PSDB webAPI documentation

Udo Eisenbarth, 17.01.2017

API version 1 (Document version 0.6)

1 Introduction

This documentation describes the web application programming interface of the PHELIX shot data base (PSDB webAPI). The purpose of this interface is to provide a way for programmatic readout of beam time data. This API thus allows for automatic data retrieval and analysis of a complete batch of data such as all camera images of a particular experiment without exporting element by element from the website's export possibilities.

For simple data export functionality to a folder or ZIP file a LabVIEW application is available. For more user specific data handling procedures a low level driver is also available for LabVIEW. Other programming environments or languages used with the following API description.

2 General

2.1 URL format

All requests are sent using the REST interface format following the syntax:

```
<base url>/api/<datatype>?parameter1=&parameter2=&...
```

where `<base url>` is the url of the website which should be in most cases `https://psdb.gsi.de`. The `<datatype>` denotes the type of information to be requested, such as "shots", "experiments", etc.

The particular parameters depend on the `<datatype>`.

Example: A valid request would be

```
https://psdb.gsi.de/api/shottypes
```

2.2 Data format

All data is sent in JSON format. Also the authentication data for the POST request uses this format. Due to limitations of the JSON format some fields (containing binary data) are first converted using Base64 encoding. Furthermore all numeric fields are sent as string fields in order not to lose precision.

2.3 Authentication

User authentication is performed by a HTTP POST message giving the fields "login" and "password" as JSON data:

HTTP POST `to https://psdb.gsi.de/api/user_sessions`

with the payload

```
{"user_session":{"login":<user_login>,"password":<user_password>}}
```

If authentication was successful, the response has an HTTP status 200 and is a JSON object with the user data in the following format:

```
{
"id":<user_id>,"login":<user_login>,"realname":<user_realname>,
"email":<user_email>,"single_access_token":<user_token>
}
```

The single_access_token is necessary for all following calls needing authentication. In this case this token must be added as a parameter to the URL in the following way (see 3):

```
https://psdb.gsi.de/api/experiments?user_credentials=<single_access_token
>
```

If the authentication was unsuccessful the HTTP status 401 with a JSON error object is returned.

3 API Reference

In the following all available API calls are described in detail. All following calls are HTTP GET messages.

As described in 2.3, all calls requiring authentication need the parameter `user_credentials=` followed by the single_access_token added to the URL. A successful response returns with HTTP status 200. If the authentication failed, the status is 401 and a JSON error `{"errors":"Authentication failed"}` is returned.

API calls to experiments, shots and instancevalues only return the data elements the current user is allowed to access, e.g. a user with the role “experimentalist” only receives data of its own experiment.

Most of the API calls return an array of JSON objects in the format:

```
[{<JSON object 0>},{<JSON object 1>},...]
```

Several API calls allow for parameters with are mostly used for filtering the results. In this case parameters can also be used in the array form in order to merge filter conditions.

Examples:

```
https://psdb.gsi.de/api/instances?user_credentials=<user_token>
```

returns all available instances.

```
https://psdb.gsi.de/api/instances?subsystem_id=1&user_credentials=<user_t
oken>
```

returns all instances belonging to subsystem with database id 1.

```
https://psdb.gsi.de/api/instances?subsystem_id[]=1&subsystem_id[]=2&user_
credentials=<user_token>
```

returns all instances belonging to subsystem with database id 1 or 2.

```
https://psdb.gsi.de/api/instances?subsystem_id[]=1&subsystem_id[]=2&class
type_id=5&user_credentials=<user_token>
```

returns all instances belonging to subsystem with database id 1 or 2 and belonging to class type 5.

3.1 Data types

format:	`<base url>/api/datatypes`
needs authentication:	no
parameters:	none
description:	returns all available measurement data types (numeric, string, etc.)
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** name of the data type (such as “string” or “image”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.2 Shot types

format:	`<base url>/api/shottypes`
needs authentication:	no
parameters:	none
description:	returns all available shot types (“experiment shot”, “test shot”, etc.)
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** name of the shot type (such as “snapshot” or “test shot”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.3 Subsystems

format:	`<base url>/api/subsystems`
needs authentication:	no
parameters:	none
description:	returns all available subsystems (“MAS”,”COS”, etc.)
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** name of the subsystem (such as “MAS” or “COS”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.4 Class types

