Testing Utility Toolkit for Güralp Data Box

16/03/2026 15:27:39

Designed and manufactured by
Güralp Systems Limited
3 Midas House, Calleva Park
Aldermaston RG7 8EA
England

Table of Contents

1. Overview

Once the Güralp Data Box (GDB) has been connected to the network (following the provided Quick Start Guide), the additional steps outlined in this document aim to guide the user through configuring and testing the registry and Discovery-hosted Modbus server using our Testing Utility Toolkit.

After following this document you should have:

- A Güralp Data Centre ready to record and archive MiniSEED records
- A Discovery pre-configured to
    - Start automatically (if closed opens again after 20 seconds)
    - Start a Modbus server automatically (port **11502**)
    - Start a CAP Receiver automatically (port **11900**)
- Example Python scripts used to query the Modbus server, found at:
    - `/home/guralp/query_modbus_server`
    - `/home/guralp/monitor_modbus_server`
- Our Testing Utility Toolkit used for:
    - Adding 3rd Party devices to the Registry
    - Sending Test CAP messages
    - Found at: `/home/guralp/testing_utility_toolkit`

The Testing Utility Toolkit can be run by double-clicking the executable file on the desktop, or by running

```
./testing_utility_toolkit
```

from a terminal on the GDB.

2. Add a Fake Device to Registry

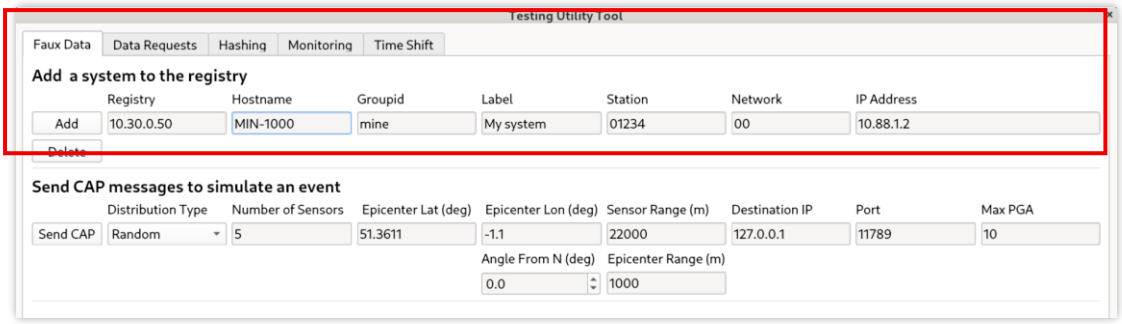

Figure 1 Testing Utility Toolkit can be used to test registry configuration as well as to test the CAP receiver tool in Discovery

The Testing Utility Toolkit can be used to test registry configuration by adding fake devices to a registry. Fake CAP messages can then be sent from these devices to test the CAP receiver and Modbus server in Discovery.

Enter the Registry IP (the IP address of the GDB, discoverable by running the command `ifconfig` in the GDB terminal) and change the Hostname field to MIN-1000. Groupid, Label, Station, Network and IP Address can be left as default. Click Add and ensure that the fake sensor is now visible in the Registry view of the main Discovery window. Repeat this process as many times as desired, incrementing the Hostname by 1 each time (i.e.: MIN-1001, MIN-1002... etc.).

