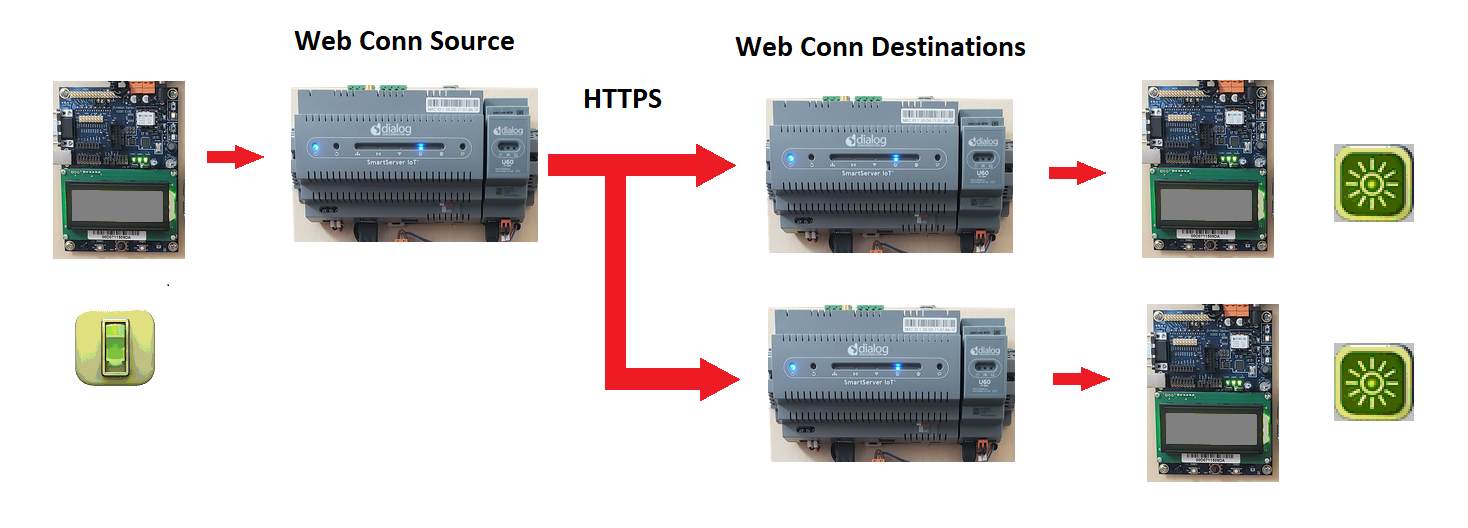
Node-RED Web Connector

(Web Conn)

This document describes how to create a Web connection (Web Conn) between two or more SmartServers. Web connections can be used if you want to send datapoint or configuration information from one SmartServer (source) to one or more other SmartServers (destinations). That is, Web Connections can push data from one SmartServer to other SmartServers. A Web Connection can be created by using Node-RED HTTP nodes in which there is one HTTP source and one or more HTTP destination.



In a test with one Source SmartServer sending a SNVT\_switch value to two other SmartServers we have seen a datapoint change on the source SmartServer device being propagated (to two destination Smartserver each with a single edge device) within 2-3 seconds. That is, pressing a button on a external device connected to a Source SmartServer, turned on or off a LEDs on a devices connected to two destination SmartServers within 2-3 seconds (each destination SmartServer had one external device).

The more destination SmartServers that you have will most likely add more delay. In addition, for each destination SmartServer that you have, the more edge devices you must write to the more delay you may see.

To implement Web Connection between two or more SmartServers, two Node-RED example flow tabs have been created: Web Conn Source flow tab and a Web Conn Destination flow tab. To use the flow tabs, you import the provided flow tabs into the SmartServer Sequence Widget on to the respective SmartServers, modify the flow tab for your specific SmartServers and datapoints and deploy it.

