



xpert**TIMER**
start. stop. track time.

Dokumentation Xpert-Timer REST-API (XTCloudserver)

Version: 8.0

DB-Patchlevel: PL 90

API-Revision: R1

Inhaltsverzeichnis

Inhaltsverzeichnis	2
1 Allgemeines	4
2 Anbinden über Programmiersprachen	5
2.1 Allgemeines.....	5
2.1.1 REST-API.....	5
2.1.2 JSON-RPC	5
2.2 Python.....	5
2.3 Aufrufmethoden.....	5
2.3.1 REST.....	5
2.3.2 JSON-RPC (JRPC).....	5
2.4 Allgemeine Verfahrensweisen.....	7
2.4.1 Primärschlüssel (GUID).....	7
2.4.2 Datum / Zeit	7
2.4.3 Löschen von Datensätzen.....	7
2.4.4 Datenobjekte abrufen / Funktionen aufrufen	7
2.4.5 Hinweise zum Login.....	7
2.4.6 Das Rechtesystem	8
2.4.7 Die Sichtbarkeiten.....	8
2.4.8 Die Sicherheitsstufen	8
2.4.9 Der API-Login	9
2.4.10 Der API-Key.....	9
3 Objekthierarchie.....	10
4 Login (Benutzeranmeldung)	11
5 Datenobjekte.....	13
5.1 Users (Mitarbeiter, Benutzer).....	13
5.1.1 Datenstruktur	13
5.1.2 Auflisten.....	13
5.1.3 Einlesen (GET)	14
5.1.4 Erstellen (CREATE)	14
5.1.5 Ändern (UPDATE)	15
5.1.6 Löschen (DELETE)	15
5.2 Clients (Kunden).....	16
5.2.1 Datenstruktur	16
5.2.2 Auflisten.....	16
5.2.3 Einlesen (GET)	17
5.2.4 Erstellen (CREATE)	17
5.2.5 Ändern (UPDATE)	18
5.2.6 Löschen (DELETE)	18
5.3 Contacts (Kundenadressen, Kontakte).....	19
5.3.1 Datenstruktur	19
5.3.2 Auflisten.....	20
5.3.3 Einlesen (GET)	20
5.3.4 Erstellen (CREATE)	20
5.3.5 Ändern (UPDATE)	21
5.3.6 Löschen (DELETE)	21
5.4 Projects (Projekte).....	23
5.4.1 Datenstruktur	23
5.4.2 Auflisten.....	24
5.4.3 Einlesen (GET)	25
5.4.4 Erstellen (CREATE)	25
5.4.5 Ändern (UPDATE)	25
5.4.6 Löschen (DELETE)	26
5.5 Tasks (Aufgaben)	27
5.5.1 Datenstruktur	27
5.5.2 Auflisten.....	28

5.5.3	Einlesen (GET)	29
5.5.4	Erstellen (CREATE)	29
5.5.5	Ändern (UPDATE)	29
5.5.6	Löschen (DELETE)	30
5.6	Times (Zeitstempel)	31
5.6.1	Datenstruktur	31
5.6.2	Auflisten	32
5.6.3	Einlesen (GET)	33
5.6.4	Erstellen (CREATE)	33
5.6.5	Ändern (UPDATE)	33
5.6.6	Löschen (DELETE)	34
5.7	System (Serverkonfiguration)	35
5.7.1	Datenstruktur	35
5.7.2	Einlesen (GET)	36
6	Suchfunktionen	37
6.1	GetUserByPersnr	37
6.2	GetUserByLoginname	37
6.3	GetClientByClientID	38
6.4	GetClientByClientname	38
6.5	GetProjectByProjectnumber	39
6.6	GetTaskByTasknumber	39
6.7	GetClientByUserdefinedField	40
6.8	GetProjectByUserdefinedField	40
6.9	GetTaskByUserdefinedField	41
7	Hilfsfunktionen	42
7.1	GetClientname	42
7.2	GetProjectname	42
7.3	GetTaskname	42
7.4	GetUsername	42
7.5	ProjectAssignUser	43
7.6	ProjectUnassignUser	43
7.7	TaskSetUser	43
7.8	TaskSetState	44
7.9	ProjectSetState	44
7.10	ClientGetMinutesNeeded	44
7.11	ProjectGetMinutesNeeded	44
7.12	TaskGetMinutesNeeded	45
7.13	ProjectStart	45
7.14	TaskStart	45
7.15	TimestampStop	45
8	Beispiele	47
9	Dokumenthistorie	49

1 Allgemeines

Die XTRESTAPI ist eine Schnittstelle um datenbankunabhängig auf die Datensätze des Programms zugreifen zu können.

