REFERENCE MANUAL

Portal

Reference Manual for Version 2.6.6

Doc# 116-061520-009-C000

6723 Odyssey Drive // Huntsville, AL 35806 // (877) 982-7888 // Synapse-Wireless.com

Disclaimers

Information contained in this Manual is provided in connection with Synapse products and services and is intended solely to assist its customers. Synapse reserves the right to make changes at any time and without notice. Synapse assumes no liability whatsoever for the contents of this Manual or the redistribution as permitted by the foregoing Limited License. The terms and conditions governing the sale or use of Synapse products is expressly contained in the Synapse's Terms and Condition for the sale of those respective products.

Synapse retains the right to make changes to any product specification at any time without notice or liability to prior users, contributors, or recipients of redistributed versions of this Manual. Errata should be checked on any product referenced.

Synapse and the Synapse logo are registered trademarks of Synapse. All other trademarks are the property of their owners. For further information on any Synapse product or service, contact us at:

Synapse Wireless, Inc.
6723 Odyssey Drive
Huntsville, Alabama 35806
256-852-7888
877-982-7888
256-924-7398 (fax)
www.synapse-wireless.com

License governing any code samples presented in this Manual

Redistribution of code and use in source and binary forms, with or without modification, are permitted provided that it retains the copyright notice, operates only on **SNAP®** networks, and the paragraphs below in the documentation and/or other materials are provided with the distribution:

Neither the name of Synapse nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SYNAPSE AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SYNAPSE OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SYNAPSE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Open-source Libraries Included in Portal

apy (2.4.1)	argparse (1.3.0)	configparser (3.3.0.post2)	crcmod (1.7)
NumPy (1.8.2)	passlib (1.6.2)	pycrypto (2.6.1)	wxPython (3.0.1)
pypiwin32 (217)	Python (2.6.6)	PyInstaller (2.1)	

Table of Contents

1. Introduction

Synapse **Portal** is a standalone software application that runs on a standard PC. Using an RS232 or USB interface, it connects directly to any node in a **SNAP** Wireless Network and becomes a graphical user interface (GUI) for the entire network. Using Portal, you can quickly and easily create, configure, deploy, and monitor **SNAP**-based network applications. Once connected, the **Portal** PC has its own unique **SNAP** Network Address and can participate in the **SNAP** network as a peer.

Portal can also *remotely* connect to your **SNAP** network over a TCP/IP network to applications that incorporate the **SNAP Connect 3.x Python Library**.

The **SNAP Connect 3.x Python Library** allows you to develop, configure, and deploy **SNAP** applications over the Internet. For more about this product, refer to **SNAP Connect Manual**. (Instructions on how to connect to this product are covered later in this manual).

Along with this document there are several other documents you need to be aware of:

Start with an "Evaluation Kit Users Guide"

Each evaluation kit comes with its own users guide. For example, the DK-200 kit comes with the **DK-200 Evaluation Kit Users Guide** ("**DK-200 Guide**").

Evaluation Kit Users Guides walk you through the basics of unpacking an evaluation kit, setting up wireless nodes, and installing **Portal** software on a PC. (Synapse **SNAP** nodes and even their component **SNAP** Engines are also sold separately, as well as bundled into evaluation kits.) Evaluation kit manuals are available from the Synapse support forum at https://forums.synapse-wireless.com/showthread.php?t=9.

The **SNAP Users Guide** contains general information about **SNAP** and **SNAPpy**. It covers topics like the **SNAPpy** language, and how to use it.

There is also a companion manual to this user's guide, the **SNAP Reference Manual**. In the **SNAP Reference Manual** you can find detailed information about **SNAP** and how to use it.

About This Manual

This manual assumes you have read and understood an evaluation kit guide. It assumes you have installed the **Portal** software, and are now familiar with the basics of discovering nodes, uploading **SNAPpy** scripts into them, and controlling and monitoring them from Portal.

Other Important Documentation

Be sure to check out all of the **SNAP** documentation:

If you are completely new to **SNAP**, be sure to check out the **SNAP Primer which provides a complete overview of SNAP and how it's used**.

There is a separate user manual for the **SNAP Connect 3.x Python Library**. **SNAP Connect** allows you to monitor and control your nodes from remote locations using TCP/IP networks.

There is a wealth of valuable information in the **SNAP Hardware Technical Manual**. This document covers every jumper and every connector of every type of node included in the evaluation kit.

For your convenience, much of the information available in the **SNAP Hardware Technical Manual** has been broken down into individual "Quick Start" guides. For example, there is a **SN171 Proto Board Quick Start** and a **SN132 SNAP Stick Quick Start**.

There is also a dedicated support forum at http://forums.synapse-wireless.com.

In the forum, you can see questions and answers posted by other users, as well as post your own questions. The forum also has examples and Application Notes, waiting to be downloaded.

You can download the latest **SNAP**, **Portal**, and **SNAP Connect** software from the forum. The website forums also contain the latest documentation for all Synapse products.

2. A Portal Into Your Network

Portal is a standalone software application that runs on any standard PC with Microsoft Windows 7 or higher. Using a USB or RS232 interface, it connects to any **SNAP** Engine in the **SNAP** Wireless Network, becoming a user interface for the entire network.

Note: Most of the icons shown in the previous diagram were taken directly from the **Portal** user interface.

 is used (by default) within **Portal** to generically represent a **SNAP** node.

 is used within the **Portal** user interface to represent **Portal** itself.

Once connected, Portal provides the capability to *interactively* build an *intelligent* wireless network. You can:

- Discover new **SNAP** Devices
- Upload intelligence to those Devices over the air, using **SNAPpy** scripts
- Customize **Portal** to suit your specific application

Interactively – you do all this within **Portal**, observing the results immediately.

Intelligent – the network is purpose-built for your application, with the ability to monitor and conditionally control things connected to it.

Throughout this manual, the term "**Portal** PC" is used to refer to the Windows PC that **Portal** is running on.

Note: Synapse also licenses a standalone **SNAP Connect** library, giving you access into your **SNAP** network. Import the **SNAP Connect** library into your Python application to allow your backend systems to participate seamlessly in the **SNAP** network.

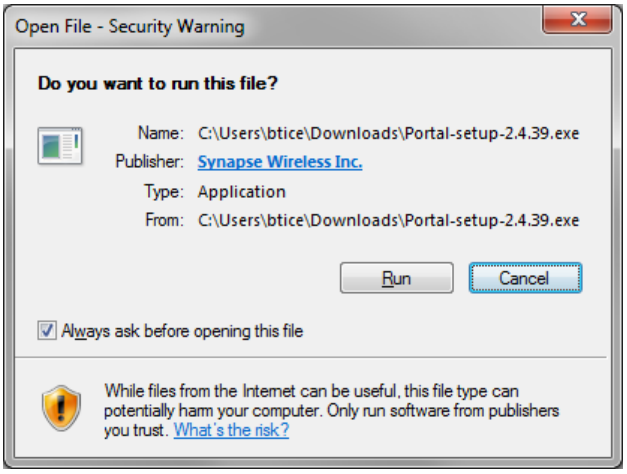
Installing Portal

You'll find the latest version of **Portal**, along with the latest **SNAP** firmware and documentation, on the Synapse Support Forum at https://forums.synapse-wireless.com/showthread.php?t=9.

Running Setup

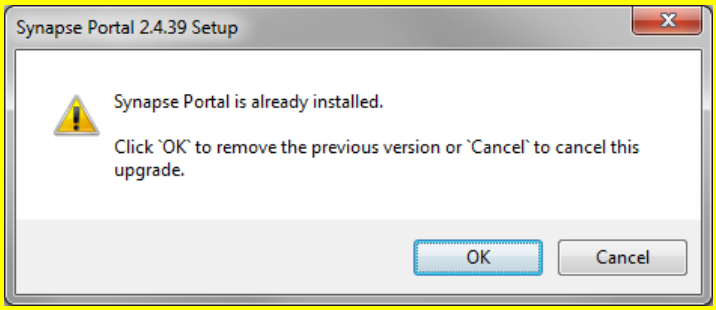
Download and run the **Portal** installer, Portal-setup-2.6.n.exe (where "n" indicates the latest release version). You should unplug any Synapse devices from your PC before installing, so that device drivers can install properly.

Depending on the version of Windows running on your PC, you may get a warning dialog similar to the following somewhere in the installation process:



The warning is harmless, and you should click **Run** or **Continue Anyway** to proceed with the installation.

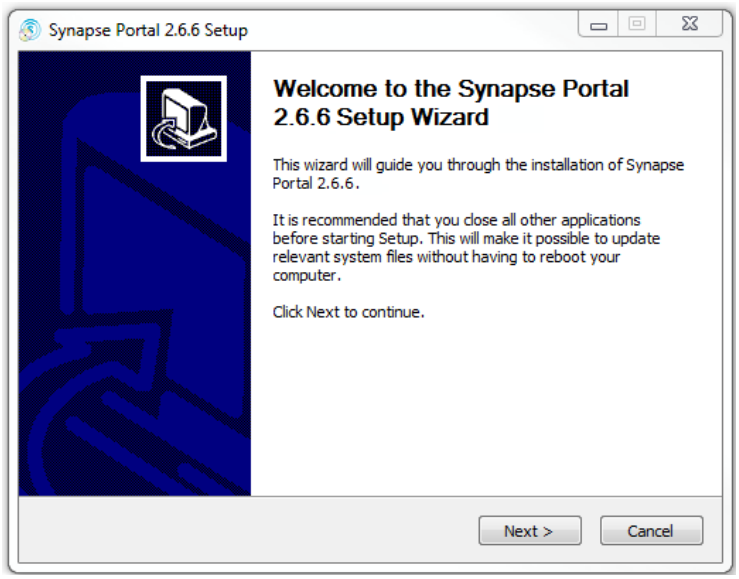
Note: If a previous version of **Portal** (for example, version 2.2.23) is already installed on your computer, you will get an initial dialog box like the following:



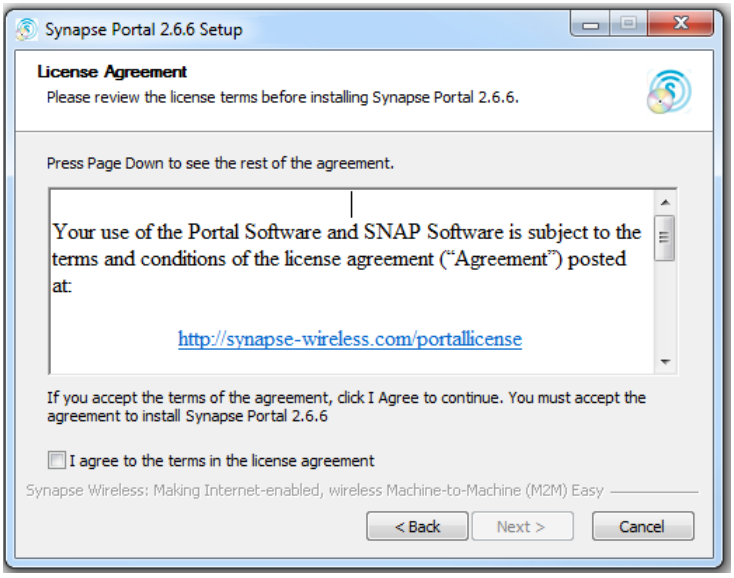
You must click the **OK** button in order to install the newer version.

The following screenshots assume you are installing version 2.6.6. Your precise **Portal** version number might be different from 2.6.6, but the process should be very similar.

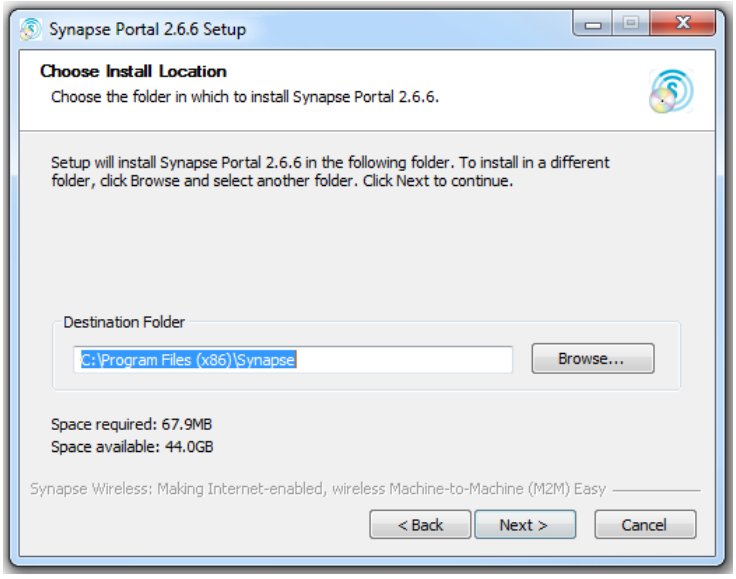
A dialog box similar to the following will appear (your version number likely will be higher).



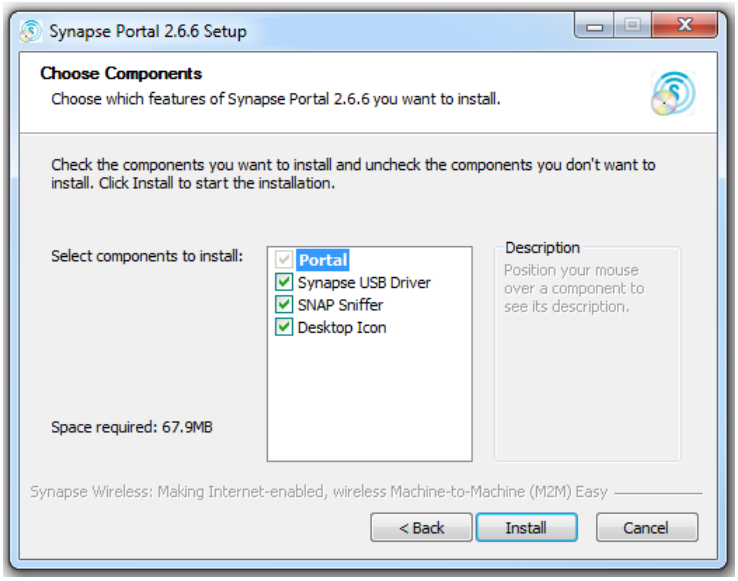
Click the **Next** button to get the following:



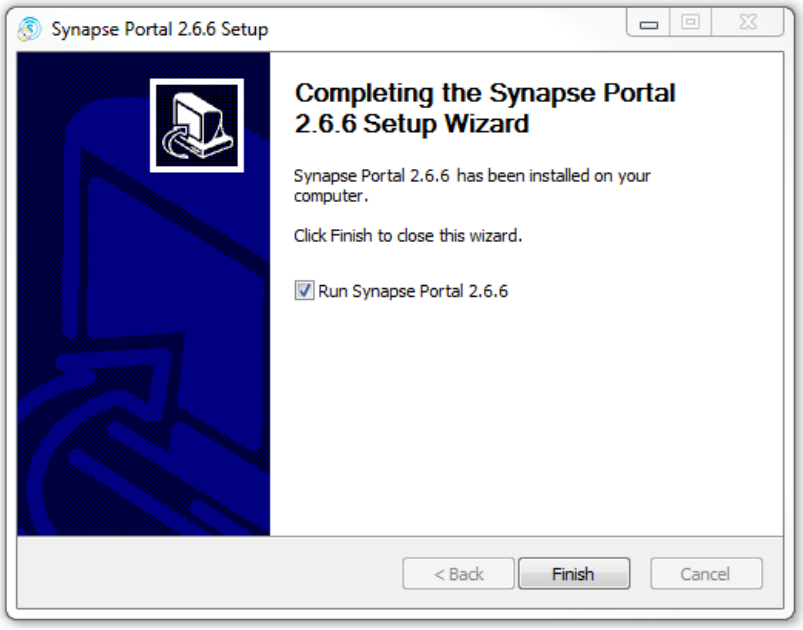
Read the license agreement at the specified URL, check the **I agree** box and then click **Next**.



You can either enter the desired destination folder manually, browse to the desired folder, or just click **Next** to accept the default.



Make sure the desired components are checked, and click **Install**.

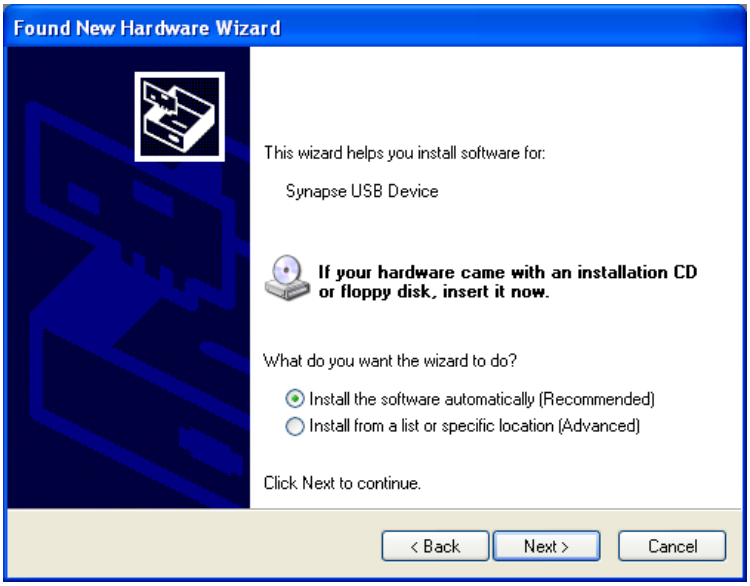


You have now successfully installed **Portal**. There will be a **Portal** icon on your Windows Desktop (if you specified there should be), as well as in the Start Menu. We recommend that you do not run **Portal** until you have completed the bridge device driver installation through the following steps, so you should uncheck the **Run Synapse Portal** checkbox before clicking **Finish**.

Plug in a Bridge Device

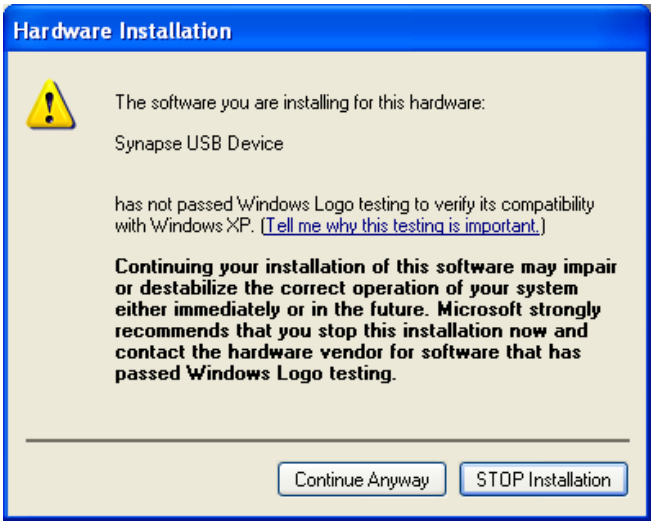
A bridge device, such as a **Synapse SS200 SNAP Stick,** should now be plugged into a USB port on the PC where you installed **Portal**. Depending on the drivers loaded on your system and which version of Windows you are using, it may be necessary for the Synapse USB drivers to complete installation. This will only occur the first time the bridge device is connected and powered up. If necessary, the following dialog box will appear:

Since the correct software is already available (you just installed it along with **Portal**), there is no need for Windows to connect to Windows Update – just select **No, not this time** and then click **Next**.

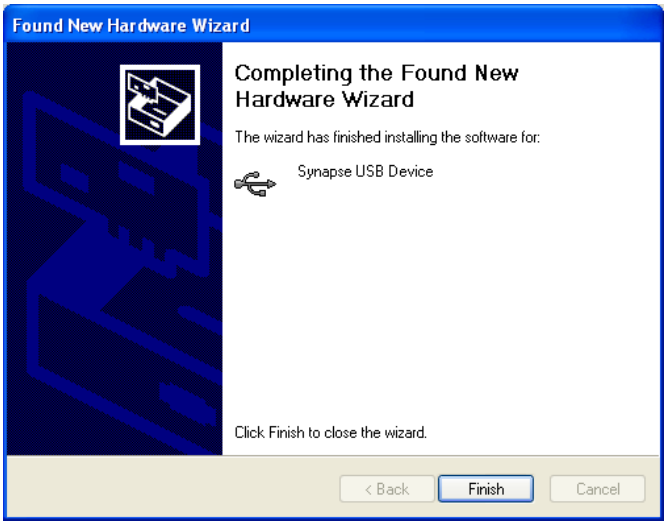


Choose **Install the software automatically…** and click **Next**.

Depending on the version of Windows you are running, you may get a warning dialog similar to the following:



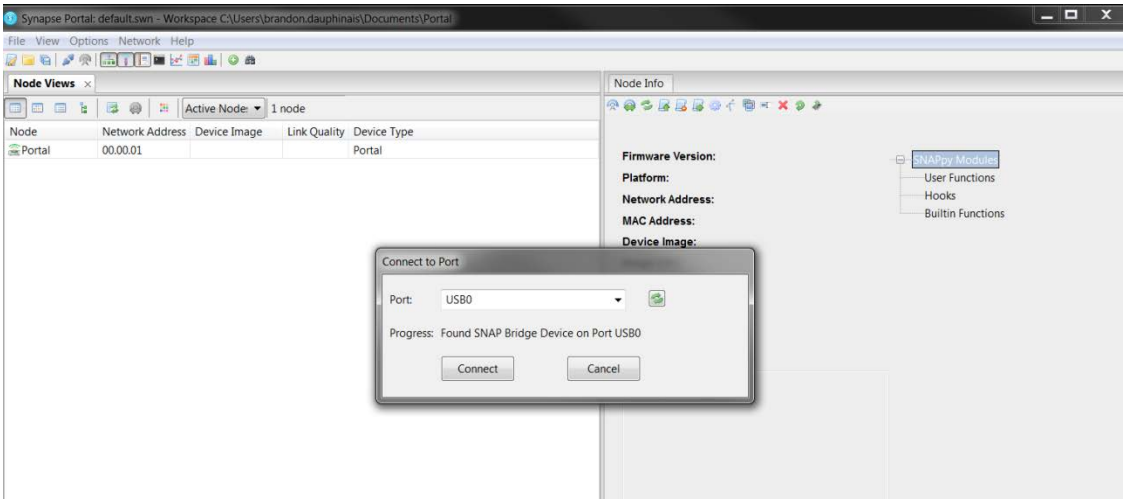
This warning is harmless, and you should click **Continue Anyway**.



Congratulations! Your Synapse USB Device is installed and ready to be used by **Portal**.

Program Launch

After installation, launch the **Portal** program. You should see a screen similar to the following:



If you do see something similar, you have now successfully installed **Portal** and detected a USB connected bridge device. The **Connect** dialog box is automatically shown at **Portal** startup. If you click the **Cancel** button, you can bring this dialog back by clicking the button on the main toolbar.

Also be aware that this toolbar button doubles as a status indicator. When you are connected, it looks like , and functions as a disconnect button. When you are not connected, it looks like and functions as a connect button.

Note: The **Portal** PC (the PC that the **Portal** software is running on) has no 802.15.4 radio of its own. One of the **SNAP** nodes must act as a "bridge" for it. **Portal** will connect directly to this bridge node, using either a USB or RS232 connection.

