Documentation
Xpert-Timer REST-API (XTCloudserver)

Version: 8.0

DB-Patchlevel: PL 90

API-Revision: R1

Inhaltsverzeichnis

1 General

The XTRESTAPI is an interface to be able to access the data records of the program independent of the database.

IMPORTANT: To be able to use the XTRESTAPI, the XTCloudserver must be installed and accessible via XTWeb.

The current version of this documentation can be found at:
http://download.xperttimer.de/helpfiles/XTRESTAPI_Doc.pdf

DThe XTRESTAPI including demo source code as ZIP can be found at:
http://download.xperttimer.de/additional/XTRESTAPIDemo.zip

The following options are provided:

- Establish / disconnect web server connection
- User login / logout
- Create, change or delete projects, clients, users, tasks, timestamps.
- Read out existing projects, clients, users, tasks, time stamp lists
- Add timestamp
- Query data about the logged in user
- Query system data
- Assign projects to users
- Read out and change clients, projects, users, tasks, timestamp details
- Start, stop or pause projects
- Calculation of time stamp sums taking into account user, project, client, period

2 Connection through programming languages

2.1 General

Access is possible with two different types of communication.

2.1.1 REST-API

Data objects can be read and written via calls to the REST API. Furthermore, data object lists can be read.

2.1.2 JSON-RPC

Server functions can be triggered via JSON-RPC. This way, for example, projects can be started or individual values or totals can be calculated..

2.2 Open-API / Swagger

The automatic Open API documentation can be viewed by using the Swagger UI. You find the option in the configuration dialog of XTCloudserver. All access objects and their methods and data models are displayed interactively.

You can access the Open-API-Definitions via:
`http://serverurl:portnumber/swagger-ui/index.html`

`e.g.` `http://localhost:9000/swagger-ui/index.html`

2.3 Python

For the connection to Python there is an **API wrapper** in the subdirectory "xtrestapi_wrapper" which provides the most common functions and offers you a lot of support for access. The API wrapper also contains various helper routines to make working with the API easier.

See example "Examples\Python"

In order to be able to execute the Python scripts, you need a Python installation on your system. This is available free of charge at: https://www.python.org/downloads/

The scripts can then be loaded and executed in the supplied IDE (IDLE). In order to be able to access your XTCloudserver installed in your company, you only have to replace the following login data in the examples:

```
# Login params to XTCloudserver
API_URL = "http://localhost:9000/api/"
API_KEY = "6D12E349-331A-4BBF-B630-A275A56308BC"
USERNAME = "demo"
PASSWORD = "demo"
```

2.4 Call methods

2.4.1 REST

Data objects can be read and written via calls to the REST API. Furthermore, data object lists can be read. The call parameters are transferred as URL parameters for read access and as POST parameters for write access.

2.4.2 JSON-RPC (JRPC)

Server functions can be triggered via JSON-RPC. This enables projects to be started, for example, or individual values or totals to be cAlld.

The call parameters are transferred in a JRPC structure. The order of the parameters is decisive.

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
persnr = 1

PARAMS = {
         "jsonrpc":"2.0",                // JRPC-Version
         "method":"GetUserByPersnr",     // Name of the remote method to be called
         "params":[ persnr ],            // Comma-separated array with parameters
         "id":924                        // Number oft he call.
                                              can be set individually, e.g. 1
         }

response = requests.request("POST",
            url = URL, params = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  result = data['result']
```

The return format:

```
{
    "jsonrpc": "2.0",
    "id": 924,
    "result": "{DA71AC04-F5AE-4DE9-B0C0-B31E42C6892D}"
}
```

2.5 General procedures

2.5.1 Primary keys (GUID)

In Xpert-Timer, data objects always own a unique identifier to be addressed to. A so-cAlld GUID (Global Unique Identifier). This GUID is a 38-digit alphanumeric identifier that makes it very easy to work without number ranges.

Example:
`{0767F4E9-CC24-4D6A-87DA-F6D9B088D503}`

A GUID can be created using the “XTRESTAPI.CreateGUID()“ function.

If a data record contains such a GUID, the corresponding field is always followed by the extension „`_nr`“ , e.g. `user_nr`, `client_nr`, `project_nr` and so on.

There are two different special cases of a GUID in Xpert-Timer to indicate the status NULL or "not linked". These are required to maintain the referential integrity for SQL JOINS and thus simplify the reading of SQL data volumes considerably.

