Technical Documentation

EU HRS Availibility Monitoring

ENDA GmbH & Co. KG — Environmental Data Management Solutions

Nicolai Buchwitz, n.buchwitz@enda.eu

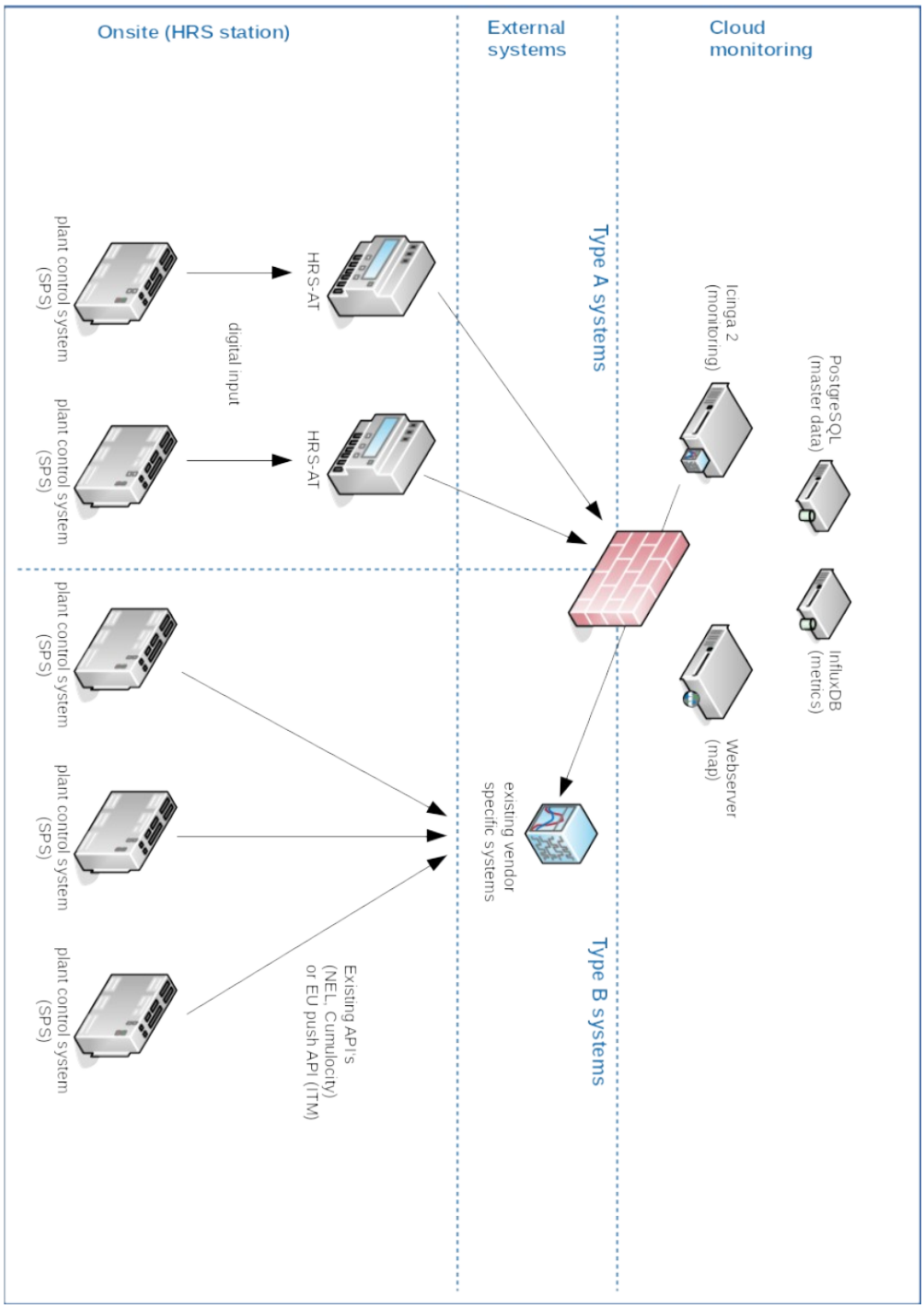
Sophy Klempert, s.klempert@enda.eu

Berlin, 2018-07-09

Rev. 1.0.0, 2018-07-09

Inhaltsverzeichnis

1 System architecture



2 On-site data acquisition

2.1 Type A: HRS-AT

Type A stations use the HRS availability transmitter (HRS-AT) to collect the availability and maintenance signals from the plant control system via 24 V digital inputs (DIO module). In order to prevent malicious operations, there is no interaction with the control system.

The HRS availability transmitter is based on an industrial grade micro-computer (Revolution Raspberry Pi) with standardised connectors for up to 10 input signals and is designed for use in typical switchgear construction, with DIN rail mounting and 24 V DC power supply. If 24 V DC power supply is unavailable on site, an external power supply unit can be mounted alongside the transmitter. Similarly, an external cell modem may be mounted wherever no local internet connection is available.

