



iDigi Dia Training

Dia102

Dia on Digi
Products



Dia Training – Dia102

Agenda

- Pre-requisites
- Deploying Dia on Digi Devices
- Running Dia on Digi Devices
- Dia & ZigBee
- Next Steps



Dia Training – Dia102

Pre-requisites (in no particular order)

- Dia101 - Basics
- PC, Mac or Linux computer
- Access to Internet from computer
- Telnet application available on computer
- Latest copy of Dia (see instructor)
- Coffee or Tea
- Curiosity and Enthusiasm



Dia Training – Dia102

Deploying the Dia on a Digi Device



Making the Dia Binary: dia.zip

Loading the iDigi Dia onto a Digi Device is easy. First, you will need to build a Dia distribution binary which reflects the functionality your device requires.

As you learned in Dia101, "Opening a Command Prompt and Running the Dia" above, open a command prompt window and navigate to the directory folder where the iDigi Dia has been installed and at the prompt type the following:

```
C:\Users\Joe\Source\dia>python make.py cfg\dia.yml
```

- Note: If python.exe is not in your environment's PATH you will have to change "python" to be "C:\Python24\python.exe" in the above.



Making the Dia Binary: dia.zip

When the `make.py` script executes it will analyze the current project from your configuration file (*.yml) and generate a file named "dia.zip" in the "bin/" directory of the iDigi Dia installation folder. This archive contains only the Python libraries necessary for running the Dia on the target Digi platform at a savings of system non-volatile flash storage.

In the previous example you used the file name "dia.yml". However, the `make.py` script will accept any valid Dia YAML file for analysis. For example:

```
C:\Users\Joe\Source\dia>python make.py cfg\joes_custom_cfg.yml
```



Making the Dia Binary: dia.zip

The output that printed to the command window when running `make.py` should resemble the following:

```
Analyzing files...  
Compiling files...  
Zipping files...  
Finished writing archive bin/dia.zip
```

Uploading the dia.zip File

After building the archive, the next task is to load the archive and the initial loader script "dia.py" onto the target Digi device. In order to do this, open a web-browser and connect to the Web User Interface of the Digi Device. At the far left there is a heading labeled "**Applications**" and underneath this heading there is a link labeled "**Python.**" Click on this "Python" link.



Uploading the Dia dia.zip

You will be presented with a listing of files as well as an upload form. Use the form to upload "dia.py" and "bin/dia.zip" to the device. There are two things to note at this point:

The first is that the configuration file "dia.yml" has been built into the dia.zip file. It does not need to be loaded to the device.

The second item of note is that if configuration changes are made, the "dia.zip" file will have to be rebuilt using the "make.py" script and re-loaded onto the device.

The "dia.py" script does not change and it does not need to be reloaded. The "dia.py" script may only change if you change it via local edits (for debugging purposes, etc.) or if you receive a new release of Dia from Digi.



Dia Training – Dia102

Running Dia on a Digi Device



Running Dia on Digi Devices

With the Dia Framework loaded on the Digi device you can now start the iDigi Dia application. Connect to the Digi device's command line interface using a telnet client to the IP address of the Digi device with the default port (TCP port 23). You should see a prompt that looks like:

```
#>
```

In order to start the iDigi Dia application type the following:

```
#> python dia.py dia.yml
```

You can override the compiled dia.yml (“yml”) file (within dia.zip) by supplying your own configuration file during start up. Devices and presentations provided by your optional “yml” file must only include functions defined with the dia.zip file.



Running Dia on Digi Devices

You should see the following output be printed to the terminal:

```
#> python dia.py dia.yml  
  
Determining platform type...Digi Python environment found.  
  
iDigi Dia Version 0.9b  
Using settings file: WEB/python/dia.yml  
Starting Channel Manager...  
Starting Device Driver Manager...  
Starting Presentation Manager...  
Core services started.
```

As you can see, the Dia is an I/O management engine that initializes in phases, from configuration file evaluation to the subsequent initialization of the core Channel, Device Driver, and Presentation services.



