Presenter-Service

Types

PresenterResult

```
type PresenterResult = object | PresenterError;
```

Exceptions

PresenterException

```
Exception PresenterException(message: string);
```

Interfaces

Presenter

```
Interface Presenter {
    presenter: string;
    data: object;
}
```

This interface specifies what presenter is used with the payload for the call.

PresenterError

```
Interface PresenterError {
    error: object;
}
```

A common format for errors.

Methods

handle_request

```
handle_request(payload: Presenter[], user_id: Id): PresenterResult[]
```

Executes some presenting function on the server. The term "presenting" means a non writing (or modifying) idempotent request There may be some side-effect allowed (like tracking calls to one presenter), but the main purpose is to get data of the server, which is not autoupdate-, projector-, or icc-data. Some purposes may be:

- aggregating of login data

- managing of whoami, if additional data is needed, that the auth service doesn’t provide

- To get history points

- To get history data

- To get the installed version

- To query statistics

- To calculate recursive trees (e.g. origins of motions; not possible with the autoupdate service)

- to synchronize the servertime for countdowns

@throws PresenterException This exception might be thrown, if there was an server error (Http-500-equivalent). For user error (e.g. wrong data) use the PresenterError interface.