30 Example Code

The `\Examples_Source` directory in the repository contains three different scenarios highlighting different use cases:

Generic Networking Example

This is the original example, showing GenNet's default use cases with two modules:

GenNet-Proxy

The `GenNet-Proxy` module is a Generic-Networking-enabled DQMH Singleton module, which means it is altered to allow for (near) zero-coupled networking functionality. It showcases the transparent use case

- Call the *Enable Network Forwarding* request to make this module a proxy. By loading a GenNet-Client clone module and redirecting communication, all requests are sent via TCP to another `GenNet-Proxy.lvlib` running in another application and/or another PC accessible via TCP/IP.
- Call the *Enable Network Listening* request to make this module accept messages via network. By loading a GenNet-Server clone module, let this module accept messages from `GenNet-Proxy.lvlib` running in another application and/or on another PC via TCP/IP communication.

RemoteControl

The `RemoteControl` module showcases the manual use case of the `GenNet-Client` and `GenNet-Server` modules, and how any generic DQMH module can make use of the generic networking functions, albeit not transparently to the user. The message that's sent via the network needs to be constructed manually, and in a way that the opposite site can decode it.

How to test GenNet-Proxy as client and server

This showcase demonstrates the transparent use case and how to use a module's API in one application and have its code executed in another application. Therefore, two LabVIEW projects are used that contain the same module. Check the Application Instance on the bottom left of the API Tester, to ensure in which Application it was loaded. Follow these steps to configure `GenNet-Proxy` in the `Windows Application` as a client which forwards requests to the `GenNet-Proxy` in the `Realtime Application`, where the `GenNet-Proxy` is configured as a server, receiving the request, executing the MHL-case and sending the payload back over the network:

1. Open the `\Generic Networking Example_Source\Windows Application.lvproj` file
2. Start the `Test GenNet-Proxy API.vi` Tester, start the module and *Enable Network Forwarding*
3. Open the `\Generic Networking Example_Source\Realtime Application.lvproj` file

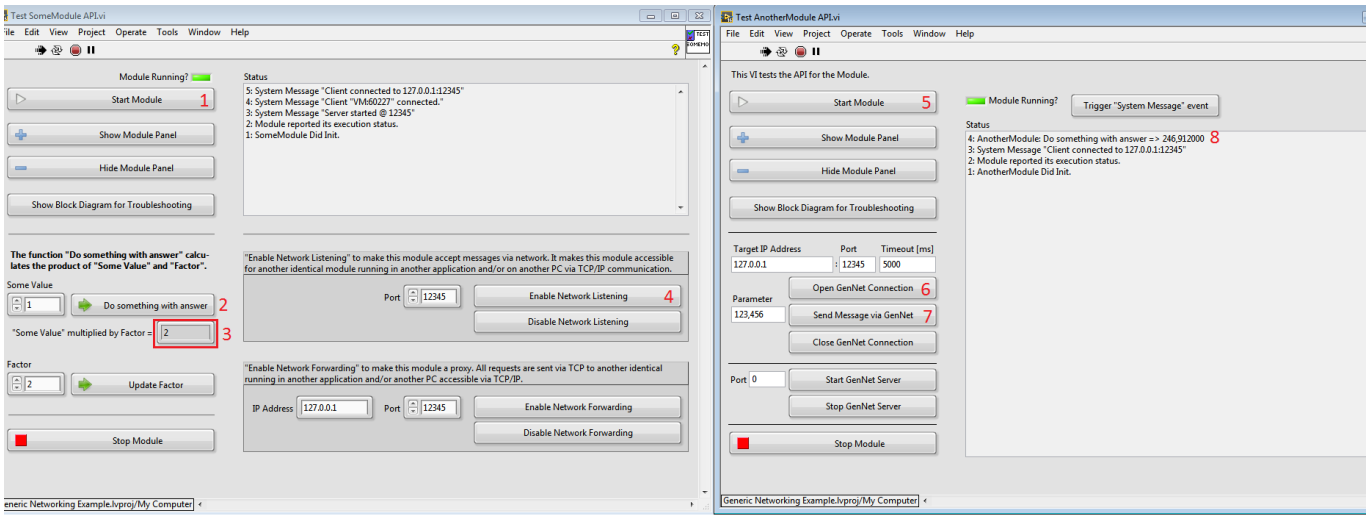
4. Start the `Test GenNet-Proxy API.vi` Tester, start the module and *Enable Network Listening*
5. In the API Tester of the `Windows Application` trigger *Do something with answer* requests with different `Some Values`
6. The API Tester of the `Realtime Application` displays multiple *Did something* messages in the status log
7. In the API Tester of the `Windows Application` change the `Factor` and trigger the *Update Factor* request
8. The API Tester of the `Windows Application` displays the "`Some Value" multiplied by Factor` = result correctly on the front panel

How to test RemoteControl as a client with GenNet-Proxy as a server

1. Open the `\Generic Networking Example_Source\Generic Networking Example.lvproj` file
2. Start the `Test GenNet-Proxy API.vi` Tester, start the module and *Enable Network Listening*
3. Start the `Test RemoteControl API.vi` Tester, start the module and *Open GenNet Connection*

Now, you have a network connection between the two modules: `RemoteControl` is using a `GenNet-Client` to send requests and receive broadcasts. `GenNet-Proxy` is using a `GenNet-Server` to receive requests and send broadcasts. Let us send a request from `RemoteControl` to the `GenNet-Proxy`:

- If you *Send Message via GenNet* in the `Test RemoteControl API.vi` Tester, it will actually call the *Do something with answer* request of the `GenNet-Proxy` module.
- Now, *Update Factor* in the `Test GenNet-Proxy API.vi` Tester and see how the result in the `Test RemoteControl API.vi` Tester changes when you call *Send Message via GenNet* again.



GenNet Roundtrip Example

This project contains a Generic-Networking-enabled module that showcases the use of broadcasts.

GenNet-RoundTrip

The `GenNet-RoundTrip` module is like the `GenNet-Proxy` module, but with a new feature for sending DQMH *Round Trip* events over the network. The normal *Round Trip* event also includes a DQMH broadcast, but this broadcast will only be fired locally. With the new feature, the normal broadcast will be encapsulated in a generic broadcast and the remote side can identify it. The `GenNet-RoundTrip` module showcases the concrete implementation.

PlainString Example

The PlainString example again showcases how communication between two separate applications that are connected through a network can be implemented with GenNet. In this case, we assume a scenario where an application running on Windows shall communicate with another application running on a PXI system (a real-time controller).

The two applications - we'll call them the “Windows Side” and “Real-Time Side” - use the PlainString protocol implementation for communicating data. Both DQMH modules implement the Manual Use Case.

Windows Side

The `PlainString Example - Windows Side.lvproj` contains a single, non-GenNet-enabled DQMH module: The *Windows Client module*. As the name suggests, it acts as the client that connects to its counterpart.

Real-Time Side

The `PlainString Example - Real-Time Side.lvproj` contains another non-GenNet-enabled DQMH module: The *PXI Server module*. This is the server-side that opens a network port and waits for incoming connections.