Dia CLI on a Digi Device

Once the Dia has started you can connect a telnet client to port 4146 and see your data just as you had when the Dia was running locally on your PC:

```
Connected to 10.8.113.64.  
Escape character is '^]'.  
Welcome to the iDigi Dia CLI.  
=>> channel_dump
```

```
Device instance: template
```

Channel	Value	Unit	Timestamp
adder_reg1	0.0		2008-11-05 18:28:57
adder_reg2	0.0		2008-11-05 18:28:57
adder_total	0.0		2008-11-05 18:28:57
counter	701		2008-11-05 18:41:00
counter_reset	(N/A)		
global_reset	(N/A)		

```
Device instance: template_transforms
```

Channel	Value	Unit	Timestamp
count_by_two	1402		2008-11-05 18:41:00

```
=>>
```



Dia Training – Dia102

Dia & ZigBee



Dia / ZigBee Introduction

The real power of the Dia is in device and data presentation abstraction. This power may be leveraged with wireless devices quite easily. The ConnectPort X series of wireless gateways from Digi incorporate a wireless XBee module for connectivity to many different types of wireless networks.

The iDigi Dia contains an abstraction model for communicating with wireless devices. The abstraction is expressed in what is known as the XBee Driver Stack for the iDigi Dia.

The XBee Driver Stack is divided into two items:

- 1) The XBee Device Manager
- 2) XBee device driver implementations.



The Dia XBee Device Manager

A single instance of the XBee Device Manager is required when using one or more XBee device drivers on the Dia. The XBee Device Driver Manager is responsible for multiplexing data communications and managing node configurations. Each XBee device driver will use this single instance of the XBee Device Driver in turn to communicate with the wireless network.

A demonstration configuration file is included with the iDigi Dia to illustrate how to use the Dia to communicate with a wireless network. The configuration file includes the necessary statements to load an instance of the XBee Device Manager and a XBee device driver for a wirelessly connected XBee XBIB development board. The configuration file is located in the file `demos/getting_started/dia_xbib.yml`



dia.yml with XBee Support

Xbee Manager Entry:

```
# First we specify an XBee Device Manager.  Since the XBee is considered,  
# a shared resource on our system, we need a device which can manage  
# requests from other devices relating to the XBee.  
- name: xbee_device_manager  
  driver: devices.xbee.xbee_device_manager.xbee_device_manager:XBeeDeviceManager
```

The “#” character represents comments.

The entry “- name:” defines the name tag for the XBee Manager.

The entry “driver:” defines the actual driver for the XBee Manager. The portion “**devices.xbee.xbee_device_manager.xbee_device_manager**” describes the location of the driver (path and file), which is under directory “**/src/devices/xbee/xbee_device_manager**” and file **xbee_device_manager**.

XBeeDeviceManager is the actual class name within the Python driver.



Defining the XBee Device

You can use the included xbib configuration file to communicate to a wireless device using the iDigi Dia. One of the first activities you need to do is to edit the `dia_xbib.yml` configuration file to contain the proper address of the XBee module installed in the XBIB development board.

In order to determine this address, you must first either read the address from the sticker at the bottom of the module or use the ConnectPort user interface to discover the address on our own:

- If you use a web-browser and connect to the IP address of the Digi device, you can click on a link under the "Configuration" heading labeled either "XBee Network" or "Mesh Network" (depending on the Digi device firmware version). If you click on the "Refresh" button, the list of discovered devices will update and it will resemble the following:



Defining the XBee Device

Node ID	Network Address	Extended Address	Node Type ...
COORD	[0000]!	00:13:a2:00:40:34:02:30!	coordinator
...			
DEVBRD	[ef69]!	00:13:a2:00:30:0a:4a:b5!	router

The "Extended Address" of each node is static and unchanging. This value, "00:13:a2:00:30:0a:4a:b5!" in the above example is what we will update the "extended_address" setting of the xbib0 instance in the dia_xbib.yml configuration file.

