SAMS Data Warehouse

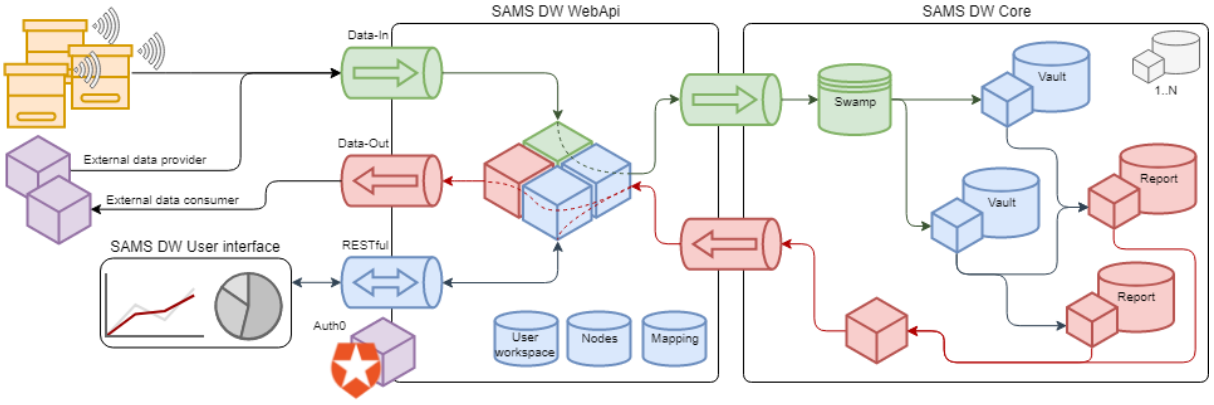
Instruction on how to connect general bee colony monitoring hardware to SAMS Data Warehouse

SAMS is a project funded by the European Union within the H2020-ICT-39-2016-2017 call. SAMS enhances international cooperation of ICT (Information and Communication Technologies) and sustainable agriculture between EU and developing countries in pursuit of the EU commitment to the UN Sustainable Development Goal “End hunger, achieve food security and improved nutrition and promote sustainable agriculture”.

To get more information about the SAMS project please visit: https://sams-project.eu/

The SAMS Data Warehouse (DW) is available online at https://sams.science.itf.llu.lv/

The SAMS DW is a universal system, which can operate with different data inputs and have flexible data processing algorithms. Its warehouse architecture is demonstrated below:



Users can sign up into the DW using their own email/password or google account. Each user is associated with one or more workspaces.

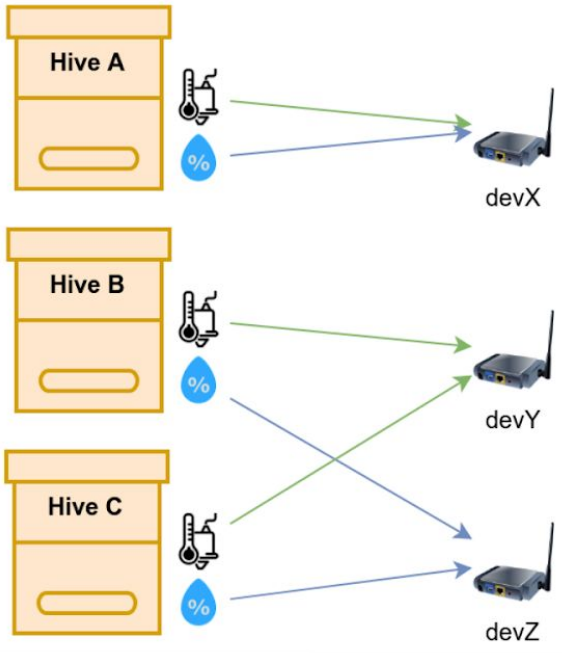
The SAMS DW solution uses authentication and authorization services provided by the *Auth0* universal platform (https://auth0.com/). *Auth0* is a ready-to-use platform with a wide range of built-in authorization related functionality and integration options. In particular, the SAMS DW uses specific authorization flows for Web and non-interactive applications, and

completely delegates user credential handling and access administration functionality to the platform. From a development perspective, usage of the *Auth0* platform removes the need to create custom solutions for secure user credential storage, provides user experience for Sign-in and Log-in flows, and simplifies administration tasks.