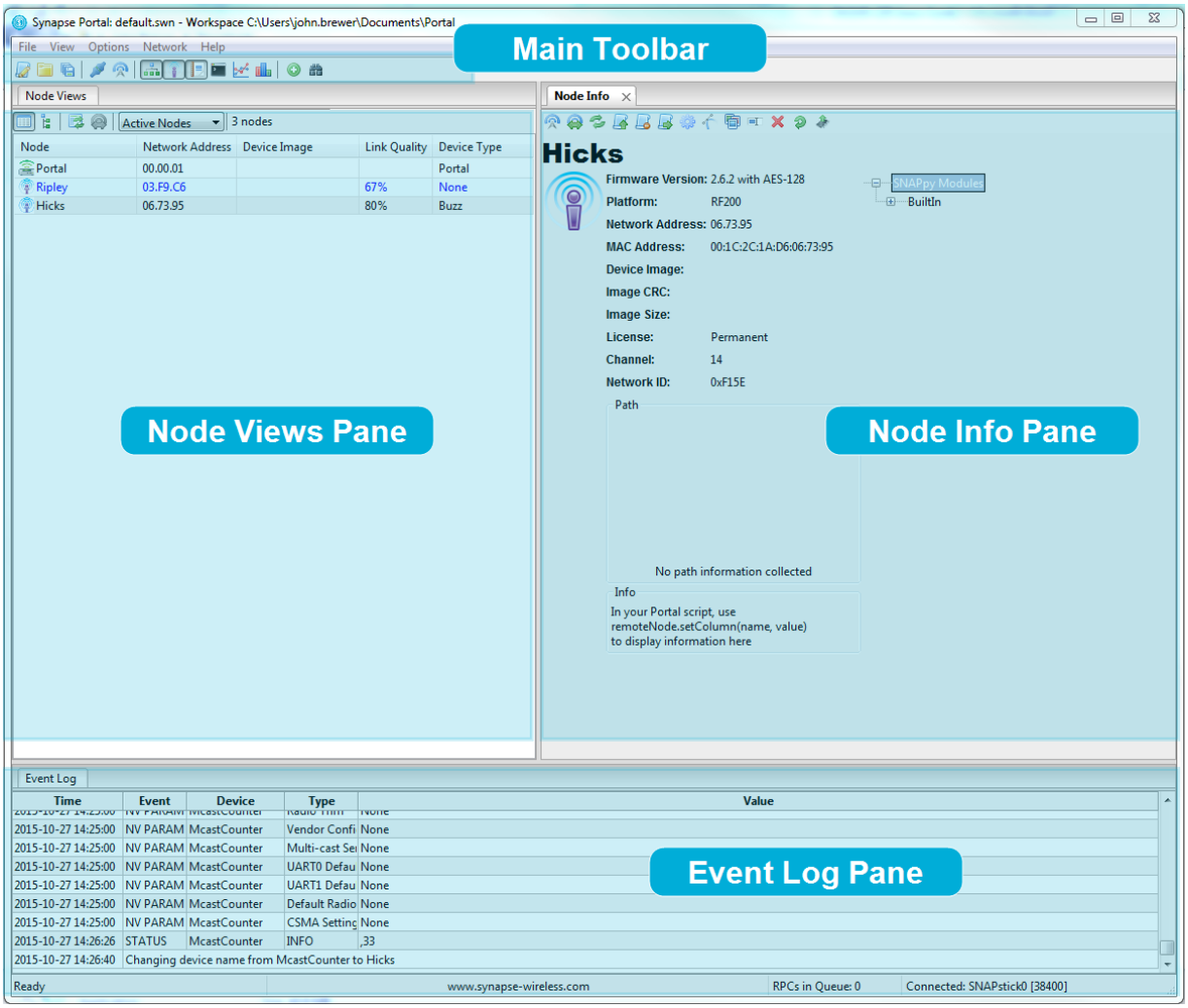
Portal will then be able to communicate to the rest of the **SNAP** nodes indirectly by sending packets across the directly connected bridge node.

3. The Portal Environment

Starting **Portal** for the first time displays a blank network configuration and a dialog asking what port to use to connect to the **SNAP** bridge device. Be sure you have a **SNAP** bridge device connected, and select it in the drop-down window. Press **Connect** once the bridge device is detected (this is probably going to be USB0).

Portal is the desktop environment used to configure and deploy a **SNAP** network, and it provides a user-customizable interface to aid in developing your **SNAPpy** scripts.

Starting **Portal** for the first time shows **Portal's** default window configuration:



From this user interface you can start viewing and configuring your **SNAP** network. The nodes in your network are displayed in the **Node Views** window as they start responding to queries made by **Portal**. Selecting a node by clicking it in the **Node Views** window will display detailed information about that node in the **Node Info** window. Finally, the **Event Log** window displays event messages about things that have occurred in the system and in the **SNAP** network. More information on these windows and their functions is provided later in this document.

Navigating within Portal

Portal is straightforward to use. However, since it is extremely customizable by the user, your screen layout may not always match the screen shots in this manual. For that reason it is important to understand the fundamental concepts used in navigating the **Portal** GUI.

Pull-down Menus

At the top of the **Portal** GUI are pull-down menus for **File**, **View**, **Options**, **Network**, and **Help** operations.

Clicking one of these top-level menu choices will pull down a sub-menu of additional choices. For example, clicking **Network** will present a sub-menu from which you perform actions like **Broadcast Ping**, **Find Nodes…**, or **New Configuration**. Similarly, clicking **Help** will display choices for **SNAP Reference Manual**, **Portal Reference Manual**, etc.

The convention is that menu choices ending in "…" usually display additional menus or dialog boxes, and menu choices not ending in "…" cause immediate action to be taken, with no further prompting.

Tool Bar

Below the pull-down menus is a horizontal Tool Bar from which you can initiate several actions. Hovering the cursor over each button will display a short "tool-tip" help message, and clicking each button will initiate the action displayed by the tool-tip.

Tabbed Windows

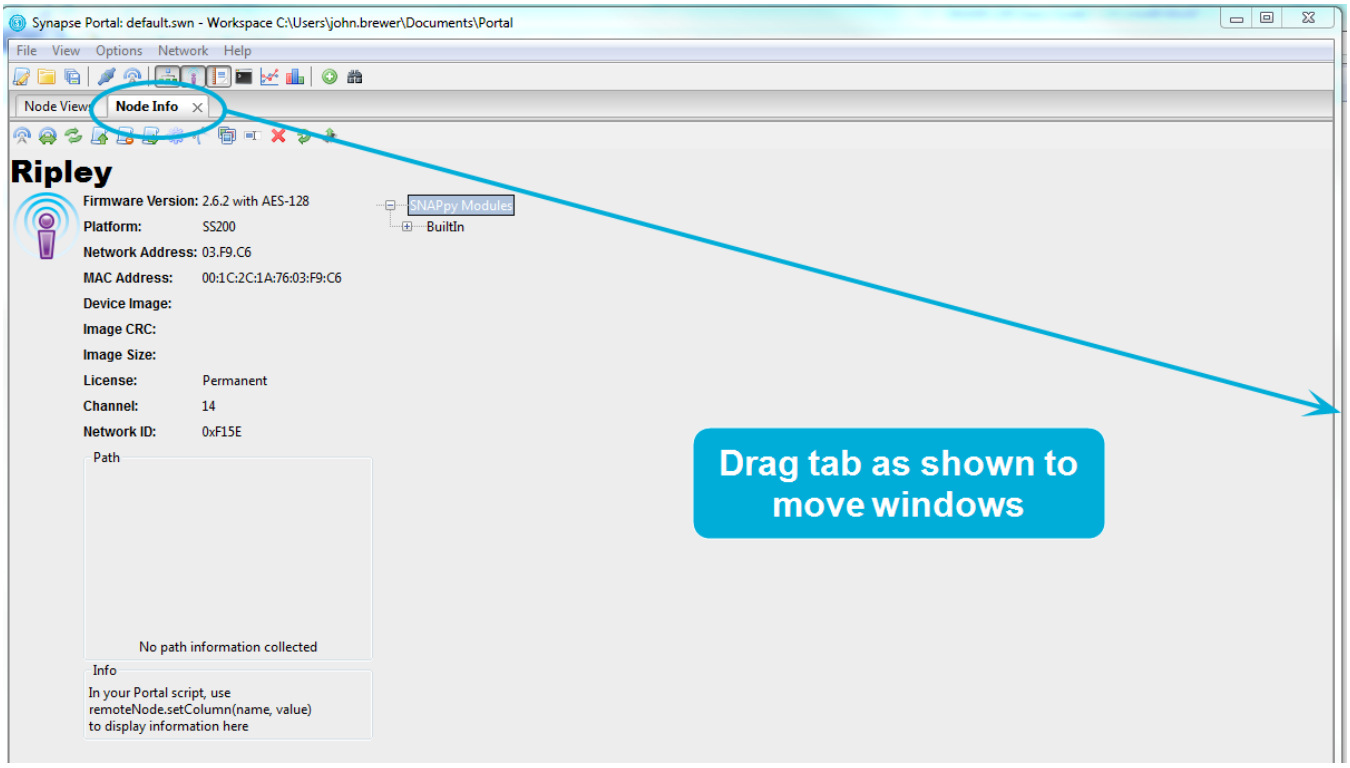
The remainder of the **Portal** GUI is taken up by a changeable collection of tabbed windows. Each of these windows has a name, which is displayed in the tab for that window.

Many of the tabbed windows have toolbars of their own, located in the horizontal region just below their labeled tab.

Portal starts out with an initial set of tabbed windows visible. Sometimes clicking certain controls within one tabbed window will open and/or switch to another tabbed window. You can also open additional tabbed windows by choosing them from the **View** menu. Finally, many of the tabbed windows can be launched from the main tool bar.

Rearranging Windows

The tabbed windows in **Portal** can be dragged and repositioned on the screen. To do this, press and hold the left mouse button while the cursor is positioned over the tab label you want to move. While holding the button down, drag the tab until you see a light blue "shadow" indicating a possible new position for the window. When you've found a suitable new position, release the mouse button and the move will be complete.



Resizing

Windows may be resized by clicking and dragging the horizontal and vertical borders separating them.

Closing Tabs

You can close tabbed windows that you no longer want by clicking the small **X** located to the right of the name in the tab.

Now that you know the basics of navigating within **Portal**, we can continue with the tour.

Discovery

We first need to look at the **Node Views** tabbed window. If this window is not already open, you can click **Views**, and then choose **Node Views window**. Alternatively, you can click the icon on the toolbar.

You will notice that the **Node Views** window has its own toolbar. Ignore all but the first four buttons for now.

The **Node Views** tabbed window lets you look at your nodes in two ways:

- **Report View**
- **Tree View**

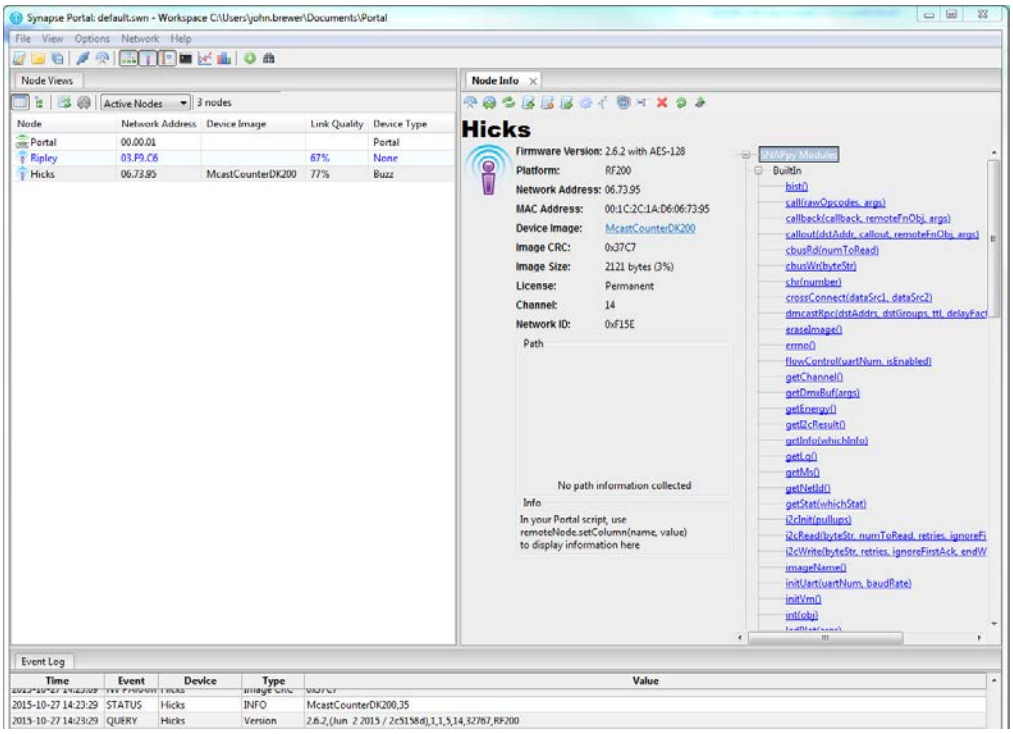
Both views are just that, "views" of the same network information.

Note: If a node isn't showing up in the list, you can click the **Ping** button to refresh it.

Click the **Report View** button.

You should see that two devices have been discovered. The directly connected “bridge” device should be shown in blue, while the remote device should be displayed in black.

When you double-click a node in one of the **Node Views**, **Portal** displays basic information about that node in a separate **Node Info** pane.

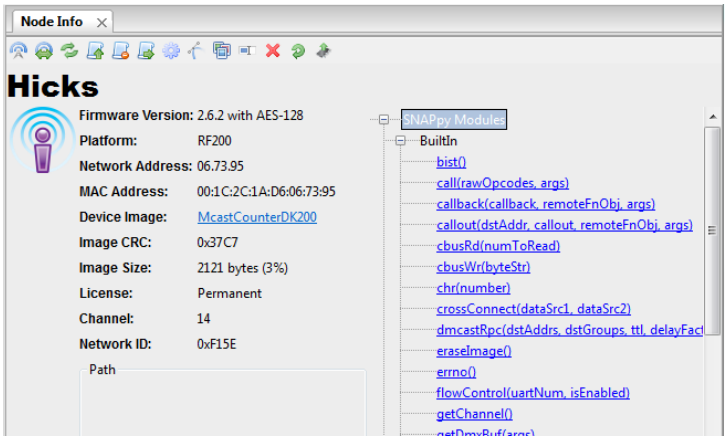


Note that the node names are based on the names of the **SNAPpy** scripts loaded into those same nodes at the time of discovery, but with additional trailing digits (2, 3, 4, etc..) added to enforce uniqueness.

If a node doesn’t currently have a script loaded it will have a name like “Node” or “Node2”.

Node Info

Lots of information is shown in the **Node Info** tabbed window. However, **Portal** may not be set to automatically query the node for its information. (This is a configurable preference in **Portal**.) To be sure **Portal** knows everything important about your node, click the **Refresh Node Information** icon in the toolbar that runs across the top of the **Node Info** tab.

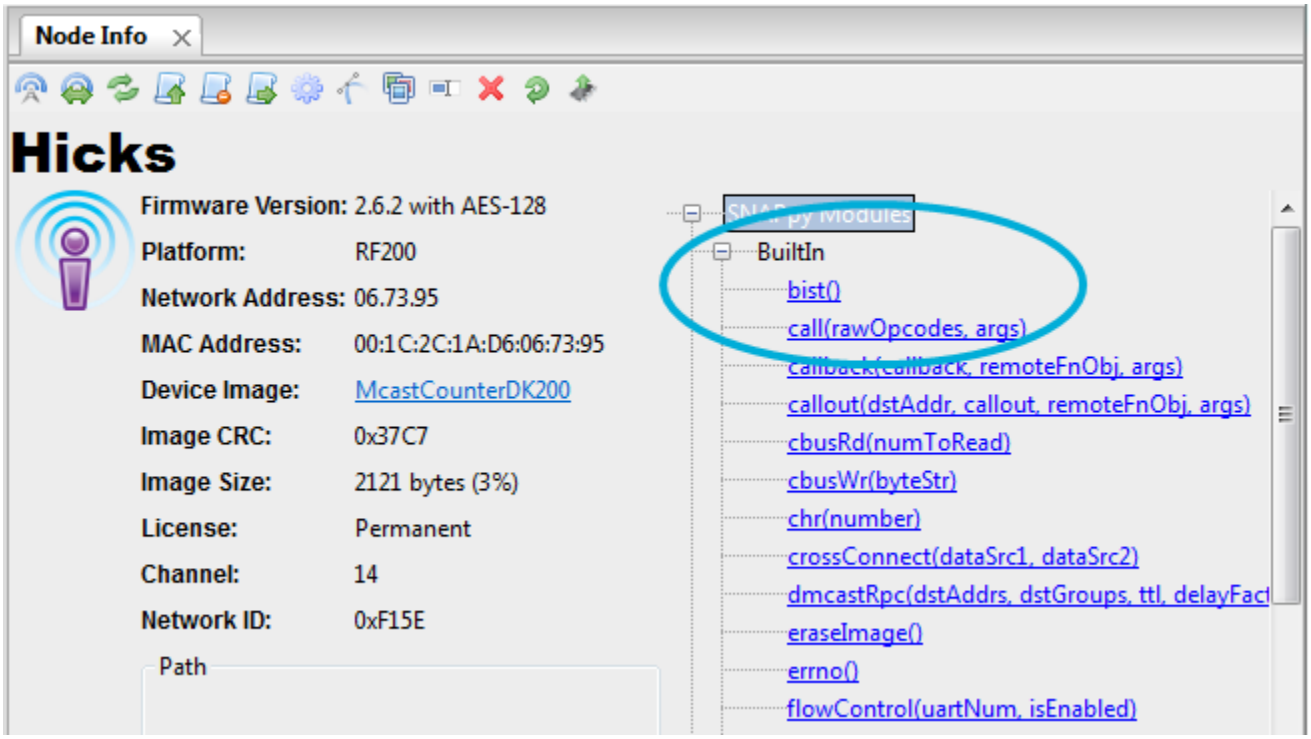


Across the top, a toolbar provides easy access to node-specific functions.

On the left-hand side, the **Firmware Version**, **Platform**, **Network Address**, **MAC Address**, **Device Image**, **Image CRC**, **Image Size**, **License**, **Channel**, and **Network Id** are shown. Below that is a block where **Path** information (the path to/from the node) can be displayed. Below that there is an **Info** field that can be controlled from **Portal** scripts to add your own custom field(s) to the **Node Info** panel.

Device Image refers to the **SNAPpy** script (also referred to as a **SNAPpy** image) loaded into the node. Here you can see that the script/image "McastCounter.py" has been loaded into the node. You can click the device image name shown (McastCounter), and automatically display that script in **Portal**'s built-in source code editor.

On the right hand side, a collapsible tree of available functions is shown. In this next screenshot, you can see the **BuiltIn** tree (the tree of built-in functions) in expanded form.

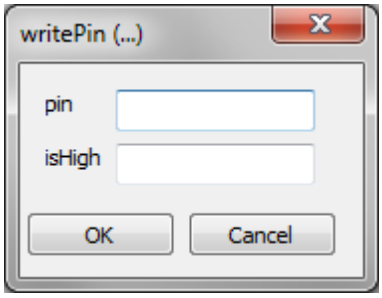


Notice that there is a scroll-bar on the right-hand side of the pane – there are too many built-in functions to fit on the screen at one time.

Hovering the cursor over a function name will display a tool-tip for that function. More importantly, you can click any function to invoke that function **directly on the selected node**.

Functions that do not require any parameters (for example, the reboot() function) will be executed immediately.

If the function requires any parameters, **Portal** will automatically prompt you for them.



For example, clicking the writePin() function will prompt you to enter the actual value to output on that GPIO pin.

You can either:

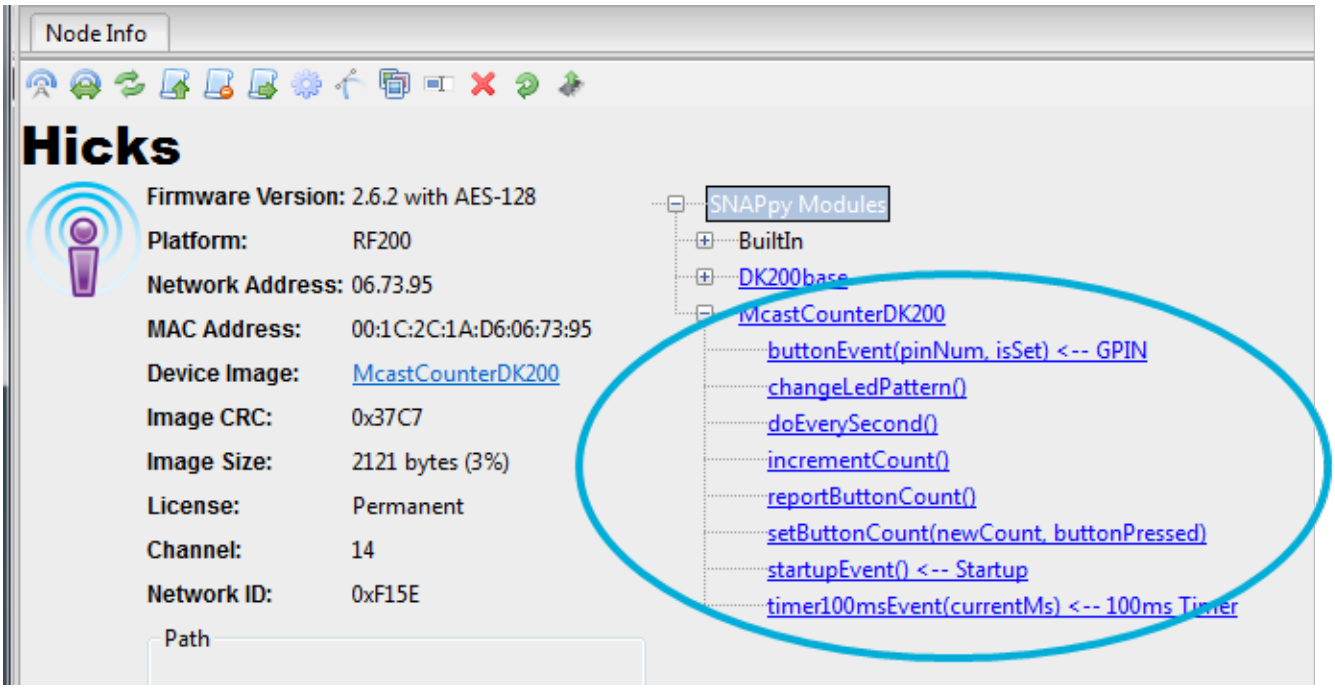
1) Enter a value (ex. pin = 1, isHigh = True) and press OK, or

2) You can press Cancel to abort the function invocation.

Don't forget that in the Python language 'True' and 'False' are case sensitive (with the first letter capitalized)

You can also expand the tree of functions defined by each **SNAPpy** source file.

As of **Portal** version 2.5, **Portal** tracks the history of the last eight entries into each field. The last value used will default into the field, and the previous seven values are available from a drop-down selection, making it easier to repeat common requests.



Here you can see the various functions defined in the “McastCounter.py” **SNAPpy** script.

Like the built-in functions, these can also be directly invoked by clicking them (and entering any needed parameters).

4. SNAP Node Views

The starting point for managing **SNAP** nodes is the **Node Views** tab. If this window is not already open, you can click **View**, then choose **Node Views window**. Alternatively, you can click the icon on the toolbar.

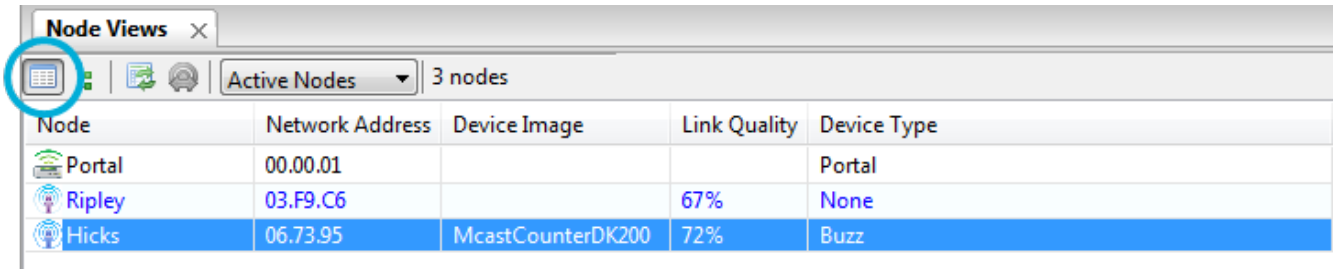
You will notice that the **Node Views** window has its own toolbar.

