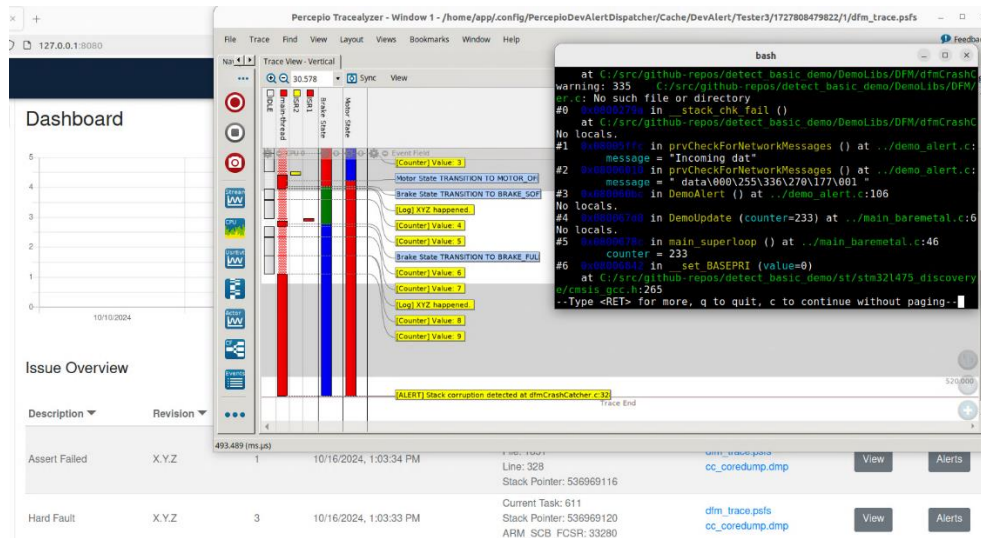


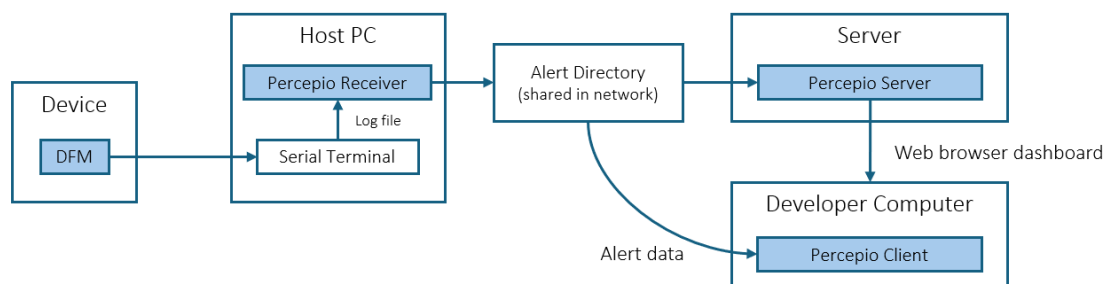
# Percepio Detect Demo Guide (Linux)

Percepio Detect™ is the new addition to Percepio's portfolio of continuous edge observability tools for embedded systems, specifically targeting embedded test monitoring.



This is designed to capture crashes, errors and anomalies, also sporadic issues that otherwise are difficult to analyze, and provide deep observability to simplify debugging.

The Percepio Detect solution consists of four parts, as described below.



- **Percepio DFM**, a C library for use in the device software. This outputs "alert data" to a host computer, for example via a serial port. The device output should be saved to a log file on the host computer.
- **Percepio Receiver** ("Receiver"). Reads the log files from the device, extracts the alert data and saves it as alert files. This typically runs on a lab computer or developer computer, directly connected to the device via serial, ethernet or similar.
- **Percepio Detect Server** ("Server"). Reads alert files from Receiver and presents a summary in the web browser (the "Dashboard"). Provides access to alert payloads (e.g. traces and core dumps) for deeper analysis/debugging.
- **Percepio Payload Viewer Client** ("Client"). An integrated set of developer tools for debugging alerts, including Tracealyzer and tools for viewing core dumps. Runs on each user's computer to make it easy to debug reported issues.

## Typical Setups

Percepio Detect is designed to be a team solution that runs on a local server, where all team members have easy access to the data. All data stays on the local server. However, the whole solution can easily be deployed on a single computer if so desired.