```
{11111111-1111-1111-1111-111111111111} = Link to empty data object
{00000000-0000-0000-0000-000000000000} = Empty field content
```

2.5.2 Date / Time

Dates or times are converted to ISO8601. Any existing time zone is not taken into account.

For example:
`2012-04-23T18:25:43.511Z`

2.5.3 Deleting records

Data records are never really deleted within the XT database, but only marked as "deleted".

If data objects are deleted, all subordinate data objects are also deleted.

2.5.4 Reading data objects / calling functions

The syntax of the examples corresponds to the Python notation for simplicity.

The following base url is used for all calls: `apiurl = "http://localhost:9000/api/"`

2.5.5 Notes in regards to the login

In order to be able to use the XTRESTAPI for automation, you should set up your own user. E.g. with the name "XTAPIUSER" or "SYSTEM". You can also use an existing user, but this user is then entered as the creator or person who made the changes for all created and changed data objects.

For security, you should create a separate **API key** for each external application that uses the API on the XTCloudserver. The API key is then sent with each call in the header of the http command. If the key is not known on the server, the calls are rejected.

A user login must take place before a command can be sent to the XTRESTAPI. The return token (JSON web token) from the registration must then be sent with each subsequent call in the header of the http command in order to be able to identify the user on the server.

2.5.6 The user access right system

Xpert-Timer has an extensive rights system to control the access rights of the individual users or user groups. As a rule, there are rights for each data object that allows reading, writing, deleting, or changing. If a function is called for which there are insufficient rights, it is acknowledged with an error message.

In addition to the standard rights, there are a large number of special rights that, for example, influence the project start behavior or the procedure for adding a manual entry.

Rights can only be assigned in Xpert-Timer Pro (Windows).

2.5.7 The visibility of records (visibility)

The Xpert-Timer has an extensive system to control the visibility of data records. As a rule, it refers mainly to only being able to see your own, or seeing all team data records. If the work groups are used in Xpert-Timer, they are also included in the visibility settings..

The following levels of visibility are available:

- Only own data sets
- Records of other employees
- If project manager
- If in the same working group
- If working group leader
- All

The visibility level can usually be transferred as a parameter when calling a reading function.

The possible visibilities are regulated by the rights system.

The following values are possible as call parameters for the visibility:

- vmOwn = 0
- vmWorkgroup = 1
- vmAll = 2
- vmInChargeOfProject = 3
- vmInChargeOfWorkgroup = 4

2.5.8 The security levels

In addition to the standard visibilities, there is also the option of controlling the visibility via security levels. This means that confidential customers, projects, documents and tasks can only be made accessible to certain employees. The security levels range from 0 = none to 5 = strictly confidential. By default, level 2 = medium is assigned to all employees and data objects.

The visibility via the security level is completely handled by the server and cannot be influenced for individual queries.

2.5.9 Pagination of queries (PageSize, PageNumber)

If data record lists are queried, the amount of data is usually returned page by page. You can influence this with the following parameters:

- PageSize: Number of data records that are transmitted in response to a query.
  If you set the PageSize to 0, all data records are transferred at once

- PageNumber: Number of the page to be transmitted.
  Important: The counter starts with 0. I.e. the first page is queried with PageNumber = 0.

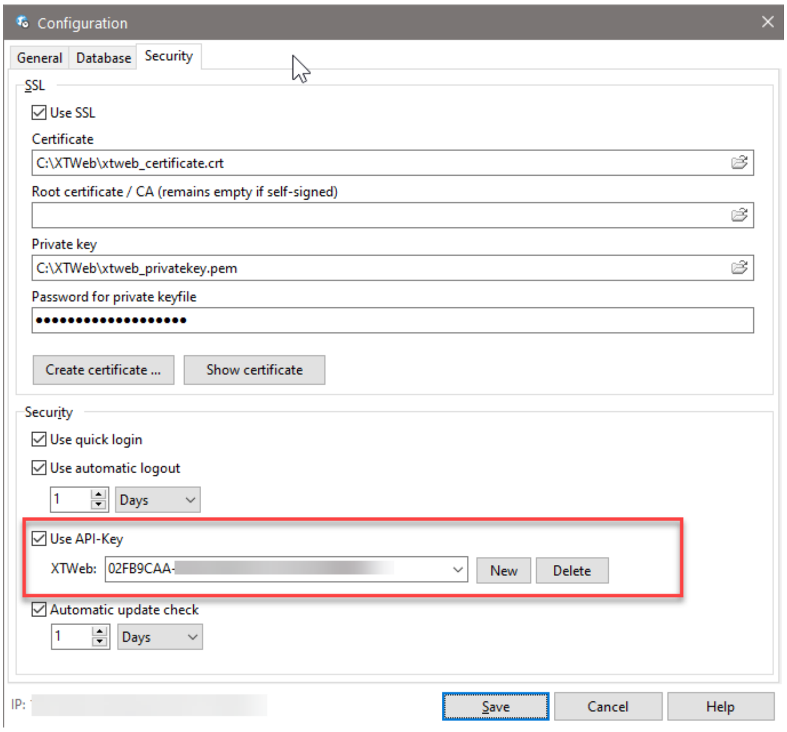
In the response header of the return quantity, you can use the "xt-total-count" field to find out how many data records are in total in the paginated output. If you divide this by the current page size, you get the total number of pages available for this query.

2.5.10 The API-Login

The API-Login is an additional security level to protect the API against unauthorized logins. It is based on the Basic-Auth-Method.

2.5.11 The API-Key

In order to be able to access the XTCloudserver via the REST API, an API key should be generated on the server for each application for security reasons. The server functions can only be accessed via the API if the API key has been activated on the server. If you want to prevent API access at a later point in time, it is sufficient to deactivate the API key.



☑ Use SSL

Certificate: C:\XTWeb\xtweb_certificate.crt

Root certificate / CA (remains empty if self-signed):

Private key: C:\XTWeb\xtweb_privatekey.pem

Password for private keyfile: •••••••••••••••••••

☑ Use quick login

☑ Use automatic logout

1 Days

☑ Use API-Key

XTWeb: 02FB9CAA-[redacted]

☑ Automatic update check

1 Days

IP: [redacted]

3 Object hierarchy

The following master data objects form the core of Xpert-Timer.

- User
- Clients / Client contacts
- Projects

The following acquisition data objects can be created for these master data objects:

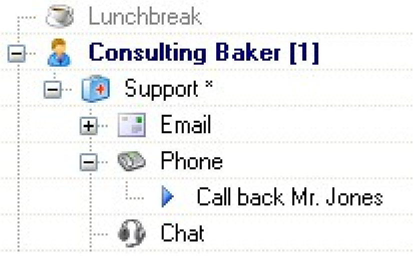
- Tasks
- Notes
- Timestamps

The hierarchical structure is always as follows:

[Client]
    Project
        [Sub-projects]
            User
                Task
                Note
                Time stamp

Objects in square brackets "[]" are optional. That means a project does not have to be attached to a customer and a sub-project level is not mandatory. However, a maximum of one sub-project level is possible.

e.g.



4 Login (User login)

Login of a user on the web server.

Parameter:

`jwtusername`	Login name of the user
`jwtpassword`	Password of the user in plain text

Return:

JSON web token (JWT) if the login could be carried out.

Example:

```
import requests
import json

# Base-URL of XTWeb-Api
apiurl = http://localhost:9000/api/
apikey = “11233456-…”

# Endpoint: Login  ======================================================
url = apiurl + "login"

HEADERS = {
    "content-type": "application/json",
    "accept": "application/json",
     "apikey": apikey,
    "jwtusername": "demo",
    "jwtpassword": "demo"
}

response = requests.request("POST", url, headers=HEADERS)

print(f"\nAUTHENTICATE with POST {url} : {HEADERS} ")
print(f"{response.status_code}: {response.reason}")
print(response.text)

data = json.loads(response.text)

# JWT Token to send with every header
try:
    token = data['token']
    print("\nToken: " + token)

    # Header to send with every future request
    HEADERS = {
        "content-type": "application/json",
        "accept": "application/json, text/plain, */*",
        "Accept-Encoding": "gzip, deflate, br",
         "apikey": apikey,
        "authentication": "Bearer " + token,
        "Access-Control-Request-Headers": "authentication"
    }
except KeyError:
    print(f"\nERROR: AUTHENTICATION FAILED")
```

The return token in JSON web token format (JWT) looks like this:

```
{"token":"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJYcGVydC1UaW1lciIsImV4cCI6MTYxNTA1MTQyNSwibm
JmIjoxNjE0OTY0NzI1LCJpYXQiOjE2MTQ5NjUwMjUsInh0dXNlcm5hbWUiOiJEaWV0ZXIgRGVtbyIsInh0c2VjdXJpdHlsZXZlbCI
6IjAiLCJ4dGlzYWRtaW4iOiIxIiwidXNlcm5hbWUiOiJkZW1vIiwieHR3b3JrZ3JvdXBtb2RlIjoiMCIsInh0dXNlcl9uciI6IntG
NDdGRDAwMS1EQzc4LTQ0MzUtQTQ0NS0xRjRGQzVGRjQ4REZ9Iiwicm9sZXMiOiIiLCJ4dG1haW53b3JrZ3JvdXBfbnIiOiJ7Njg1N
DU5NkMtMjU2Mi00QUEyLTg5NjMtMDA5ODA2RjhDQTVFfSJ9.jKt0hmEI2BoxxbBXijHiG5qtCDAbf5IYd05eRVv-
9ayS_e4AtaLpShHNhSVRzB9vjbAKqSZFbKakyUKJhquW-w"}
```

If you look at it decoded, e.g. via https://www.jsonwebtoken.io then you get the following data fields:

```
{
 "iss": "Xpert-Timer",
 "exp": 1614969120,
 "nbf": 1614964725,
 "iat": 1614965025,
 "xtusername": "Dieter Demo",
 "xtsecuritylevel": "0",
 "xtisadmin": "1",
 "username": "demo",
 "xtworkgroupmode": "0",
 "xtuser_nr": "{F47FD001-DC78-4435-A445-1F4FC5FF48DF}",
 "roles": "",
 "xtmainworkgroup_nr": "{6854596C-2562-4AA2-8963-009806F8CA5E}",
 "jti": "acd87c51-a4b5-4c60-a304-fbdc93c1e23f"
}
```

5 Data objects

Data objects are usually queried using standard REST calls. The data records are output in JSON format. The following endpoints are available for this.

5.1 Users

A user is an employee in the database. It is usually referenced with its unique `user_nr`. The creator of a data record is usually addressed via `creator_nr`.

5.1.1 Data structure

```
export class User {
  user_nr: string;                   // Unique GUID                       [Required field]
  firstname: string;                 // First Name                        [Required field]
  lastname: string;                  // Last Name                         [Required field]

  loginname: string;                 // Login name for user login         [Required field]
  persnr: number;                    // Personnell number

  title: string;                     // Titel
  nickname: string;                  // Nickname

  active: boolean;                   // Is the user active?
  admin: boolean;                    // Is the user an administrator?
  changepwonlogin: boolean;          // Change password on next login?

  pricegroup_nr: string;             // Standard price group
  mainworkgroup_nr: string;          // What main work group does the user belong to?
  costcenter: string;                // Cost center
  securitylevel: number;             // Security level
  email: string;                     // Email address
  hoursday: number;
  daysweek: number;

  loginmode: number;                 // Login through Desktop and/or Web

  lastlogin: Date;                   // Last Login                        [Read-Only]

  date_created: Date;                // Date created                      [Read-Only]
  lastchange: Date;                  // Last change                       [Read-Only]
  creator_nr: string;                // Creator                           [Read-Only]

  usergroups: string[];              // Assigned access right groups
}
```

5.1.2 List

Reads in a list of user data records.

Type of call: `REST, GET`

Parameter:

`pageNumber`	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0

`pageSize`	Number of records per query 0 if no pagination is to be used.
`sortfield`	Sort column
`sortdirection`	Sort order

`visibility`	Visibility
	vmOwn = 0, vmWorkgroup = 1, vmAll = 2,
	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4

`project_nr`	Assigned to project
`not_in_project_nr`	Not assigned to project
`username`	Search value

Example:

```
endpoint = "users"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
           'pagenumber': 0,
           'pagesize': 10
          }

response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  firstname = data[0]['firstname']
```

5.1.3 Read

Reads a user record

Type of call: `REST, GET`

Parameter:

`user_nr`	GUID of the user

Example:

```
endpoint = "users"
user_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + user_nr

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  firstname = data['firstname']
```

5.1.4 Create

Creates a new user record

Type of call: `REST, POST`

Parameter:
none

Example:

```
endpoint = "users"
user_nr = '{1234-...}'
URL = apiurl + endpoint
```

```
DATA = {
         'firstname': 'Max',
         'lastname': 'Muster'
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  user_nr = data['message']
```

5.1.5 Update

Creates a new user record. The entire user data set is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`user_nr`	GUID of the user

Example:

```
endpoint = "users"
user_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + user_nr

DATA = {'firstname': 'Max',
         'lastname': 'Muster',
        …
         }

response = requests.request("PUT", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.1.6 Delete

Deletes a user data record with all subordinate data objects (e.g. time stamp or tasks)

Type of call: `REST, DELETE`

Parameter:

`user_nr`	GUID of the user

Example:

```
endpoint = "users"
user_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + user_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.2 Clients

The client level is not required for time recording, but lets you structure your data much easier. A client is referenced via the `client_nr`.

5.2.1 Data structure

```
export class Client {
  client_nr: string;                    // Unique GUID            [Required field]
  clientid: number;                     // Client ID              [Required field]
  clientname: string;                   // Clientname             [Required field]
  info: string;                         // Comment
  active: boolean;                      // Active/Inactive
  securitylevel: number;                // Securitylevel
  color: number;                        // Color
  hasvat: boolean;                      // Has VAT
  clientreference: string;              // Reference number
  vatid: string;                        // VAT-ID

  maincontact_nr: string;               // GUID of main contact
  pricetable_id_nr: string;             // GUID of price table
  created_workgroup_nr: string;         // GUID of main workgroup

  userdefined1: string;                 // Freely assignable user fields 1-10
  userdefined2: string;
  userdefined3: string;
  userdefined4: string;
  userdefined5: string;
  userdefined6: string;
  userdefined7: string;
  userdefined8: string;
  userdefined9: string;
  userdefined10: string;

  date_created: Date;                   // Created on             [Read-Only]
  lastchange: Date;                     // Last change            [Read-Only]
  creator_nr: string;                   // GUID Creator           [Read-Only]
}
```

5.2.2 List

Reads in a list of customer records.

Type of call: `REST, GET`

Parameter:

`pageNumber`	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0
`pageSize`	Number of records per query 0 if no pagination is to be used.
`sortfield`	Sort column
`sortdirection`	Sort order
`visibility`	Visibility vmOwn = 0, vmWorkgroup = 1, vmAll = 2,

	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4
`user_nr`	User

`state`	0 = All, 1 = Only active, 2 = Only with assigned projects
`favoritesonly`	Only if a client project has been marked as a favorite
`clientname`	Search term

Example:

```
endpoint = "clients"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
           'pagenumber': 0,
           'pagesize': 10
          }

response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  clientname = data[0]['clientname']
```

5.2.3 Get

Reads a client record

Type of call: `REST, GET`

Parameter:

`client_nr`	GUID of the client

Example:

```
endpoint = "clients"
client_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + client_nr

response = requests.request("GET",
            url = URL, headers = HEADERS)

data = response.json()
clientname = data['clientname']
```

5.2.4 Create

Creates a new customer record

Type of call: `REST, POST`

Parameter:

none

Example:

```
endpoint = "clients"
URL = apiurl + endpoint

DATA = {
         'clientname': 'Kunde 1',
         'clientid': 1
        …
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  client_nr = data['message']
```

5.2.5 Update

Creates a new client record. The entire data record is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`client_nr`	GUID of the client

Example:

```
endpoint = "clients"
client_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + client_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.2.6 Delete

Deletes a client data record with all subordinate data objects (e.g. projects, time stamps or tasks)

Type of call: `REST, DELETE`

Parameter:

`client_nr`	GUID of the client

Example:

```
endpoint = "clients"
client_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + client_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.3 Contacts (Addresses, Contacts)

The client addresses (contacts) are used to manage different contact persons in a company. A contact address always belongs to a client and is referenced via the contact_nr.

5.3.1 Data structure

```
export class Contact {
  contact_nr: string;         // Unique GUID             [Read-Only]
  client_nr: string;          // GUID of the client      [Required field]
  firstname: string;          // First name
  name: string;               // Last name               [Required field]
  description: string;        // Description of position in company
  comment: string;            // Comment

  phone1: string;                // Phone 1
  phone2: string;                // Phone 2
  fax: string;                   // Fax
  mobile: string;                // Mobile
  email1: string;                // Email 1
  email2: string;                // Email 2
  website: string;               // Website
  address1: string;              // Address 1
  address2: string;              // Address 2
  zip: string;                   // Zip
  city: string;                  // City
  state: string;                 // County
  country: string;               // Country

  title: string;              // Titel
  salutation: number;         // Salutation
  birthday: Date;             // Date of birth
  contacttype: number;        // Type of contact (e.g. billing address,
                                                          Company address, etc.)
  textualaddress: string;     // Free postal address
  clientnameaddress: string;  // Customer name that should appear on the mailing address

  active: boolean;            // Active/Inactive

  userdefined1: string;       // User defined fields 1-10
  userdefined2: string;
  userdefined3: string;
  userdefined4: string;
  userdefined5: string;
  userdefined6: string;
  userdefined7: string;
  userdefined8: string;
  userdefined9: string;
  userdefined10: string;

  date_created: Date;        // Created on               [Read-Only]
  lastchange: Date;          // Last change              [Read-Only]
  creator_nr: string;        // GUID of creator          [Read-Only]
}
```

5.3.2 List

Reads a list of contact records.

Type of call: `REST, GET`

Parameter:

`pageNumber`	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0
`pageSize`	Number of records per query 0 if no pagination is to be used.
`sortfield`	Sort column
`sortdirection`	Sort order

`visibility`	Visibility
	vmOwn = 0, vmWorkgroup = 1, vmAll = 2,
	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4

`user_nr`	User

`client_nr`	Only if a client project has been marked as a favorite
`contactname`	Search term

Example:

```
endpoint = "contacts"
URL = apiurl + endpoint
client_nr = "{12345678-…"

PARAMS = {'sortdirection': 'asc',
           'client_nr': client_nr,
           'pagenumber': 0,
           'pagesize': 10
          }

response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  name = data[0]['name']
```

5.3.3 Read

Reads a contact record.

Type of call: `REST, GET`

Parameter:

`contact_nr`	GUID of the contact

Example:

```
endpoint = "contacts"
contact_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + contact_nr

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  name = data['name']
```

5.3.4 Create

Creates a new contact record.

Type of call: `REST, POST`

Parameter:
none

Example:
```
endpoint = "contacts"
URL = apiurl + endpoint

DATA = {
         'client_nr': client_nr,
         'name': 'Muster',
         'firstname': 'Max',
        …
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  contact_nr = data['message']
```

5.3.5 Update

Creates a new contact record. The entire data record is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`contact_nr`	GUID of the contact

Example:
```
endpoint = "contacts"
contact_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + contact_nr

DATA = {
         'name': 'Muster',
         'firstname': 'Max',
         'persnr': 1,
        …
         }

response = requests.request("PUT",
            url = URL, data = DATA, headers = HEADERS)
data = response.json()
name = data['name']
```

5.3.6 Delete

Deletes a customer data record with all subordinate data objects (e.g. projects, time stamps or tasks)

Type of call: `REST, DELETE`

Parameter:

`contact_nr`	GUID of the contact

Example:

```
endpoint = "contacts"
contact_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + contact_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.4 Projects

The projects are the core component of the Xpert-Timer and therefore a `project_nr` is necessary for almost all actions.

5.4.1 Data structure

```
export class Project {
  project_nr: string;          // Unique GUID                    [Required field]
  parent_project_nr: string;   // Unique GUID of the main project
  client_nr: string;           // Unique GUID of the client

  projectname: string;      // Name of the project               [Required field]
  projectnumber: string;    // Alphanumeric project number

  comment: string;          // Project description
  securitylevel: number;    // Security level
  color: number;            // Color

  projecttype_nr: string;   // GUID of the project type
  minutesneeded: number;    // Time required in minutes. This field is READ-ONLY
                               and is calculated from the time stamps
  minutesestimated: number;

  state: number;               // Project state: 0=psRunning, 1=psPaused, 2=psCancled,
                                 3=psFinished, 4=psInactive, 5=psAccounted
                                 6=psToBeAccounted, 100-109=psUserdefined1-10
  progress: number;        // Progress in percent
  priority: number;        // Priority
  timeaccount: boolean;    // Time account Yes/No
  accountmode: number;     // Type of account mode
  teamproject: boolean;    // Team project. Is the project visible for all?
  subprojectorder: number; // Sort order of the subproject

  getpricefrom: number;    // Billing rate
  flatrateprice: number;   // Flat rate
  billingunit: number;     // Billing unit
  priceperbillingunit: number; // Price per unit

  addtimestampmode: number;

  startdate: Moment;       // Planned project start
  enddate: Moment;         // Planned project end
  datestarted: Moment;     // Real project start
  datefinished: Moment;    // Real project end

  maincontact_nr: string;        // GUID of main contact
  user_nr: string;               // Person in charge
  secondincharge_nr: string;     // 2. Person in charge
  pricetable_id_nr: string;      // Price table
  created_workgroup_nr: string;  // Main work group

  userdefined1: string;          // User defined fields 1-10
  userdefined2: string;
  userdefined3: string;
  userdefined4: string;
  userdefined5: string;
  userdefined6: string;
  userdefined7: string;
  userdefined8: string;
  userdefined9: string;
  userdefined10: string;

  lastchange: Moment;            // Last change                  [Read-Only]
  creator_nr: string;            // Creator                      [Read-Only]
  date_created: Moment;          // Created on                   [Read-Only]

  // Read-Only-Area
  clientname: string;            // Name of client (Read-Only)
  clientid: number;              // Client number (Read-Only)
  mainprojectname: string;    // Name of main project (Read-Only)
  mainprojectnumber: string; // Number of main project (Read-Only)
}
```

5.4.2 List

Reads a list of project data records.