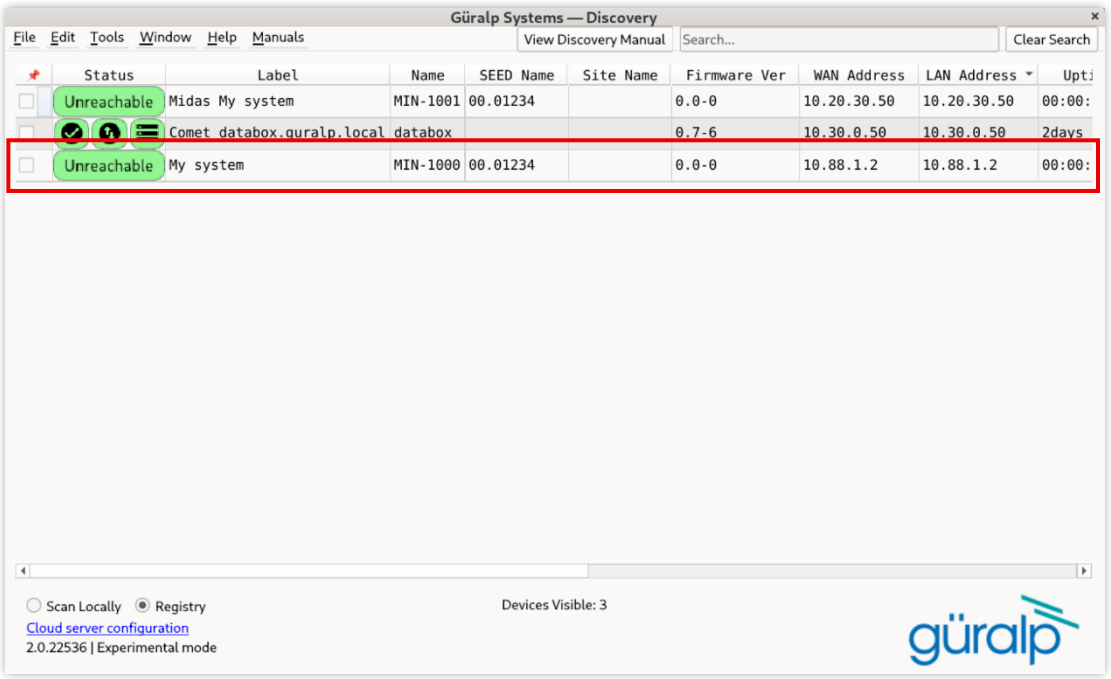

Figure 2 Fake devices can be added to a registry to test configuration (MIN-1000, MIN-1001)

3. Testing the CAP Receiver in Discovery

Discovery software should automatically start when the GDB receives power. When the main interface has loaded, the CAP receiver function can be accessed using the menu bar under Tools. The CAP receiver listens on a specified UDP port for incoming CAP messages.

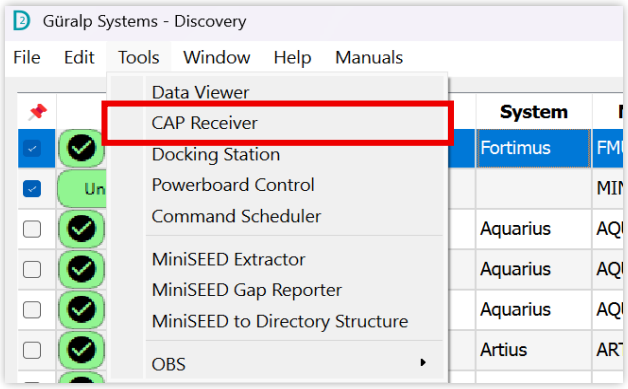

Figure 3 CAP receiver widget in Discovery is accessible through the Tools menu on the main page

When the CAP receiver window opens, navigate to the *Settings* menu. Under the Network tab, take note of the Port Number. When hosting a CAP receiver in Discovery on a GDB, the default port number should be **11900**. To start listening for CAP messages on the specified UDP port, click Start.

From within Testing Utility Toolkit, head to the section named "Send CAP messages to simulate an event". Change the number of sensors to 1. Additional sensors may be used, but first must be added to the registry following the same naming convention.

In the Destination IP field, enter the address of the device hosting the CAP receiver. As the Testing Utility Tool is being run on the same device as the server is being hosted, default localhost address **127.0.0.1** should be applicable for this testing procedure. Ensure the port number also matches the one configured in the CAP receiver (default **11900**).

Once configured, press Send. The CAP receiver should receive a message and display information in the table on the right-hand side.

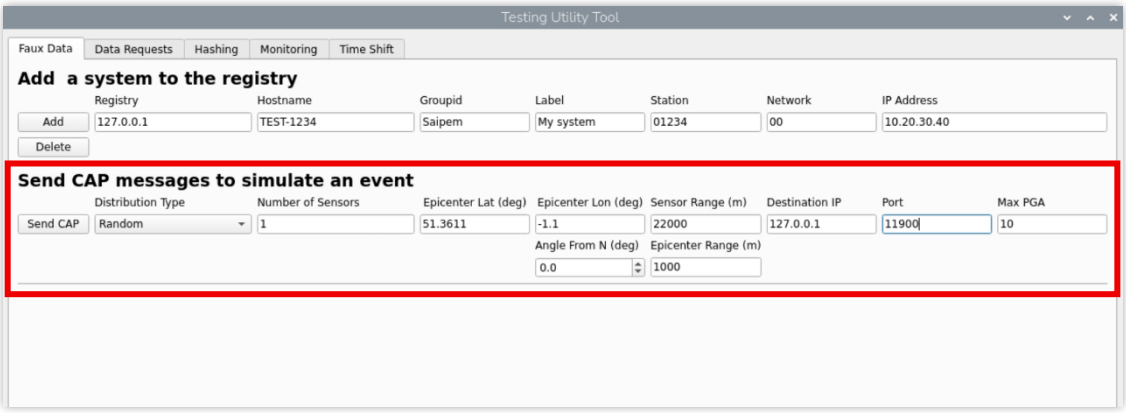

Figure 4 Testing Utility Toolkit can be configured to send fake CAP messages to a receiver from the fake devices added to the registry