The **Node Views** tabbed window lets you look at your nodes in two different ways:

- **Report View**
- **Tree View**

These two options provide different views of the same network information.

Click the **Change to Report View** button in the **Node Views** toolbar (not the main toolbar). Several columns of information are shown about each node found.



The **Node** column shows an icon and a name for each node. Both the icon and name can be changed by the user. What you see here are just the default icons.


is used within **Portal** to generically represent any **SNAP** node.


is used within the **Portal** user interface to represent **Portal** itself.

(Remember, **Portal** is able to participate in the network as if it were a wireless node.)

The name for the node is assigned when the node is first discovered, and can come from three possible sources:

- As of version 2.1, it is possible to assign a name to a node
- If you don't give a node a name, it will use the name of any loaded script
- If no name given and no script loaded, **Portal** will use "Node" as a base name

If more than one node will have the same name based on this arrangement, a trailing numeric identifier is appended to the base name so that all nodes have unique names. This is a **Portal** requirement. You can rename your units but each name must be unique.

The **Network Address** column shows the three-byte Network Address for each node. The Network Address is simply the last three bytes of the node's MAC Address, and is not user definable.

The **Device Image** column shows the **SNAPpy** script/image loaded into the device. If there is no script loaded into the node, then this field is blank. Notice that this field tracks the currently loaded script. If you upload a different script into a node after it has been discovered/named, then this column can be different from the **Node** column.

The **Link Quality** column shows a snapshot of the radio receive level. By default, it is expressed as a percentage, with 0% representing the weakest possible signal and 100% representing a maximum strength signal. (You can configure the system preferences to display the direct signal strength in dBm in the system preferences. See **Preferences** for more details.)

It is important to understand three things about the displayed Link Quality:

1) Normally this field **is not** continuously refreshed – **Portal** does not "poll" nodes unless you tell it to. You *can* use the **Broadcast PING** button to update the Link quality fields of all active units. There is also a **Refresh** button you can click to force a refresh of a single node's Link Quality. (This button is on the **Node Info** toolbar.) Finally, there is a **Watch Nodes** button that essentially turns on automatic broadcast pings.

2) The value shown is based on the received signal strength of the most recent message *from **any** other wireless node*. It **does not** represent the signal strength between **Portal** and the node. (It **is not** an indication of the USB or RS232 quality between **Portal** and **Portal's** bridge node.)

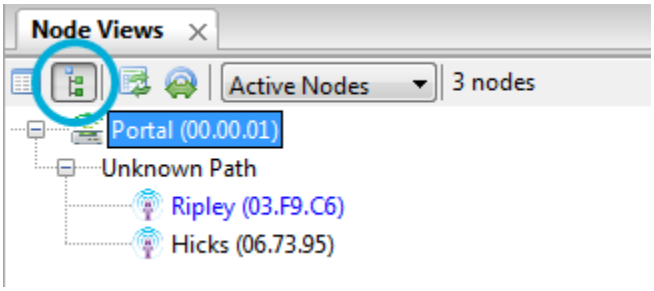
3) It is possible that *at the time* the **Link Quality** field was read from the unit, it had not yet received any radio messages from any other node. In this case, a value of 0 is reported. This does not mean the unit has a faulty radio; it simply has not done any radio communications yet. This is most often seen with the node that is acting as a "bridge" for **Portal**, because **Portal** can be interacting with this *directly attached* node *without* necessarily generating any **radio** traffic.

The **Device Type** column shows one of the non-volatile (NV) configuration parameters of the unit. **Device Type** is a second string label that can be applied to a node. Unlike the **Node Name**, this label does not have to be unique, and is often used to show what job or role a node is filling. This parameter can also be read by **SNAPpy** scripts, allowing a single script to act differently on different nodes by setting the **Device Type** of each node appropriately.

You may notice the text for one of the nodes is blue. This is **Portal's** way of showing which node it thinks is its bridge onto the **SNAP** network.

NOTE: If you change cabling on nodes it is possible to trick **Portal**. The blue highlighting is only accurate when you first discover your **SNAP** network. (You can use ***Network - New Configuration…*** to force a re-discovery.)

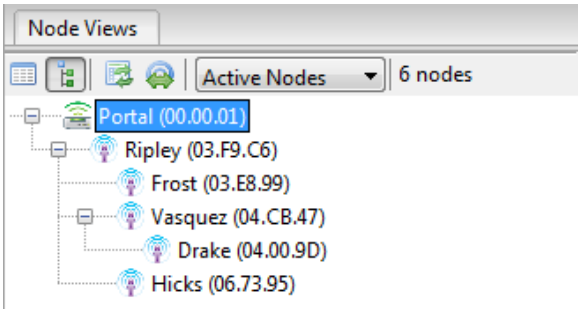
Clicking **Tree View** displays the next **Node View** option.



The tree view can show a hierarchical representation of how nodes reach each other. By default, **Portal** does not collect the path information to display a full tree view of a **SNAP** network. However the **Traceroute** button, described later, can assist in getting this information.

The **Watch Nodes** button has already been mentioned briefly. This button toggles **Portal's** "Node Watcher" (automatic periodic polling of nodes) feature. If you need to "see" nodes coming and going, you can turn on the Node Watcher instead of manually clicking **Broadcast PING** periodically. To accommodate increased network traffic, **Portal** extends its timeout for listening for ping replies by one second when **Broadcast Ping** is on. You can set this timeout (the Ping Response Spread) from the **Properties** window.

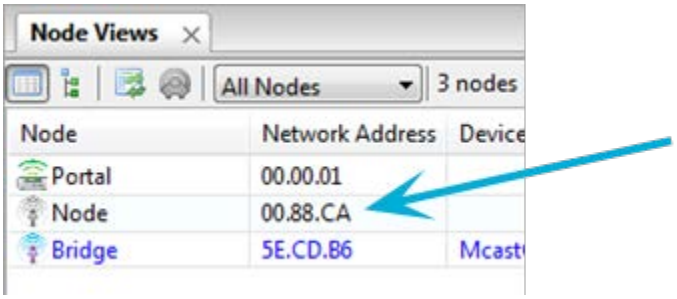
The **Traceroute** button enables **Portal** to collect path information about currently displayed nodes. This option is only available while in the tree view mode. Once **Portal** has collected the path information on how to reach each node the tree view is updated:



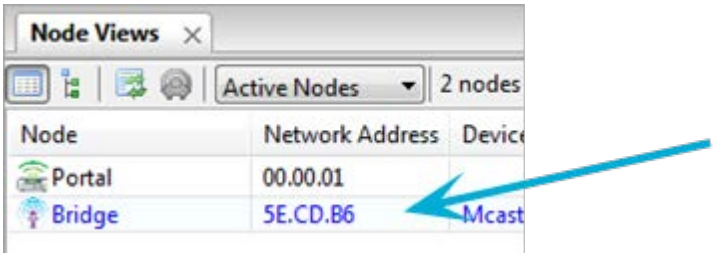
In this example network we can see that node 03.E8.99 routes packets through the bridge node, 03.F9.C6, to reach **Portal**, while node 04.00.9D is routed through node 04:CB.47 to reach **Portal**. As nodes are spread apart and it is necessary for nodes to "hop" through other nodes, the tree view can become more and more branched. This view enables you to see a snapshot of how nodes route messages through each other through the **SNAP** mesh network.

The last control in the **Node Views** toolbar is a drop-down selector box that lets you choose to see only the **Active Nodes** or **All Nodes**. Both views are identical as long as all nodes are online and communicating.

With the filter set to show **All Nodes**, when **Portal** sees that a unit has stopped responding, it is grayed-out in all three views.



With the filter set to show only **Active Nodes**, when **Portal** sees that a unit has stopped responding it is completely removed from all three views.



Screenshots are only shown for the **Report View**, but both views behave similarly.

5. SNAP Node Configuration

The configuration of an individual node, including the **Portal** node, can be viewed and changed from the **Node Info** tab.

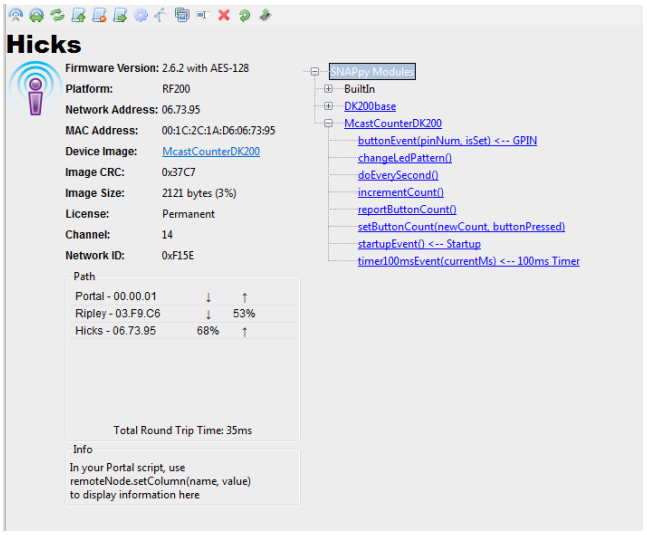
If the **Node Info** tab is already visible within Portal, selecting a node in the **Node Views** tab will display that specific node's info in the **Node Info** tab.

If the **Node Info** tab is not already visible, you can display it by clicking **View - Node Info Window** or by double-clicking the desired node in the **Node Views** tab.

Starting in the upper left-hand corner, the **Node Info** tab shows the:

- "Logical name" of the node
- Firmware (or software, for the **Portal** node) version and type, if special (Debug, AES-128)
- Some network configuration parameters
- Device Image and Image CRC for remote nodes, or Image Modified date for **Portal** images
- Image size information, indicating how much space is in use in the node for **SNAPpy** scripts
- License information (Permanent or Demo)
- Some more network configuration parameters

The left-hand side continues with **Path** and **Info** panes. The right-hand side shows a tree view of the callable scripted functions.



Network Configuration Parameters

There are four network configuration parameters: **Network Address**, **MAC Address**, **Channel**, and **Network ID**.

SNAP MAC addresses are standard 64-bit IEEE MAC addresses. MAC addresses are assigned to the node at the factory.

The **Network Address** is the three-byte identifier that is actually used by the **SNAP** wireless protocol. It matches the last three bytes of the eight-byte MAC address.

There are 16 allotted **Channels** for Synapse products. In the 2.4 GHz band used under the 802.15.4 specification, each channel maps to a particular specification channel. See the **SNAP Reference Manual** for more details on the frequencies used by Synapse products.

The **SNAP Network ID** can be thought of as a "logical channel." This 16-bit integer may be assigned any hexadecimal value from 0x0001 through 0xFFFE. Devices in a **SNAP** network must share both the same Channel and same Network ID in order to communicate. This allows multiple **SNAP** networks to share the same channel if required, although it is preferred to place independent networks on separate *physical* channels to reduce collisions.

Node Info – Tasks Pane

There are thirteen toolbar options in this pane. (Not all are active for the **Portal** node.)

- **Ping –** poll a node for connectivity.
- **Traceroute** – collect path information to the node.
- **Refresh** – poll a unit for the values shown in the Node Info tab.

- **Upload SNAPpy Image** – program a node with a new script. (This button is labeled **Change Portal Base File** for the **Portal** node.)
- **Erase SNAPpy Image** – remove previously loaded script. (This button is labeled **Clear Portal Base File** for the **Portal** node.)
- **Export SNAPpy Image** – export a **SNAPpy** image as a .SPY file.
- **Change Configuration Parameters** – more on this below.
- **Intercept Node Output** – divert the node's script output to **Portal**.
- **Change Icon** – substitute an alternate PNG graphic file as the node's icon.
- **Rename Node** – replace the node's current logical name.
- **Remove Node** – remove the node from the various **Node Views**.
- **Reboot Node** – reboot the node.
- **Upload SNAP Firmware** – send new **SNAP** firmware to the node via RPC calls. (This capability is only available nodes based on the AT128RFA1 platforms as of this release.)

Ping

Clicking this action causes **Portal** to make a quick connectivity test to the selected node. Observe the Event Log for confirmation of **Ping** activity.

The **Ping** action is not available for the **Portal** node (**Portal** cannot "ping" itself).

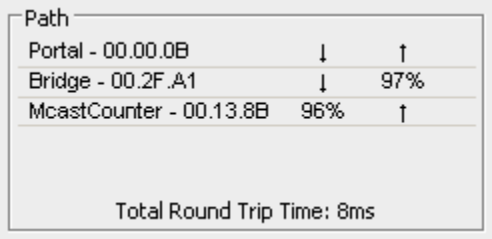
A **Broadcast Ping** command (seen by all nodes) is also available in the **Portal** toolbar, here:

Traceroute

This action sends a **SNAP** traceroute packet to the current node. A traceroute packet collects the **SNAP** network address and link quality of each node that it travels through to reach its destination. For example, a traceroute from **Portal**, at address 00.00.0B, to another node, at address 00.13.8B, might take the following path:

00.00.0B *to* 00.2F.A1 *to* 00.13.8B *back to* 00.2F.A1 *and finally back to* 00.00.0B

This path would be visualized in the **Path** pane as:



The arrows represent a hop in the path where no link quality information needed to be collected. The link between **Portal**, 00.00.0B, and the bridge node, 00.2F.A1, is a serial connection, which has no link quality. The 96% link quality shown in the example represents the link quality as received by node 00.13.8B when sent from 00.2F.A1. Another way to describe it is the link quality shown is what the listening node "heard" from the sending node.

Lastly, the *Total Round Trip Time* is the time it took for the bridge to send the traceroute request and then receive the response back. You might notice that the time varies between traceroute requests to the same node. This is due to external factors such as mesh network route discovery, network collisions in a crowded channel, random backoff, or waiting for another node to finish transmitting.

The **Traceroute** action is not available for the **Portal** node.

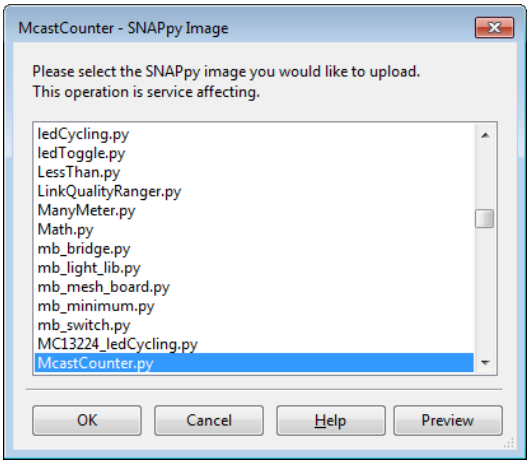
Refresh

Clicking this action will poll the node for the information shown on the **Node Info** tab. Check the **Event Log** for confirmation of **Refresh** activity.

The **Refresh** action is not available for the **Portal** node.

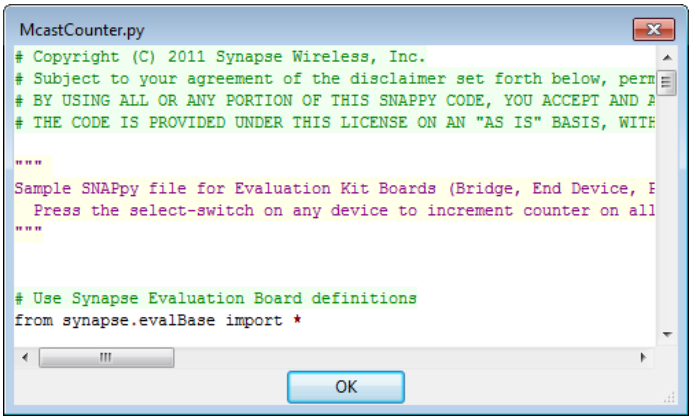
Upload SNAPpy Image

Clicking this action will display a dialog box allowing you to choose which script to upload into the node.



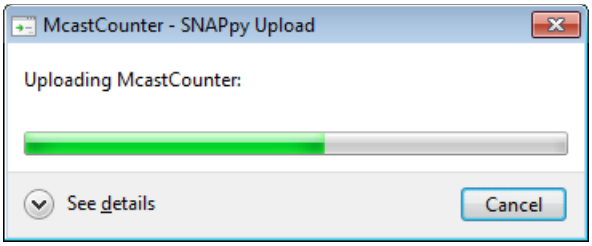
A node can only contain one script at a time, but the script can actually consist of multiple source files, using the Python "import" functionality.

If you are not sure which script you want, you can click the **Preview** button to take a quick look at the currently selected script. Click **OK** when you are done looking at it.



Once the script is chosen, **Portal** will erase any previous script, and then upload the new script to the node.

NOTE: What is uploaded is the compiled byte-code form of the script, not the actual source code.



When the entire script has been uploaded to the node **Portal** will automatically reboot the node, which in turn will cause the node to execute any script code contained within the HOOK_STARTUP event handler of that script.

If you click the name of a script within the **Node Info** window, an editor window containing that script opens.

For the **Portal** node, this action is named **Change Portal Base File**. The **Portal** node is not restricted to **SNAPpy** scripts, and can run any Python 2.6 script. Therefore the button opens a standard Windows dialog allowing you to browse to the Python script of your choice.

For more information about **SNAPpy** scripting, please consult the **SNAP Reference Guide**.

Erase SNAPpy Image

Clicking this action will erase any currently loaded (and running) script from the selected node.

NOTE: You do not have to manually erase a previous script to load a new one. **Portal** will do this for you. This button is so misbehaving scripts can be shut down while you work on making corrections to them.

For the **Portal** node, this action is named **Clear Portal Base File**.

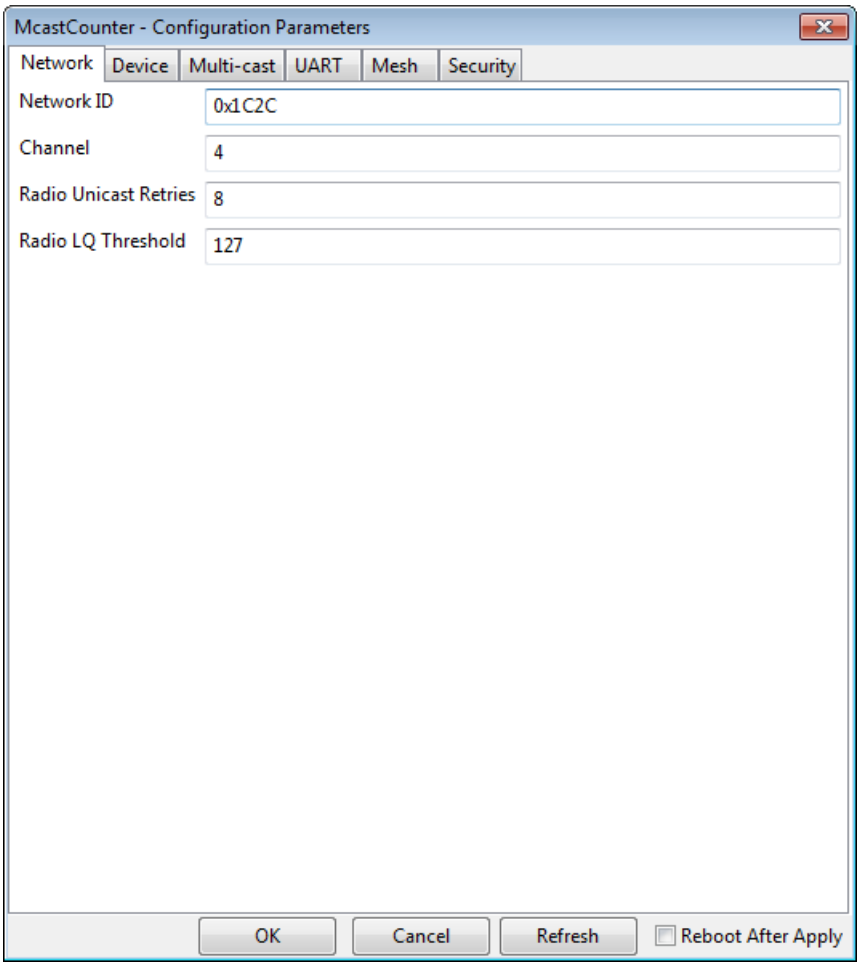
Export SNAPpy Image

This toolbar option allows you to export the **SNAPpy** script as a **SNAPpy** Export (.SPY) file. The .SPY files are used to save the current local copy of the script as a compiled script ready to upload to a node. For example, **SNAP Connect** cannot upload .py files because it requires the scripts to already be compiled. This option allows you to create the necessary .SPY file for **SNAP Connect** to upload to your nodes. Another use for the .SPY files is if you want to provide a script to another user of **Portal** without giving them the source code.

The **Export SNAPpy Image** action is not available for the **Portal** node, because **Portal** can run Python 2.6 scripts that are not valid in another node.

Change Configuration Parameters

Clicking this task will display the following (tabbed) dialog box:



From this dialog box, you can view and edit the Configuration Parameters of the current **SNAP** Node.

Configuration Parameters are also known as Non-Volatile (NV) Parameters, and they are described in detail in the **SNAP Reference Manual**.

NOTE: If you hover over any field in this dialog box, you will get a tool-tip containing that NV Parameter's description (and NV ID number).

You can see the different NV Parameters broken down by category by clicking each of the six named tabs within the dialog box.

The **Network** tab lists parameters relating to general radio operations.

The **Device** tab lists parameters relating to node "identity," and some hardware features. As of **Portal** release 2.5, the tab includes a breakout of the Feature Bits, indicating what each bit represents and allowing you to specify the numerical value to use by checking individual features you desire.

The **Multi-cast** tab lists parameters relating to multicast communications.

The **UART** tab lists parameters relating to the buffering of data received from the system's universal asynchronous receiver/transmitter(s) for serial communications.

The **Mesh** tab lists parameters relating to the mesh routing capabilities of the node.

Synapse nodes communicate with each other using mesh routing. This means that any unit can communicate with any other unit (they are all *logical* peers within the mesh), even though in some cases units that are out of radio range of each other might have to pass messages through intermediate nodes to do so. Only units that are within radio range of each other can communicate directly (*physical* peers).

The "route discovery" and "packet forwarding" necessary to make this work are all done automatically by the node.

There are several **Mesh Routing Timeouts**. The default values should cover most network topologies. Be careful if changing these values on remote nodes: you could make it impossible to communicate with the node remotely.

See the **SNAP Reference Manual** for more info about tuning these timeout parameters.

There are also several "non-timeout" parameters. Refer to the **SNAP Reference Manual** for information about using them.

The **Security** tab lists parameters relating to encryption. Specify an Encryption Type of 0 for no encryption, 1 for AES-128 encryption, or 2 for Basic encryption. If encryption is enabled, the Encryption Key must be exactly 16 characters long, and can be any combination of text string and hexadecimal values.

NOTE: Not all nodes support AES-128 encryption. This is determined by the **SNAP** firmware currently loaded into the node. However all nodes running firmware version 2.4 or newer will have Basic encryption available.

To change one or more of the editable fields, just type the replacement value(s) in before clicking **OK**. The new values are sent to the node and stored.

NOTE: While new values are immediately sent to the node and stored, any changes to configuration parameters will not take place until the node has been rebooted.