Type of call: `REST, GET`

Parameter:

`pageNumber`	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0
`pageSize`	Number of records per query 0 if no pagination is to be used.
`sortfield`	Sort column
`sortdirection`	Sort order

`visibility`	Visibility
	vmOwn = 0, vmWorkgroup = 1, vmAll = 2,
	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4

`user_nr`	Users

`client_nr`	Only find projects of a certain client
`parent_project_nr`	Only show sub-projects of a certain main project
`projectname`	Search term
`mainprojectsonly`	Mainprojects only BOOL
`timeaccountonly`	Timeaccounts only BOOL
`teamprojectsonly`	Teamprojects only BOOL
`assignedtouser`	Only when assigned to user (user_nr) BOOL
`favoritesonly`	Favorites only BOOL
`projecttype_nr`	Projecttype GUID
`projectstate`	Project state INTEGER
`accountmode`	Accountmode INTEGER
`priority`	Priority INTEGER

Example:

```
endpoint = "projects"
URL = apiurl + endpoint
```

```
PARAMS = {'sortdirection': 'asc',
           'pagenumber': 0,
           'pagesize': 10
          }

response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  projectname = data[0]['projectname']
```

5.4.3 Get

Reads a project record.

Type of call: `REST, GET`

Parameter:

`project_nr`	GUID of the project

Example:

```
endpoint = "projects"
project_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + project_nr

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  projectname = data['projectname']
```

5.4.4 Create

Creates a netw project record.

Type of call: `REST, POST`

Parameter:

`client_nr`	GUID of the client the poject was create for
`parent_project_nr`	GUID of the main project the new project is created for

Example:

```
endpoint = "projects"
URL = apiurl + endpoint

DATA = {
         'client_nr': client_nr,
         'parent_project_nr': parent_project_nr,
         'projectname': 'Projekt 1',
         'projectnumber': '12345',
        …
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  project_nr = data['message']
```

5.4.5 Update

Creates a new project data set. The entire data record is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`project_nr`	GUID of the project

Example:

```
endpoint = "projects"
project_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + project_nr

DATA = {
         'projectname': 'Projekt 1',
         'projectnumber': '12345',
        …
         }

response = requests.request("PUT",
             url = URL, data = DATA, headers = HEADERS)
data = response.json()
projectname = data['projectname']
```

5.4.6 Delete

Deletes a project data record with all subordinate data objects (e.g. projects, time stamps or tasks)

Type of call: `REST, DELETE`

Parameter:

`project_nr`	GUID of the project

Example:

```
endpoint = "projects"
project_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + project_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.5 Tasks

Using the tasks, users can be assigned specific activities that should be completed on a key date. They are referenced by the `todo_nr`.

5.5.1 Data structure

```
export class Task {
  todo_nr: string;            // Unique GUID                                                    [Read-
Only]

  project_nr: string;         // GUID of the project               [Required field]
  user_nr: string;            // GUID of the user (If empty, To-Do task pool)
  creator_nr: string;         // GUID of the creator                [Required field]
  todocategory_nr: string;    // GUID of the todo category

  subject: string;            // Subject                             [Required field]
  body: string;               // Text
  itemdate: Moment;
  donedate: Moment;           // Done date
  readdate: Moment;           // Read date

  tobedonefrom: Moment;       // To do from
  tobedonetill: Moment;       // To do until

  progress: number;           // Progress in %
  priority: number;           // Priority (0=Low;...;5=Highest)

  tasknumber: number;         // Running task number                     [Read-Only]

  notecount: number;          // Number of notes                         [Read-Only]
  attachmentcount: number;    // Number of documents                     [Read-Only]

  lastchanged: Moment;        // Last change                             [Read-Only]
  date_created: Moment;       // Created on                              [Read-Only]

  minutesneeded: number;      // Time needed in minutes (Is calculated from the timestamps)
  minutesestimated: number;   // Time estimated in minutes
  taskstate: number;          // Task state: 0=xttdstNotStarted, 1=xttdstChecking,
                                 2=xttdstInProgress, 3=xttdstPaused, 4=xttdstWaiting,
                                 5=tsDeclined, 6=tsDone
                                 7=tsTesting, 8=tsTestFailed, 9=TestOk
                                 10=tsProblem, 11 = tsReady
                                 100-104=tsUserdefined1-5
  taskstatecomment: string;   // Comment on task state

  textformat: number;         // Type of text format (xttdtfRVF=0, xttdtfRTF=1,
                                 xttdtfText=2, xttdtfHtml=3);
  // Read-Only-Area
  client_nr: string;          // GUID of client  (Read-Only)
  parent_project_nr: string;  // GUID of main project  (Read-Only)
  clientname: string;         // Client name (Read-Only)
  clientid: number;           // Client number (Read-Only)
  mainprojectname: string;    // Main project name (Read-Only)
  mainprojectnumber: string;  // Main project number (Read-Only)
  projectname: string;        // Project name (Read-Only)
  projectnumber: number;      // Project number (Read-Only)
  firstname: string;          // First name of editor (Read-Only)
  lastname: string;           // Last name of editor (Read-Only)
  creatorfirstname: string;    // First name of creator (Read-Only)
  creatorlastname: string;     // Last name of creator (Read-Only)

  projectstate: number;        // Project state (Read-Only)
  hastimeaccount: boolean;     // Is the project a timeable project? (Read-Only)
}
```

5.5.2 List

Reads a list of task records.

Type of call: `REST, GET`

Parameter:

pageNumber	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0
pageSize	Number of records per query 0 if no pagination is to be used.
sortfield	Sort column
sortdirection	Sort order

visibility	Visibility
	vmOwn = 0, vmWorkgroup = 1, vmAll = 2,
	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4

client_nr	Show tasks of a certain client only
project_nr	Show tasks of a project only
parent_project_nr	Show tasks of a main project only
comment	Search term
taskstate	Taskstate INTEGER
todocategory_nr	Task category GUID
priority	Priority INTEGER
taskpoolmode	0 = All, 1 = No pool, 2 = Pool only
favoritesonly	Only from project favorites
daterange	Date filter 0 = All, 1 = Today, 2 = Yesterday, 3 = This week, 4 = Last week, 5 = This month, 6 = Last month 7 = This year, 8 = Last year, 9 = Free entry 10 = This quarter, 11 = Last quarter, 12 = Last 3 days 13 = Last 7 days, 14 = Last 14 days, 15 = Last 21 days 16 = Last 30 days
daterangefrom	Daterange From (Only when daterange = 9) ISODATESTRING
daterangetill	Daterange To (Only when daterange = 9) ISODATESTRING

Example:

```
endpoint = "tasks"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
           'pagenumber': 0,
           'pagesize': 10
          }
```

```
response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  subject = data[0]['subject']
```

5.5.3 Read

Reads a task record.

Type of call: `REST, GET`

Parameter:

`todo_nr`	GUID of the task

Example:

```
endpoint = "tasks"
todo_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + todo_nr

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  subject = data['subject']
```

5.5.4 Create

Creates a new task record.

Type of call: `REST, POST`

Parameter:

`project_nr`	GUID of the project the task is created for

Example:

```
endpoint = "tasks"
URL = apiurl + endpoint
project_nr = "{12345678-…}"

DATA = {
         'project_nr': project_nr,
         'subject': 'Aufgabe 1',
        …
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  todo_nr = data['message']
```

5.5.5 Update

Creates a new task data set. The entire data record is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`todo_nr`	GUID of the task

Example:

```
endpoint = "tasks"
todo_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + todo_nr

DATA = {
         'subject': 'Aufgabe 1',
        …
         }

response = requests.request("PUT", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  message= data['message']
```

5.5.6 Delete

Deletes a task data record with all subordinate data objects (e.g. time stamps or notes)

Type of call: `REST, DELETE`

Parameter:

`todo_nr`	GUID of the task

Example:

```
endpoint = "tasks"
todo_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + todo_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message= data['message']
```

5.6 Times (Time stamps)

A time stamp contains a single recording data record, which is always assigned to a user and a project.

5.6.1 Data structure

```
export class Timestamp {
  times_nr: string;             // Unique GUID                  [Read-Only]
  project_nr: string;           // GUID of the project          [Required field]
  todo_nr: string;              // GUID of the task (optional)
  user_nr: string;              // Editor of the time stamp     [Required field]
  itemdate: Moment;             // Date of the time stamp (without time)
  fromtime: Moment;             // Date+time start              [Required field]
  tilltime: Moment;             // Date+time dend               [Required field]
  pausedsince: Moment;          // Time since pause is running
  comment: string;              // Comment
  minutesneeded: number;        // total of minutes needed (automatically calculated)
  minutespause: number;         // Pause in minutes
  manualentry: boolean;         // Is this a manual entry?
  state: number;                // State: 0=xttssAccountNormal, 1=xttssDontAccount,
                                     2=xttssIsAccounted
  creator_nr: string;           // GUID of the creator          [Read-Only]

  fulltimestampmode: number;      // Was the time stamp rounded?
  tilltimebeforerounding: Moment; // Not rounded end of time stamp

  timezonename: string;         // Name of time zone

  location: string;             // Location
  location_lat: number;         // GPS-Coordinates Latitude
  location_lng: number;         // GPS- Coordinates Longitude

  pricegroup_nr: string;        // GUID of the price group
  activity_nr: string;          // GUID of activity

  // Read-Only-Area
  clientname: string;         // Client name (Read-Only)
  clientid: number;           // Client number (Read-Only)
  mainprojectname: string;    // Main project name (Read-Only)
  mainprojectnumber: string;  // Main project number (Read-Only)
  projectname: string;        // Project name (Read-Only)
  projectnumber: number;      // Project number (Read-Only)

  taskname: string;           // Subject of task (Read-Only)
  tasknumber: number;         // Number of task (Read-Only)
  isteamproject: boolean;     // Is the project a team project (Read-Only)
  projectstate: number;       // Project state (Read-Only)

  firstname: string;          // First name of editor
  lastname: string;           // Last name of editor
  persnr: string;             // Personalnummer of editor
  client_nr: string;          // GUID of the client (Read-Only)
  parent_project_nr: string;  // GUID of the main project (Read-Only)

  activityname: string;      // Description of activity (Read-Only)
  activityshortcut: string;  // Short description of activity (Read-Only)
  activitycolor: number;     // Color of activity (Read-Only)
  pricegroupname: string;    // Name of the price group (Read-Only)
}
```

5.6.2 List

Reads a list of timestamps.

Type of call: `REST, GET`

Parameter:

`pageNumber`	Number of the requested page Starting with "0". I.e. page 1 corresponds to PageNumber = 0
`pageSize`	Number of records per query 0 if no pagination is to be used.
`sortfield`	Sort column
`sortdirection`	Sort order

`visibility`	Visibility
	vmOwn = 0, vmWorkgroup = 1, vmAll = 2,
	vmInChargeOfProject = 3, vmInChargeOfWorkgroup = 4

`client_nr`	Show time stamps of a client
`Parent_project_nr`	Show time stamps of a main project
`project_nr`	Show time stamps of a project
`todo_nr`	Show time stamps of a task
`user_nr`	Show time stamps of a user
`search`	Search term for comment
`projectname`	Search term for projectname
`projecttype_nr`	Project type GUID
`projectstate`	Project state INTEGER
`timestampstate`	Timestamp state INTEGER
`accountmode`	Account mode INTEGER
`hasLocation`	Has location data BOOL
`daterange`	Date filter 0 = All, 1 = Today, 2 = Yesterday, 3 = This week, 4 = Last week, 5 = This month, 6 = Last month 7 = This year, 8 = Last year, 9 = Free entry 10 = This quarter, 11 = Last quarter, 12 = Last 3 days 13 = Last 7 days, 14 = Last 14 days, 15 = Last 21 days 16 = Last 30 days
`daterangefrom`	Daterange From (Only when daterange = 9) ISODATESTRING
`daterangetill`	Daterange To (Only when daterange = 9) ISODATESTRING

Example:

```
endpoint = "times"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
            'pagenumber': 0,
            'pagesize': 10
           }

response = requests.request("GET", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  subject = data[0]['subject']
```

5.6.3 Read

Reads a time stamp record.

Type of call: `REST, GET`

Parameter:

`times_nr`	GUID of the timestamp

Example:

```
endpoint = "times"
times_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + times_nr

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  subject = data['subject']
```

5.6.4 Create

Creates a new time stamp.

Type of call: `REST, POST`

Parameter:

`project_nr`	GUID of the project the timestamp is created for
`todo_nr`	GUID of the task the timestamp is created for
`user_nr`	GUID of the user the timestamp is created for

Example:

```
endpoint = "times"
URL = apiurl + endpoint

DATA = {
         'project_nr': project_nr,
         'user_nr': 'user_nr,
         'fromtime': '2021-04-15T11:45:00.000Z',
         'tilltime': '2021-04-15T12:00:00.000Z',
         'comment': 'Kommentar zum Zeitstempel',
        …
         }

response = requests.request("POST", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  times_nr = data['message']
```

5.6.5 Update

Creates a new timestamp. The entire data record is overwritten with the specified values. Omitting fields leads to data loss.

Type of call: `REST, PUT`

Parameter:

`times_nr`	GUID of the timestamp

Example:

```
endpoint = "times"
times_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + times_nr

DATA = {
         'fromtime': '2021-04-15T11:45:00.000Z',
         'tilltime': '2021-04-15T12:00:00.000Z',
         'comment': 'Kommentar zum Zeitstempel',
        …
         }

response = requests.request("PUT", url = URL, json = DATA, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.6.6 Delete

Delete a time stamp.

Type of call: `REST, DELETE`

Parameter:

`times_nr`	GUID of the time stamp

Example:

```
endpoint = "times"
times_nr = '{1234-...}'
URL = apiurl + endpoint + '/' + times_nr

response = requests.request("DELETE", url = URL, headers = HEADERS)
if response:
  data = response.json()
  message = data['message']
```

5.7 System (Server configuration)

Various configuration parameters that may be necessary or informative in further processing can be read out via the system data. The system data record cannot be changed via the XTREST-API.