- **Single-user setup:** All parts of the solution run on the same computer, except for the DFM library on the device. This is suitable for small projects where a single user is sufficient, and for a first test of the solution.
- **Multi-user setup:** The Server runs on a shared server computer. Each user runs the Client on their local development computer. The Receiver runs on a "lab computer", connected to the device with a serial terminal or similar. To monitor multiple devices, you can have multiple Receivers running on the same computer. If needed, you can also run the Receiver on multiple lab computers, each monitoring one or multiple devices.

## Preparation Steps (Linux)

1. Make sure Docker Engine is installed. See <https://docs.docker.com/engine/install/>.
2. Run "docker run hello-world" to see that Docker works as expected.
3. Install "x11-xserver-utils" using e.g. "sudo apt-get install x11-xserver-utils"

## Running the Demo

After you have completed the preparation steps above, follow these steps to run the demo.

1. Extract the provided installation file (.tgz) to any suitable location on your computer.
2. Add your Detect license key in the server start script, **percepio-server.sh**. Locate the assignment of the LICENSE variable and update it like:

```
LICENSE="ABCD-ABCD-ABCD-ABCD"
```

Note that Tracealyzer and Detect Server have separate license keys, don't mix them up!

3. Make sure the test-data/alert-files directory is accessible also for "other" users, like the server and client Docker containers. For example, by running:

```
sudo chmod -R 777 test-data/alert-files
```

4. Make sure the two start scripts are executable by running:

```
sudo chmod +x percepio-server/percepio-server.sh
sudo chmod +x percepio-client/percepio-client.sh
```

5. Enter the percepio-server folder and start the server by running:

```
./percepio-server.sh start
```

Docker will download multiple images from Docker Hub, which may take a while.

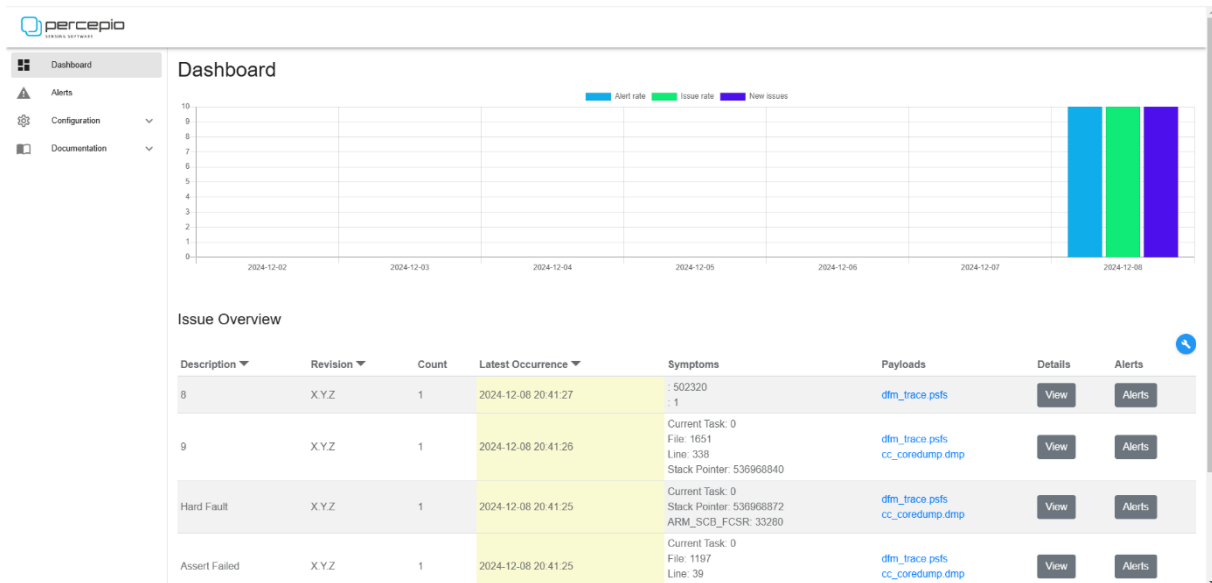
6. Open a second terminal, navigate to the percepio-client folder and run:

```
./percepio-client.sh
```

Docker will download the client image from Docker Hub, which may take a while.