The **Change Configuration Parameters** action is not available for the **Portal** node, except for when the **Portal** node is connected to a **SNAP Connect** instance. Then a subset of relevant parameters are available for adjusting how **Portal** interacts with the network. If you have a network that includes more than one instance of **Portal**, the remote **Portal** node will not report useful information for the **Change Configuration Parameters** window.

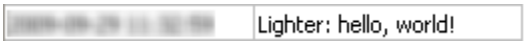
Intercept Node Output

Script output (from "print" statements) normally goes to the "data sink" specified by the script itself. For example, the script might be outputting text to serial port 0.

For testing purposes, it is sometimes handy to "intercept" the script output from the node (which might be remotely located), and display it within **Portal** instead.

Clicking this action will send the necessary commands to the node to accomplish this.

The intercepted text will appear in the **Portal Event Log**.



In the above example, the script did a print "hello, world!" The node name ("Lighter") was pre-pended by Portal, providing an easy way to see which node printed the text.

You must specify which types of messages you want your node to redirect to **Portal** using the **Preferences** window available from the **File** menu in **Portal**. You can have your nodes send STDOUT, error messages, both, or neither to the **Portal** log. If you have both the **Intercept STDOUT** and **Intercept ERROR** checkboxes unchecked on the **Preferences** window, no output from the node will be redirected to the **Portal** log.

The **Intercept Node Output** action is not available for the **Portal** node. Anything printed by **Portal** will always appear in the **Event Log**.

Change Icon

You can replace the default icon **Portal** uses to represent each node with your own 60x60 PNG file. This might be used to make it easier to track which nodes are serving what purpose. The graphic is only associated with the node while **Portal** is aware of the node. If you remove the node from the network, you will need to reassign your graphic icon to it. If you save a configuration that includes a node for which you have changed the icon, that icon will be restored when you reload that configuration data.

Rename Node

The default "logical name" for a node with no name provided by the user, and no script loaded, is "Node" (for the first one found) or "NodeX" (for subsequent nodes), where "X" is a number (starting at 2) that represents the order in which **Portal** discovered the nodes.

The **Rename Node** action allows you to choose your own name for a node, instead of using the default name assigned by **Portal**. This "given name" is stored in the node itself (in NV Parameter 8), and will automatically be reported by **Portal** (instead of "Node") if **Portal** re-discovers the node in the future.

Node names may not contain spaces, punctuation or any special characters other than underscore. You should not specify a node names longer than 64 characters.

Remove Node

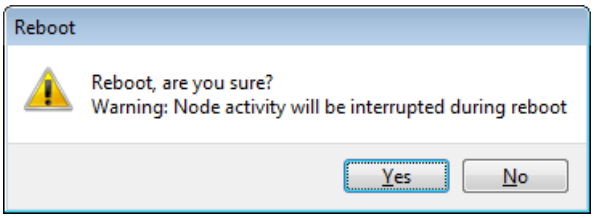
Clicking this action will remove the node from Portal, including removing the node from the **Node Views**.

Be aware that if the node really does still exist. It may be re-discovered by **Portal** on the next broadcast ping.

The **Remove Node** action is not available for the **Portal** node.

Reboot Node

Clicking this action will display the following prompt.



If you really do want to reboot (restart) the node, select **Yes**, otherwise select **No**.

The **Reboot Node** action is not available for the **Portal** node.

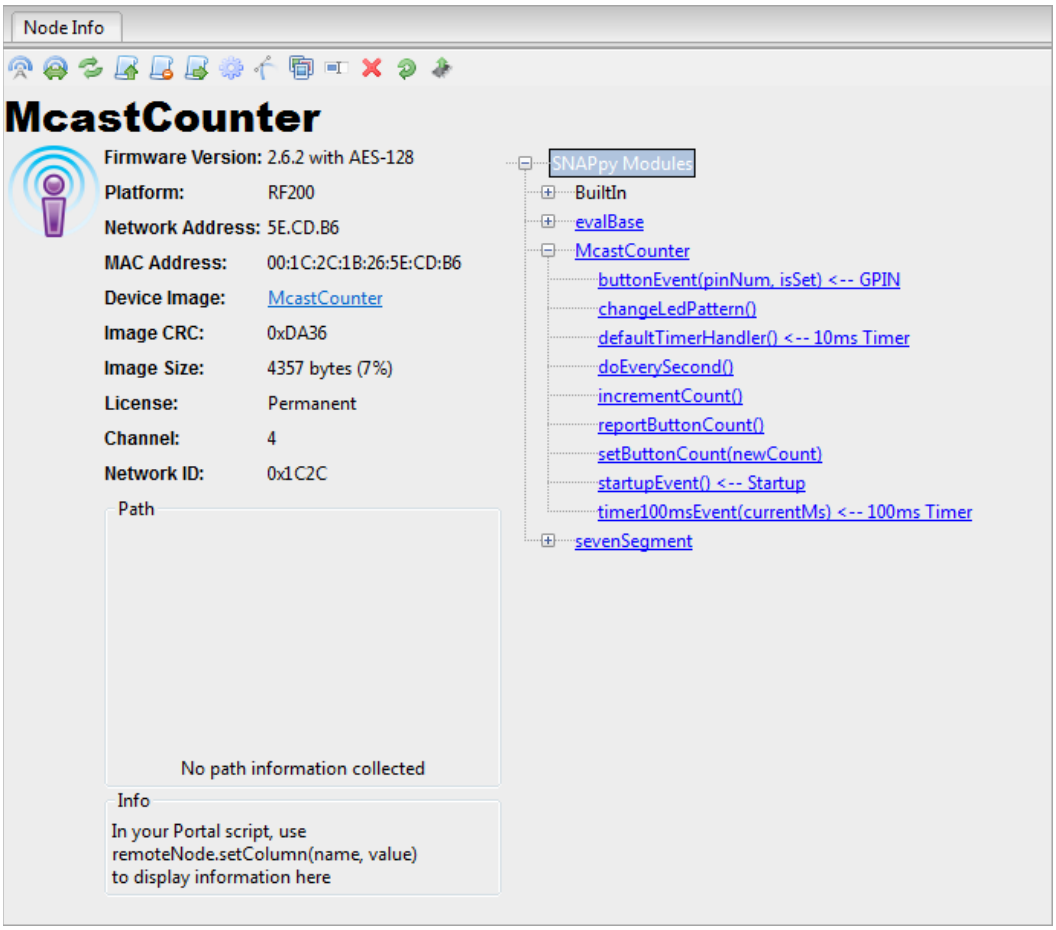
Upload SNAP Firmware

This toolbar button is a shortcut to the **Upgrade Over The Air…** feature that can also be found under the **Options** menu.

Refer to section **Firmware Updates via Remote Connection** for more information.

Node Info – “SNAPpy Scripts” Section

The right hand side of the **Node Info** tab shows a tree view of all the callable functions within that node.



The **BuiltIn** functions represent the core **SNAPpy** feature set, and are functions that can be called from **Portal** even when no user script has been uploaded into the node. These same functions are also always available for use *within* scripts.

Refer to the **SNAP Reference Guide** for details on these built-in **SNAPpy** functions.

The other branch(es) of the tree view show(s) the uploaded script’s name, with all of the callable functions implemented by that script. (If the script imports other scripts, each imported script’s functions will be shown under that script’s name in the tree. In this example, the McastCounter script imports both the evalBase and sevenSegment scripts.) Functions with names beginning with an underscore character are considered “private” by **SNAPpy**; they can be called by other functions in the script, but cannot be called from outside the node.

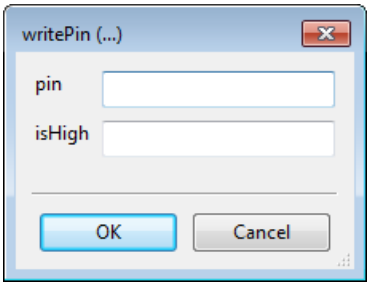
Hovering the mouse cursor over the links in any branch of the tree will display a short tool-tip on that function, if one has been defined.

NOTE: For this to be useful for the functions defined in *your* scripts, be sure to put a Python “docstring” as the second line of your function definitions. For example:

```
def mySuperFunction():
    '''This is the text that will appear as a tool-tip'''
    pass
```

Clicking any function name in the tree will cause the selected node to execute that function.

If the function takes parameters, a dialog box will prompt the user for the needed values.

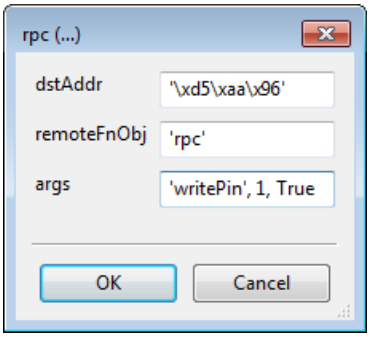


In the above example, clicking the `writePin()` function activated a dialog box asking for "pin" (which IO pin), and for a Boolean (True or False) value to make the pin high (True) or low (False). As of **Portal** version 2.5, **Portal** tracks the history of the last eight entries into each field. The last value used will default into the field, and the previous seven values are available from a drop-down selection, making it easier to repeat common requests.

Once you fill in the values, click the **OK** button to invoke the function *on the actual node*.

This can be very useful for learning the built-in functions, as well as for testing and debugging scripts.

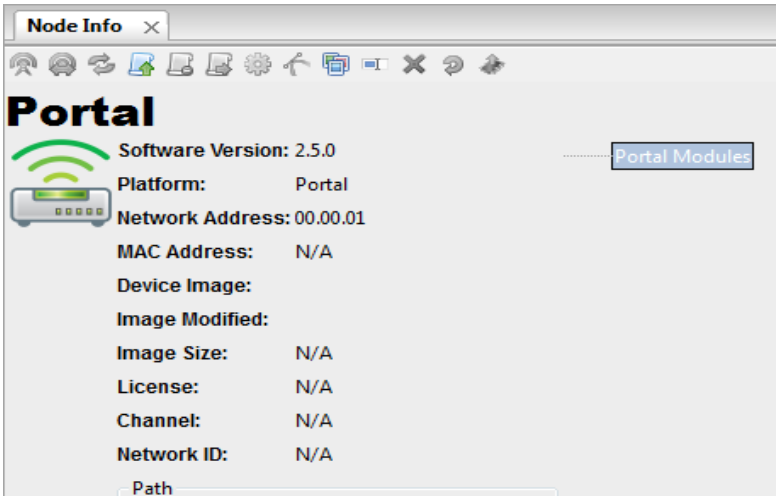
Some functions have a variable number of parameters, it may be necessary to include more than one parameter, separated by commas, in the args field of the window. For example, you can use the rpc() function from **Portal** to have one remote node use `rpc()` to request that a different node perform some function:



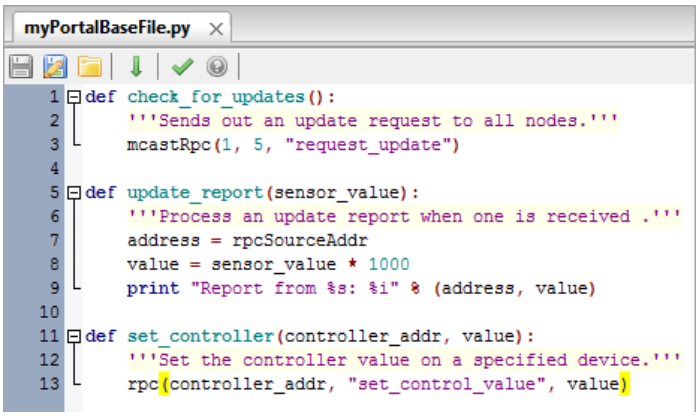
Portal is a Node Too

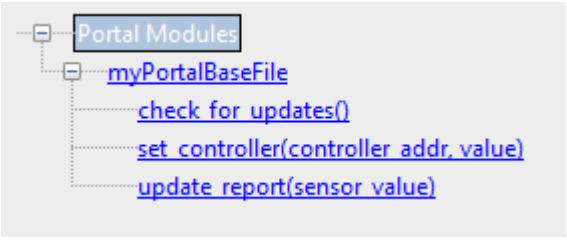
Most of this section has been focused on using **Portal** to interact with remote **SNAP** nodes. However, **Portal** itself can function as a node on the network. Many of the actions available in the toolbar work with the **Portal** node, with exceptions and clarifications noted above.

If you select **Portal** in the **Node Views** pane, the **Node Info** pane will show info similar to the following.



If a **Portal** script is loaded, you will see it in the **Portal Modules** function tree . You can invoke any functions defined in that script by clicking them, just as you can with the functions defined in the remote **SNAP** nodes.





NOTE: If your **Portal** script is broken up into *multiple source files* (i.e. a main script that imports one or more "helper scripts,") and you change one or more of the imported scripts, you may have to restart **Portal** to force it to recognize those changes.

6. Portal Tools

The main toolbar provides quick access to Portal's tabbed windows. These windows can be shown, hidden, or rearranged to suit a variety of operational modes. Some of the toolbar buttons invoke immediate commands instead of opening new tabbed windows.

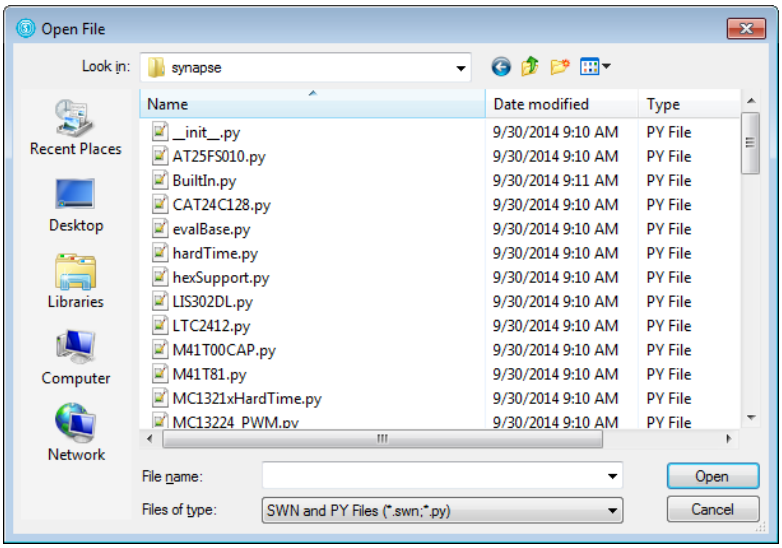
New Script

Clicking this button will open a new **Edit Window** containing a new (empty) source file.

The default name of the script will be "NewScriptX" where "X" is a trailing number to enforce uniqueness.

Open File

Clicking this button will display a file chooser dialog, from which you can select a previously saved source or **Portal** configuration file.



Source (.py) files can be **SNAPpy** scripts (these go into your **SNAP** nodes) or full Python files (these can be loaded into your Portal).

A **Portal** configuration specifies things like which nodes are available in the network and which Python script is loaded as the **Portal** base file. **Portal** configurations are saved in files having a .SWN (Synapse Wireless Network) extension, and are created using the **Save All** toolbar button, or the **Save Configuration As…** option from the **Network** menu.

Save All

This toolbar button will save any open modified scripts and initiate creation of an .SWN file containing all currently active **Portal** settings. The most recently saved configuration file will automatically be re-loaded by **Portal** when it restarts.

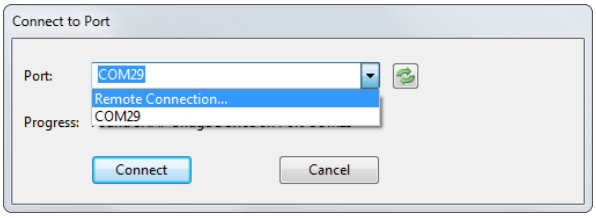
Connect Serial Port / Disconnect Serial Port

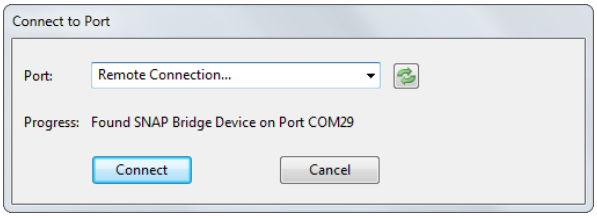
This modal toolbar button is used to connect and disconnect from the attached bridge node. Because the icon (along with its tool-tip) changes when you connect or disconnect, it also serves as a quick status indicator.

If your network does not seem to be responding, this is the first thing to check.

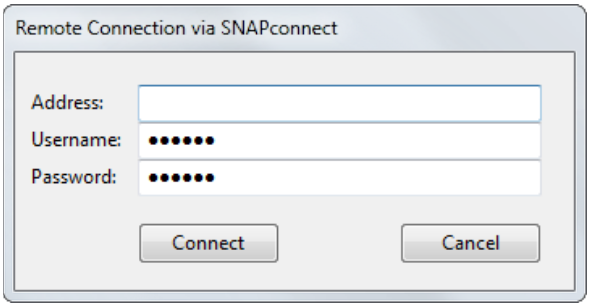
Although serial port (including USB serial) access is the most common case, the **Connect** menu also supports remote connection to a **SNAP Connect** application, assuming your PC has TCP/IP connectivity (for example, Ethernet or WiFi).

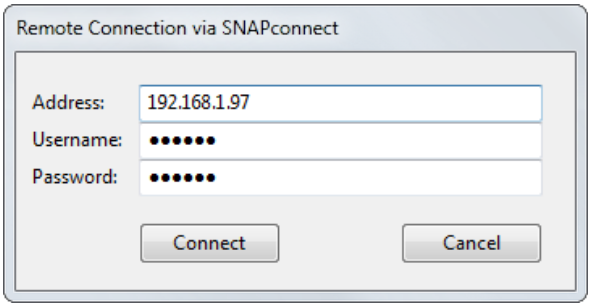
If you click the **Port** pull-down, you will see that there is an additional option, **Remote Connection…**





If you choose **Remote Connection…** after clicking the **Connect** button, you will see an additional options dialog (not displayed when using serial port connectivity).





The **SNAP**/TCP protocol is used with desktop PCs and embedded devices like the Synapse **SNAP Connect E10 Gateway** that are running the Python-specific **SNAP Connect** 3.x library.

> For more about this library, refer to the **SNAP Connect Python Package Manual**, and for more about the **SNAP Connect E10 Gateway**, refer to the **SNAP Connect E10 Users Guide**.

Enter the correct IP Address of the E10 (or PC) that is running **SNAP Connect** (and configured to accept TCP/IP connections). In the screenshot above, we are connecting to an E10 at IP Address 192.168.1.97. The IP Address can optionally be followed by a TCP/IP port number, using the usual ":" separator. The default port for **SNAP**/TCP is 48625, so typing "192.168.1.97" and "192.168.1.97:48625" are equivalent.

NOTE: You can only establish remote connections to **SNAP Connect** instances that have been configured to accept TCP/IP connections.

The **Username:** and **Password:** fields default to the same default values that **SNAP Connect** uses, namely **Username**="public" and **Password**="public". If you have not changed those in **SNAP Connect** then you don't have to enter anything into these two fields (it has already been done for you).

If you *have* changed the **SNAP Connect** settings from their default of "public", "public", then you must enter matching values (including uppercase/lowercase) into these fields before the connection will work.

If **Portal** is having remote connectivity problems, it will print the message…

```
Communications attempt failed, please check your settings (username, password, IP address)
```

…into the **Portal Event Log** as a reminder.

To take down this remote connection, just use the **Disconnect** button. You can then use the **Connect** button to establish a different serial or remote connection.

NOTE: Portal only supports one active connection at a time. You cannot be connected to a local serial bridge and to a remote E10 Gateway (or PC) simultaneously.

Broadcast Ping

Clicking this toolbar button will cause **Portal** to broadcast a special "answer if you hear me" message to all nodes. When the nodes answer, any nodes that **Portal** did not already know about will be individually queried for additional information (You can override this behavior under the **Preferences** menu). You might use this button if you just added one or more new nodes to your network. **Broadcast Ping** will only find nodes using the same channel and network ID as the bridge node.

Node Views

This tab provides a graphical display of the nodes in **Portal's** internal device directory. Internally, **Portal** indexes all nodes by network address. When a new network address is discovered, a corresponding icon in the **Node Map** will be created.

Deleting a node (select the node and click **Remove Node** from the **Node Info** tab) removes the device from Portal's internal database. This removes all knowledge of the device from **Portal**. If the device is subsequently re-discovered, usually via a ping, it will reappear in the **Node Map**.

Node Info

The **Node Info** tab provides node information, a mini-toolbar for common tasks, and **SNAPpy** script information about the currently selected node.

Event Log

The **Event Log** captures real-time event history for **Portal**. All network-affecting activity is recorded here. Log entries are time-stamped to a 1-second resolution.

Events Captured Include:

- Configuration save/load
- Script STDOUT and STDERR (print statements, errors)
- Status messages
- RPC communication messages

Command Line

The **Command Line** window provides interactive access to **Portal's** Python-based scripting engine. Using this tool, you can invoke the Device API on all discovered and active devices. This is useful for testing and debugging commands interactively before incorporating them into **Portal** scripts.

Most of what can be done through the command line can also be done through the other parts of the GUI. For example, you can type the command:

```
rpc('\x01\x02\x03', 'reboot')
```

… ***or*** you can just click on the node's `reboot()` function in the **Node Info** tree.

Within the **Command Line** pane, you can use the up and down cursor arrows to re-select previously executed commands. Position the cursor on a previously executed command and press the **<Enter>** key, and it will be re-displayed (but not executed yet) at the bottom of the **Command Line** window. You can then edit the recalled command (using the left and right cursor keys to move the cursor horizontally within the command). When the command line has been edited to your liking, press the **<Enter>** key again to execute it.

To re-execute the most recently executed command as-is (no changes), just press the keys **<Up>**, **<Enter>**, **<Enter>**. The first **<Enter>** chooses the command for editing, and the second **<Enter>** executes it.

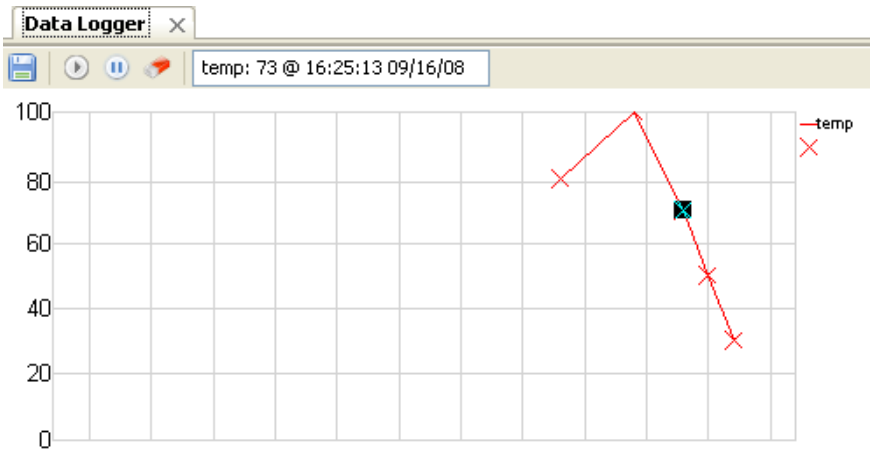
Within the **Command Line** window, you can type `shell.help()` to learn more of the tricks the **Command Line** window supports.

Data Logger

The Data Logger window gives a strip-chart view of a set of values plotted over time. It is very simple for scripts in nodes to add completely new categories of data. Just use a unique name, and you've created a new strip. You can either send an RPC function call from your node to **Portal's** address, invoking the `logData()` function with your desired data, or you can RPC your data back to some other function in **Portal**, where you can use Python code to evaluate and interpret your data before that function invokes `logData()`.