The Web Conn Source flow tab is used for the SmartSever that is sending a datapoint update. The Web Conn destination flow tab is used on one or more SmartServers that is supposed to receive a datapoint update.

Both Web Conn flow tabs can be used on the same SmartServer. That is, a SmartServer can have a flow to push data to other SmartServers, and it can have a flow to receive data from other SmartServers.

Data is sent as a REST request using endpoint “/node-red/webbind” with a JSON payload like the one shown below which consists of a dps array. In most cases only one datapoint will be sent. If the flow has to resend data, then it will resend all dp values in one message that have not already been sent to that SmartServer.

{"dps":"[{\"dp\":\"nvoLampFb\",\"value\":{\"value\":0,\"state\":0}}]"}

Resending occurs after logging in or after recovering after temporary lost communication.

The Web Conn is designed to support sending a couple of datapoints which have unique names. Datapoint names are extracted from the IAP-input node msg.payload.message which looks like

msg.payload.message = <datapoint name>/value

The “/value” string is stripped off of msg.payload.message to create the datapoint name used for the dps array.

For example:

msg.payload.message = “nvoLampFb/value” the datapoint name is “nvoLampFb”

You need to make sure all the datapoints sent to the HTTPS PUT node have a unique datapoint name (msg.payload.message). If not, then use another Function node before the HTTPS PUT node to change the datapoint name to make it unique.

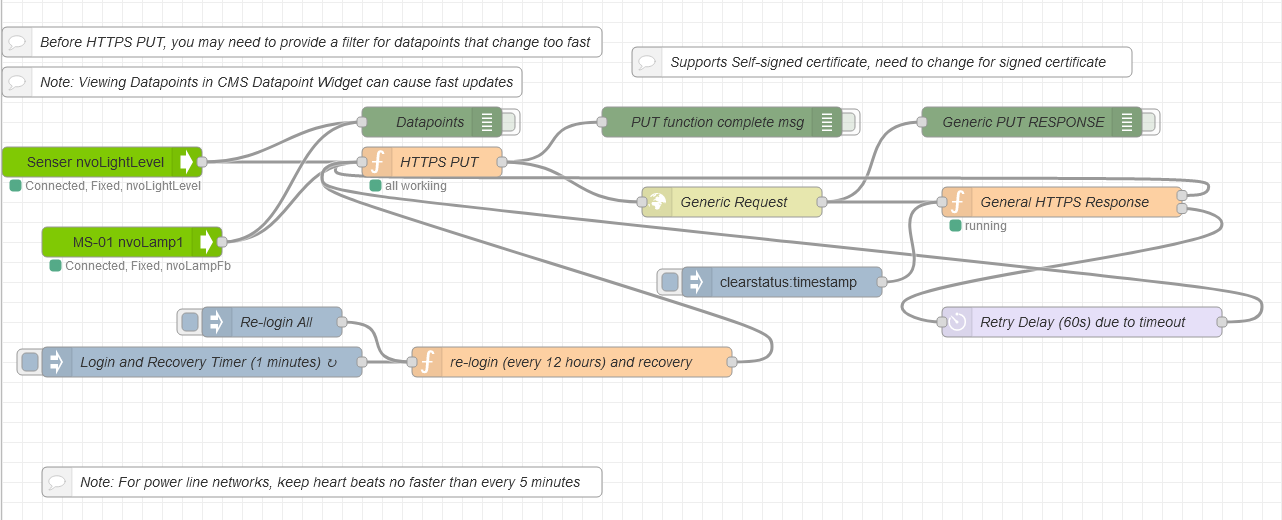
So if two IAP-input Nodes have the same <datapoint name> then you will need to use a function between the IAP input node and the HTTPS PUT node to change the msg.payload.message datapoint name.

The Web Conn Source Flow SmartServer needs to log into the Web Conn Destination Flow SmartServer before it can push the datapoint updates, so you should create a new owner login for each Server Flow SmartServer. The Web Conn Source Flow needs to re-log into the destination SmartServers roughly every 12 hours, as login secession times out after 24 hours.

