GST Borehole

Service Manual

GiGa infosystems 2026

Contents

1 Introduction

The GST Borehole service lets you connect your existing borehole data with a GST Web instance.

To receive the borehole data, the service connects to an existing database instance and reads the required information from a set of user defined tables or views with a predefined schema. Currently supported Database management systems are Postgres and Oracle.

The service currently supports the following data.

* Boreholes. Describes a borehole by name, location and total length.
* Layers. Describes a borehole layer by length, name and color. Supports Stratigraphy and Petrography.
* Directional data. Describes a borehole segment length, its inclination and azimuth angles. This allows the specification of directional boreholes.

See the schema description for the full details.

2 Installation

The particular installation instructions will be specific to each instance and will be provided with the installation package.

For information about how to setup the service in GST Web please see the GST Web documentation.

3 Configuration

3.1 Service Configuration

The GST Borehole service requires some configuration to setup the database connection. You can also define the IP and post it will listen on for incoming connections.

Make sure the service has read access to the configuration file.

config.toml:

```
[server]
ip = "::"
port = 50066
enforce_user_rights = false

[database]
url = "postgresql://localhost/boreholes"

[environment]
default_locale = "de"
texture_path = ""
```

Configuration parameters

[server]

* **ip**. The IP address to listen for incoming connections.
* **port**. The port number to listen for incoming connections.
* **enforce_user_rights**. Specifies whether the server should check access for boreholes and borehole logs via the **borehole_rights** and **log_rights** database tables/views. This setting can be set to **true** or **false**. If it's not provided, a value of **false** is assumed. If it is set to **true** the mentioned tables need to exist and grant the relevant users rights to see the desired data. If it is set to **false** the mentioned tables do not need to exist.
* **license_path** The path to the license file used by GST Borehole.

[database]

* **url**. The connection string to the database instance.
  Postgres: postgres://[user[:password]@][host][:port][/dbname]
  Oracle: oracle://<user>/<password>@//<host>:<port>/<service>

[environment]

* **texture_path**. Path to the directory containing the textures used for visualizing layer lithologies.
  The file names of these textures **must** match the names used in the **texture** column of the **lithologies** table.
* **default_locale**. The default locale used as fallback for translations. This setting is optional. If it's not set **de** (german) is used.
* **internal_gdal_path**. The path used to temporarily store gdal and proj configuration files bundled with GST-Borehole server. This setting is optional. If it's not set **$TEMP_DIR/giga-infosystems-{randstr}** is used. The configured directory needs to be writeable for the gst-borehole-server application.

To make the configuration file available, provide the path with the **config** parameter.

```
gst-borehole-server --config /path/to/config
```

3.2 Database Configuration

The following are the requirements for a working service.

* The service database user must have login rights to connect to the database instance.
* The service database user must have read access on all referenced tables.
* The service database user must be able to reference the borehole information tables / views without schema.

4 Table Schema

The following sections specify the schema of required tables that the service accesses. The information can be read from either tables or (materialized) views. This document will refer to table for simplicity. It is important that the names and types match what is specified below.

4.1 Boreholes

The boreholes table contains the general information of a borehole. The starting point, total length and srs.

Postgres

```
create table boreholes
(
  id TEXT PRIMARY KEY,
  srs TEXT NOT NULL CHECK (srs <> ''),
  easting DOUBLE PRECISION NOT NULL,
  northing DOUBLE PRECISION NOT NULL,
  elevation DOUBLE PRECISION NOT NULL,
  total_length DOUBLE PRECISION NOT NULL,
  metadata JSONB DEFAULT '[]'::JSONB
);
```

Oracle

```
create table boreholes
(
  id VARCHAR2(2000) PRIMARY KEY,
  srs VARCHAR2(2000) NOT NULL,
  easting BINARY_DOUBLE NOT NULL,
  northing BINARY_DOUBLE NOT NULL,
  elevation BINARY_DOUBLE NOT NULL,
  total_length BINARY_DOUBLE NOT NULL,
  metadata CLOB DEFAULT '[]'
);
```

Column Explanation

* **id**. The unique identifier (e.g. Primary key) of the borehole. Used as reference in other tables.
* **srs**. The SRS. Format "EPSG:ID". For example "EPSG:31469".
* **easting**. The easting of the starting point.
* **northing**. The northing of the starting point
* **elevation**. The elevation of the starting point.
* **total_length**. The total length.
* **metadata**. Optional Metadata see 4.14 for the expected json format

4.2 Borehole Translations

The **borehole_translations** table contains localised data for boreholes. Several entries can exit for the same borehole for different languages.