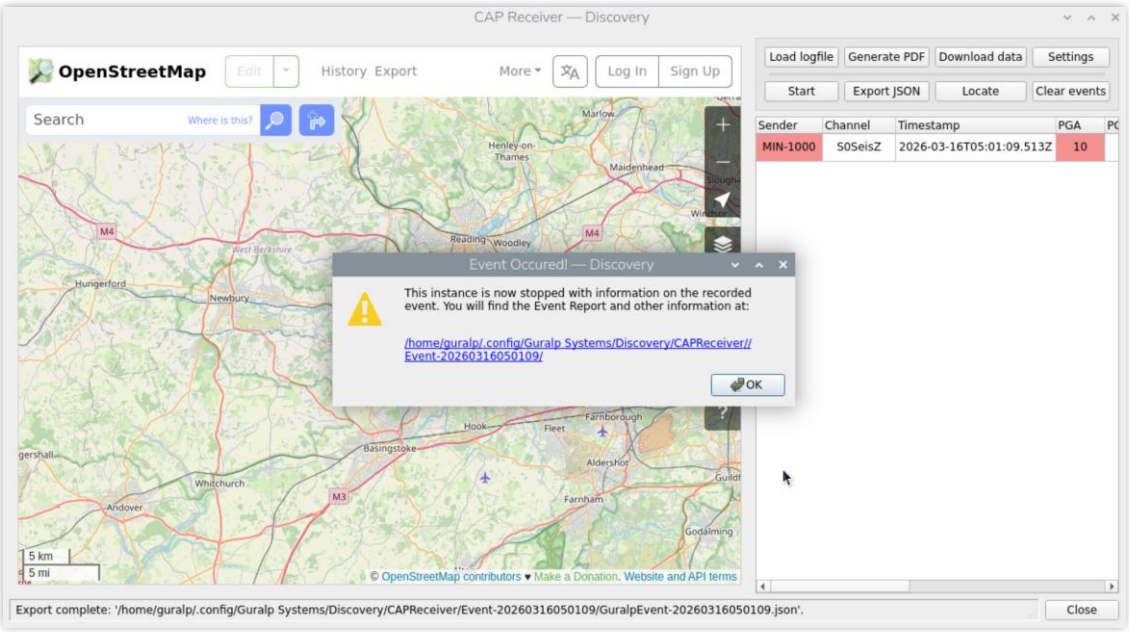

Figure 5 The fake device sends a fake CAP message to the specified address and port. The Discovery CAP receiver will receive this message and record the event.

4. Testing the Modbus Server in Discovery

The *Modbus Server* tool within Güralp Discovery allows for a Modbus TCP Server to be hosted from a device running the Discovery software. When run at startup, Discovery automatically begins hosting a Modbus server on the default port **11502**. To access configuration, the Modbus Server widget is accessible from the Tools menu in the main interface.

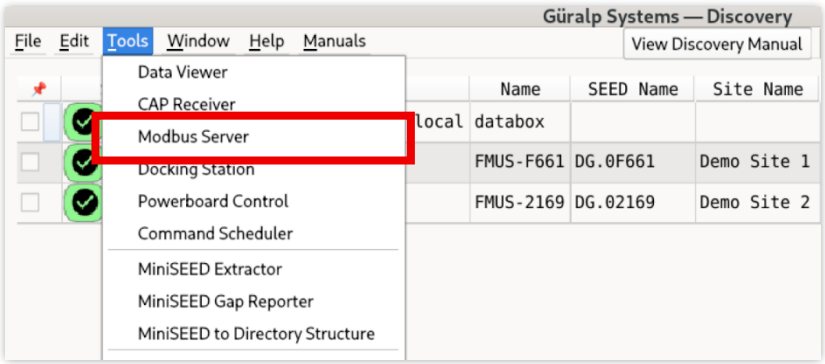

Figure 6 Discovery features a Modbus Server widget which enables communication of information to an outside client

To send a test query to the Modbus server, the GDB contains a Python script to print information currently held on the server. Open a terminal window and enter:

```
./query_modbus_server --ip [IP]
```

Where `[IP]` is the address of the server host. As the script is being run on the same device as the server is being hosted, the default localhost address **127.0.0.1** should be applicable for this testing procedure. The script takes additional arguments in the form of:

```
./query_modbus_server --ip [IP] --port [PORT] --slave [SLAVE]
```

Where `[PORT]` is the same port configured in the Modbus Server widget (default **11502**) and `[SLAVE]` refers to the register being queried (default 1). The script can be found in

```
/home/guralp/query_modbus_server
```

on the GDB for inspection.