format:	`<base url>/api/classtypes`
needs authentication:	no
parameters:	none
description:	returns all available class types (“PH_gentec_Powemeter”, etc.)
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** name of the class type (such as “PH_Switch” or “PH_IEEE1394Camera”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.5 Instances (List)

format:	`<base url>/api/instances`
needs authentication:	yes
parameters:	• **shot_id** return only instances involved for the given shot (see notes) • **subsystem_id** return only instances belonging to the specified subsystem. • **classtype_id** return only instances belonging to the specified class type.
description:	returns system instances / devices (“PA_Input_Nearfield_Cam”, etc.). If no parameters are given, all instances are returned. Otherwise the result is

	filtered by the given parameters.
response:	array of JSON objects with the following fields: • **id** database id of the object • **classtype_id** the class type the instance belongs to • **subsystem_id** the subsystem the instance belongs to • **name** the name of the instance • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)
notes:	If more than one shot_id is given, an *inclusive* list of instances involved in the given shots is returned. This means a particular instance in the list might not be available for each shot.

3.6 Instances (Single)

format:	`<base url>/api/instances/<id>`
needs authentication:	yes
parameters:	none
description:	returns a single system instances / devices with the given database id.
response:	single JSON object with the following fields: • **id** database id of the object • **classtype_id** the class type the instance belongs to • **subsystem_id** the subsystem the instance belongs to • **name** the name of the instance • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)
notes:	If <id> does not exist, the HTTP status 404 together with a JSON error is returned.

3.7 Experiments

format:	`<base url>/api/experiments`
needs authentication:	yes
parameters:	none
description:	returns all experiments available to the user
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** experiment name such as “P212” • **active** boolean field, which denotes if the experiment has an “active” status (e.g. showing up in the PHELIX control system) • **description** description of the experiment • **experimentarea_id** id of the experiment location (e.g. PTA, Z6, etc.) • **created_at** timestamp of creation (UTC format)

	• **updated_at** timestamp of latest update (UTC format)

3.8 Experimentareas

format:	`<base url>/api/experimentareas`
needs authentication:	no
parameters:	none
description:	returns all available experiment areas (“PTA”, etc.)
response:	array of JSON objects with the following fields: • **id** database id of the object • **name** name of the experiment area (such as “PTA” or “Z6”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.9 Shots (List)

format:	`<base url>/api/shots`
needs authentication:	yes
parameters:	• **id** return only shots with the specified shot id. • **experiment_id** return only shots belonging to the specified experiment • **shottype_id** return only shots with the specified shot type • **from_date** return only shots with a created_at field newer than specified • **to_date** return only shots with a created_at field older than specified • **from_shot_id** return only shots with shot ids higher than specified • **to_shot_id** return only shots with shot ids lower than specified
description:	returns all shots available to the user
response:	array of JSON objects with the following fields: • **id** database id of the object • **description** the shot comment (might be null) • **experiment_id** the experiment it belongs to • **shottype_id** the shot type • **status** multi-purpose field: null = “normal shot”, 1 if it was manually declared as a “failed shot”. • **sysversion** the system version of the shot. This value reflects the general version of the overall laser system. It is changed if major parts of the laser system are modified / updated. • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)

3.10 Shots (Single)

format:	`<base url>/api/shots/<id>`
needs authentication:	yes
parameters:	none
description:	returns a single shots with the given id
response:	single JSON objects with the following fields: • **id** database id of the object • **description** the shot comment (might be null) • **experiment_id** the experiment it belongs to • **shottype_id** the shot type • **sysversion** the system version of the shot. This value reflects the general version of the overall laser system. It is changed if major parts of the laser system are modified / updated. • **status** multi-purpose field: null = “normal shot”, 1 if it was manually declared as a “failed shot”. • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)
notes:	If <id> does not exist or the shot is not available for the current user, the HTTP status 404 together with a JSON error is returned.

3.11 Instancevalues

format:	`<base url>/api/instancevalues`
needs authentication:	yes
parameters:	• **instance_id** return only measurement data belonging to the specified instance • **shot_id** return only measurement data belonging to the specified shot • **experiment_id** return only measurement data belonging to the specified experiment • **chunk** The data chunk number. See notes below.
	returns a list of actual measurement data available to the user
response:	array of JSON objects with the following fields: • **instance_id** the id of the instance which created the data • **shot_id** the shot this measurement data belongs to • **datatype_id** the type of measurement data the object contains. This defines how the following data_* values have to be interpreted (see section 4). • **name** the name of the measurement value • **data_numeric** contains the value for numeric data, might be null (see section 4) • **data_string** contains the value for string data, might be null (see section 4) • **data_binary** contains normally large-sized objects dependent on the data type, might be null (see section 4) • **created_at**

	timestamp of creation (UTC format)
notes:	Because of the possibly huge amount of data, instancevalues are transmitted in sets of 10 values (= 1 chunk). The chunk parameter denotes the chunk number starting with 1 for the first 10 data values, Chunk parameter value 2 hence returns the second 10 values and so on. If the chunk parameter id less than 1 or greater than the number of available chunks (see next command) an empty array is returned.