Postgres

```
create table borehole_translations
(
  id TEXT REFERENCES boreholes(id),
  lang_id TEXT NOT NULL,
  long_name TEXT NOT NULL,
  short_name TEXT,
  primary key (id, lang_id)
);
```

Oracle

```
create table borehole_translations
(
  id VARCHAR2(2000) REFERENCES boreholes(id),
  lang_id VARCHAR2(2000) NOT NULL,
  long_name VARCHAR2(2000) NOT NULL,
  short_name VARCHAR2(2000),
  primary key (id, lang_id)
);
```

Column Explanation

* **id**. The unique identifier (e.g. Primary key) of the borehole. Used as references the borehole in the **boreholes** table.
* **lang_id** An local identifier. Use `de` for german, `en` for english and so on
* **long_name** The long name of the borehole.
* **short_name** A shorter version of the long name. Used in places where there is less space to show long names. Recommended to be less than 20 characters. If not set GST falls back to the long name

4.3 Layers

The layers table contains information about the borehole layers. It references information about stratigraphy, lithology and additional metadata.

Postgres

```
create table layers
(
 layer_id TEXT NOT NULL,
 borehole_id TEXT NOT NULL REFERENCES boreholes(id),
 depth_from DOUBLE PRECISION NOT NULL,
 depth_to DOUBLE PRECISION NOT NULL,
 stratigraphy TEXT,
 metadata JSONB DEFAULT '[]'::JSONB,
 PRIMARY KEY(layer_id)
);
```

Oracle

```
create table layers
(
 borehole_id VARCHAR2 NOT NULL REFERENCES boreholes(id),
 layer_id VARCHAR2 NOT NULL,
 depth_from BINARY_DOUBLE NOT NULL,
 depth_to BINARY_DOUBLE NOT NULL,
 stratigraphy TEXT REFERENCES stratigraphy(key),
 metadata CLOB DEFAULT '[]',
 PRIMARY KEY(layer_id)
);
```

Column Explanation

* **borehole_id**. The borehole of the layer.
* **layer_id**. The layer identifier. Must be unique.
* **depth_from**. The start depth of the layer. Measured depth [>= 0] from the starting point.
* **depth_to**. The end depth of the layer. Measured depth [>= 0, > depth_from] from the starting point.
* **stratigraphy**. The referenced stratigraphy entry
* **metadata**. Optional Metadata see 4.14 for the expected json format

4.4 Stratigraphy

The **stratigraphy** table contains stratigraphic information for all borholes.

Postgres

```
create table stratigraphy
(
  key TEXT PRIMARY KEY NOT NULL,
  color TEXT NOT NULL
);
```

Oracle

```
create table stratigraphy
(
  key TEXT PRIMARY KEY NOT NULL,
  color TEXT NOT NULL
);
```

Column Explanation

* **key**. The stratigraphy key used to uniquely identify a stratigraphy
* **color**. The color used to display a stratigraphy. Format Hexcolor "#RRGGBB". For example "#FF00C3".

4.5 Stratigraphy Translations

The **stratigraphy_translations** table contains any localized stratigraphy data like the name. This table can contain multiple entries of the same stratigraphy using different language ids.

Postgres

```
create table stratigraphy_translations
(
  key TEXT NOT NULL REFERENCES stratigraphy(key),
  lang_id TEXT NOT NULL,
  short_name TEXT,
  long_name TEXT NOT NULL,
  primary key (key, lang_id)
);
```

Oracle

```
create table stratigraphy_translations
(
  key VARCHAR2(20) NOT NULL REFERENCES stratigraphy(key),
  lang_id VARCHAR2(20) NOT NULL,
  short_name VARCHAR2(200),
  long_name VARCHAR2(2000) NOT NULL,
  primary key (key, lang_id)
);
```

Column Explanation

* **key**. The stratigraphy key from the **stratigraphy** table.
* **lang_id** An local identifier. Use `de` for german, `en` for english and so on
* **long_name** The long name of the stratigraphy.
* **short_name** A shorter version of the long name. Used in places where there is less space to show long names. Recommended to be less than 20 characters. If not set GST falls back to the long name

4.6 Lithologies

The **lithologies** table contains all untranslatable information about lithologies belonging to any borehole.