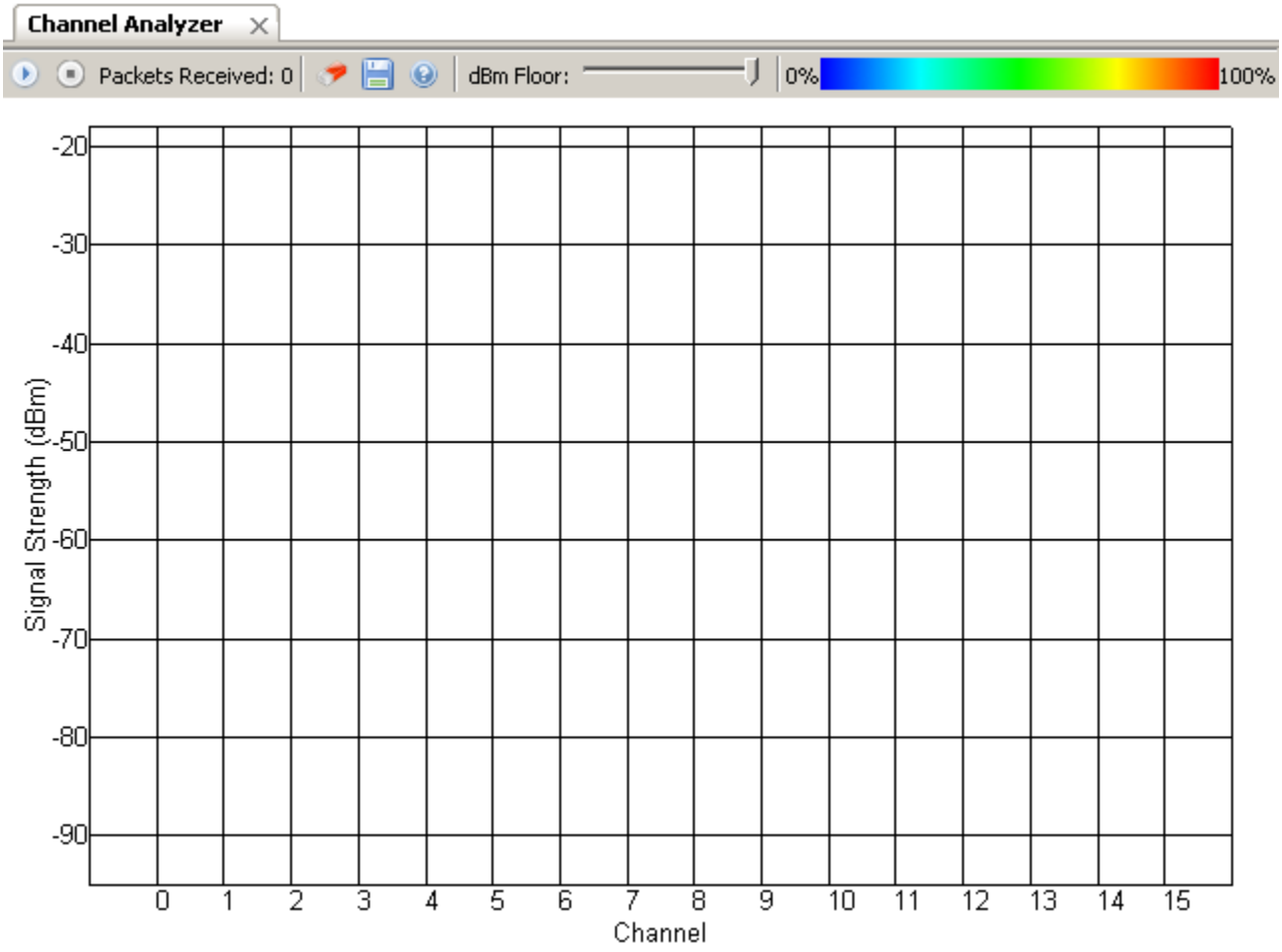
Scaling

The vertical axis is fixed at 100 units. Since a heterogeneous collection of data types may occupy the same plot, generic units are used. However, the `logData()` function accepts a scaling factor so that each value can map its full-scale range onto the vertical axis. The horizontal axis is a continuous timeline, with automatic scaling. You can plot multiple values, from the same or multiple nodes. If you select a data point in the graph, the window will tell you the point's data and time.



Channel Analyzer

Clicking this button will open the **Channel Analyzer** pane within Portal.



Performing a channel analysis (energy detection scan) consumes significant resources on your bridge node, so the function is only performed on demand. Performance of other functions within **Portal** (such as uploading scripts to a node or running a **Portal** script) may be significantly affected while the Channel Analyzer is running. It may be best to **Stop** the channel analyzer while you perform other functions, and then **Start** it again afterward.

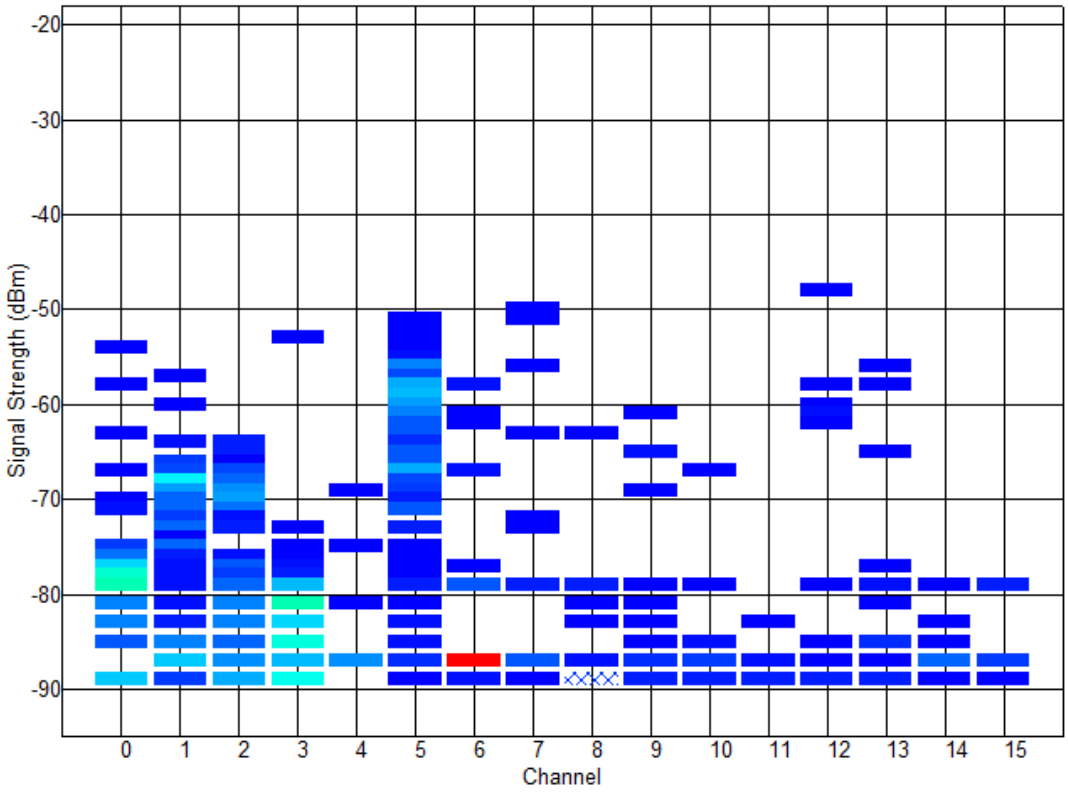
Like many panes, the **Channel Analyzer** pane has its own mini-toolbar.

- The **Start** button begins continuous channel scanning.
- Once started, the **Stop** button can be used to stop channel scanning.
- The **Packets Received** indicator show the number of energy detection packets received from the bridge.
- The **Clear Values** button can be used to erase any existing data and start a fresh plot.
- The **Save Graph** button can be used to save the currently graphed data as a graphic.
- The **Channel Help** button brings up a list of channels with an indication of how clear each channel is. This can help you choose the best channel for your nodes, based on the observed radio traffic in your environment.
- A **dBm Floor** slider lets you choose a cutoff value for the graphed data. This lets you ignore low-level signals and focus on stronger ones.

While channel scanning is in progress, the **dBm Floor** slider actually filters the incoming data (incoming readings below the current cutoff value will be discarded). When channel scanning is not in progress, this slider just controls what already captured data is visible, and thus controls what data is included in the **Channel Help** calculation. To include more or less data in the calculation, adjust the **dBm Floor** slider and then select again.

Here is a screenshot taken from a system while scanning was actually in progress.

NOTE: You may not have as much active traffic as in the example shown.



The colored bars in the chart provide three pieces of information simultaneously:

1) The horizontal position of each bar tells you the channel on which the signal was seen.
2) The vertical position tells you the signal strength seen for that signal.
3) The color of each bar tells you the number of times a signal of that level has been seen (in % form). A color legend near the top of the pane shows you the scaling.

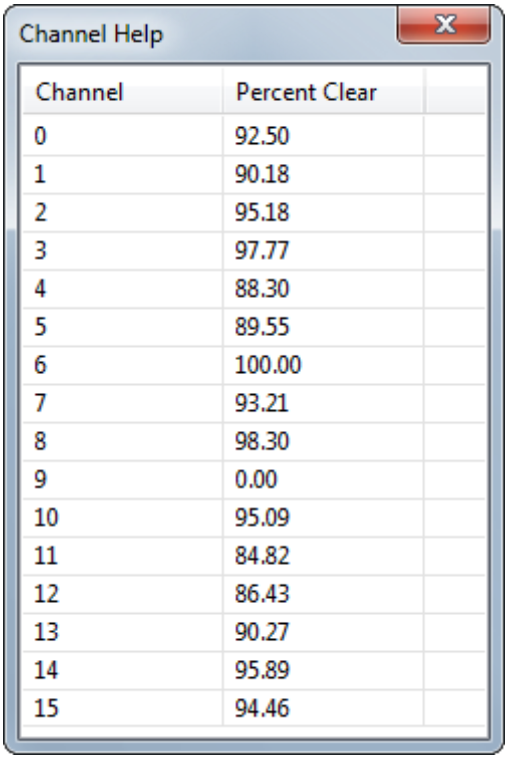
In the example above, you can see that the strongest signals have been detected on channel 12 (around -48 dBm), but that the most frequently seen signal is on channel 6 (at around -87 dBm).

NOTE: The Channel Analyzer takes "live" readings, and can only detect signals from nodes that are actually powered on, awake, and communicating at the time of the scan.

You can restart your graph at any time with the **Clear Values** button, and you can save as many graphs as you want using the **Save Graph** button.

Be sure to use the **Stop** button when you are done. This will give your bridge node more time to handle other network traffic.

Clicking **Channel Help** brings up a dialog similar to the following:



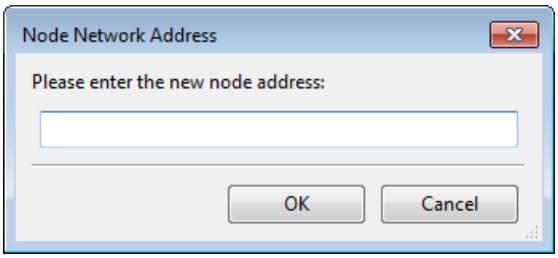
Channel	Percent Clear
0	92.50
1	90.18
2	95.18
3	97.77
4	88.30
5	89.55
6	100.00
7	93.21
8	98.30
9	0.00
10	95.09
11	84.82
12	86.43
13	90.27
14	95.89
15	94.46

From the example chart, you would expect that channel 6 would be the best choice for newly deployed nodes (as it is quietest right now), followed by channels 8 and 3. You can click column headers to sort the list to quickly find the weakest signal seen (most clear channel) or strongest signal seen (most crowded channel).

Add Node

Normally **Portal** only adds nodes to its internal database when it discovers them via actual **SNAP** communication. The **Add Node** function allows you to manually add a new node to Portal's node database. This can be handy if the node is not installed yet (or simply powered off), but you know it will be online later.

Clicking the **Add Node** button will display the following dialog box:



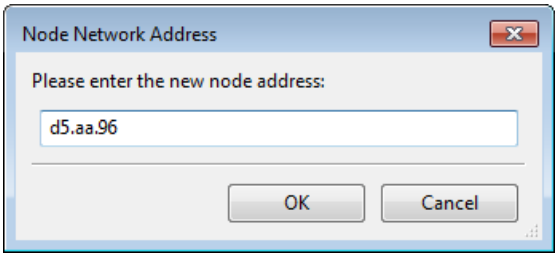
Enter the **SNAP** Address for the node, in any of the following formats:

- 12.34.56 (using . separators)
- \x12\x34\x45 (using the same format used in Python strings)
- 123456 (raw hexadecimal characters only)

All three of the above examples will result in the same three-byte **SNAP** address.

NOTE: Quotation marks are **not** used in any of the three formats. Also, you should just enter the 3-byte **SNAP** Address, not the entire 8-byte MAC Address.

Reminder: The **SNAP** address is the last 3 bytes of the MAC Address, which is usually printed on the node somewhere (look for a label).

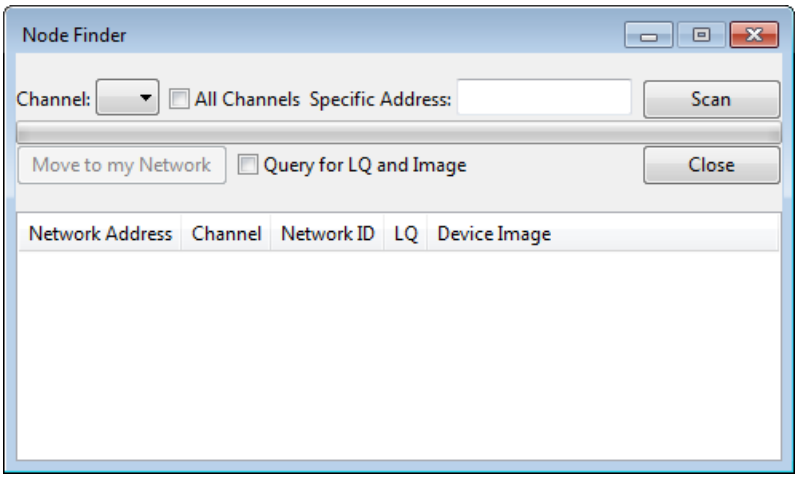


After entering the correct **SNAP** Address, click the **OK** button and the node will appear in Portal's **View Nodes** pane.

NOTE: You have to select **All Nodes** in the pull-down on this pane to see manually-entered nodes; by definition the **Active Nodes** filter only shows nodes that have *actually communicated to* Portal.

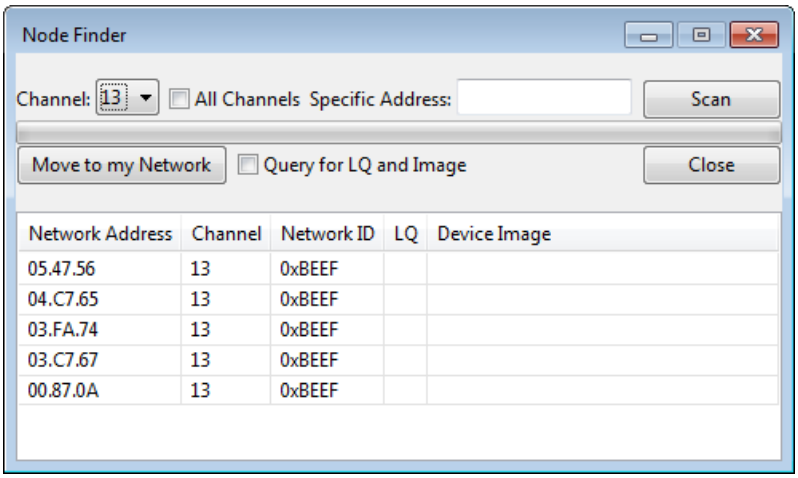
Find Nodes

Clicking this button will display the following interactive dialog:



First choose a specific channel to scan, or check the ***All Channels*** flag. Next click the ***Scan*** button.

Portal will switch the bridge node to a special "wildcard" Network ID value that all nodes will respond to. **Portal** will then broadcast pings on the specified channel (or all channels) and build up a list of all **SNAP** nodes that respond.



NOTE: This is an active scan, and **Portal** can only find nodes that are *powered on*, and *awake*. If you check the **Query for LQ and Image** checkbox, **Portal** will query each found node for its link quality and to determine what code image the node contains when it completes its node search.

In the example shown here, five nodes were found on Channel 13, on three separate networks.

Because the broadcast ping process uses (unacknowledged) broadcast messages, some nodes may not respond. You may have to hit the ***Scan*** button more than once to find the node you are looking for.

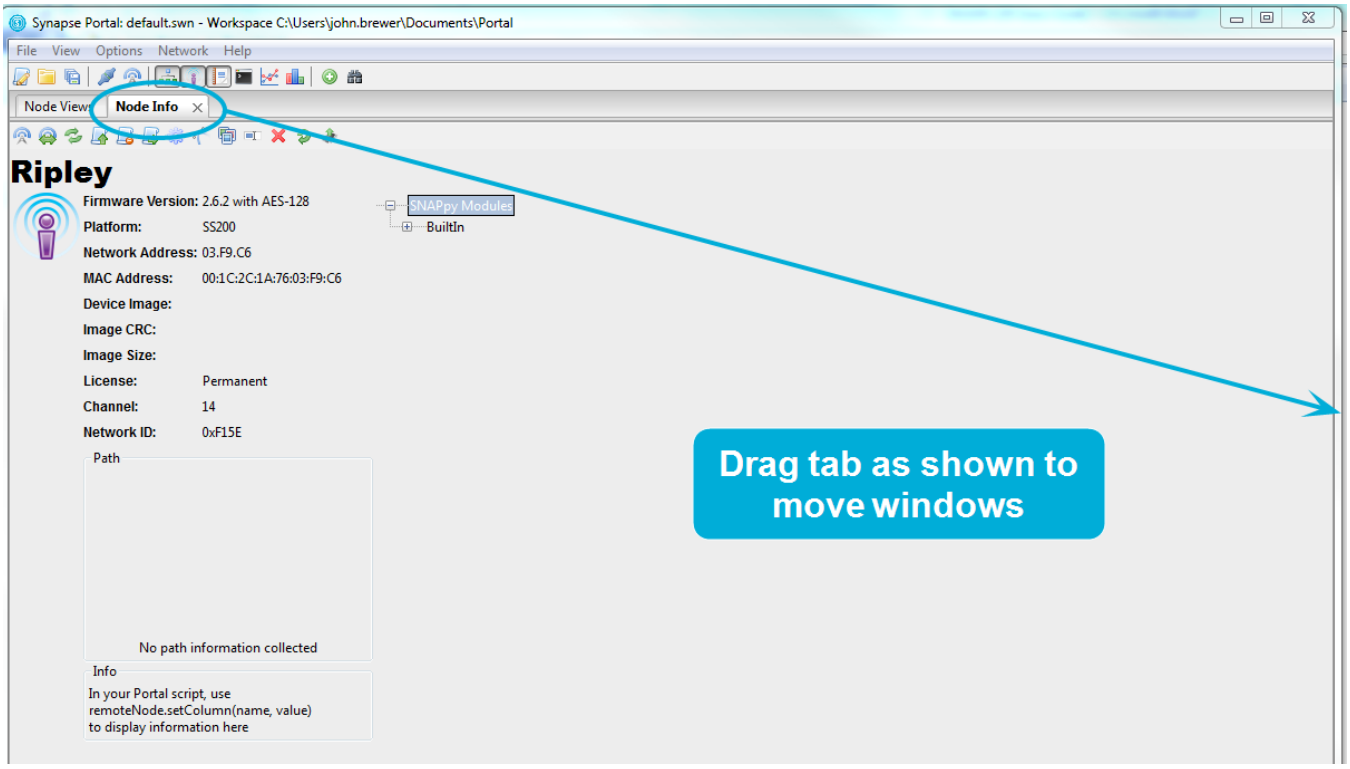
If the ***Node Finder*** locates a node on a different channel or using a different network ID than Portal's bridge node is using, you can select the node within the list and click the ***Move to my Network*** button. **Portal** will then command the node to change channel and/or network ID as needed.

Press the ***Close*** button to close this dialog and resume normal **Portal** operations.

If you are looking for a specific node with a known network address, you can use the ***Specific Address*** textbox to enter its network address. This allows you to search for that specific node and move it to your current network if needed, instead of getting a list of all nodes.

Rearranging Windows

The tabbed windows in **Portal** can be dragged and repositioned on the screen. To do this, press and hold the left mouse button while the cursor is positioned over the tab label you want to move. While holding the button down, drag the tab until you see a light blue “shadow” indicating a possible new position for the window. When you’ve found a suitable new position, release the mouse button and the move will be complete.



Window panes may be resized by clicking and dragging the horizontal and vertical borders separating them. If you have made changes to your window placement and sizing and wish to have **Portal** return to its default layout, select **View - Rearrange to Default View** and you will be returned to a view with the **Node Views** window on the left, the **Node Info** window on the right, and the ***Event Log*** window across the bottom of the screen.

7. Built-in Editor

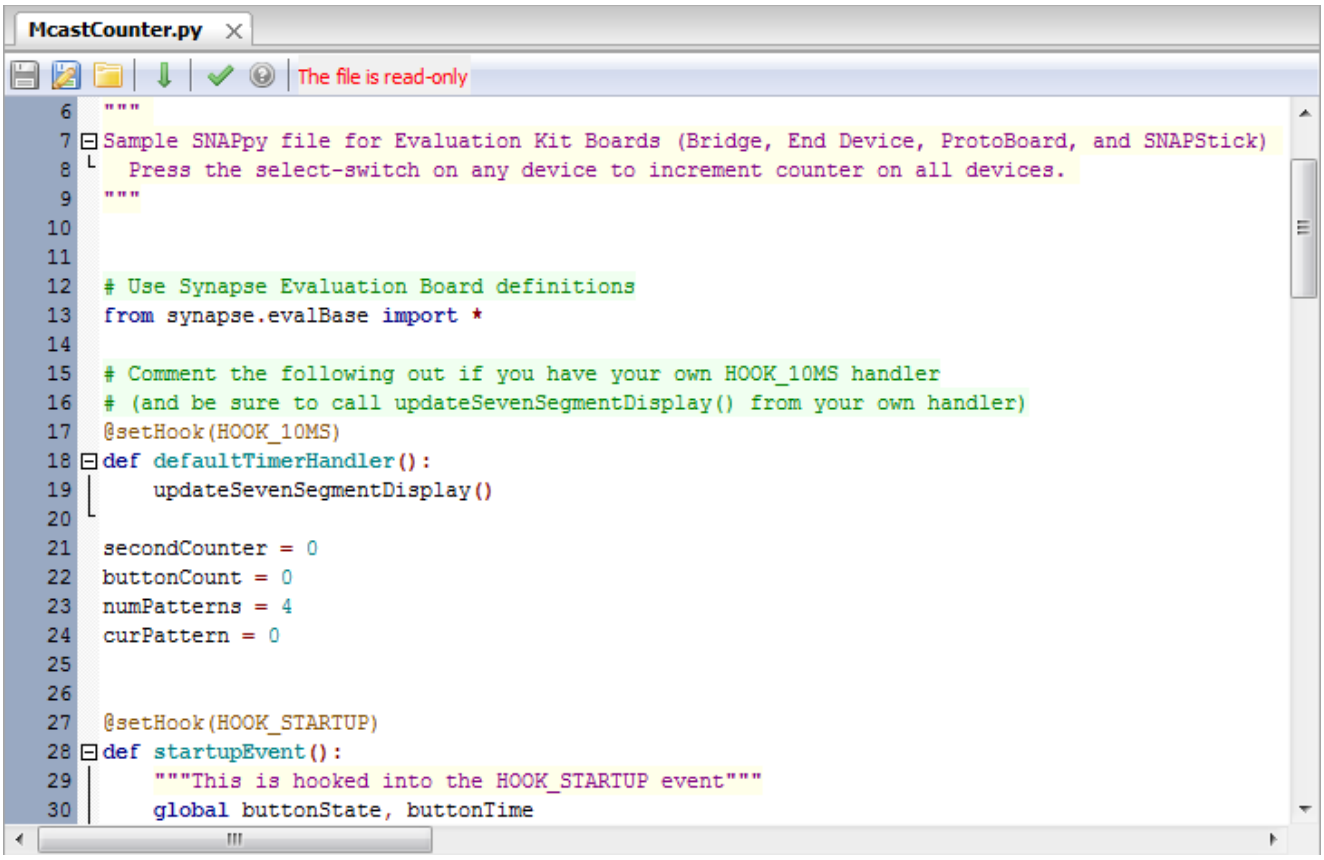
Portal includes a built-in code editor, making it an all-in-one solution. The code editor includes useful features such as line numbering, automatic indentation, code folding, and text auto completion[1], making it easy to develop scripts that meet your needs.