General concepts

Within the system following concepts are used:

- **Workspace** is a logical grouping of entities and can be considered as a customized working environment. By default, each user has one workspace, but more workspaces can be created.
  The main benefit of workspaces is the possibility to share them with other users. This way multiple users can work on the same objects in a manageable way.
- **Nodes** are any objects or groups of objects relevant for the beekeeping business. These include apiaries, hives, hive elements, devices, etc.
  Usually nodes are organized into hierarchical structures for logical overview, however this is not mandatory and users can arrange nodes according to their need. In the DW all nodes are associated with one workspace.
  Usually users want to receive data about the node (e.g. temperature and weight about hive, power level for the solar panel, income rate about apiary, etc). Data can be different for each node.
- **Device** is a special type of a node used to communicate with the DW and to push measurements into the system. It contains additional information about credentials (Client ID) which are used for data exchange.
- **Client ID** is a special type of username used by the device and recognized by DW during communication sessions. This ID is not easily readable by the user and contains a string of symbols, like [sb75X2K0dHt1Xl0cr5crcL0T8cvUB52j].
- **Data source** is indicated by a globally unique ID and represents a channel of data (usually measurements) which is sent by the device. In general, any random ID can be used as a data source identifier. But it is recommended to use sensor prefix and Client ID of the device for data source ID, like [ds18-sb75X2K0dHt1Xl0cr5crcL0T8cvUB52j].
- **Mapping** is a relation between incoming data source IDs, target Node and value meaning. The main purpose of the mapping is to ease the hive/device reorganization without having to change the embedded code - mapping is used to route the incoming data stream to a related node (see figure below).



Source	Object	Meaning
temp@devX	Hive A	temperature
hum@devX	Hive A	humidity
temp1@devY	Hive B	temperature
temp2@devY	Hive C	temperature
hum1@devZ	Hive B	humidity
hum2@devZ	Hive C	humidity

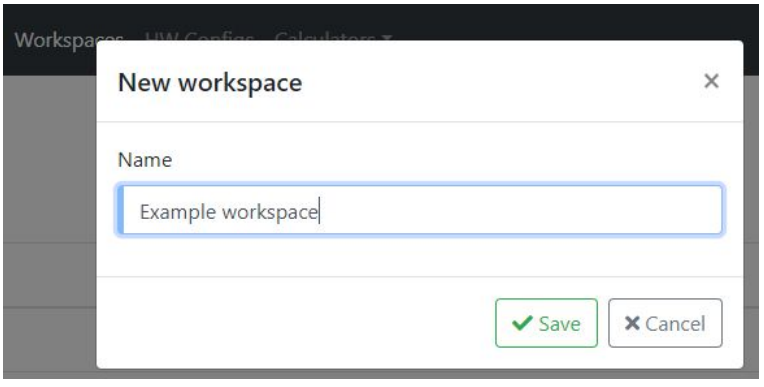
Node configuration example

As an example, let's consider the following scenario:

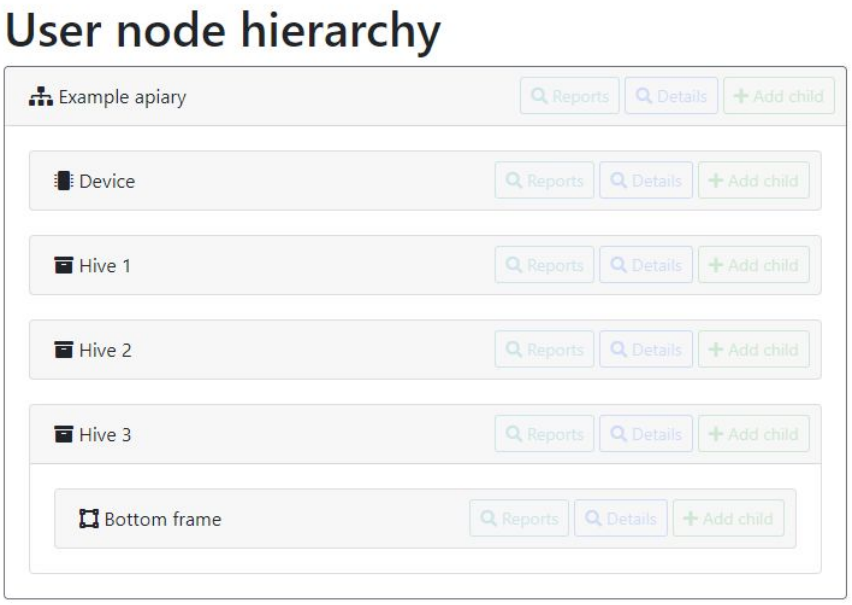
- a user has small an apiary with 3 hives,
- two hives have temperature sensors installed above the frames,
- one hive has two temperature sensors on the top and bottom parts of frames,
- all sensors are connected to a single device,
- the user works in this apiary with his friend.

The configuration process might be as follows (but not necessarily):

- As the user wants his friend to have access to the apiary, he creates a separate workspace for the apiary and share it with his friend.



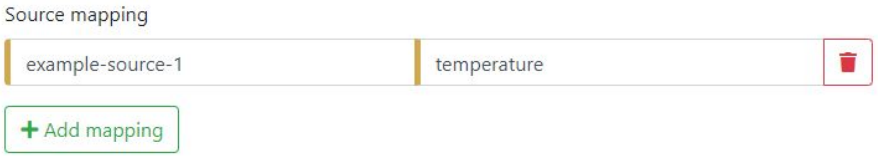
- Node hierarchy example:



  Note: The 3rd hive has two temperature sensors which should be distinguished, so that a separate element for the bottom frame is created.