The signals are transmitted via an encrypted channel to the cloud based EU HRS platform.

2.1.1 Icinga 2 check scripts

There are standard check scripts for hardware health and operational metrics in place, such as CPU load, disk usage and update status. Furthermore, two custom check scripts are used to fetch signals from the Revolution PI DIO module.

Both scripts are located in the project source tree at *`hrs-at/hrs-at-check/`*.

2.2 Type B: Cumulocity, NEL, ITM

Type B stations do not require additional hardware to transmit a signal from the site to the HRS availability platform. The existing vendor-specific monitoring systems are already connected to on-site control systems and are used to send a periodic availability signal to the EU HRS availability platform. The signals of both the german legacy system (Cumulocity) and NEL are fetched by their preexisting pull interfaces, while ITM uses the standardised EU-Push-API. The metrics are gathered directly on the monitoring server via Icinga 2 check commands.

2.2.1 EU PUSH API

HRS operators and providers who already have an availability monitoring system in operation may send the metrics via a standardised REST-API, according to the following specifications:

Set availability status

Method	POST
URL	api.h2-map.eu/v2/availability/{{ STATION_UUID }}
Example Payload (JSON)	{ "type": "700bar-car", "available": true, "limited": false }
Allowed values	
type	700bar-car, 350bar-car, 350bar-bus
available	yes, no, true, false, 0, 1
limited *(optional)*	yes, no, true, false, 0, 1

Set availability status (bulk mode)

Method	POST
URL	api.h2-map.eu/v2/availability
Example Payload (JSON)	["{{ STATION_UUID }}": { "type": "700bar-car", "available": true, "limited": true }, "{{ STATION_UUID }}": { "type": "700bar-car", "available": false }]
Allowed values in each station object	
type	700bar-car, 350bar-car, 350bar-bus
available	yes, no, true, false, 0, 1
limited *(optional)*	yes, no, true, false, 0, 1

Set maintenance mode

Method	POST
URL	api.h2-map.eu/v2/maintenance/{{ STATION_UUID }}

Example Payload (JSON)	{
	"maintenance": true
	}

Allowed values	
maintenance	yes, no, true, false, 0, 1
begin *(optional)*	ISO 8601 date with time (e.g. *2007-08-31T16:47+00:00)*
end *(optional)*	ISO 8601 date with time (e.g. *2007-08-31T16:47+00:00)*
message *(optional)*	Description of the planned maintenance

Set maintenance mode (bulk mode)

Method	POST
URL	api.h2-map.eu/v2/maintenance

Example Payload (JSON)	[
	"{{ STATION_UUID }}": {
	"maintenance": true
	},
	"{{ STATION_UUID }}": {
	"begin": "*2017-08-13T10:00+00:00*",
	"begin": "*2017-08-15T12:00+00:00*",
	"message": "*annual maintenance*",
	}
	]

Allowed values in each station object	
maintenance	yes, no, true, false, 0, 1
begin *(optional)*	ISO 8601 date with time (e.g. *2007-08-31T16:47+00:00)*
end *(optional)*	ISO 8601 date with time (e.g. *2007-08-31T16:47+00:00)*
message *(optional)*	Description of the planned maintenance

3 Data processing and visualization

A typical Icinga 2 monitoring stack with InfluxDB, Grafana and Icinga Web 2 with addons is used for the server-side data processing.

3.1 Monitoring server

Icinga 2 is the state-of-the-art open source monitoring suite, which can be used both in a distributed environment (satellite) and directly on a master server. In the EU HRS availability plattform Icinga 2 is installed on the cloud monitoring server, as well as on each HRS-AT. Separate InfluxDB and PostgreSQL databases are used to store the performance data (metrics).

Since the installation of the components depends strongly on the underlying architecture, for which there are no restrictions in place, we refer to the respective official documentations for installation.

Icinga 2	https://www.icinga.com/docs
Icinga Web 2	https://www.icinga.com/docs/icingaweb2/
InfluxDB	https://docs.influxdata.com/influxdb/
PostgreSQL	https://www.postgresql.org/docs/
PostGIS	https://postgis.net/documentation/
Grafana	https://docs.grafana.org/

In addition to the basic Icinga Web 2 installation the following modules are used to configure the systems as well as to display and export status data:

Icinga Director	https://www.icinga.com/docs/director/
Grafana module	https://github.com/Mikesch-mp/icingaweb2-module-grafana
Map module	https://github.com/nbuchwitz/icingaweb2-module-map
Business Process	https://github.com/Icinga/icingaweb2-module-businessprocess

Although databases are often installed on the monitoring server for demonstration purposes, we recommend to use separate hosts in order to simplify scalability with increasing distribution. Since gathering the performance data demands writing many small files within short time intervals, we also recommend a fast server with first-grade SSD storage.