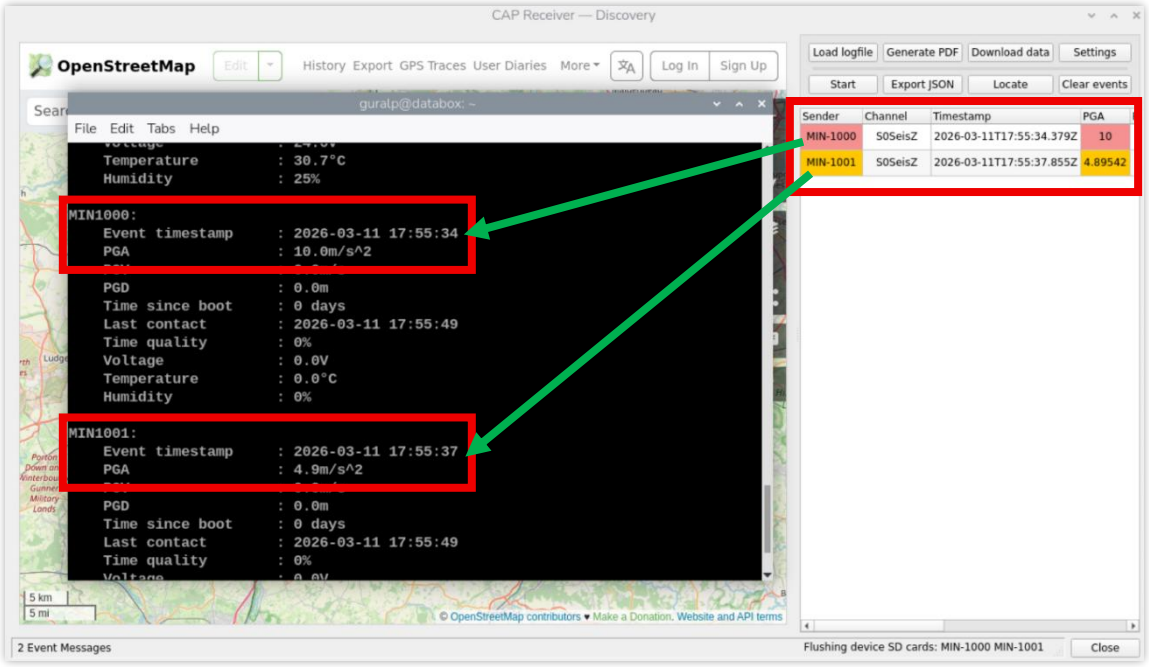

Figure 7 Fake events received by the CAP receiver can then be queried via the Modbus server using a Python script on the GDB.

Verify that the Sender name, Timestamp and PGA obtained from the query match the values configured in the Testing Utility Tool. If these values match, then the CAP receiver and Modbus server are correctly communicating between each other.

5. Site Fragility and Modbus System Alarms

The Modbus server system alarms are raised when significant ground motion is detected by the CAP receiver, according to the configured site fragility. Under the default configuration, the Modbus Server will set up three different alarms, which will alert the user to a change in circumstances that demand the user's attention. These alarms are as follows:

- **System Alarm High**: this indicates that ground movement has been detected, with PGA exceeding the user-defined "Safe" threshold but below the "Danger" threshold.
- **System Alarm High-High:** this indicates that ground movement has been detected with PGA exceeding the user-defined "Danger" threshold.
- **System Fault Alarm**: this indicates that a communication, timing, or storage issue has been identified with one or more of the devices from which data is being put in the server. It is tied to the state of health information for each device, displayed in the status column of the Discovery main window.

Site Fragility Threshold	Colour	Corresponding Modbus Alarm
Safe	Green	None
Warning	Yellow	System Alarm High
Warning-High	Orange	System Alarm High
Danger	Red	System Alarm High-High

Figure 8 table of different configurable site fragility thresholds, with corresponding CAP receiver colour codes and Modbus System Alarm

Site fragility can be tested using an example file included with the GDB file by enabling site fragility from within the settings widget and uploading the fragility test file. The file is configured with arbitrary safe, warning and danger PGA thresholds for up to 5 test devices added via the Testing Utility Toolkit. (MIN-1000, MIN-1001, MIN-1002, MIN-1003, MIN-1004). The file is found at

```
/home/guralp/fragility_test.csv
```

When an alarm is raised, it will remain raised until the Modbus server is next queried, or based on a timeout (configurable in Discovery). Ensure the alarm is being raised by sending a fake CAP message with significant ground motion (e.g.: PGA = 10) and ensure that the corresponding system alarm is raised by querying the Modbus server.