3.12 Instancevalues (chunks)

format:	`<base url>/api/instancevalues/chunks`
needs authentication:	yes
parameters:	• **instance_id** return only measurement data belonging to the specified instance • **shot_id** return only measurement data belonging to the specified shot • **experiment_id** return only measurement data belonging to the specified experiment
description:	returns the number of data chunks (see command above)
response:	number of chunks as string (no JSON object)

3.13 Attachments

format:	`<base url>/api/attachments`
needs authentication:	yes
parameters:	• **experiment_id** return only attachments belonging to the specified experiment (see notes) • **shot_id** return only attachments belonging to the specified shot (see notes) • **chunk** The data chunk number. See notes below.
description:	returns a list of file attachments available to the user
response:	array of JSON objects with the following fields: • **id** database id of the object • **filename** the file name of the attachment while it was uploaded to the website (with the path stripped) • **filetype** the file type in MIME format such as “application/pdf” • **description** the description as entered at upload dialog of the website • **content** the actual (binary) file content (see notes) • **attachable_id** the id of the object it belongs to (either shot or experiment) • **attachable_type** the type of object it belongs to (either “shot” or “experiment”) • **created_at** timestamp of creation (UTC format) • **updated_at** timestamp of latest update (UTC format)
notes:	• Because of the possibly huge amount of data, attachments are transmitted in sets of 10 values (= 1 chunk). The chunk parameter denotes the chunk number starting with 1 for the first 10 attachments, Chunk parameter value 2 hence returns the second 10 attachments and so on. If the chunk parameter id less than 1 or greater than the number of available chunks (see next command) an empty array is returned. • Attachments can be only filtered by *either* experiment_id *or* shot_id

	since an attachment can only belong to one of these groups. If parameters are mixed the result might be undefined! • The content field is Base64 encoded (see section 4)

3.14 Attachments (chunks)

format:	`<base url>/api/attachments/chunks`
needs authentication:	yes
parameters:	• **experiment_id** return only attachments belonging to the specified experiment (see notes) • **shot_id** return only attachments belonging to the specified shot (see notes)
description:	returns the number of data chunks (see command above)
response:	number of chunks as string (no JSON object)

3.15 Instancevaluesets (count)

format:	`<base url>/api/instancevaluesets/count`
needs authentication:	yes
parameters:	• **instance_id** consider only measurement data sets belonging to the specified instance • **shot_id** consider only measurement data sets belonging to the specified shot
description:	returns the number of measurement value sets that would be downloaded with the given filter parameters
response:	number of data sets as string (no JSON object)

4 Appendix: Measurement data format

The measurement data returned by the instancevalue API call (section 3.11) need to be interpreted depending on the data type (which is given in the `datatype_id` field). Currently the following data types are defined in PSDB:

4.1 String data

In this case the information is simply written to the `data_string` field. The other data fields are null.

4.2 Numeric data

For numeric data the actual value is written to the `data_numeric` field as a decimal string in order preserve precision. The `data_string` field contains the physical unit (if available).

4.3 Boolean data

Boolean data is encoded in the `data_numeric` field as either 0 or 1. The other data fields are null.

4.4 Image data

For images only the `data_binary` field is used. This field is Base64 encoded (for details see https://en.wikipedia.org/wiki/Base64). After decoding, the first 4 bytes contain the length of the following data. The remaining byte stream represents actual image data as a PNG file.

4.5 2D data

Current curves, spectra, etc. are stored as 2d Data. This field is Base64 encoded (for details see https://en.wikipedia.org/wiki/Base64). After decoding, the first 4 bytes contain the length of the following data. The remaining bytes of the `data_binary` field represent the actual data as string in

CSV format. The `data_string` field contains a comma-separated list of column headers in the format:

```
<column X name>,<column X unit>,<column Y name>,<column Y unit>
```