Identify the extended address value of your module and then update file demos/getting_started/dia_xbib.yml using the text editor of your choice and modify the following line to contain the value of your XBee module's address:

Once this change has been made, save the file and quit your text editor.

```
# Instantiate and XBee XBIB development board instance:
- name: xbib0
  driver: devices.xbee.xbee_devices.xbee_xbib:XBeeXBIB
  settings:
    # Note: the xbee_device_manager setting is being set to
    # "xbee_device_manager", which is the name of the instance of the
    # XBeeDeviceManager, above.
    xbee_device_manager: "xbee_device_manager"
    # Change the below extended_address setting to match the 64-bit
    # hardware address of the radio installed in your development
    # board:
    extended_address: "00:13:a2:00:40:0a:4a:b5!"
```



Re-Building dia.zip

As you did earlier, rebuild the dia.zip, this time using the xbib configuration file:

```
C:\Users\Joe\Source\dia>python make.py demos\getting_started\dia_xbib.yml
```

Now upload the new dia.zip file to the Digi device as you did earlier and reboot the gateway for the changes to fully take effect.

Next, telnet into the Digi device and start the Dia using command:

```
#> python dia.py dia_xbib.yml
```

After Dia starts (look for the “Core services started...” message), open another telnet window from your PC and telnet to your Digi device with port 4146 (Dia CLI presentation). At the prompt, issue the channel_dump command.

In addition to the template and template_transformations device instances, you’ll see a new one labeled: xbib0.



The new xbib0: Device Instance

Notice the addition of the xbib0 device (XBee).

```
Device instance: template_transforms
```

Channel	Value	Unit	Timestamp
count_by_two	50		2008-11-10 17:50:59

```
Device instance: xbib0
```

Channel	Value	Unit	Timestamp
sw1	True		2008-11-10 17:50:34
sw2	True		2008-11-10 17:50:36
sw3	True		2008-11-10 17:50:38
sw4	True		2008-11-10 17:50:38

```
=>>
```



Interacting with the XBee Device

From the channel dump you can see the new device instance of the XBee XBIB device driver named "xbib0". The driver contains four channels which are representative of the four momentary push buttons on the XBee XBIB development board. The four buttons on this development board will pull the logic level to ground when they are pressed and the value of the channel in the Dia will transition from "True" to "False". In the above snapshot, all four buttons are in their open state.

```
=>> channel_dump xbib0
```

```
Device instance: xbib0
```

Channel	Value	Unit	Timestamp
sw1	False		2008-11-10 17:51:28
sw2	True		2008-11-10 17:50:36
sw3	True		2008-11-10 17:50:38
sw4	True		2008-11-10 17:50:38

In the below example, you should be holding a button down while dumping the channels of the xbib0 device: You can see the xbib0.sw1 channel has transition from "True" to "False".



Extending the Dia with Additional Devices

We can enable more wireless devices by following the same pattern presented in this section:

- Add configuration lines to a configuration file.
- Re-generate dia.zip and upload it to the gateway.
- Restart the iDigi Dia.

Settings for each driver may be found in the API reference for each driver's source file. Either open the API reference documentation or use a text editor to read the documentation from each driver source file directly.



Other Device Driver Examples

```
# Here we are including a Digi LTH (Light, Temperature, Humidity)
# XBee Sensor in our system. Configuration is similar to the Smart Plug.
# The main addition is that since the LTH Sensor will likely be battery
# powered, we have set the XBee to sleep between samples.
- name: lth_sensor_1
  driver: devices.xbee.xbee_devices.xbee_sensor:XBeeSensor
  settings:
    xbee_device_manager: xbee_device_manager
    extended_address: "00:13:a2:00:40:4a:ba:7f!"
    sleep: True
    sample_rate_ms: 1000
    awake_time_ms: 320

# This is an XBee Analog IO Adapter. The setting power decides whether
# the Adapter should provide power through terminal 6 of the device. The
# following channel modes let us decide whether terminals 1-4 measure
# voltage (from zero to ten), four to twenty milliamp current loop,
# or the voltage differential between terminal pairs (1/2, 3/4).
- name: aio_adapter_1
  driver: devices.xbee.xbee_devices.xbee_aio:XBeeAIO
  settings:
    xbee_device_manager: xbee_device_manager
    extended_address: "00:13:a2:00:40:52:2c:4a!"
    sample_rate_ms: 1000
    power: On
    channel1_mode: TenV
    channel2_mode: TenV
    channel3_mode: TenV
    channel4_mode: TenV
```



Next Steps

This ends the introduction (Dia102) course. You should now have a general understanding of the Dia product and know how to deploy it on Digi products. In Dia103, we'll continue exploring the functionality of Dia by working with Presentations and Transforms.