NOTE: If you already have a favorite editor, you can use it instead.

There are several ways to open an editor pane:

- Click on a script name from the **Node Info** pane
- Use the ***Open File*** button on the main toolbar
- Use the ***New Script*** button on the main toolbar

Regardless of how you opened it, all the edit windows work the same way.



A mini-toolbar provides buttons to ***Save***, ***Save As***, ***Open File***, ***Find Next***, (show path), and ***Test SNAPpy Script***.

Save

The ***Save*** button saves the script shown in the edit window. If the script has never been saved before, this button will be disabled (greyed-out), and you will have to use the ***Save As*** button instead.

[1] Auto completion of text requires that the synapse.BuiltIn.py file be present in your scripts directory. This file is present when Portal is installed, and you should not move, delete or rename it.

Save As

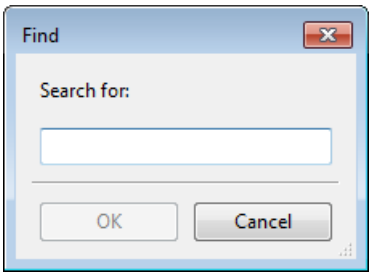
The ***Save As*** button is used when saving a script for the first time (so you can give it a name), or when you want to save a copy of an existing script to disk under a new name. Clicking this button will cause a file dialog to appear, allowing you to choose the directory and filename to be used for the newly created file.

Open File

The ***Open File*** button can be used to change the file being edited in the text window below the toolbar. Clicking this button will cause a file dialog to appear, allowing you to choose the directory and file within that directory to load into the edit window.

Find

The **Find** function does not have a toolbar button. Like most text editors, **Portal** uses Ctrl-F as the keyboard shortcut for **Find**. Typing Ctrl-F will display a dialog box asking what text to search for.



If you highlight some text within the text file before typing Ctrl-F, that text will automatically be filled into the dialog box as the default value.

Find Next

The ***Find Next*** button is used to continue the text search.

NOTE: Function key [F3] is an alias for the ***Find Next*** button.

Clicking the ***Find Next*** button (or pressing [F3]) will advance to the next instance of the search text, and highlight it in the edit window below.

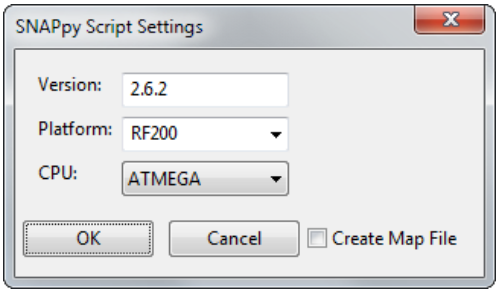
If you click this button, or press the [F3] key without ever doing an initial **Find** (Ctrl-F), **Portal** will automatically display the **Search for:** dialog box.

(show path)

Hovering the mouse over this question mark just brings up the full filename (including directory path) of the file in the edit window as a tool-tip (handy when you want to confirm you are editing the correct source file).

Test SNAPpy Script

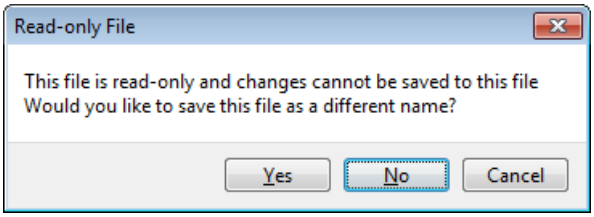
The ***Test SNAPpy Script*** button lets you validate **SNAPpy** scripts before you try to upload them into a node. Starting in version 2.2 the platform and version can affect how a script is compiled for a node. When you click the button to compile the script, you will be prompted for the platform and version appropriate for your target **SNAP** node.



For the version, enter the firmware version for your node (e.g., 2.6.2). Select the platform and the appropriate CPU. (The platform selected will control which code is compiled into your script. It matches the Platform specified for the node. If you exclusively use the "default" platforms specified by Synapse Wireless, the CPU will automatically default for you. Refer to the platform selections listed in the platforms.py file included in the **SNAPpy**Images\synapse directory for examples of the default platform values.)

You will notice a **Create Map File** checkbox next to the **Cancel** button. If you select this before clicking the **OK** button, an additional diagnostic file will be created that has the same base filename as the **SNAPpy** script, but has an extension of ".map". (For example, MCastCounter.py will result in a map file named MCastCounter.map). This file will automatically be opened in a new editor window so that you may review it.

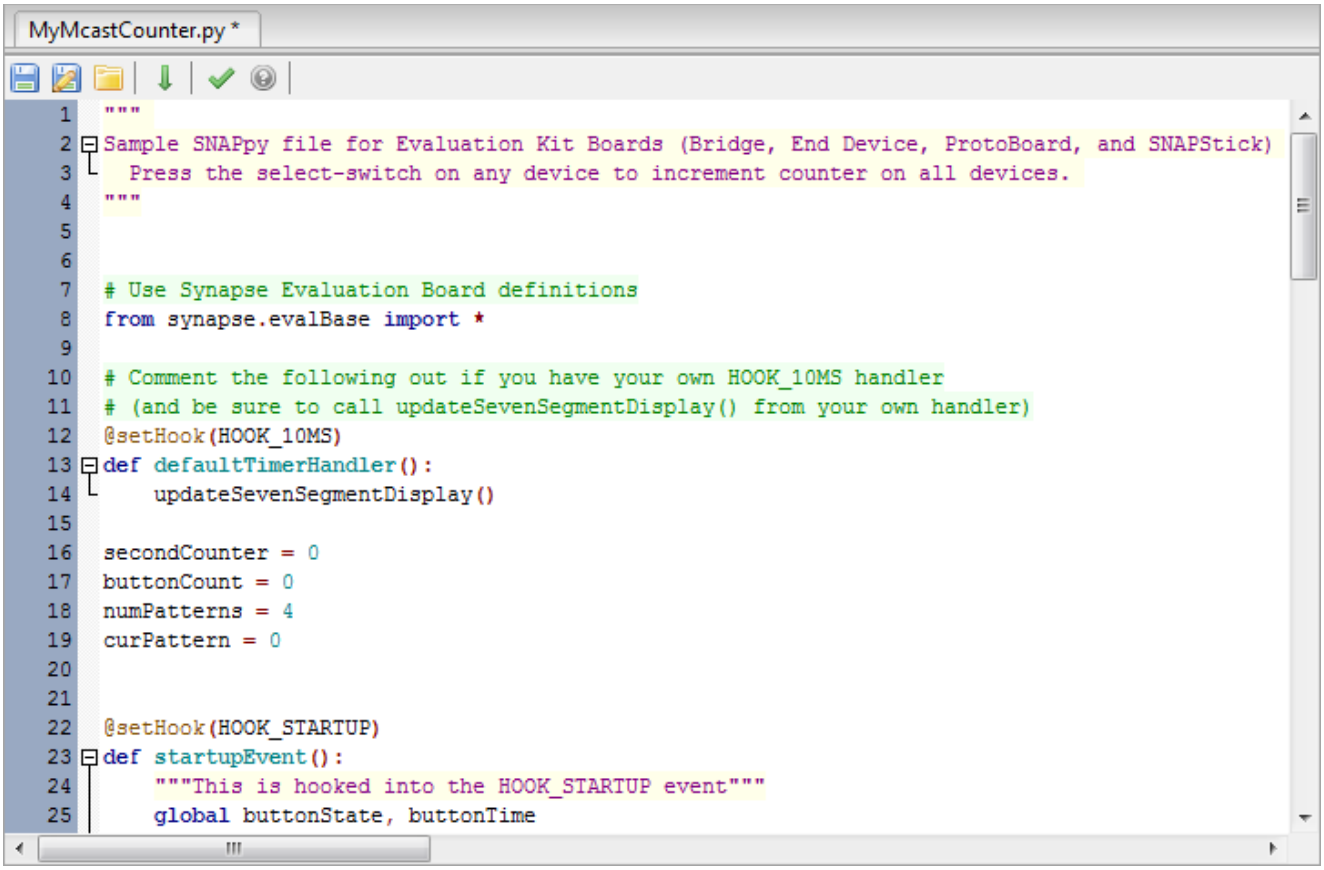
If you have opened a read-only file, a message on the toolbar will tell you so. You can perform a ***Save As*** to create a separate editable copy. This lets you start with an existing demo script and make your own custom variations. If you *try* to type characters into the edit window of a read-only file, **Portal** volunteers to save the file to a writable copy for you.



If you did not mean to be editing the read-only file, just click on ***No***.

If you *do* want to edit a copy of the original read-only file, click on ***Yes***.

The usual file dialog box will allow you to specify the new file name. The edit window will automatically switch to editing the newly created file.



Be sure to ***Save*** your edits when you are done. Your new script can be uploaded into one or more **SNAP** nodes using the **Upload SNAPpy Image** button in the **Node Info** window.

Keyboard Shortcuts

There are several keyboard shortcuts available in the Code Editor window:

- Ctrl-A — Select all text in the window
- Ctrl-B — Increase font point size for the current window
- Ctrl-C — Copy selected text to the clipboard
- Ctrl-D — Duplicate the current line
- Ctrl-F — Invoke the Find dialog. (Once a search has executed, F3 repeats the search for the next instance.)
- Ctrl-L — Delete the current line
- Ctrl-N — Decrease font point size for the current window
- Ctrl-S — Save the current code file
- Ctrl-T — Swap the current line with the previous line
- Ctrl-V — Paste from the clipboard
- Ctrl-X — Cut to the clipboard
- Ctrl-Y — Redo the last undo
- Ctrl-Z — Undo the last action
- Ctrl-[— Move cursor to the beginning of the previous (or current) paragraph
- Ctrl-] — Move cursor to the beginning of the next paragraph
- Ctrl-Home — Move cursor to the beginning of the file
- Ctrl-End — Move cursor to the end of the file
- Ctrl-Delete — Delete from the cursor to the beginning of the next word

- Ctrl-Shift-Delete — Delete from the cursor to the end of the line
- Ctrl-Right Arrow — Move cursor to the beginning of the next word
- Ctrl-Left Arrow — Move cursor to the beginning of the previous (or current) word
- Home — Move cursor to the beginning of the current line
- End — Move cursor to the end of the current line
- Alt-Home — Folds or Unfolds code
- Tab — When multiple lines (or sections thereof) are selected, increases indentation level for the selected lines
- Shift-Tab — When multiple lines (or sections thereof) are selected, decreases indentation level for the selected lines

Additionally, the following keyboard shortcuts work in every pane in **Portal**, as long as that pane has multiple tabs opened:

- Ctrl-Tab —Select the next tabbed window in the same pane
- Ctrl-Shift-Tab — Select the previous tabbed window in the same pane
- Ctrl-Page Up — Select the next tabbed window in the same pane
- Ctrl-Page Down — Select the previous tabbed window in the same pane

8. Firmware Updates via Direct Connection

The Firmware Version is shown in the **Node Info** pane for each **SNAP** node in your network. Upgrading your devices to a newer version of **SNAP** is easy with Portal.

NOTE: Direct Connect Upgrades use UART1 of the **SNAP** Engine (on engines with more than one UART). This means a connection to the second serial port (usually RS-232) of the device being upgraded is required. The *SN132 SNAP Stick* carrier does not support reprogramming. You must move the *SNAP Engine* from the *SN132 SNAP Stick* to some other board (like the *SN171 Proto Board*) in order to upgrade that *SNAP Engine*'s firmware.

Obtaining Firmware

Each new release of **Portal** includes the most current version of the **SNAP** firmware *at the time of that* **Portal** *release*. Intermediate "firmware only" updates may also be available on the *Synapse Support Web Site*:

http://forums.synapse-wireless.com/

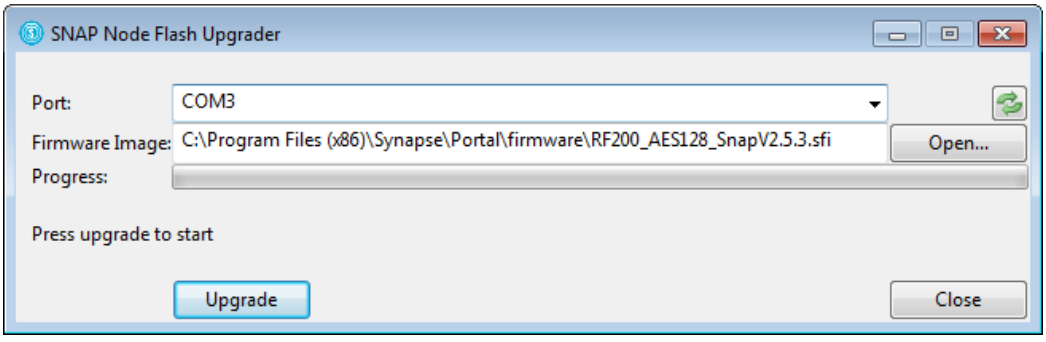
Look in the ***Latest Releases*** thread on the forum, under the ***Software Releases*** topic. If you download new firmware, put it in the "Program Files\Portal\firmware" subdirectory for ease of use.

Installing new Firmware

First make sure you have a **serial** connection to the device you are upgrading. On the SN163 Evaluation Kit board, this requires a connection to the DE9 connector as the USB connection on that board connects to UART0, which is not supported for firmware upgrades. The SNAPstick 132 USB SNAP Engine carrier board does not support serial firmware updates; it provides no means to reset the node. Other Synapse demonstration boards with USB connections do allow serial firmware updates.

Note: Firmware updates always occur on UART1 on any device that has more than one UART available, even if you have set UART0 as your default UART, or disabled UART1 using your node's feature bits.

From Portal's ***Options*** menu, select ***Firmware Upgrade…*** The following dialog box will display:



Portal will list the available COM ports on your system under the ***Port*** dropdown. Manually select the correct COM port to which that your **SNAP** node is connected. (You may need to click the  ***Scan*** button for **Portal** to locate your serial connection.)

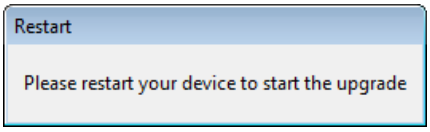
Select the Firmware Image to be used for the upgrade. Starting with version 2.1, there are now **four types** of firmware images from which to choose:

- Firmware images with "debug" in the title have extra error checks to help debug user scripts and configurations. These extra diagnostic checks come at the expense of a slight decrease in speed and **SNAPpy** image space.
- Firmware images without "debug" in the title do not have these extra error checks, which allow them to run faster, and have more room for **SNAPpy** images.
- Firmware images with AES128 in their name have built-in support for AES-128 over-the-air encryption. These images are not included with Portal, but can be ordered separately from Synapse. (There is an additional charge for AES-128 support.)
- Firmware images without AES in the title do not support AES-128 encryption.

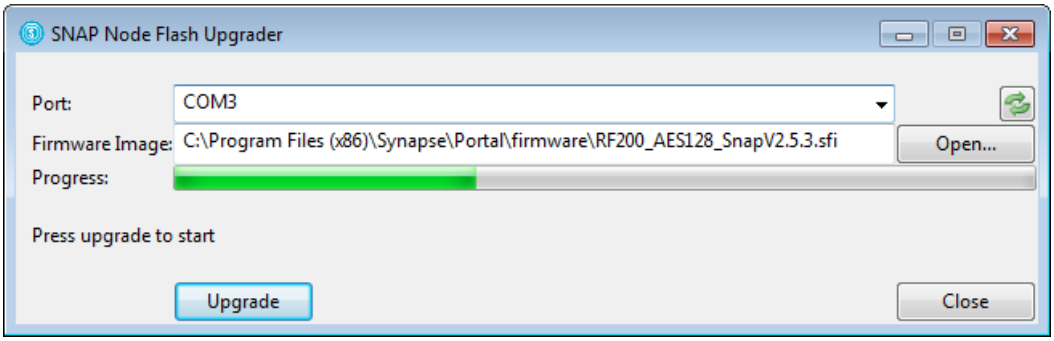
The firmware image names also indicate the version of that firmware. For example, 2.4.10 is newer firmware than 2.4.6, and both are newer than 2.2.16.

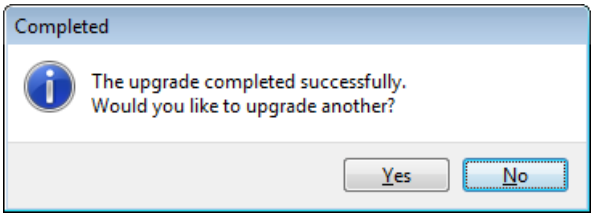
After choosing the correct firmware image to load, press the ***Upgrade*** button to begin.

Follow the instructions given in the dialog, which may have you restart (power-cycle or reset) your device to complete the process.



New firmware will be sent to the **SNAP** node over the serial connection.





Choose ***Yes*** to download the same image into another **SNAP** node, or ***No*** to return to Portal.

Troubleshooting

There is a possibility of interruption (power, serial connection, etc.) in the midst of the upgrade process. This may leave the **SNAP** node in a state where the **SNAP** application firmware is unusable. However, upgrade is still possible, since the **SNAP** boot-loader firmware is protected in a "locked" region of the node. Simply restart the upgrade process. This time you'll need to *manually select the appropriate COM port* for RS232 communication, since **Portal** won't be able to detect a running *SNAP Engine* on the serial port.

9. Firmware Updates via Remote Connection

The previous section covered the most common case – upgrading a node that is directly connected to **Portal**. Some **SNAP** platforms can also be upgraded remotely (over-the-air, AKA "OTA").

The ability of a particular platform to support OTA firmware update depends not only on what hardware it is based on, but in some cases also which version of **SNAP** firmware it is already running.

At the time of this writing, the following versions of **SNAP** support OTA firmware update:

- **SNAP** on ATMEL ATmega128RFA1 (2.4.32 or higher)
- **SNAP** on Synapse RF200 (2.4.32 or higher)
- **SNAP** on Synapse SM200 (2.4.32 or higher)
- **SNAP** on Synapse SM220 (any)
- **SNAP** on Synapse RF266 (2.4.32 or higher)
- **SNAP** on Synapse SS200 (2.4.32 or higher)

The platforms that gained OTA capability in a particular version must already be running that version in order to be upgraded remotely. So for example, version 2.4.32 must be loaded via a direct connection into an RF200, but then it will be able to be wirelessly upgraded to a future version.

NOTE: The OTA firmware upgrade process uses the "**SNAPpy** script area" as a temporary holding area for the new version of firmware. This means that performing an OTA firmware update **will erase** any existing **SNAPpy** script that may be residing in the node, and you will have to re-upload that script after the upgrade process is completed.

Obtaining Firmware

Refer to the previous section of this document. The firmware images used for OTA firmware update are identical to the ones used for direct serial update.

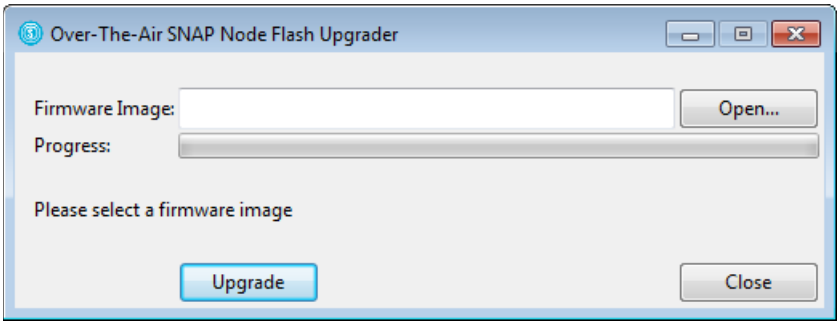
Installing new Firmware (One Node at a time)

First make sure you have connectivity (either wireless or direct serial) to the node you wish to upgrade.

Second, make sure it is selected in the **Node Views** pane, and perform a **Refresh Info**  .

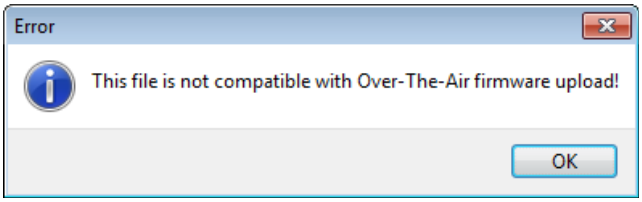
Third, go to the **Options** menu and choose **Upgrade Over The Air…**

You should get a dialog box like the following:



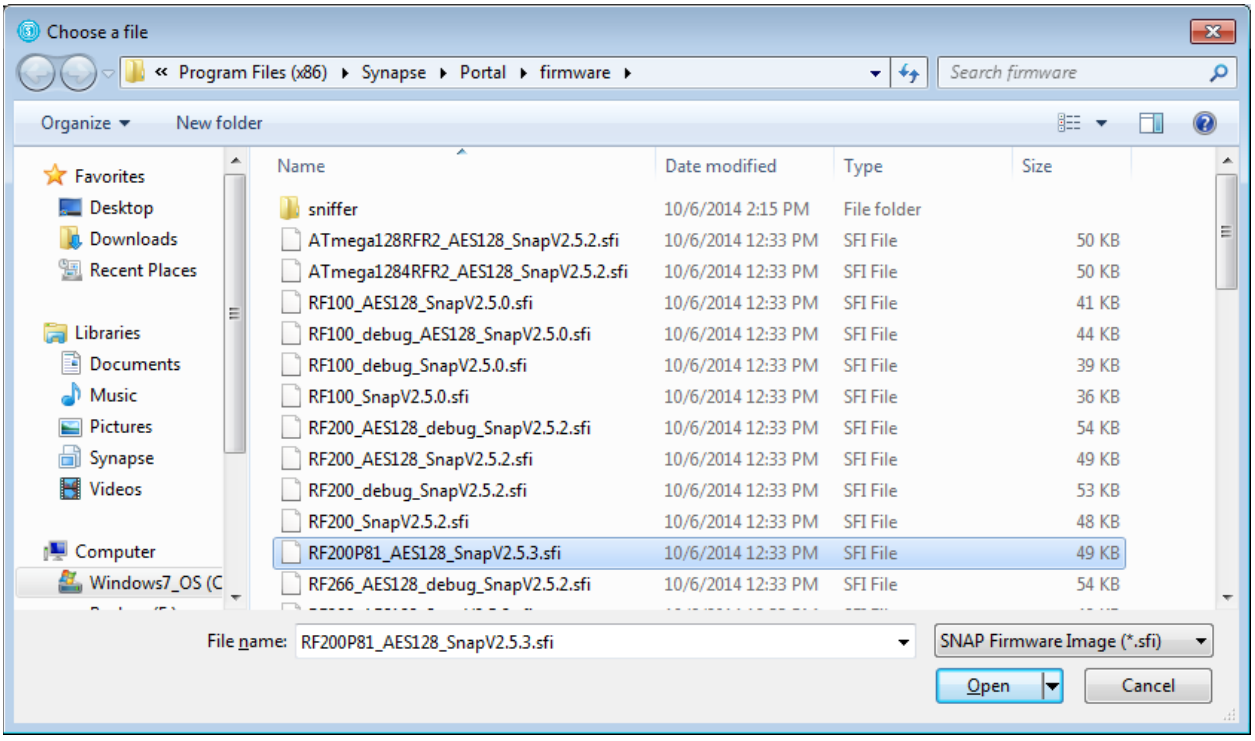
Click the **Open…** button next to the **Firmware Image:** field, and choose the version of software that you wish to upgrade the unit to.