Postgres

```
create table lithologies
(
  key TEXT PRIMARY KEY NOT NULL,
  color TEXT,
  texture TEXT
);
```

Oracle

```
create table lithologies
(
  key VARCHAR2(2000) PRIMARY KEY NOT NULL,
  color VARCHAR2(7),
  texture VARCHAR2(2000)
);
```

Column Explanation

* **key**. A key uniquely identifying the given lithology.
* **color** An optional color to display the lithology. If not set no color will be used. Format Hexcolor "#RRGGBB". For example "#FF00C3".
* **texture** An optional texture file name, refering to a file in the texture folder. If not set no texture will be used when the lithology is displayed.

4.7 Layer Lithologies

The **layer_lithologies** table contains a mapping between layers (defined in the **layers** table) and lithologies (defined in the **lithologies** table). It allows to define multiple lithologies for the same layer to describe the composition of the layer.

Postgres

```
create table layer_lithologies
(
  litho_rank INTEGER NOT NULL,
  layer_id TEXT NOT NULL REFERENCES layers(layer_id),
  key TEXT NOT NULL REFERENCES lithologies(key),
  percentage FLOAT8,
  percentage_from FLOAT8,
  percentage_to FLOAT8,
  metadata JSONB DEFAULT '[]'::JSONB,
  PRIMARY KEY(layer_id, key)
);
```

Oracle

```
create table layer_lithologies
(
  litho_rank INTEGER NOT NULL,
  layer_id VARCHAR2(2000) NOT NULL REFERENCES layers(layer_id),
  key VARCHAR2(2000) NOT NULL REFERENCES lithologies(key),
  percentage BINARY_DOUBLE,
  percentage_from BINARY_DOUBLE,
  percentage_to BINARY_DOUBLE,
  metadata CLOB DEFAULT '[]',
  PRIMARY KEY(layer_id, key)
);
```

Column Explanation

- **litho_rank**. Sorting order to display the different lithologies.
- **layer_id** The id of the layer this lithologies is associated with. References an entry in the **layers** table.
- **key** The lithology this entry is associated with. References an entry in the **lithologies** table.
- **percentage** A percentage value to describe the composition of the layer. Either this value or **percentage_from** and **percentage_to** needs to be set.
- **percentage_from**. A lower bound percentage value to describe the composition of the layer. Either this value and **percentage_to** or **percentage** need to be set.
- **percentage_to**. An upper bound percentage value to describe the composition of the layer. Either this value and **percentage_from** or **percentage** need to be set.
- **metadata**. Optional Metadata see 4.14 for the expected json format

4.8 Lithology Translations

The **lithology_translations** contains any translatable information about lithologies. Multiple entries for different languages can exist by using a different language key for each language.

Postgres

```
create table lithology_translations
(
  key TEXT NOT NULL REFERENCES lithologies(key),
  lang_id TEXT NOT NULL,
  short_name TEXT,
  long_name TEXT NOT NULL,
  primary key (key, lang_id)
);
```

Oracle

```
create table lithology_translations
(
  key VARCHAR2(2000) NOT NULL REFERENCES lithologies(key),
  lang_id VARCHAR2(2000) NOT NULL,
  short_name VARCHAR2(2000),
  long_name VARCHAR2(2000) NOT NULL,
  primary key (key, lang_id)
);
```

Column Explanation

* **key**. A key uniquely identifying the given lithology. References entries from the **lithologies** table.
* **lang_id** An local identifier. Use `de` for german, `en` for english and so on
* **long_name** The long name of the lithology.
* **short_name** A shorter version of the long name. Used in places where there is less space to show long names. Recommended to be less than 20 characters. If not set GST falls back to the long name

4.9 Metadata translations

The **metadata_translations** table contains translations for metadata labels and values. It can contain entries for multiple languages for the same metadata translation key.

Postgres

```
create table metadata_translations
(
    key TEXT NOT NULL,
    lang_id TEXT NOT NULL,
    value TEXT NOT NULL,
    primary key(key, lang_id)
);
```

Oracle

```
create table metadata_translations
(
    key VARCHAR2(2000) NOT NULL,
    lang_id VARCHAR2(2000) NOT NULL,
    value VARCHAR2(2000) NOT NULL,
    primary key(key, lang_id)
);
```

Column Explanation

* **key**. A key uniquely identifying the given metadata entry. This can either be the value of the **label** metadata object field or a **value** object field entry for translatable_string. See 4.14 for details.
* **lang_id** An local identifier. Use `de` for german, `en` for english and so on
* **value** The translated string that should be displayed.

