampSw |
| nvoLampFb\_sw2 | 12 | SNVT\_switch\_2 | Reports the current state from nviLampSw\_2, and the setting.value relate to the states SW\_SET\_OFF, SW\_SET\_ON, SW\_RECALL\_SCENE, SW\_SET\_LEVEL. |
| cpSceneTable | 13-16 | SNVT\_scene\_def | Implement scene presets. The simulation does not implement the unoccupied scene but may in the future. |

# Deploying the Virtual Device

The file uiTestTarget.dtp contains the XIF file for this device, and the necessary resource file in XML form to define the UFPTuiTestTarget2 profile just describe. In DMM mode, use the Device Types widget to Import the file using the Import device types action. Allow time for the CMS to fully load, unpack, restart the LTE/LTX engines, and to create the device type. This might take on the order of 1 minute.

The file UI Test Target App.zip contains the node.js application to implement this device. Do the follow to install on you SmartServer IoT target.

1. Unzip the UI Test Target App archive to a location on your PC.
2. Use winscp to create the folder /var/apollo/data/apps/ui-test as user apollo.
3. Copy the files to this folder. (uiTestTarget2.js, package.json, package-lock.json).
4. Connect to the ssh console as user apollo, and type:

cd /var/apollo/data/apps/ui-test

1. Type: npm install **Your SmartServer IoT must have an internet connection.**
2. To run the application, type note that the 1 argument gives developer detailed output tracing to the SSH console: node uiTestTarget2.js 1
3. To run this application as a service, the file uiTestTarget.conf needs to be copied to the directory /etc/supervisor/conf.d
4. When you reboot the SmartServer, the application should run as a service, and be part of the collection of applications you can monitor and control using sudo supervisorctl

A few notes. This application creates itself on the SmartServer IoT as device uiDev.1. If running as a service, you need to be sure to import the DTP file so the resources are in place for the CMS to create the device. You need to do sudo supervisorctl, and restart uiTestTarget2. The best practice if you plan to do apollo-reset is this sequence:

1. In and SSH console type: sudo supervisorctl
2. In the supervisorctl type: stop uiTestTarget2
3. Do the apollo-reset normal [pwd].
4. After the apollo-reset completes, the log in to the CMS and in the Device type widget, Import Device types to import the DTP package for this example
5. In supervisorctl type: start uiTestTarget2
6. Observe the device uiDev.1 being crated and provisioned in the Device widget.
7. Note, it is like that the UID uiDev.1 will change if you are using nodeutil to exercise the interface.