If you choose an invalid firmware image, then you might see an error message like

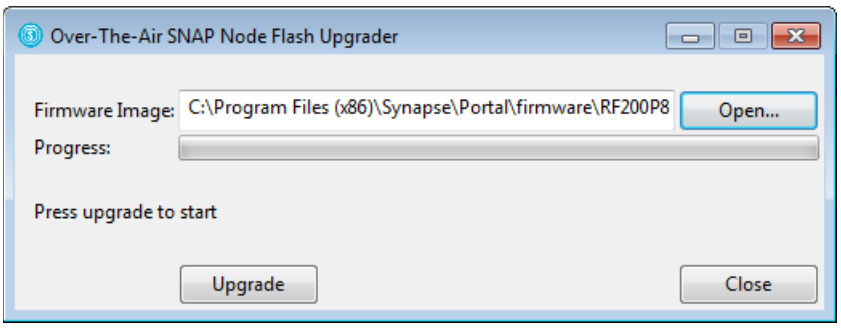


Examples of invalid files would be “sniffer” firmware images or ATmega128RFA1 **SNAP** versions prior to 2.4.32.

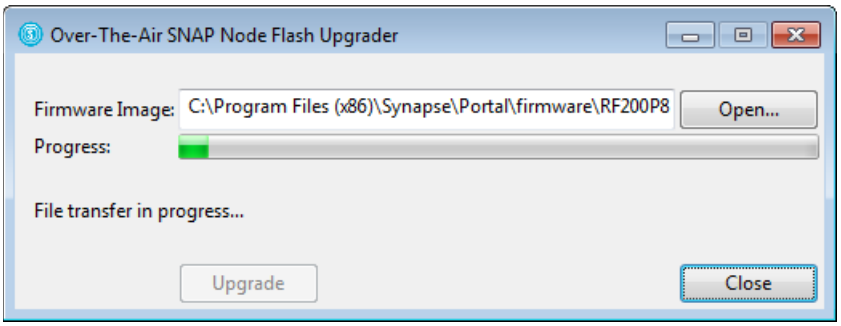
The following screenshot shows an example of upgrading the firmware on an RF200P81 module; your platform may be different.

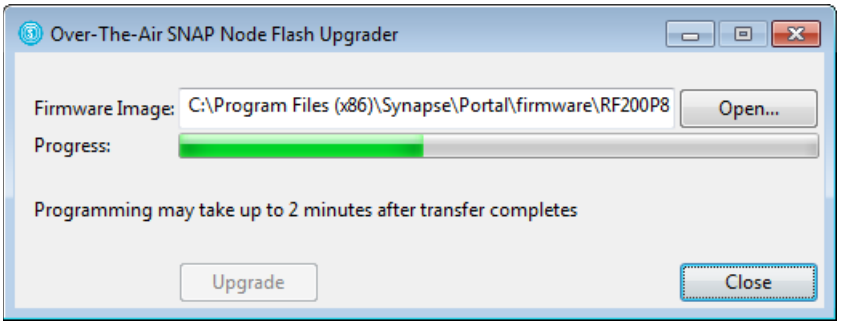


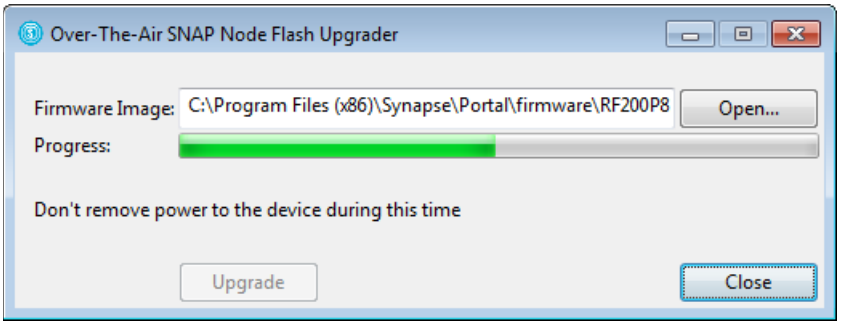
After choosing the correct filename, you will be back to the original dialog box.



Click the button labeled **Upgrade**, and wait for the upgrade process to complete. The progress bar will fill in as the **SNAP** firmware image is transferred over to the node, and stored in the temporary holding area. The message displayed under the progress bar will change as the firmware is loaded:



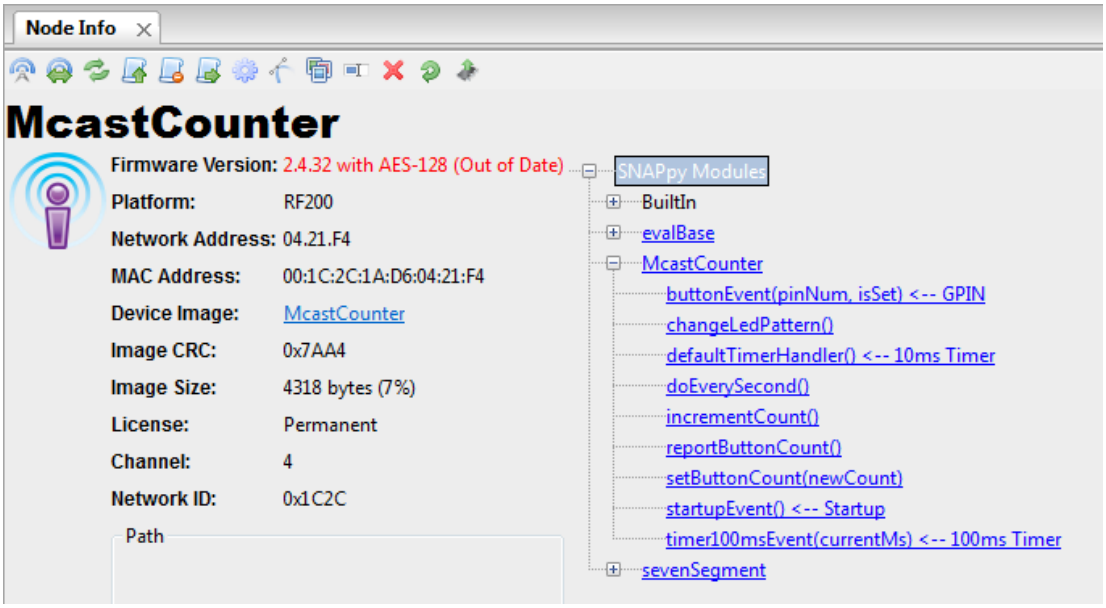




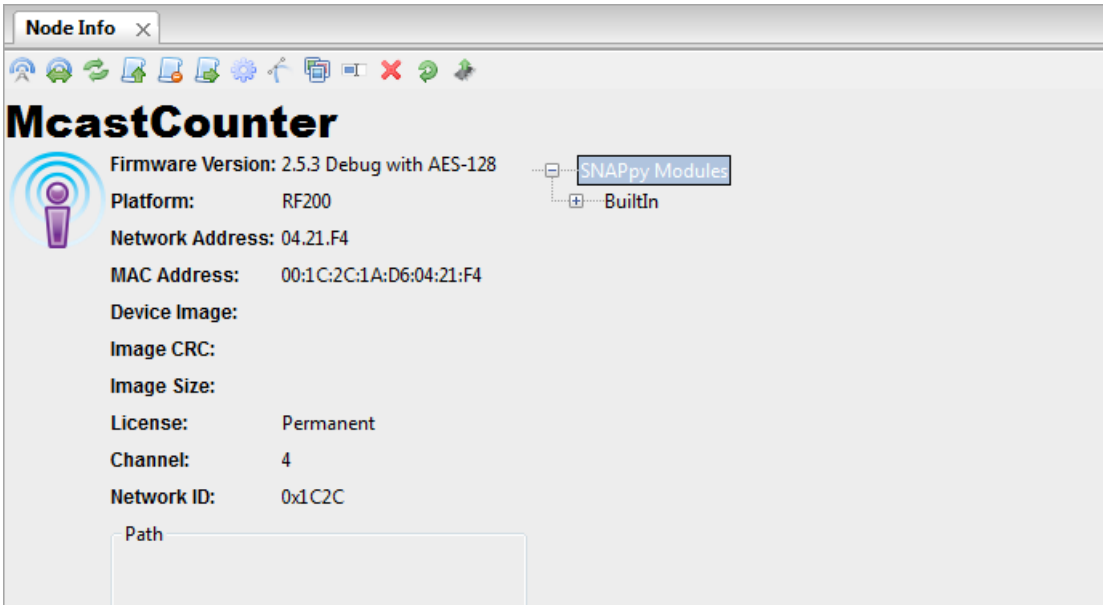
After the image is transferred, the node will reprogram its own FLASH such that the previous **SNAP** firmware image is overwritten with the new one. This will take several seconds (over two minutes for some nodes).

NOTE: If the node loses power (or is manually reset) during this final step, the interruption may result in the node no longer running **SNAP** at all. If this happens, use the "direct connect" method described in the previous section to reload the firmware serially.

You can use the **Refresh Node Info** function to confirm that the upgrade process has been successful. For example, here is the **Node Info** pane of an RF200 node before it was upgraded.



The following screen capture shows the same panel after doing an upgrade, and after doing a **Refresh Info** to update the GUI.



Notice that the firmware revision (and in this case, the firmware capabilities) have changed. Before the upgrade, the node was running a version 2.4.32 build of **SNAP**. After the upgrade, the node is running a version 2.5.3 build of **SNAP** with the “debugging checks” turned on.

Also notice that the unit no longer has a **SNAPpy** script in it. As explained previously, the **SNAPpy** script gets erased to make room for the incoming firmware image.

Simply use **Portal** to reload the script back into the node.

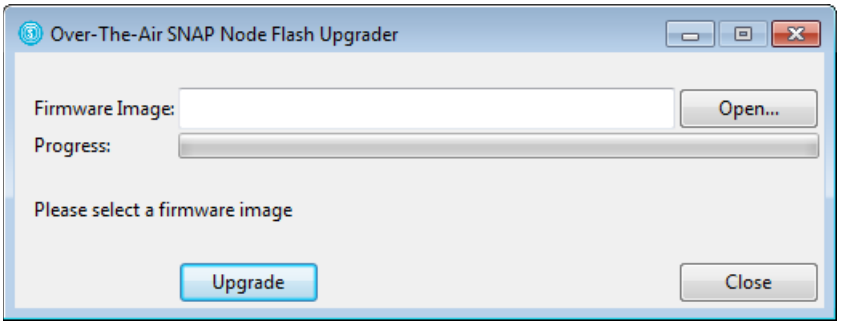
Installing new Firmware (Multiple Nodes in a batch)

Portal has supported “single node” OTA Firmware Update for some time, but now also supports “multi-select Firmware Update”, much like you can do “multi-select Script Update”.

Use the **Refresh Node** button to make sure you have connectivity (either wireless or direct serial) to all of the nodes you wish to upgrade.

Next, make sure all of those nodes are selected in the **Node Views** pane, then go to the **Options** menu and choose **Upgrade Over The Air…**

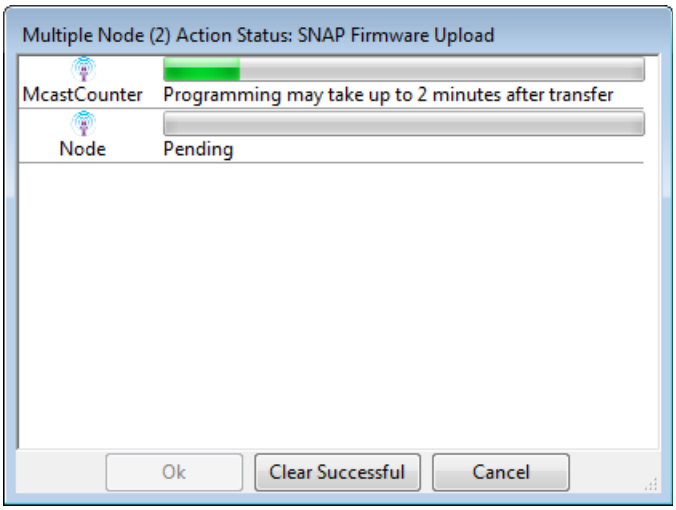
You should see a dialog box like the following:



You will notice this is the same dialog box as was shown in the “one node selected” scenario, except that in this case the Progress Bar is missing.

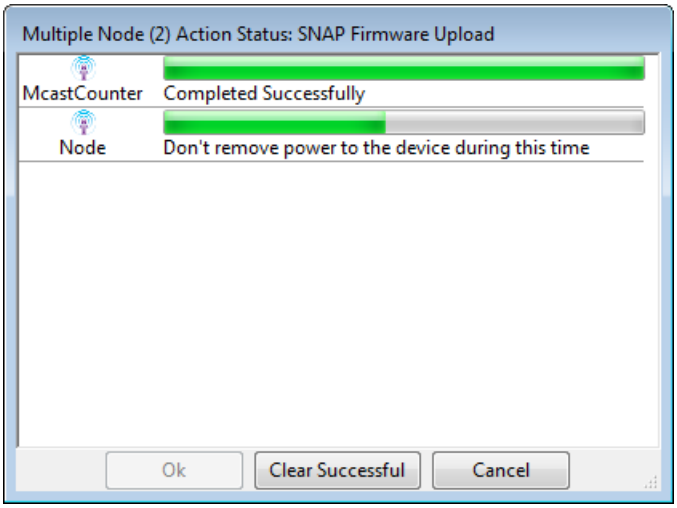
Choose the filename to be uploaded (just like in the single node case), and then press the **Upload** button.

The original dialog will be replaced with a multi-node progress dialog.

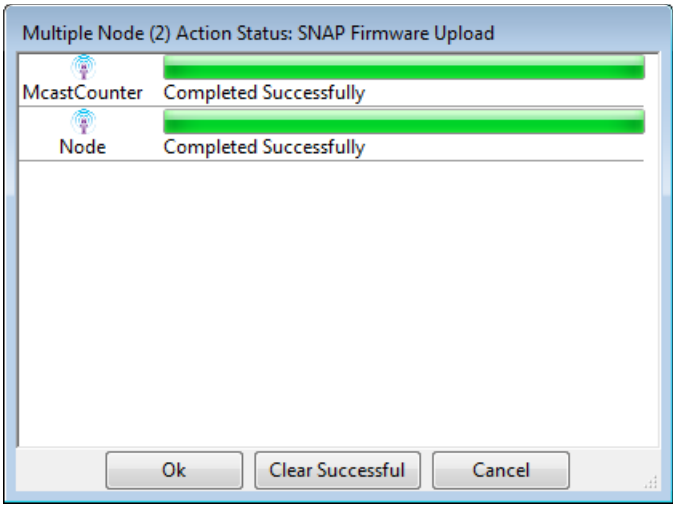


Above you can see the progress as the first of two nodes is uploaded.

After the first node has been sent a copy of the firmware, **Portal** will automatically proceed with the next node in the set.



When the last node in the set is uploaded, you can click the **Ok** button and resume normal **Portal** operations.



Remember that for some nodes, the node may not respond to inquiries for up to a little over two minutes. Do not remove power from these nodes or manually reset them during this period, or you will have to load new firmware into the nodes with a serial connection.

10. Menu Options

The following sections discuss the menu options within **Portal**.

File Menu

New Script

The **New Script** menu option will allow you to open a new **Code Editor** window with a blank script. This option has the same effect as using the **New Script** button on the main toolbar

Open File

This menu option displays a standard open file dialog that will open a **SNAPpy** or **Portal** script to edit in a code editor panel. This option has the same effect as using the **Open File** button on the main toolbar.

Save SNAPpy Script

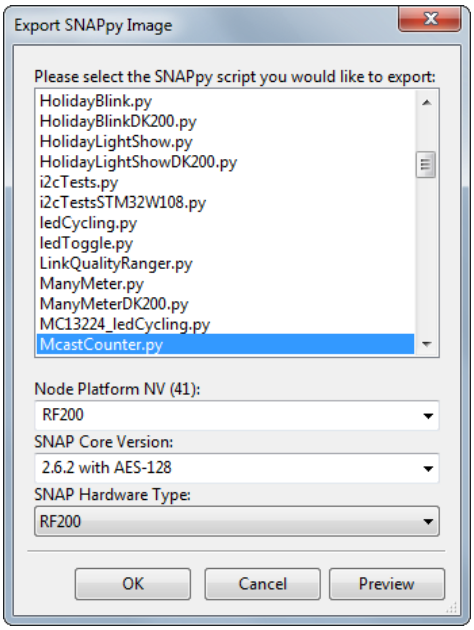
When a code editor panel is currently active this option will save the current file. This option has the same effect as using the **Save** button in the **Code Editor** toolbar.

Save SNAPpy Script As

When a code editor panel is currently active this option will save the current file as a different filename. This option has the same effect as using the **Save As** button in the **Code Editor** toolbar.

Export SNAPpy Script

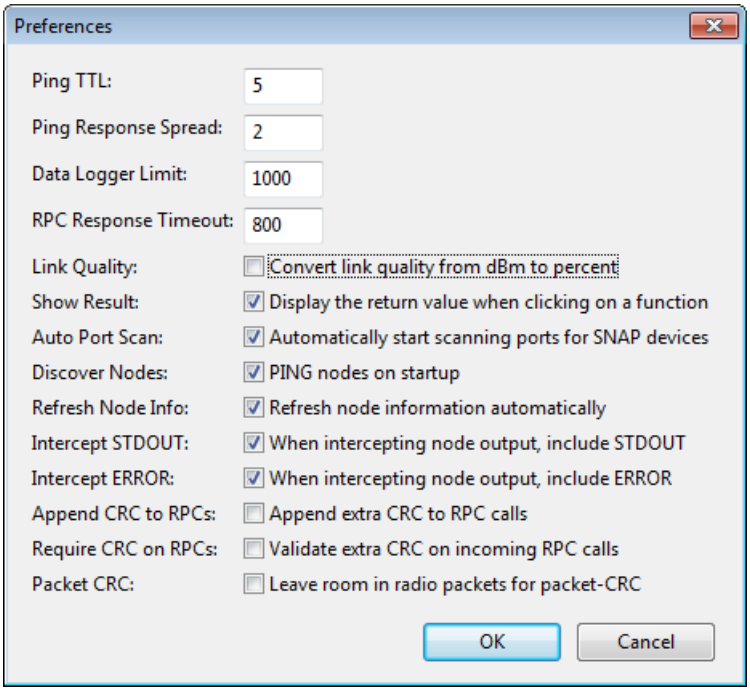
This menu option will display the following dialog box:



This option is similar to the **Export SNAPpy Image** button on the Node Info panel toolbar but this dialog allows you to manually specify the platform and version to use when exporting the script.

Preferences

The preferences dialog box allows you to configure different options on how you would like **Portal** to behave:



The following table describes each option in the ***Preferences*** dialog:

Ping TTL	The Time to Live (TTL) value to use when **Portal** sends a broadcast ping request. TTL is sometimes referred to as “number of hops”.
Ping Response Spread	The time in seconds across which to request nodes to randomize their broadcast ping responses.
Data Logger Limit	The number of data log entries to keep before dropping the oldest entry.
RPC Response Timeout	When **Portal** sends an RPC command to a node for which it expects a response (such as a callout, or a ping to the node), this parameter specifies how long, in milliseconds, **Portal** will wait for the response before sending the next packet. If your network requires many hops to reach all your nodes, you may want to lengthen this from the default value of 800.
Link Quality	Where applicable, display link quality as a percentage instead of dBm.
Show Result	When a function is clicked in the **Node Info** function tree, request that the node send the return value of that function to **Portal’s** event log.
Auto Port Scan	At startup have **Portal** automatically start scanning for **SNAP** bridge nodes.
Discover Nodes	Automatically have **Portal** send a broadcast ping once connected to a bridge node. This does not prevent **Portal** from discovering who the bridge node is.
Refresh Node Info	When **Portal** receives a message from a node not previously heard from, this option automatically retrieves the node’s information. When administering a large **SNAP** network, you may want to turn this off to reduce traffic.
Intercept STDOUT	When you select a node and use the **Intercept STDOUT** function, any messages that a node would normally output through a print statement are redirected to the **Portal** log instead. If you have both this checkbox and the **Intercept ERROR** checkbox unchecked, the **Intercept STDOUT** function does not do anything.

Intercept ERROR	When you select a node and use the **Intercept STDOUT** function, any error messages generated by **SNAP** in the node redirected to the **Portal** log instead. If you have both this checkbox and the **Intercept STDOUT** checkbox unchecked, the **Intercept STDOUT** function does not do anything. In nodes with firmware older than release 2.4, error messages are included in STDOUT automatically, and this option has no effect.
Append CRC to RPCs	When checked, **Portal** will compute a "payload CRC" and append it to all RPC and multicast RPC calls. Some **SNAP** nodes can be configured to perform this additional packet validity check. Nodes without this feature (or without this feature enabled) will simply ignore the additional information. This option reduces the maximum packet payload size by two bytes.
Require CRC on RPCs	When checked, **Portal** will ignore any RPC or multicast RPC packets it receives that do not have a valid "RPC CRC" appended to them.
Packet CRC	When checked, **Portal** will reduce its maximum packet size by two bytes to accommodate a packet-level CRC that can be applied to all radio traffic by nodes with firmware that supports this feature. This CRC covers more than the payload CRC that **Portal** can append or require based on the previous two checkboxes, in that it includes packet header information in addition to the packet payload. If you check this checkbox, **Portal** does *not* change the contents of any message it sends (as the packet-level CRC is applied only to radio traffic). But if a radio node has the packet-level CRC enabled/required and this checkbox is not checked, **Portal** may send serial packets that are too large for the radio nodes to be able to forward. (This will especially be true for script or over-the-air firmware uploads.) This feature was added in **Portal** and **SNAP** version 2.5. Checking this checkbox will not break compatibility with earlier **SNAP** versions. It will simply result in **Portal** generating packets that are two bytes smaller than they would otherwise be.

Exit

This menu option closes **Portal**, prompting to save any open and unsaved scripts first.

View Menu

The first six options in the View menu correspond to toolbar options to toggle the display of other windows:

View Menu Option	Toolbar Icon
Node View window	
Node Info window	
Event Log window	
Command Line window	
Data Logger window	
Channel Analyzer window	

If a menu item has a check mark next to it, this indicates the window is currently open.

Clear Windows

The **Clear event log window** and **Clear data logger window** menu options allow each window's respective captured data to be cleared.

Rearrange to Default View

If you have rearranged your **Portal** window layout, this option will rearrange the windows to the default layout view. It also resets any color settings you have applied to the **Node Views** pane.

Options Menu

Connection...

This menu option shows the standard **Portal** connect dialog box. This option has the same effect as using the button in the main toolbar.

Firmware Upgrade...

This menu option displays the **Firmware Upgrade** dialog to facilitate upgrading your **SNAP** nodes to a different version. You must have a serial connection to UART1 on your node to use this function.

Upgrade Over The Air...

This menu option displays the **Remote Firmware Upgrade** dialog to facilitate upgrading your **SNAP** nodes to a different version. **Portal** will disable this menu option unless you have selected a node that supports OTA upgrade, and that **Portal** has queried (The node has to tell **Portal** what it is capable of).