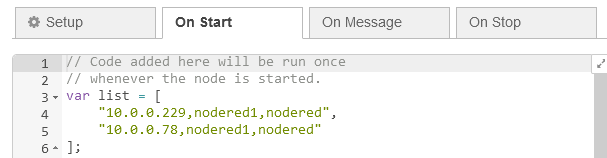
How it works:

For each Web Conn Destination SmartServer, you should create a new owner login on that SmartServer which will be used by the Web Conn Source flow. I don’t recommend using the apollo login so use something different. You will then need to modify the Web Conn Source flow with the destination SmartServer IP Address and login.

Web Conn Source flow:



The “HTTP PUT” function node is the heart of this flow. It sends messages to login or send data to the remote SmartServers. The Generic Request node (HTTP Request node) handles communication between the SmartServers. Add the list of the remote SmartServer IP addresses and username/password to Web Conn flow tab “HTTP PUT” function node “on start” tab.



This Source flow “HTTP PUT” function expects to see a datapoint name in the msg.payload.message and the datapoint value in msg.payload.data. The “HTTP PUT” node provides status information, but you may see an all devices down status even if some or all devices are working. Wait for the next datapoint update to see if the status message changes.

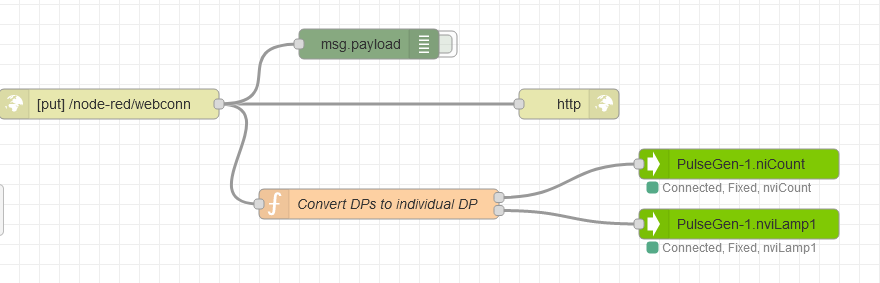
The Source flow “General HTTPS Response” node handles responses from the remote SmartServers. If a successful login request was sent to the remote SmartServer, the General HTTPS Response node will tell the HTTP Request node to resend all data not already sent. If there is a response error or timeout, the General HTTPS Response node will either send the request out again after a minute (it will re-try more than once) or set the recovery counter (controlled by a backoffCount).

The Source flow “Re-login and recovery” node is used to re-login every 12 hours (once it logs into the new REST secession it logs out of the old REST secession) and to re-try communication to remote SmartServer periodically. The login time and recovery time uses counters and so it a product of the counter and timer inject node. Using the re-login inject node, you can cause the HTTP PUT function node to immediately issue login messages to all the SmartServers.

Once a successful re-login is sent, the flow will resend any datapoint updates not sent and re-log out of the old connection with the remote SmartServer.

When a datapoint update is sent to the HTTP PUT function node, the HTTP PUT node will either send a msg to login into the remote Destination SmartServer (and a msg to log out of the Destination SmartServer) or send the data to the destination SmartServer. For datapoint updates going into the HTTP PUT node, the node uses msg.payload.message and msg.payload.data for the datapoint name, and datapoint value, respectively.

Web Conn Destination flow:



The Web Conn Destination Flow needs to be imported into the SmartServers that is supposed to receive the datapoint updates. This flow only gets data after the remote Web Conn Source flow logs into this SmartServer.

The PUT node will have a Node-RED msg.payload similar to the one below.