WICHTIG: Um die XTRESTAPI verwenden zu können muss der XTCloudserver installiert sein und über XTWeb erreichbar sein.

Die aktuelle Version dieser Dokumentation finden Sie unter:
http://download.xperttimer.de/helpfiles/XTRESTAPI_Doc.pdf

Die XTRESTAPI inkl. Demo Sourcecode als ZIP finden Sie unter:
<http://download.xperttimer.de/additional/XTRESTAPIDemo.zip>

Folgende Möglichkeiten sind vorgesehen:

- Webserververbindung aufbauen / trennen
- Benutzer einloggen/ausloggen
- Projekt, Kunden, Benutzer, Aufgaben, Zeitstempel anlegen, ändern oder löschen.
- Vorhandene Projekte, Kunden, Benutzer, Aufgaben, Zeitstempellisten auslesen
- Zeitstempel nachtragen
- Daten über den angemeldeten Benutzer abfragen
- Systemdaten abfragen
- Projekte zu Benutzern zuweisen
- Projekt- Kunden- Benutzer, Aufgaben, Zeitstempeldetails auslesen und ändern
- Projekte starten, stoppen oder pausieren
- Kunden, Projekte, Aufgaben und Benutzer nach verschiedenen Kriterien suchen
- Berechnen von Zeitstempelsummen unter Berücksichtigung von Benutzer, Kunde, Projekt, Aufgabe Zeitraum

2 Anbinden über Programmiersprachen

2.1 Allgemeines

Der Zugriff erfolgt über zwei unterschiedliche Kommunikationsarten.

2.1.1 REST-API

Über Aufrufe der REST-API können Datenobjekte gelesen sowie geschrieben werden. Des Weiteren können Datenobjektlisten eingelesen werden.

2.1.2 JSON-RPC

Über JSON-RPC können Serverfunktionen ausgelöst werden. Hierdurch können z.B. Projekte gestartet oder Einzelwerte oder Summen abgerufen werden.

2.2 Python

Für die Anbindung an Python gibt es einen **API-Wrapper** im Unterverzeichnis „xtrestapi_wrapper“ der die geläufigsten Funktionen zur Verfügung stellt und Ihnen viel Unterstützung für den Zugriff bietet. Der API-Wrapper enthält auch diverse Hilfsroutinen um die Arbeit mit der API zu erleichtern.

Siehe Beispiel „Examples\Python“

Um die Python-Skripts ausführen zu können benötigen Sie eine Python-Installation auf Ihrem System. Diese ist kostenlos erhältlich unter: <https://www.python.org/downloads/>

Die Skripts lassen sich dann in der mitgelieferten IDE (IDLE) laden und ausführen. Um auf Ihren im Unternehmen installierten XTCloudserver zugreifen zu können müssen sie nur noch folgende Anmeldedaten in den Beispielen ersetzen:

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

2.3 Aufrufmethoden

2.3.1 REST

Über Aufrufe der REST-API können Datenobjekte gelesen sowie geschrieben werden. Des Weiteren können Datenobjektlisten eingelesen werden.

Die Aufrufparameter werden bei lesendem Zugriff als URL-Parameter übergeben, bei schreibendem Zugriff als POST-Parameter.

2.3.2 JSON-RPC (JRPC)

Über JSON-RPC können Serverfunktionen ausgelöst werden. Hierdurch können z.B. Projekte gestartet oder Einzelwerte oder Summen abgerufen werden.

Die Aufrufparameter werden in einer JRPC-Struktur als POST-Parameter übergeben. Die Reihenfolge der Parameter ist hierbei ausschlaggebend.

Beispiel:

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

PARAMS = {
    "jsonrpc": "2.0",                // JRPC-Version
    "method": "GetUserByPersnr",    // Name der aufzurufenden Remote-Methode
    "params": [ persnr ],           // Kommagetrenntes Array mit Parametern
```

```
    "id":924                                     // Nummer des Aufrufs. Hilfreich bei
                                                asynchronem Aufruf
                                                Kann beliebig vorbelegt werden z.B. mit 1
    }

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

Die Rückgabe hat folgendes Format:

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

2.4 Allgemeine Verfahrensweisen

2.4.1 Primärschlüssel (GUID)

Datenobjekte haben im Xpert-Timer immer eine eindeutige Kennung über welche sie angesprochen werden. Eine sog. GUID (Global Unique Identifier). Diese GUID ist eine 38-Stellige alphanumerische Kennung die es ermöglicht sehr komfortabel ohne Nummernkreise zu arbeiten.