7. Open <http://127.0.0.1:8080> and check that you see the Server dashboard. The demo data typically show up within 5-10 seconds.

You should now see the Detect Server dashboard.



The demo uses some custom "alert types" and "symptom" IDs, that are not defined in the Server by default. As a result, you see numeric values in the Description column and in the Symptoms.

To add the missing definitions, follow the steps below:

7.1. Open **Configuration -> Alert Types**. Select **Add new Alert Type** (button in top right corner)

Type	Definition	Description	State	Updated	Actions
7	DFM_TYPE_HEARTBEAT	Heartbeat failure	Published	2024-12-08 20:40:49	Alerts Edit
6	DFM_TYPE_BAD_MESSAGE	Invalid/bad message received	Published	2024-12-08 20:40:49	Alerts Edit
5	DFM_TYPE_OVERLOAD	CPU Overload	Published	2024-12-08 20:40:49	Alerts Edit

We need to add two additional Alert Types. For the first, use


Definition: **DFM\_TYPE\_STOPWATCH** and Description: Stopwatch alert

Click "Submit"

Add another new alert type for the stack integrity alerts.

Definition: **DFM\_TYPE\_STACK\_CHK\_FAILED** and Description: Stack corrupted

Click "Submit"



Dashboard

Alerts

Configuration

Products

Devices

Alert Types

Symptoms

Code Export

Documentation

Back

New Alert Type

Definition

DFM\_TYPE\_STOPWATCH

Description

Stopwatch alert

Severity

1

Submit

7.2. Open **Configuration** -> **Symptoms** and select **Add new Symptom** (top right corner).


Use Definition: **DFM\_SYMPTOM\_HIGH\_WATERMARK** and Description: High watermark

Click "Submit"

Add a second symptom:

Use Definition: **DFM\_SYMPTOM\_STOPWATCH\_ID** and Description: Stopwatch ID

Click "Submit".



Dashboard

Alerts

Configuration

Products

Devices

Alert Types

Symptoms

Code Export

Documentation

Back

New Symptom

Definition

DFM\_SYMPTOM\_HIGH\_WATERMARK

Description

High watermark

Data type

Decimal

Significant

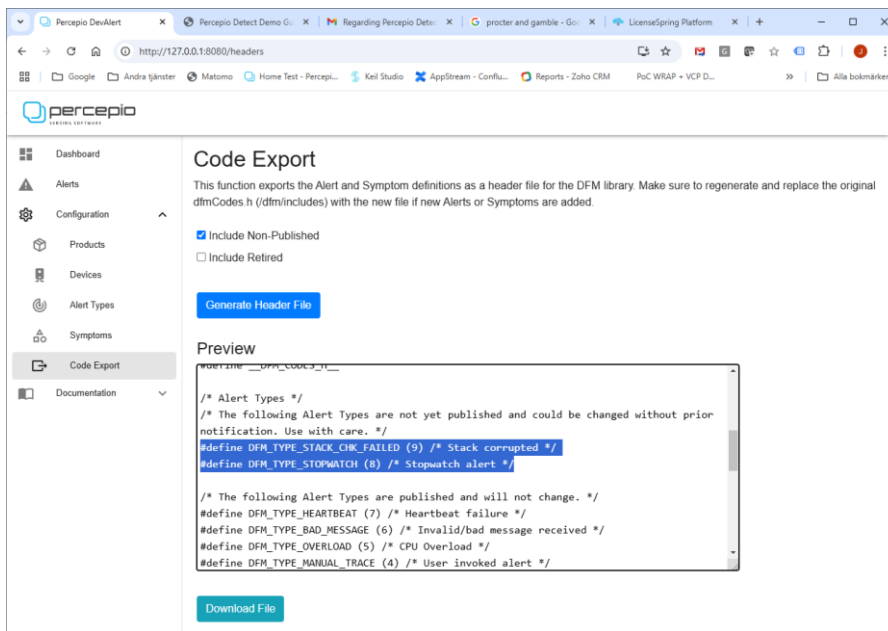
No

Submit

A symptom is an attribute of the alerts, set by the device, mainly used for grouping the alerts.