{"dps":"[{\"dp\":\"nvoLampFb\",\"value\":{\"value\":0,\"state\":0}}]"}

If you are using IAP Output nodes to send data to device datapoints then you will have to break up dps array into individual datapoints. The “Convert DPs to individual DP” Function shows how to break up the datapoints into individual outputs that are wired to a specific IP-Output node and sending just msg.payload.data.

msg.payload = {"data":{"value":100,"state":1}}

These updates are sent a normal priority. If you want to send the updates a different priority then you can either modify the Convert DPs to individual DP” function node or add another function node to add the priority.

If you are using a MQTT output node to send datapoint updates the you will have to replace the “Convert DPs to individual DP” function to a function that only uses a single output for the function node.

There is no error checking to see if the Destination Flow actually sent the data.

Once the Web Conn Source or Destination SmartServer powers up, it takes about 7.5 minutes for data to be transferred.

Design Criteria:

1. The Source SmartServer needs to log into Destination SmartServers CMS Node-Red uri.
   1. best to setup a new owner user that is only used for this flow
   2. The following REST endpoint is used “/node-red/webbind”
   3. Even though the
2. Determine how often you push the data to other SmartServers
3. Determine how you are going to limit data from being pushed too often.
   1. Looking at datapoints in a CMS Datapoint Widget may poll datapoints much faster than the IAP Input nodes or CMS Datapoint Properties widget setting.
4. Make sure you don’t constantly send requests when SmartServer not available temporarily or down.
   1. EHOSTUNREACH – timeout, resends after 1 minute for up to errCountMax times
      1. If still fails after errCountMax then uses recovery time
   2. HTTP 401 – bad password, uses recovery time
   3. HTTP 429 – too many attempts, uses recovery time
      1. Uses 5-minutes timeout if too many login attempts
      2. Uses 2-minute timeout for all other too many attempts
5. Need to log into server every 12 hours and then log out of old secession.

**Web Conn Source Instructions** (SmartServer pushing the data):

1. Import Web Conn Source Flow to SmartServer that you want to push data from.
2. Replace the IAP input nodes with the nodes that you want to use.
3. If using more than one datapoint, makes sure datapoint names are unique.
   1. Only the datapoint name is used, datapoint block name and block index are not used.
   2. If datapoint names are not unique then you can do one of the following
      1. Modify the HTTPS PUT node to use the entire Datapoint path for the datapoint name. For example: “Sensor1/Light/0/nvoLamp1” instead of “nvoLamp1”
      2. use a Function node before the HTTP PUT node to change the name of the node.
4. Make sure that datapoint updates don’t happen to fast.
   1. The HTTPS Put function tries to limit communication but doesn’t limit datapoints that have fast updates in which the datapoint value changes every update.
      1. This node limits the same value being sent to one every 30 seconds.
      2. If value changes then the new value is sent immediately.
      3. If you are looking at sending lightlevel or temperature that can change for every poll when the CMS Datapoint Widget is reading the values then you may need to add a function node between the IAP-input node and the HTTPS PUT function node that limits datapoint values based on a delta value in order to reduce the traffic.
   2. You need to make sure if someone uses the CMS Datapoint Widget that 2 second polling doesn’t overwhelm communication.

**Web Conn Destination** Instructions (SmartServer pushing the data):

1. Import Web Conn Destination Flow to SmartServer that you want to push data from.
2. Replace the IAP-output nodes with the nodes that you want to use.
3. You will need to modify the “Convert DPs to individual DP” function node to break up the datapoints into separate outputs.
   1. If you have a lot of datapoints then should consider using a MQTT-output node to send data instead of IAP-output nodes.
4. Once you set up one Destination SmartServer export the flow tab and now you can import on other destination SmartServers.

Known Issue:

1. The Web Conn Source Flow “HTTPS Put” node sometimes falsely reports all devices are down. Wait a short period of time (for at least the next datapoint update) to see if a the devices are really down.
2. The only known issue is when the Source Flow tab tries to log out of a destination SmartServer that you will see the following message in the Sequence Widget debug menu. This message can be ignored as the problem is believed to be due to having a msg.payload = “”.  
   