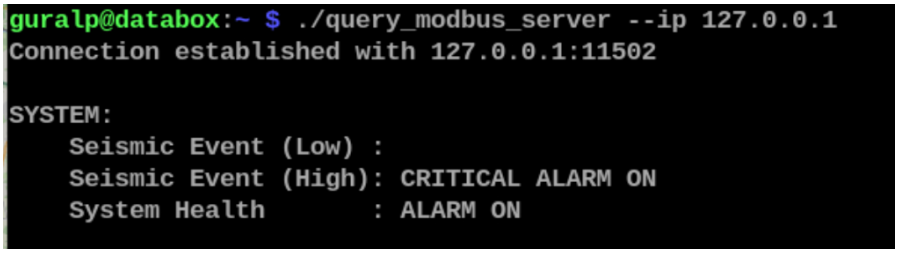

Figure 9 System alarms will be raised on the Modbus server when the CAP receiver receives an event message with significant ground movement

6. Further Options – Monitoring the Modbus Server

In addition to `query_modbus_server`, the GDB contains a Python script named `monitor_modbus_server` which automatically queries the Modbus server at a fixed time interval. This program will run in the background and do nothing until there a system alarm is raised, caused either by a system fault or by an event PGA exceeding the user-defined warning threshold using the site fragility function. When this happens, a pop-up window will display the relevant information. The script can be stopped at any time by clicking Quit. The script is stored at

```
/home/guralp/monitor_modbus_server
```

Usage of the script is identical to `query_modbus_server`, with one additional argument for query interval (in seconds).

```
./monitor_modbus_server --ip [IP] --port [PORT] --slave [SLAVE] --interval
[INTERVAL]
```

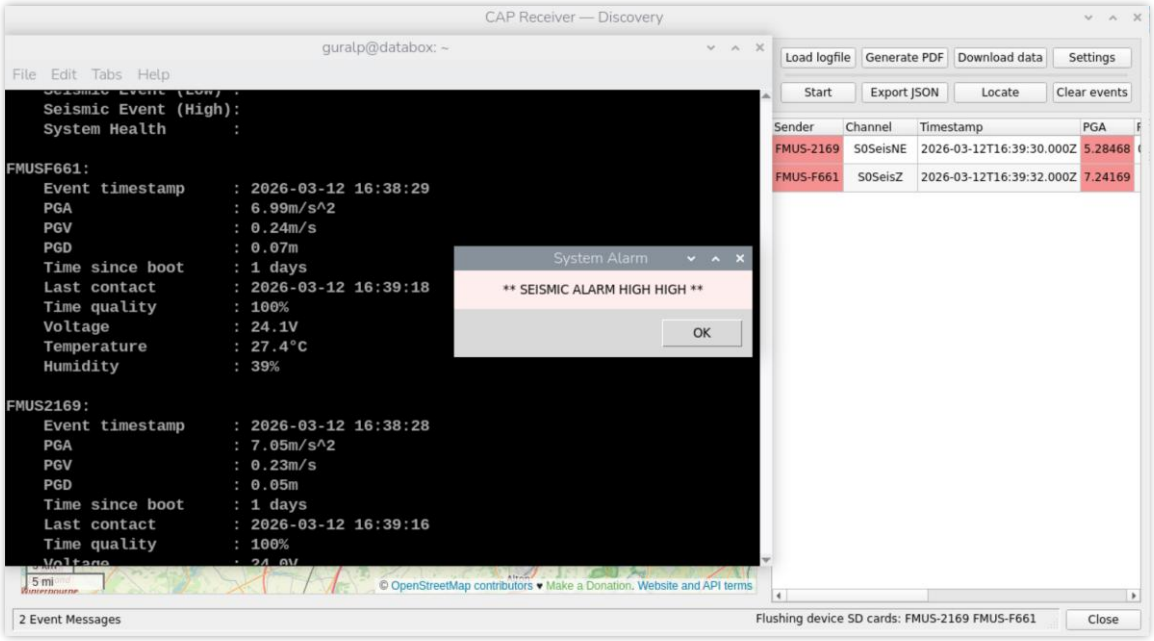

Figure 9 The GDB contains a Python script which can be used to continuously monitor the Modbus server hosted by Discovery. The script will listen for system alarms declared by the CAP receiver and generate a pop-up window warning the user when an alarm occurs