4.10 Directional Data

The **directional_data** table contains information about the borehole segments for defining directional boreholes. These are the depth values with the inclination and azimuth angle for each segment. Either **azimuth**, **inclination** and **measured_depth** or **dx**, **dy** and **tvd** need to be set.

Postgres

```
create table directional_data
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  depth_from DOUBLE PRECISION,
  depth_to DOUBLE PRECISION,
  azimuth DOUBLE PRECISION,
  inclination DOUBLE PRECISION,
  measured_depth DOUBLE PRECISION,
  dx DOUBLE PRECISION,
  dy DOUBLE PRECISION,
  tvd DOUBLE PRECISION
);
```

Oracle

```
create table directional_data
(
  borehole_id VARCHAR2(2000) NOT NULL REFERENCES boreholes(id),
  depth_from BINARY_DOUBLE,
  depth_to BINARY_DOUBLE,
  azimuth BINARY_DOUBLE,
  inclination BINARY_DOUBLE,
  measured_depth BINARY_DOUBLE,
  dx BINARY_DOUBLE,
  dy BINARY_DOUBLE,
  tvd BINARY_DOUBLE
);
```

Column Explanation

* **borehole_id**. The borehole of the segment.
* **depth_from**. The start depth of the layer. Measured depth (>= 0) from the starting point.
* **depth_to**. The end depth of the layer. Measured depth (>= 0, > depth_from) from the starting point.
* **azimuth**. The azimuth angle of the segment in degrees. 0 is north, direction is clockwise. For example, east is 90 degree, south 180, etc.
* **inclination**. The inclination angle of the segment in degrees. 0 is vertical, 90 is horizontal.
* **measured_depth**. The actual measure depth. For the case **dx**, **dy** and **tvd** is set, this value is only used to order the entries. It doesn't describe the actual depth there, that's provided by **tvd**
* **dx** Step wise derivation in x direction
* **dy** Step wise derivation in y direction
* **tvd** Step wise derivation in z direction

4.11 Borehole Logs

Borehole logs can be used in GST as well. GST is able to read the LAS 2.0 format, c.f. https://www.cwls.org/wp-content/uploads/2017/02/Las2_Update_Feb2017.pdf.

The LAS files can be stored either directly in the database or as files on the application server by for example mounting a shared directory from a file server.

GST can also read borehole log values directly from the database.

Postgres

```
create table borehole_logs
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  log_format TEXT NOT NULL,
  log BYTEA NOT NULL
);
```

Oracle

```
create table borehole_logs
(
  borehole_id VARCHAR(2000) NOT NULL REFERENCES boreholes(id),
  log_format VARCHAR(2000) NOT NULL,
  log BLOB NOT NULL

);
```

Column Explanation

- **borehole_id**. The borehole of the log.
- **log_format**. Values can be LAS_INTERNAL, LAS_EXTERNAL or IN_DATABASE.
- **log**. If log_format is LAS_INTERNAL then here should be the LAS file. If log_format is LAS_INTERNAL the path to the LAS file should be stored. If log_format is LAS_DATABASE this field should be ".

Borehole Logs in Database

Postgres

```
create table borehole_log_info
(
  log_id TEXT NOT NULL,
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  name TEXT NOT NULL,
  description TEXT NOT NULL,
```

```
  unit TEXT
);

create table borehole_log_value
(
  value_id TEXT NOT NULL,
  log_id TEXT NOT NULL,
  depth DOUBLE PRECISION NOT NULL,
  value DOUBLE PRECISION
);
```

Oracle

```
create table borehole_log_info
(
  log_id VARCHAR2(2000) NOT NULL,
  borehole_id VARCHAR2(2000) NOT NULL REFERENCES boreholes(id),
  name VARCHAR2(2000) NOT NULL,
  description VARCHAR2(2000) NOT NULL,
  unit VARCHAR2(2000)
);

create table borehole_log_value
(
  value_id VARCHAR2(2000) NOT NULL,
  log_id VARCHAR2(2000) NOT NULL,
  depth BINARY_DOUBLE NOT NULL,
  value BINARY_DOUBLE
);
```

Column Explanation

* **log_id**. The id of the log.
* **borehole_id**. The borehole of the log.
* **name**. Name of the log.
* **description**. Further description of the log.
* **unit**. The physical unit of the log.
* **value_id**. The id of the value.
* **depth**. The depth at which the value has been measure.
* **value**. The actual measured value.