These software components are provided in their currently employed versions in the projects source tree.

3.2 Map visualization

HRS availability metrics are exported via a custom version of the map module in the geojson format. This geojson data can also be used for third party API integrations.

3.2.1 Export API

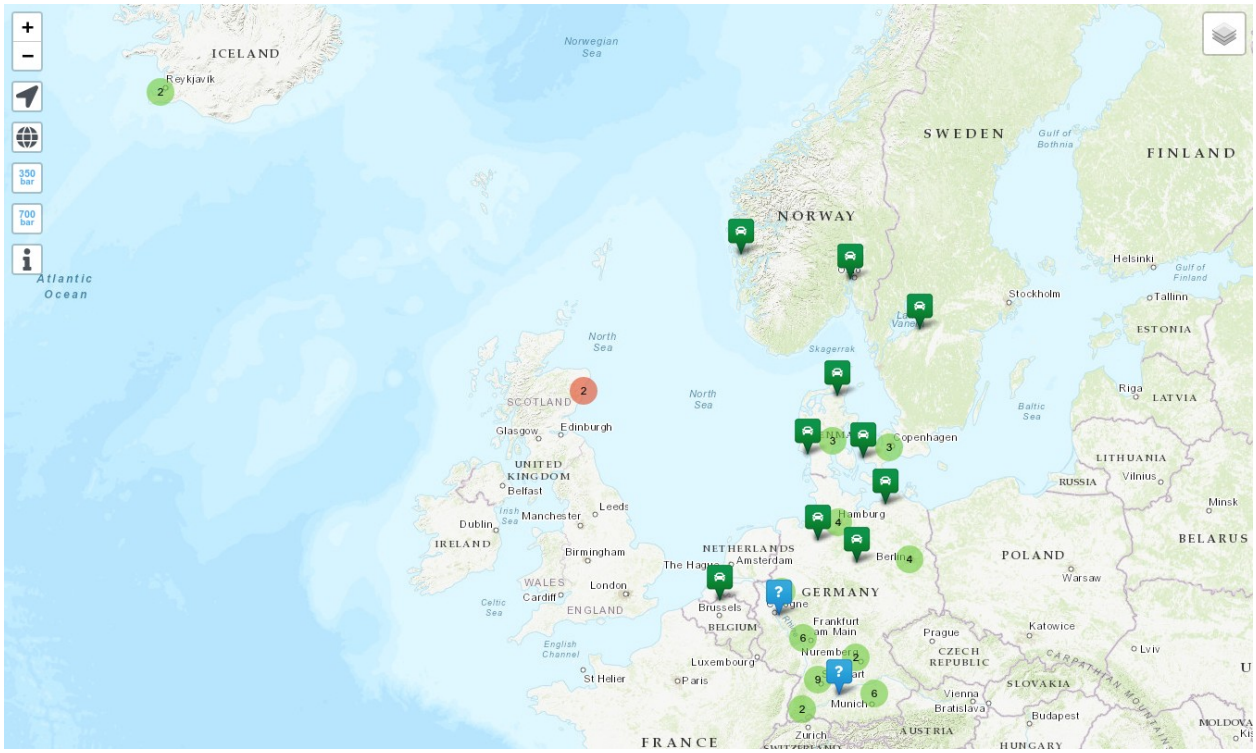
Method	GET (x-form/url-endcoded)
URL	h2-map.eu/data/

Example Payload (JSON)	{
	"pressure": 700,
	"vehicle": car
	}
	
	or
	
	{
	"pressure": 700,
	"vehicle": car,
	"stations": [12, 41, 102, 39]
	}

Allowed filters	
pressure *(optional)*	700, 350
vehicle *(optional)*	*car, bus*
stations *(optional)*	*(list of station IDs)*
Data fields	
id	HRS station id
geometry	*Geojson encoded point coordinates*
properties	*HRS station master data*

last_update	Timestamp of the last signal transmission
status	0 = OK, 1 = Limited availability, 2 = Maintenance, 3 = Malfunction, 4 = Unknown because no connection to the HRS (type A) or last Status transmission too long ago (Type B) 5 = Filling station is closed (opening hours)

3.2.2 Map



All stations are displayed on an interactive map based on OpenStreetMap tiles, which is built on the following components:

Leaflet	https://leafletjs.com/
Jquery	https://jquery.com/
Leaflet Easy Button	https://github.com/CliffCloud/Leaflet.EasyButton
Font Awesome	https://fontawesome.com/

Every modern web server (e.g. nginx or Apache 2) together with a recent PHP version (>7.0) should be sufficient to run the map frontend. We recommend to use nginx with php-fpm built from the official sources.

The source code for the interactive map is located in the project source tree at *server/h2-map/*.