Note that the “Definition” only affects the name used in the Code Export (for creating dfmCodes.h). You can use other names if you like. Just make sure the IDs match the alert data.

### 7.3. Open Code Export and select Generate Header File (blue button).

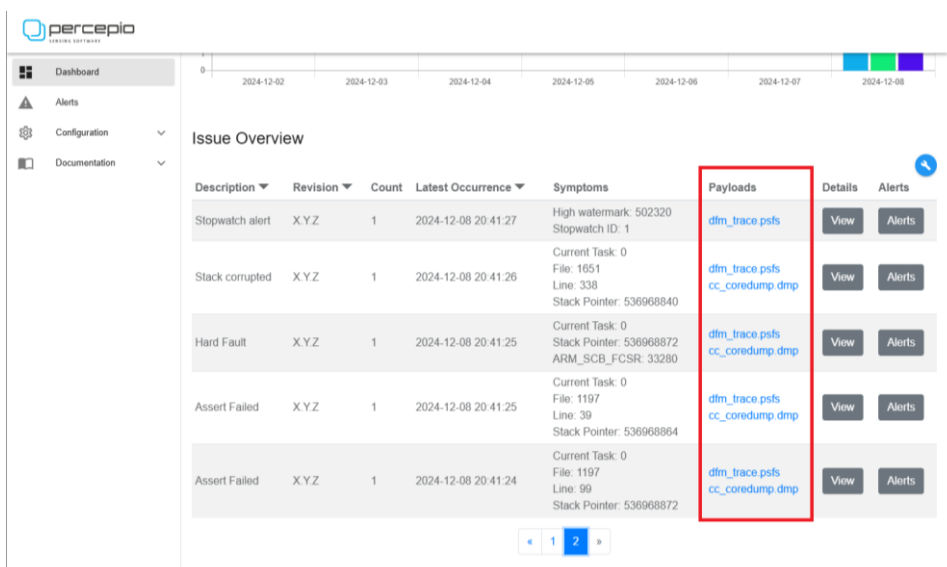


Inspect the code in Preview and verify that the right numeric IDs have been set, so they match the already created alert data in the demo. (If not, you can rename the entries by selecting Edit).

- DFM\_TYPE\_STACK\_CHK\_FAILED should be (9)
- DFM\_TYPE\_STOPWATCH should be (8)
- DFM\_SYMPTOM\_STOPWATCH\_ID should be (9)
- DFM\_SYMPTOM\_HIGH\_WATERMARK should be (8)

When adding new alerts and symptoms, copy these definitions to DFM/include/dfmCodes.h. But in this case, dfmCodes.h is already updated.

**8.** In the lower part of the dashboard, you find "Issue Overview" showing the reported issues, including the names added in the previous step.



The "Payloads" column provides links to the alert payloads, i.e., the debugging data included with the alerts from the DFM library on the device.

Note that "Issue Overview" is an aggregated view. All alerts with the same alert type and symptoms are considered as the same "issue" and represented by one row in the Issue Overview. The payload links points to the latest alert of each issue. (You can see all individual alerts on the "Alerts" page.)

**9.** Click on one of the Payload links. This will notify the Client to show the selected data in the appropriate tool. The tools are included in the Client and ready to run.

- Percepio Tracealyzer for showing TraceRecorder traces (the "dfm\_trace.psfs" links)
- Core dump viewer for showing CrashCatcher core dumps (the "cc\_doredump.dmp" links)

**10.** To install your Tracealyzer license key, first click on a "dfm\_trace.psfs" trace link to start Tracealyzer. On the welcome screen, select the option "Activate License". Enter your Tracealyzer license key, close the application and click the link again to verify that it works.

Note: If you want to update Tracealyzer, make sure to overwrite the bundled Tracealyzer installation in the `percepio-client-windows/tracealyzer` directory.

More information on how to set up and customize Percepio Detect is found in:

- `readme.txt`
- `percepio-client-linux-docker/readme-client.txt`
- `percepio-server/readme-server.txt`
- `percepio-receiver/readme-receiver.txt`

For assistance and technical questions, please contact [support@percepio.com](mailto:support@percepio.com).

For sales and licensing questions, please contact [sales@percepio.com](mailto:sales@percepio.com).

See <https://percepio.com/detect/> for more information about Percepio Detect.