4.12 Borehole Rights

The **borehole_rights** table contains information about which borehole is visible for which GST User, GST Group or GST Web User. This table is only expected to exist if the **enforce_access_rights** setting is set to **true**

Postgres

```
create table borehole_rights
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  gst_user TEXT,
  gst_group TEXT,
  gst_web_user TEXT
);
```

Oracle

```
create table borehole_rights
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  gst_user TEXT,
  gst_group TEXT,
  gst_web_user TEXT
);
```

Column Explanation

* **borehole_id**. The id of the borehole accessible by the specified user, group or webuser
* **gst_user**. The name of the GST User this borehole should be accessible to.
* **gst_group**. The name of the GST Group this borehole should be accessible to.
* **gst_web_user**. The name of the GST Web User this borehole should be accessible to.

4.13 Log Rights

The **log_rights** table contains information about which log is visible for which GST User, GST Group or GST Web User. This table is only expected to exist if the **enforce_access_rights** setting is set to **true**

Postgres

```
create table log_rights
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  log_id TEXT NOT NULL,
  gst_user TEXT,
  gst_group TEXT,
  gst_web_user TEXT
);
```

Oracle

```
create table log_rights
(
  borehole_id TEXT NOT NULL REFERENCES boreholes(id),
  log_id TEXT NOT NULL,
  gst_user TEXT,
  gst_group TEXT,
  gst_web_user TEXT
);
```

Column Explanation

* **borehole_id**. The id of the borehole the log belongs to
* **log_id**. For the **LAS_INTERNAL** and **LAS_EXTERNAL** log variants this field contains the name of the log values from the data section of the LAS file. For the **IN_DATABASE** variant this field should contain the **log_id** from the **borehole_log_info** table that should be accessible for this GST User, GST Group or GST Web user.
* **gst_user**. The name of the GST User this borehole should be accessible to.
* **gst_group**. The name of the GST Group this borehole should be accessible to.
* **gst_web_user**. The name of the GST Web User this borehole should be accessible to.

4.14 Metadata

GST Borehole allows to provide arbitrary custom metadata at different locations. All share the same json format. Any such column must be either NULL or contain an array of json objects with the following keys:

Required Json Object keys

* **label**. The display label used to describe the metadata value. The corresponding object value is expected to be a string. This string is translated using entries from the **metadata_translations** table.
* **type**. The type of the metadata value. The field value is expected to be one of literal_string, translatable_string, bool, integer, float, timestamp, date, or link. This field controls how the **value** field is handled.
* **value**. The actual metadata value. Depending on the value of the **type** field this value is expected to be a string, float, boolean, integer or object. For values of the **type** literal_string a string value is expected. It's displayed as it is in the frontend. For values of the **type** translatable_string a string value is expected. This value is used as a translation key for the **metadata_translations** table. For values of the **type** bool a boolean value is expected and displayed without transformation. For values of the **type** integer an integer value is expeceded and displayed without transformation. For values of the **type** float a floating point number is expected and displayed without transformation. For values of the **type** timestamp a string containing a RFC-3339 formatted timestamp is expected. The timestamp is displayed according to the configured local. For values of the **type** date a string containing RFC-3339 formatted date is expected. The value is displayed according to the configured locals. For values of the **type** link an object containing a require **href** field with a link as string value and an optional **label** object containing another metadata object is expected. The **label** field is used as label for the link.

Required Json Object keys

* **positions**. A list of positions where the metadata are displayed in GST Web. Depending on the kind of metadata different positions are accepted here:
  For borehole metadata the following values are accepted:
  * **metadata**. Display the metadata entry in a general ``Metadata'' section
  * **masterData**. Display the metadata entry as part of the always displayed master data record

  For layer metadata the following values are accepted:
  * **metadata**. Display the metadata entry in a general ``Metadata'' section
  * **stratigraphy**. Display the metadata entry as part of the stratigraphy information for this layer
  * **lithology**. Display the metadata entry as part of the lithology information for this layer

  This field is optional if it does not exist a default value of **metadata** is assumed.