Beispiel:

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

Eine neue GUID kann über die Funktion „XTRESTAPI.CreateGUID()“ erstellt werden, falls erforderlich.

Falls ein Datensatz eine solche GUID enthält, so ist das entsprechende Feld immer durch die Endung „_nr“ gekennzeichnet. Z.B. `user_nr`, `client_nr`, `project_nr` usw.

Es gibt im Xpert-Timer zwei verschiedene Sonderfälle einer GUID um den Zustand NULL bzw. „nicht verknüpft“ zu kennzeichnen. Diese werden benötigt um die referenzielle Integrität für SQL-JOINS zu erhalten und vereinfacht somit das Auslesen von SQL-Datenmenge erheblich.

```
{11111111-1111-1111-1111-111111111111} = Verknüpfung mit leerem Datenobjekt  
{00000000-0000-0000-0000-000000000000} = Leerer Feldinhalt
```

2.4.2 Datum / Zeit

Datums oder Zeitangaben werden nach ISO8601 umgewandelt. Eine evtl. vorhandene Zeitzone wird dabei nicht berücksichtigt.

Bsp.:

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

2.4.3 Löschen von Datensätzen

Datensätze werden innerhalb der XT-Datenbank niemals wirklich gelöscht, sondern nur als „gelöscht“ gekennzeichnet (Storniert).

Werden Datenobjekte gelöscht, so werden ebenfalls alle untergeordneten Datenobjekte gelöscht.

2.4.4 Datenobjekte abrufen / Funktionen aufrufen

Die Syntax der Beispiele entspricht zur Vereinfachung der Python-Notation.

Für alle Aufrufe wird folgende Base-Url verwendet: `apiurl = "http://localhost:9000/api/"`

2.4.5 Hinweise zum Login

Um die XTRESTAPI für Automatisierungen verwenden zu können, sollten Sie sich einen eigenen Benutzer einrichten. Z.B. mit dem Namen „XTAPIUSER“ oder „SYSTEM“. Sie können auch einen vorhandenen User verwenden, allerdings wird dieser dann für alle erzeugten und geänderten Datenobjekte als Ersteller, bzw. Änderer eingetragen. Dieser XTAPIUSER sollte nur so viele Rechte zugewiesen bekommen wie für die Aufgaben benötigt wird.

Sie sollten zur Sicherheit für jede externe Anwendung die das API verwendet auf dem XTCloudserver einen eigenen **API-Key** erstellen. Der API-Key wird dann bei jedem Aufruf im Header des http-Befehls mitgeschickt. Falls der Key auf dem Server nicht bekannt ist, werden die Aufrufe abgewiesen.

Vor dem Abschicken eines Befehls an die XTRESTAPI muss eine Benutzeranmeldung stattfinden. Das Rückgabebetoken (JSON-Web-Token) aus der Anmeldung muss dann mit jedem folgenden Aufruf im Header des http-Befehls mitgeschickt werden um den User auf dem Server identifizieren zu können.

2.4.6 Das Rechtssystem

Der Xpert-Timer besitzt ein umfangreiches Rechtssystem um die Zugriffsrechte der einzelnen Benutzer bzw. Benutzergruppen steuern zu können. So existieren i.d.R. für jedes Datenobjekt Rechte die das Lesen, Schreiben, Löschen oder Ändern erlauben. Falls eine Funktion aufgerufen wird für die nicht ausreichend Rechte vorhanden sind, so wird dies mit einer Fehlermeldung quittiert.

Zusätzlich zu den Standardrechten gibt es eine Vielzahl an Sonderrechten die z.B. das Projektstartverhalten oder die Verfahrensweise beim Nachtrag beeinflussen.

Die Rechtezuweisung kann nur im Xpert-Timer Pro (Windows) vorgenommen werden.