5.7.1 Data structure

```
export class System {
  system_nr: string;

  database_nr: string;         // GUID of the database
  databasename: string;        // Name of the database
  databasetype: string;        // Type of the database

  patchleveldb: number;        // Patchlevel of the database
  patchlevelserver: number;    // Patchlevel of XTCloudServers
  restapilevel: number;
  workgroupmode: number;       // Work group mode

  licence_nr: string;          // GUID of the license
  licenceholder: string;       // Name of the license holder

  currentusercount: number;    // Number of active users in the database
  licenceusercountuniversal: number;    // Number Universal users
  licenceusercountdesktoponly: number;  // Number Only-Desktop-User
  licenceusercountwebonly: number;      // Number Only-Web-user
  licenceusercountmobileonly: number;   // Number Only-Mobile
  licenceusercountnologin: number;      // Number No-Login-User
  licencextcloudserverdateend: Date;
  licencextcloudserverdateendmessage: string;
  licencesubscriptiondateend: Date;
  licencesubscriptiondateendmessage: string;

  version: string;             // Current version of server
  versiondate: string;         // Current version date of server

  versionavailable: string;     // Version available online for download
  versiondateavailable: string; // Date of online available version
  updateavailable: boolean;     // Is an update available?

  isdemodb: boolean;            // Is the database a demo database?
  dblanguageid: number;         // Language of database
  timezonebias: number;
  timezonename: string;         // Name of time zone of server
  currentservertime: Date;      // Current server time on server

  projectssetprojectnumbertomp: boolean; // Set the number of a new project to the
                                               same number of the main project
  projectsdeleteaskforcode: boolean;     // Ask for code before deleting clients or projects
  projectsautoincnumber: boolean;        // Automatically add project number
  manualentryrangeactive: boolean;       // Is the manual entry for timestamps active?
  manualentryrange: number;              // Add manual time stamp for days
  edittimesrangeactive: boolean;         // Is the date range for a manual entry for
                                                timestamps active?
  edittimesrange: number;                // Manually edit timestamps - days

  vatpercentage: number;                 // Standard tax rate

  passwordminimumlength: number;         // Minimum Password length
  passwordcomplexity: number;           // Password complexity

  pricetable_id_nr: string;             // Base price table

  moduleactiveclients: boolean;         // Module client management active?
  moduleactivetasks: boolean;           // Module tasks active?
  moduleactivehistory: boolean;         // Module activity report active?
  moduleactivenotes: boolean;           // Module notes active?
  moduleactivedms: boolean;             // Module document management active?
  moduleactiveinvoice: boolean;         // Module billing active?
  moduleactiveprojectitems: boolean;    // Module reimbursables active?
  moduleactivereminders: boolean;       // Module reminders active?
  moduleactivepricegroups: boolean;     // Module price groups active?
  moduleactiveworkgroups: boolean;      // Module work groups active?
  moduleactivebookingarchive: boolean;  // Module booking archive active?

  haswebcam: boolean;
}
```

5.7.2 Get

Reads a system record.

Type of call: `REST, GET`

Parameter:
none

Example:

```
endpoint = "system"
URL = apiurl + endpoint

response = requests.request("GET", url = URL, headers = HEADERS)
if response:
  data = response.json()
  databasename = data['databasename']
```

6 Search functions

6.1 GetUserByPersnr

Find user by personnel number.

Type of call: `JRPC, POST`

Parameter:

`persnr`	Personnel number of the user you are looking for

Return:

GUID of the user (`user_nr`)

Example:
```
endpoint = "jsonrpc"
URL = apiurl + endpoint
persnr = 1

PARAMS = {"jsonrpc":"2.0",
           "method":"GetUserByPersnr",
           "params":[ persnr ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  user_nr = data['result']
```

6.2 GetUserByLoginname

Find user by login name.

Type of call: `JRPC, POST`

Parameter:

`loginname`	Loginname of the user

Return:

GUID of the user (`user_nr`)

Example:
```
endpoint = "jsonrpc"
URL = apiurl + endpoint
loginname = "DEMO"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetUserByLoginname",
           "params":[ loginname ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  user_nr = data['result']
```

6.3 GetClientByClientID

Find client using the client number.

Type of call: `JRPC, POST`

Parameter:

`clientid`	Client number of the client you are looking for

Return:

GUID of the client (`client_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
clientid = 1

PARAMS = {"jsonrpc":"2.0",
           "method":"GetClientByClientID",
           "params":[ clientid ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  client_nr = data['result']
```

6.4 GetClientByClientname

Find client using the client name.

Type of call: `JRPC, POST`

Parameter:

`clientname`	Name of the client

Return:

GUID des Kunden (`client_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
clientname = "ABC Inc."

PARAMS = {"jsonrpc":"2.0",
           "method":"GetClientByClientname",
           "params":[ clientname ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  client_nr = data['result']
```

6.5 GetProjectByProjectnumber

Find a project by its project number.

Type of call: `JRPC, POST`

Parameter:

`projectnumber`	Alphanumeric project number of the project you are looking for

Return:

GUID of the project (`project_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
projectnumber = "1"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetProjectByProjectnumber",
           "params":[ projectnumber ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  project_nr = data['result']
```

6.6 GetTaskByTasknumber

Find a task using its task number.

Type of call: `JRPC, POST`

Parameter:

`tasknumber`	Task number of the task you are looking for

Return:

GUID of the task (`todo_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
tasknumber = "1"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetTaskByTasknumber",
           "params":[ tasknumber ],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  todo_nr = data['result']
```

6.7 GetClientByUserdefinedField

Find a client using a user defined field.

Type of call: `JRPC, POST`

Parameter:

`fieldnumber`	Number of the user defined field (1-10)
`search`	Search phrase

Return:

GUID of the client (`client_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
fieldnumber = 1
search = "12345"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetClientByUserdefinedField",
           "params":[ fieldnumber, search],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  client_nr = data['result']
```

6.8 GetProjectByUserdefinedField

Find a project by a user defined field.

Type of call: `JRPC, POST`

Parameter:

`fieldnumber`	Number of the user defined field (1-10)
`search`	Search phrase

Return:

GUID of the project (`project_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
fieldnumber = 1
search = "12345"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetProjectByUserdefinedField",
           "params":[ fieldnumber, search],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  project_nr = data['result']
```

6.9 GetTaskByUserdefinedField

Find a task by a user defined field.

Type of call: `JRPC, POST`

Parameter:

`fieldnumber`	Number of the user defined field (1-5)
`search`	Search phrase

Return:

GUID of the task (`todo_nr`)

Example:

```
endpoint = "jsonrpc"
URL = apiurl + endpoint
fieldnumber = 1
search = "12345"

PARAMS = {"jsonrpc":"2.0",
           "method":"GetTaskByUserdefinedField",
           "params":[ fieldnumber, search],
           "id":924
         }

response = requests.request("POST", url = URL, json = PARAMS, headers = HEADERS)
if response:
  data = response.json()
  todo_nr = data['result']
```

7 Help functions

7.1 GetClientname

Query the name of a client via his client_nr.