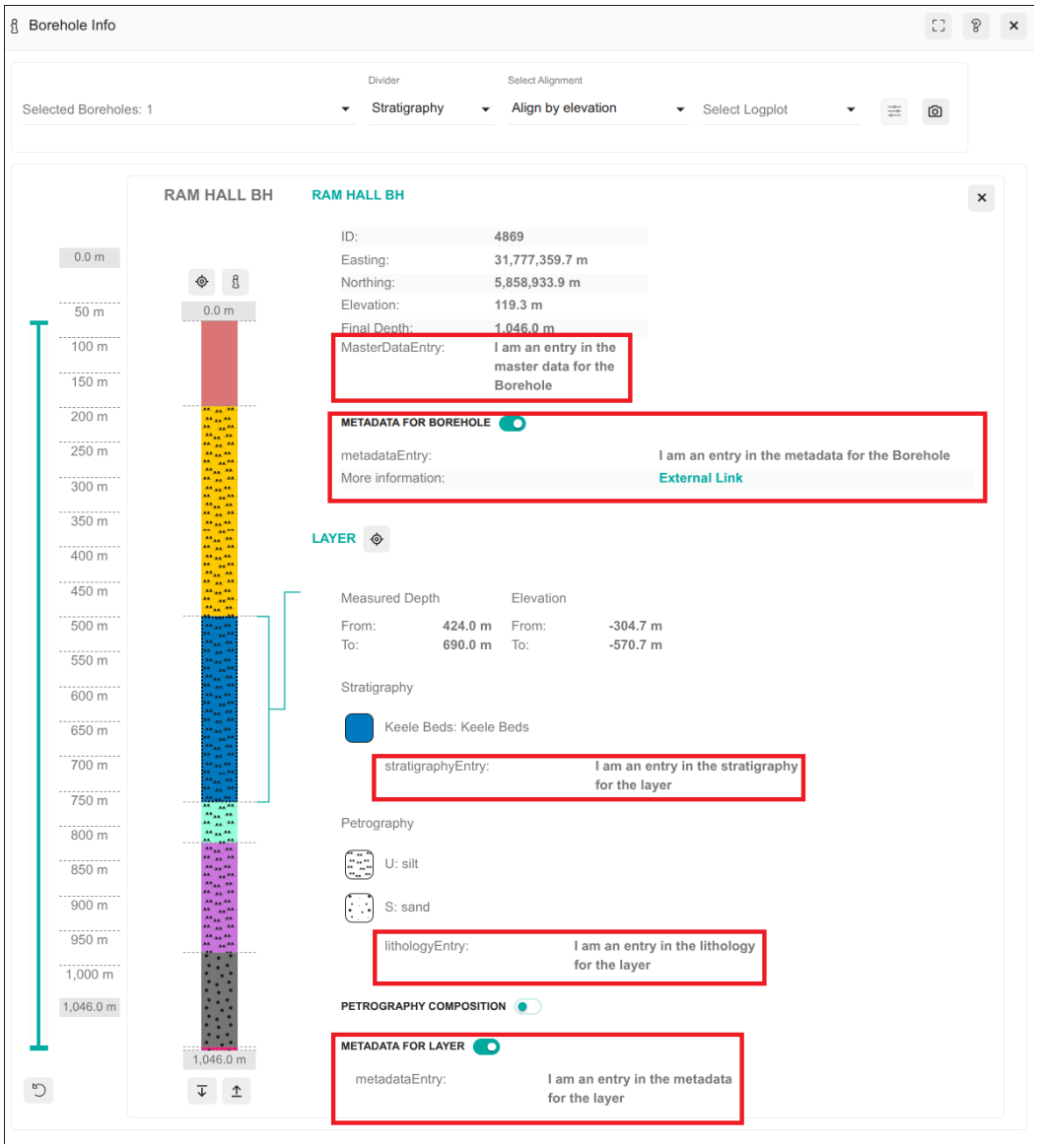

Figure 4.1
This picture shows the metadata positions and where the metadata is displayed.

Example entries

```
[
{
  "label": "borehole_web_link",
  "type": "literal_string",
  "value": "https://nibis.lbeg.de/DetailseitenKartenserver/DetailSeitenBohrsaeulenGeodir
},
{
  "label": "borehole_status",
  "type": "translatable_string",
  "value": "ka"
},
{
  "label": "depth",
```

```
    "type": "integer",
    "value": 42,
    "positions": ["metadata"]
  },
  {
    "label": "width",
    "type": "float",
    "value": 42.1
  },
  {
    "label": "drilling_date",
    "type": "date",
    "value": "2025-03-05"
  },
  {
    "label": "access_time",
    "type": "timestamp",
    "value": "2025-03-05T09:32:10"
  },
  {
    "label": "link",
    "type": "link",
    "value": {
      "href": "https://giga-infosystems.com",
      "label": {
        "type": "translatable_string",
        "value": "Foo"
      }
    }
  }
]
```