NOTE: In some cases not only is the hardware platform important, but also the firmware version too. For example, if **Portal** is connected to an ATmega128RFA1-based node running a version prior to 2.4.32, then this menu choice will be unavailable.

Factory Default NV Params...

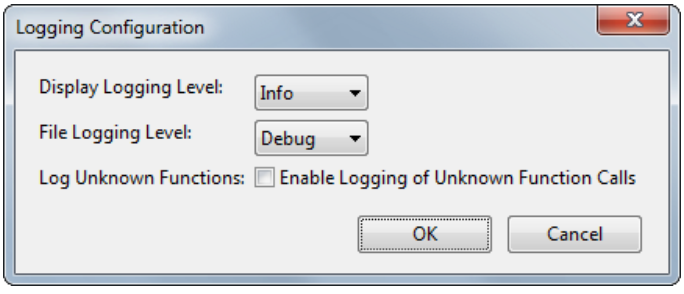
This menu option displays the **Factory Default NV Params** dialog, which allows you to reset all NV parameters to the defaults as shipped from the factory. You must have a serial connection to UART1 on your node to use this function.

Erase SNAPpy Image...

This menu option displays the **Erase SNAPpy Image** dialog, which allows you to erase the currently loaded **SNAPpy** script in your node. This option is useful if your script has inadvertently been programmed to lock you out of your node over the air. You must have a serial connection to UART1 on your node to use this function.

Configure Logging...

The **Logging Configuration** dialog enables the configuration of different logging levels and methods for Portal.



The **Display Logging Level** field controls the level of detail printed to the **Portal** Event Log.

The **File Logging Level** field controls the level of detail logged to disk.

The **Log Unknown Functions** checkbox enables logging of unknown function calls into the **Portal** Event Log. In this context, an unknown function call is a function that is not either a **Portal** built-in function, nor a function defined in the currently loaded **Portal** script.

NOTE: Portal sees function calls addressed directly to it (via unicast RPC), as well as multicast messages that **Portal's** bridge node is configured to pass along.

For example, if a remote node with **SNAP** Address 00.86.f9 were broadcasting

```
mcastRpc(1, 5, 'doSomething', 1, 2)
```

calls, and **Log Unknown Functions** were enabled (checked), then you would see events like the following in the **Portal** Event Log:

```
Node 0086F9 called function doSomething('1', '2') which does not exist
```

If you created a Python script that defined a doSomething(a, b) function, and loaded that into **Portal**, that function would be invoked instead of the "unknown function" event being generated.

Depending on the logging level specified, you may sometimes see messages in the log that are intended for advanced users and Synapse developers to assist with troubleshooting complicated network configurations. Typically if the error message is not explicitly documented, it can be ignored for most usages.

Set Working Directory...

The standard scripts that ship with **Portal** reside in a directory "tree" with the following naming conventions and structure:

The top level directory is named "Portal", and the source files in it are meant to be loaded into **Portal** itself.

Underneath the "Portal" directory is another directory named "**SNAPpy**Images". The source files in this directory are meant to be loaded into actual **SNAP** nodes, such as the RF100 or SM220.

Underneath the "**SNAPpy**Images" directory is a subdirectory named "synapse". These source files get imported by some of the example scripts in the "**SNAPpy**Images" directory.

By default, this source code tree is "rooted" in the *standard user files location* for your particular operating system. For example, on Windows the script source code tree can be found at

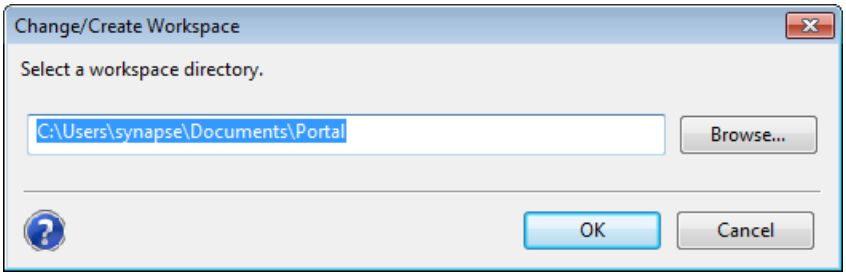
```
C:\Users\{PC_User}\Documents\Portal
```

The full tree is:

```
C:\Users\{PC_User}\Documents\Portal
C:\Users\{PC_User}\Documents\Portal\SNAPpyImages
C:\Users\{PC_User}\Documents\Portal\SNAPpyImages\synapse
```

You can add your own custom **Portal** and **SNAPpy** scripts into these same directories. However, users have requested the ability to work with alternate directories. Making that possible is the purpose of this menu option.

The **Set Working Directory** menu option displays a standard directory dialog box to allow you to choose a different "root" directory for your **Portal** scripts and **SNAPpy** scripts.



Use the dialog box to browse to the desired alternate "root" directory, and click the **Select Folder** button, and then the **OK** button.

If you change your mind *before* you click the **OK** button, you can click the **Cancel** button and the active working directory will not be changed. If you change your mind *after* choosing an alternate directory, you can use the **Set Working Directory** menu option again to browse back to the original (default) location. However, refer to the next menu option, **Restore Original Working Directory** for an easier way to do this.

To maintain a streamlined work-flow, **Portal** will automatically create a "**SNAPpy**Images" directory under the directory you specify, if one does not already exist. This is because the **Upload SNAPpy Image** function always defaults to this subdirectory when you choose to upload a script into a **SNAP** node other than **Portal**. For example, if you decide to keep your script work in "C:\MyProject", then the full tree will become:

```
C:\MyProject
C:\MyProject\SNAPpyImages
```

Remember that auto-completion of function names and variable names requires that **Portal** can locate the BuiltIn.py file within the "synapse" directory. If you do not copy that file into your new directory structure, **Portal** will locate it and use it from the default location – as long as you have not created a "synapse" directory in your own structure that does not contain a copy of the BuiltIn.py file.

A few words about adding additional custom subdirectories (Python Modules)

Just as Synapse used the "synapse" subdirectory to reduce the number of script source files contained in the "**SNAPpy**Images" directory, you can add additional subdirectories to better organize your working environment.

For example, you might set up something like:

```
C:\MyProject
C:\MyProject\SNAPpyImages
C:\MyProject\SNAPpyImages\MySensors
```

Your "MySensors" directory might contain scripts like "Temperature.py", "Pressure.py" etc. This will allow you to do things like "from MySensors.Temperature import *" in your top-level scripts.

However, in order for this to work, Python (not **Portal**) requires a *special file* to be present in every subdirectory that you intend to use in this way.

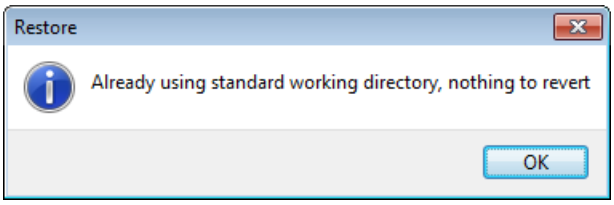
This special file is required to be named "__init__.py".

"__init__.py" (notice that there are two leading and two trailing underscore characters) can be an empty file (0 bytes long), but it is required to be present to tell Python (which **Portal** is running internally) that a directory is to be considered a "module", and thus imports from it should be allowed.

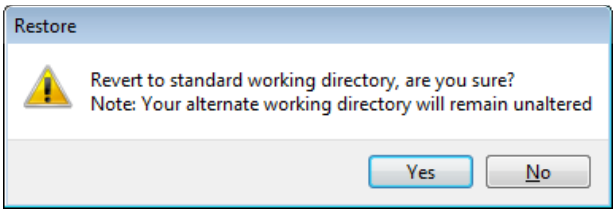
For more on this topic, please refer to the standard Python programming documentation. (www.python.org is always a good place to start.)

Restore Original Working Directory...

The **Restore Original Working Directory** menu option provides an easy way to get back to the default set of script source files. If you choose this menu option when you are in fact already using the original settings, **Portal** will tell you so.



If you were in fact using an alternate directory tree, **Portal** will prompt you to make sure you want to change back.



Choose **Yes** to revert back to the original directory tree, or **No** to continue using the alternate directory tree.

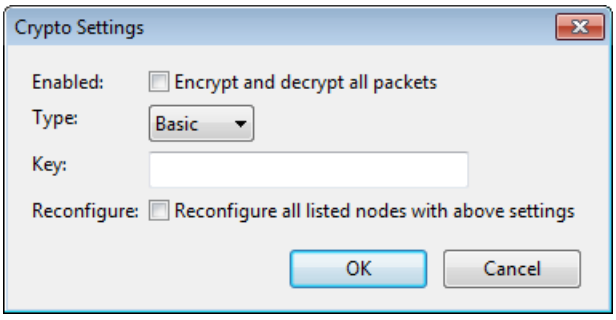
As the dialog box indicates, the files in your alternate directory tree are unaffected by this change, and you can resume working with them later by once again using the **Set Working Directory** menu option.

Configure Python Library Directory...

The **Configure Python Library Directory** menu option displays a standard directory dialog box to choose the directory where you have optionally installed a local copy of Python 2.6. This enables you to use the power of full desktop Python in your **Portal** scripts. If you have installed Python 2.6 on Windows in its default directory, this will likely be something like C:\Python26\Lib. Versions of Python other than 2.6.x are not supported in **Portal** and should not be selected.

Configure Crypto Settings...

The **Crypto Settings** dialog enables the configuration of encryption for the **Portal** node. Beginning with firmware release 2.4, all serial communication between nodes is encrypted when encryption is enabled, so **Portal** must be configured with the same encryption options used in your other nodes.

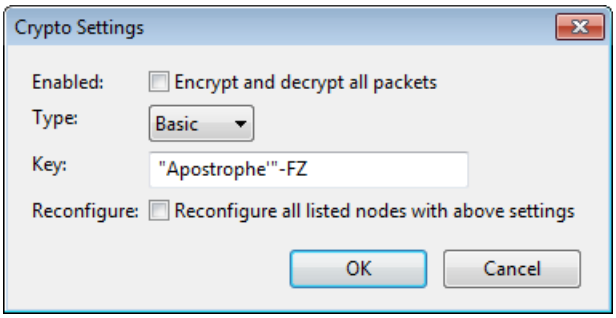


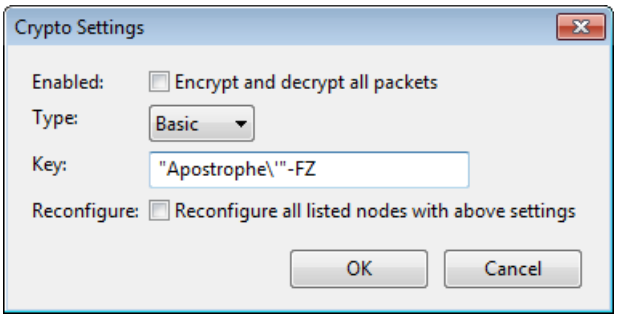
Check the **Enabled** checkbox to indicate that you want to use encryption. Select the appropriate encryption type (either **Basic** or **AES128**, if available), and specify a 16-character encryption key. The encryption key and the encryption type must exactly match the key and type used in your other nodes or they will not be able to communicate.

The key must be exactly 16 bytes long. You can specify the key as a simple string (e.g., ThEeNcRyPtIoNkEy), as a series of hex values (e.g., \x2a\x14\x3b\x44\xd7\x3c\x70\xd2\x61\x96\x71 \x91\xf5\x8f\x69\xb9) or as some combination of the two (e.g. \xfbOF\x06\xe4\xf0Forty-Two!). Standard security practices suggest you should use a complicated encryption

key that would be difficult to guess. If your hex values include characters that can be displayed, **Portal** will convert them to displayable characters when you next view the key. For example, "\x37" would display as "7".

If your encryption key includes both an apostrophe (') and quotation marks ("), **Portal** will display the apostrophe as an “escaped” character the next time you view the key.





If you check the **Reconfigure** checkbox, **Portal** will set all available nodes to use the same encryption settings, using traceroute information to work from the outer edges of your network, in. Refer to the **SNAP Reference Manual** and **SNAP Users Guide** for more information about encrypted **SNAP** networks.

Clear GUI Component History

Starting with **Portal** version 2.5 , text fields maintain a history of the values that have been entered in them to facilitate making a particular function call or node configuration again. When you invoke an RPC window or a configuration window, **Portal** will show you the current setting or the most recent previous value used. If more than one value has been specified in the field since **Portal** was installed or since the GUI component history was cleared, **Portal** will display a down arrow indicating that there are other options available for selection.

Using this option from the Options menu removes this history data, leaving **Portal** with a fresh lack of knowledge about what calls have been made in the past.

Network Menu

Broadcast PING

This menu option sends a broadcast PING request to the **SNAP** network. This option has the same effect as using the **Broadcast PING** button on the main toolbar.

Find Nodes...

This menu option opens the Find Nodes dialog box to search for nodes across different **SNAP** networks. This option has the same effect as using the **Find Nodes** button on the main toolbar.

Generate Topology .DOT File

This option generates a topology.dot file to view a snapshot of your **SNAP** mesh network. This is accomplished by having the **SNAP** nodes query for any nodes that are one hop away and report that information for **Portal**.

To view the .DOT file, which is saved in your **Portal** documents directory, you will need to use a .DOT file viewer. There are several freely available viewers on the Internet.

Change Portal Address...

This option changes the network address of your **Portal** node. This may be necessary if you have an environment with multiple computers running **Portal** where the nodes can interact. If more than one node has the same network address (the default is 00.00.01), you can have unpredictable results.

The valid network addresses for **Portal** range from 00.00.01 through 00.00.0F. It is not necessary to restart **Portal** for this change to take effect.

New Configuration

This option creates a new **Portal** configuration, which deletes any locally saved **Portal** information. Your window layout and all your preferences will be preserved, however scheduled scripts and saved node information will be deleted. For example all the nodes in the **Node Views** will be deleted and a broadcast ping will be performed to rediscover the network.

Open Configuration...

This option opens a previously saved .SWN file that contains saved information about nodes.

Save Configuration As...

This option saves a Synapse Wireless Network (.SWN) file that contains information about your nodes. This includes a list of all the nodes about which **Portal** is currently aware, plus changes such as custom node icons.

Export Node List as CSV...

This option saves a text file of comma-separated values to the destination of your choice, listing all the nodes currently included in your **All Nodes** view. The CSV file includes a header and three pieces of data for each entry: Node Name, **SNAP** Address, and file path to an icon file (if you have specified an icon for the node).

Import Node List as CSV...

This option loads a text file of comma-separated values, either adding to replacing the nodes in your **All Nodes** view (depending on whether you check the “Clear Network First” checkbox on the dialog). The CSV file should include a header and three pieces of data for each entry: Node Name, **SNAP** Address, and file path to an icon file (if you have specified an icon for the node).

Launch SNAP Sniffer...

The **SNAP** Sniffer is an external program that helps to analyze and debug your **SNAP** network. Please consult the **SNAP Sniffer Users Guide** for detailed information.

NOTE: You can run **Portal** and the SNAP Sniffer on the same computer at the same time, **but** they cannot use the same serial port.

Help Menu

Portal Reference Manual

This option will open the **Portal Reference Manual** (this document) in your default PDF viewer.

DK-200 Users Guide

This option will open the **DK-200 Users Guide** in your default PDF viewer.

SNAP Primer

This option will open the **SNAP Primer** in your default PDF viewer.

SNAP Users Guide

This option will open the **SNAP Users Guide** in your default PDF viewer.

SNAP Reference Manual

This option will open the **SNAP Reference Manual** in your default PDF viewer.

SNAP Sniffer Users Guide

This option will open the **SNAP Sniffer Users Guide** in your default PDF viewer.

SNAP Support Forums

This option will open the Synapse Support Forums website in your default web browser. This website is the preferred place to ask any technical questions you may have while using our products.

About

The about dialog displays information about the current version of **Portal** you are running.

11. Portal API

This section details the API presented to **Portal** Scripts. The **Command Line** window of **Portal** also has access to this API.

When a node makes a call to a **Portal** function, two variables are set within the scope of the **Portal** function that allow easy identification of and access to the calling node. The `remoteNode` variable is set to the calling node's node object, and the `remoteAddr` variable is set to the address of the calling node. These variable values persist only for the duration of the called function. They can be used to invoke node methods or read node attributes on the node, or to send messages or commands back to the node.

Node Methods

The following are functions bound to each network device (node) known to Portal, i.e. visible in one of the **Node Views**, or appearing in the **Node Info** window.

To invoke one of these methods, you must prefix the desired function name with a variable containing a reference to the node that you want to invoke the function on, followed by a "." character. For example:

```
MyNode.getDottedNetAddr()
```

In order to get a reference to a node, you can use the `remoteNode` property in your **Portal** script. The `remoteNode` variable contains a reference to a node object any time **Portal** is running a function invoked by an RPC call from another node.

`getColumn(`*`name`*`)`	Returns the current value stored for the specified column name
`getDottedNetAddr()`	Returns this node's network address as a string in the format '##.##.##'
`getHexNetId()`	Returns a string representing the node's network ID, if known, e.g., '0x1C2C'
`ping(toggleResponding=True)`	PINGs the node. Parameter *toggleResponding* determines how the result of the PING affects the Node Views and Node Info windows
`refresh(`*`ping`*`=True, `*`toggleResponding`*`=True)`	Queries the node for information to display in its Node Info tab.
`setColumn(`*`name`*`, `*`value`*`)`	Stores the specified value for the specified name
`setNvParam(`*`id`*`, `*`value`*`)`	Tells the node to set the NV parameter *id* to the specified *value*.
`setStdoutPortal()`	Tells the node to redirect its STDOUT to **Portal's** network address. Whatever the node prints will appear in the **Portal *Event Log***
`linkQualityAsk()`	**Deprecated:** Queries the node for its currently link quality value
`linkQualityPercentVal()`	**Deprecated:** Returns an integer value in the range 0-100 based on the node's last known link quality
`macAsk()`	**Deprecated:** Queries the node for its MAC Address
`netParamAsk()`	**Deprecated:** Queries the node for its channel and network ID
`reboot()`	**Deprecated:** Tells the node to reboot
`versionAsk()`	**Deprecated:** Queries the node for its current version string

Node Attributes

The following are attributes bound to each node known to **Portal**, i.e. visible in one of the **Node Views**.

Like the node methods, these must be prefixed by a reference to the node you want to examine, and a "." character. For example:

```
print MyNode.channel
```

NOTE: These are data fields, not functions. Do not put parentheses "()" after them.

`MyNode.channel` returns a number.

`MyNode.channel()` returns an error.

channel	An integer corresponding to the node's channel
dottedNetAddr	The node's network address as a string in the format '##.##.##'
hexNetId	A string representing the node's network ID, if known, e.g., '0x1C2C'
isResponding	A Boolean value indicating whether the last query was successful
lqVal	A float value of the node's last queried link quality in dBm
name	A string that describes this node in Portal's node views
netAddr	A string containing this node's unformatted network address
networkId	An integer corresponding to the node's network id
version	A string containing the node's last queried version

Portal Methods

The following methods are independent of **SNAP** Nodes. They provide a system-level interface between **Portal** scripts and system-wide capabilities. Because they are not node specific, you do not prefix them with a node name and a "." character.

Some of these routines do take a node parameter. You can specify the correct node in two different ways:

- By explicit address
- By node name (the node's name as shown in the Node Info pane within Portal)

Explicit addresses are three-byte strings. **Portal** shows node Network Addresses in hexadecimal, but omits any number base prefixes or suffixes. If **Portal** shows a node's Network Address as 12.34.56, then the individual address bytes are 0x12, 0x34, and 0x56 (hex), not 12, 34, and 56 (decimal).

To specify hexadecimal constants within Python or **SNAPpy** strings, you use a leading "\x". Going back to our example, if you want to specify the node with address 12.34.56, you would use "\x12\x34\x56" in your script.

Portal also assigns names to nodes when it first discovers them. If the node with address 12.34.56 has a displayed name of "MyNode" then instead of specifying an explicit address string, you can just use the text MyNode instead.

NOTE: There are no quotation marks used when referencing nodes *by name* within **Portal** scripts (or from the **Portal** command line).

So, assuming node MyNode4 is at address AB.CD.EF, the following two commands are equivalent:

```
sendData("\xAB\xCD\xEF", "This is a test")
sendData(MyNode4, "This is a test")
```

dmcastRpc(*nodes*, *groups*, *ttl*, *delay*, *function*, *args*)	Sends a directed multicast RPC request to the specified addresses in the *nodes* address list. Parameter *function* is the routine to invoke on each node in the specified *groups*, and the remaining parameters are passed to that routine.
logEvent(*message*)	Log the specified *message* to the Event Log

`logData(name, value, fullscale=100.0)`	Plot the specified *value* to a named strip chart in the Data Logger window
`mcastRpc(groups, ttl, function, *args)`	Sends a multicast RPC request to the specified *groups*. Parameter *function* is the routine to invoke on each node in the specified *groups*, and the remaining parameters are passed to that routine.
`multicastRpc(…)`	The same as the mcastRpc function above but left for backwards compatibility with older versions of Portal
`rpc(node, function, *args)`	Sends an RPC request to the specified node. Parameter *function* is the routine to invoke on the node, and the remaining parameters are passed to that routine.
`sendDataModePkt(node, data)`	Sends *data* to the specified *node* in a way that mimics TRANSPARENT MODE. This lets **Portal** insert data into TRANSPARENT MODE links.
`sendData(node, data)`	This is the shorthand form of `sendDataModePkt()` (**Portal** Command Line only)
`sendMcastDataModePkt(group, ttl, data)`	Sends *data* to the specified multicast *group* in a way that mimics TRANSPARENT MODE. This lets **Portal** insert data into TRANSPARENT MODE links. *Group* can be specified as a two-byte binary string, or as an integer. For example, to reach the **broadcast group**, specify a *group* parameter of "\x00\x01" or 0x0001. To reach **all** groups, specify a *group* parameter of "\xFF\xFF" or 0xFFFF. (The "string form" of the parameter is supported for backward compatibility; most users will prefer integer notation.)
`sendMcastData(group, ttl, data)`	This is the shorthand form of `sendMcastDataModePkt()` (**Portal** Command Line only)
`sendTraceRouteRequest(node, sequence)`	Sends a traceroute request packet to the specified *node* with the specified *sequence* number

There are also some commands that must be prefixed by "root."

`root.clearEventLog()`	Clears the **Portal** Event Log
`root.displayTraceFile()`	Pops up a tabbed window with the contents of an internal log file sometimes used in debugging.
`root.setPortalNetAddr(newAddr)`	Change **Portal's** network address from the default value of "\x00\x00\x01" (00.00.01). Portal's network address can range from \x00\x00\x01 to \x00\x00\x0F.
`setup_usb_as_vcp()`	Convert an SN132 SNAPStick **SNAP** Engine carrier to respond to your computer as a virtual COM port rather than as a USB device. For most users there will never be a need to do this.
`setupSynapseUsb()`	Revert an SN132 SNAPStick that has been set to respond as a virtual COM port back to responding as a USB device. For most users there will never be a need to do this.

There are several commands that are deprecated or for Synapse internal use only. They are listed here only for completeness; you **should not** be calling these functions.

`cmdDispatcher`	Synapse use only
`notifyOnComplete()`	Synapse use only

SNAPpyErrorDecode(*node, errorMsg*)	Synapse use only

Contextual Variable

Within the scope of a running function in **Portal**, you can use the `remoteAddr` variable to determine the **SNAP** address of the node that invoked the running function. This serves much the same function in **Portal** that the `rpcSourceAddr()` function serves in **SNAPpy** scripts at the module level. If the function is invoked from the Node Info pane in **Portal** or by a scheduled event, the `remoteAddr` variable will contain None.