Type of call: `JRPC, POST`

Parameter:

`client_nr`	GUID of the client

Return:

Name of the client

7.2 GetProjectname

Query the name of a project via its project_nr.

Type of call: `JRPC, POST`

Parameter:

`project_nr`	GUID of the project

Return:

Name of the project

7.3 GetTaskname

Query the subject of a task via its todo_nr.

Type of call: `JRPC, POST`

Parameter:

`todo_nr`	GUID of the task

Return:

Name of the task

7.4 GetUsername

Query the name of a user via their user_nr.

Type of call: `JRPC, POST`

Parameter:

`user_nr`	GUID of the user

Return:

Name of the user

7.5 ProjectAssignUser

Assign user to a project.

Type of call: `JRPC, POST`

Parameter:

`project_nr`	GUID of the project
`user_nr`	GUID of the user
`assignsubprojects`	Assign sub-projects if available?

Return:

Status 0 = Error, 1 = Ok

7.6 ProjectUnassignUser

Delete user from a project.

Type of call: `JRPC, POST`

Parameter:

`project_nr`	GUID of the project
`user_nr`	GUID of the user

Return:

Status 0 = Error, 1 = Ok

7.7 TaskSetUser

Set the user for a task.

Type of call: `JRPC, POST`

Parameter:

`todo_nr`	GUID of the task
`user_nr`	GUID of the user

Return:

todo_nr and user_nr

7.8 TaskSetState

Set the state of a certain task.

Type of call: `JRPC, POST`

Parameter:

`todo_nr`	GUID of the task
`taskstate`	INTEGER task state (0 – 11, 100 – 104)

Return:

todo_nr and taskstate

7.9 ProjectSetState

Set project status of a project.

Aufrufart: `JRPC, POST`

Parameter:

`project_nr`	GUID of Project
`projectstate`	INTEGER Project state (0 – 6, 100 – 109)

Rückgabe:

project_nr und projectstate

7.10 ClientGetMinutesNeeded

Get needed time of a user in minutes.

Type of call: `JRPC, POST`

Parameter:

`client_nr`	GUID of the client

Return:
Minutes INTEGER

7.11 ProjectGetMinutesNeeded

Get needed time of a projet in minutes.

Type of call: `JRPC, POST`

Parameter:

`project_nr`	GUID of the project

Return:
Minutes INTEGER

7.12 TaskGetMinutesNeeded

Get needed time of a task in minutes.

Type of call: `JRPC, POST`

Parameter:

`todo_nr`	GUID of the task

Return:
Minutes INTEGER

7.13 ProjectStart

Start a project.

Type of call: `JRPC, POST`

Parameter:

`project_nr`	GUID des Projekts
`user_nr`	GUID des Users
`comment`	Zeitstempelkommentar (optional)

Return:
JSON of timestamp

7.14 TaskStart

Start a task.

Type of call: `JRPC, POST`

Parameter:

`todo_nr`	GUID der Aufgabe
`user_nr`	GUID des Users
`comment`	Zeitstempelkommentar (optional)

Return:
JSON of timestamp

7.15 TimestampStop

Stops a running time stamp for a user.

Type of call: `JRPC, POST`

Parameter:

`user_nr`	GUID des Users
`comment`	Zeitstempelkommentar (optional)

Return:
JSON of timestamp

8 Examples

The following code examples, created in Python, are intended to explain the interaction of the individual XT functions. The examples are designed as console applications to avoid unnecessary overhead.

Further examples of other programming environments can be found in the folder "\Examples\".

If you make the API connection using Python, it is best to use the API wrapper from the folder "\Examples\Python\xtrestapi_wrapper". Examples are under "\Examples\Python\"

```
# ================================================================================
# EXAMPLE
# Demonstrates how to access the XTREST-API without using the API-Wrapper
# ================================================================================

import requests
import json

# Base-URL of XTWeb-Api
apiurl = "http://localhost:9000/api/"
apikey = "12345678-..."
API_USERNAME = "demo"
API_PASSWORD = "demo"

# Endpoint: Login
url = apiurl + "login"

headers = {
    "content-type": "application/json",
    "accept": "application/json",
    "apikey": apikey,
    "jwtusername": "demo",
    "jwtpassword": "demo"
}

response = requests.request("POST", url, headers=headers, auth=(API_USERNAME, API_PASSWORD))

print(f"\nAUTHENTICATE with POST {url} : {headers} ")
print(f"{response.status_code}: {response.reason}")
print(response.text)

if not response: exit
data = json.loads(response.text)

# JWT Token to send with every header
try:
    token = data['token']
    print("\nToken: " + token)

    # Get User list

    # Endpoint: Login
    url = apiurl + "users"

    headers = {
    "content-type": "application/json",
    "accept": "application/json, text/plain, */*",
    "Accept-Encoding": "gzip, deflate, br",
    "apikey": apikey,
    "authentication": "Bearer " + token,
    "Access-Control-Request-Headers": "authentication"
    }

    params = {
    "state":1,                # Only active users
    "sortdirection":"asc",    # Default: asc
    "pageNumber":0,           # Default: 0
    "pageSize":10             # Default: 10
    }

    response = requests.request("GET", url, json=params, headers=headers,
        auth=(API_USERNAME, API_PASSWORD))

    print(f"\nGET {url}?{params}")
    print(f"\n{response.status_code}: {response.reason}")
    print(response.text)

    if not response: exit
    data = response.json()
    firstname = data[0]['firstname']
    print('=> Firstname: ', firstname)

    # Get user_nr of first user in the list
    user_nr = data[0]['user_nr']

    # Get User =======================================================

    # Endpoint: user
    url = apiurl + "users" + "/" + user_nr

    response = requests.request("GET", url, headers=headers, auth=(API_USERNAME, API_PASSWORD))

    print(f"\nGET {url}")
    print(f"\n{response.status_code}: {response.reason}")
    print(response.text)

    if not response: exit
    data = response.json()
    firstname = data['firstname']
    print('=> Firstname: ', firstname)

    # Get list of timestamps for user ================================

    # Endpoint: timestamps
    url = apiurl + "times"

    params = {
    "user_nr":user_nr   # Only times of current user
    }

    response = requests.request("GET", url, json=params, headers=headers,
        auth=(API_USERNAME, API_PASSWORD))

    if not response: exit
    print(f"\nGET {url}?{params}")
    print(f"\n{response.status_code}: {response.reason}")
    print(response.text)

    data = response.json()
    # Pick first timestamp from list
    minutesneeded = data[0]['minutesneeded']
    print('=> minutesneeded: ', minutesneeded)

except KeyError:
    print(f"\nERROR: AUTHENTICATION FAILED")
```

9 Document history

All changes made to this document are recorded here in chronological order.

Datum	Beschreibung
01.03.2021	First version of the documentation