- The device is configured with a Client ID (the suffix *@clients* should be added to the client ID):

- All hives and bottom frame element have similar data source mappings (but different source IDs):



Sending data to the DW

The following steps have to be performed on the device to send data to the system:

1. **Acquire access token**. The access token is used by the DW to authenticate and authorize the request. In order to acquire the token, the device should send a POST request to https://sams.science.itf.llu.lv/api/token with its Client ID and secret.
   Each device has its ***unique credentials*** and they have to be ***requested individually*** from the SAMS DW maintainers (see contacts below).
   The request body should have the following schema:

```
{
    "client_id": "AsD...QwE",
```

```
    "client_secret": "LkJ...MnB",
    "audience": "sams-dwh-web-api",
    "grant_type": "client_credentials"
}
```

It is recommended to provide content type header as well. An example of full *cURL* command for token request:

```
curl --request POST \
  --url https://sams.science.itf.llu.lv/api/token \
  --header 'content-type: application/json' \
  --data
'{"client_id":"AsD...QwE","client_secret":"LkJ...MnB","audience":"sams-dwh-web-api",
"grant_type":"client_credentials"}'
```

As response, the device gets an access token, which is typically valid for 24 hours (the exact expiration time is provided in the response).

```
{
    "access_token": "eyJ0e...1J6dA",
    "scope": "device",
    "expires_in": 86400,
    "token_type": "Bearer"
}
```

2. **Post the data to DW**. There is a single endpoint for posting data to the DW – https://sams.science.itf.llu.lv/api/data. The access token should be provided in the Authorization header and the request body can contain multiple data packages.

   An example of *cURL* command for posing the data:

```
curl -X POST \
  https://sams.science.itf.llu.lv/api/data \
  -H 'Authorization: Bearer eyJ0e...1J6dA' \
  -H 'Content-Type: application/json' \
  -d '[{
        "sourceId": "temperature-97834523476534654",
        "values": [
            {
                "ts": "2020-02-17T10:15:00Z",
                "value": 44.4
            }
        ]
      }]
```

   In response a report about each posted data source is provided, similar to this:

```
{
    "temperature-97834523476534654": "Ok",
    "humidity-97834523476534654": "NotFound"
}
```

   Statuses might be also:

   - *NotFound* – mapping for given source is not defined;

- *AccessDenied* – there is a mapping, the device is not authorized to post data for the mapped node (the Client ID is not registered among user devices, or the device is not active).
- *CoreTemporarilyUnavailable* – the posted data package was received, but was not stored in the DW due to temporary service outage (e.g. maintenance).
- *ValidationError* – the posted data package was received but failed schema validation (see below).

3. **Reports** about posted measurements are immediately available in the UI under the Reports section. Additional debugging information is available in the Dashboard (last 10 received measurements) and Devices (last events and errors).

Data package schemas

General data package schema used for posting data to the DW is as follows:

`$root: array`	required	one or more packages for different data sources
`sourceId: string`	required	unique source ID
`values: array`	required	one or more measurements from given data source
`ts: string`	required	zoned timestamp of measurement (ISO 8601)
`value: number`	required one	scalar measurement value (e.g. temperature)
`values: number[]`	required one	array of measurement values (e.g. audio spectrum)

An example of data package:

```
[{
    "sourceId": "temperature-123",
    "values": [{
        "ts": "2020-02-10T10:15:00Z",
        "value": 44.4
      },
      {
        "ts": "2020-02-10T10:20:00Z",
        "value": 41.2
      }
    ]
  },
  {
    "sourceId": "audio-123",
    "values": [{
      "ts": "2018-10-10T10:15:00Z",
      "values": [10.1, 20.2, 30.3]
    }]
  }
]
```

For devices **without hardware clock** another package schema is allowed:

`$root: array`	required	one or more packages for different data sources
`sourceId: string`	required	unique source ID
`tint: number`	required	measurement interval in seconds
`values: array`	required	one or more measurements from given data source
`value: number`	required	scalar measurement value (e.g. temperature)

In this case, the DW will use server local time as a time reference point (the moment when the request received in server) and calculate timestamps for all provided measurement values taking into account measurement interval. The DW assumes that the value array is ordered by measurement time and the request was sent immediately after the latest measurement.

* For any questions related to the Data Warehouse and connection of the hardware to it please contact the team from Latvia University of Life Sciences and Technologies, Faculty of Information Technologies:

Vitalijs Komasilovs: vitalijs.komasilovs@llu.lv

Aleksejs Zacepins: aleksejs.zacepins@llu.lv

Armands Kviesis: armands.kviesis@llu.lv

This project receives funding from the Horizon 2020 European Union Research and Innovation Framework under **Grant Agreement Nr. 780755 - SAMS**