2.4.7 Die Sichtbarkeiten (Visibility)

Der Xpert-Timer besitzt ein umfangreiches System um die Sichtbarkeit von Datensätzen zu steuern. I.d.R. bezieht sich die Steuerung darauf nur eigene, oder auch Team-Datensätze sehen zu können. Werden im Xpert-Timer die Arbeitsgruppen verwendet, so werden auch diese in die Sichtbarkeiten einbezogen.

Folgende Sichtbarkeitsstufen sind vorhanden:

- Nur eigene Datensätze
- Datensätze anderer Mitarbeiter
- Wenn Projektleiter
- Wenn in selber Arbeitsgruppe
- Wenn Arbeitsgruppenleiter
- Alle

Die Sichtbarkeitsstufe kann i.d.R. beim Aufruf einer Lesenden Funktion als Parameter übergeben werden.

Die möglichen Sichtbarkeiten werden über das Rechtssystem geregelt.

Als Aufrufparameter sind für die Visibility folgende Werte möglich:

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

2.4.8 Die Sicherheitsstufen (Security level)

Zusätzlich zu den Standardsichtbarkeiten gibt es noch die Möglichkeit die Sichtbarkeit über Sicherheitsstufen zu steuern. Dadurch können vertrauliche Kunden, Projekte, Dokumente, Aufgaben nur für bestimmte Mitarbeiter zugreifbar gemacht werden. Die Sicherheitsstufen gehen von 0 = Keine bis 5 = Streng vertraulich. Standardmäßig wird allen Mitarbeitern und Datenobjekten die Stufe 2 = Mittel zugewiesen.

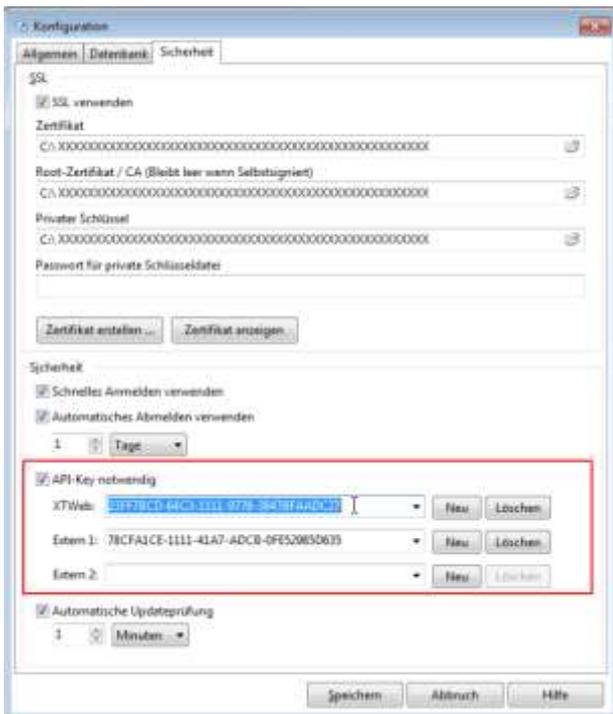
Die Sichtbarkeit über die Sicherheitsstufe wird vollständig vom Server gehandhabt und kann nicht für einzelne Abfragen beeinflusst werden.

2.4.9 Der API-Login

Der API-Login stellt eine zusätzliche Sicherheitsebene für den Zugriff auf den XTCloudserver dar um missbräuchliche Anmeldungen zu unterbinden. Ist der API-Login aktiviert, so muß vor jedem Zugriff auf eine Ressource des Server eine Anmeldung mit Benutzername und Passwort (Basic-Auth) erfolgen.

2.4.10 Der API-Key

Um auf den XTCloudserver über die REST-API zugreifen zu können sollte aus Sicherheitsgründen für jede Anwendung ein API-Key am Server erzeugt werden. Nur wenn der API-Key am Server freigeschaltet wurde kann über die API auf die Serverfunktionen zugegriffen werden. Falls man einen API-Zugriff dann zu einem späteren Zeitpunkt unterbinden möchte, reicht es den API-Key zu deaktivieren. Falls der Key auf dem Server nicht bekannt ist, werden die Aufrufe abgewiesen.



3 Objekthierarchie

Folgende Stammdatenobjekte bilden den Kern des Xpert-Timers.

- Benutzer
- Kunden / Kundenadressen
- Projekte

Für diese Stammdatenobjekte können folgende Erfassungsdatenobjekte erstellt werden:

- Aufgaben
- Notizen
- Zeitstempel

Der hierarchische Aufbau ist immer wie folgt:



Objekte in eckigen Klammern „[]“ sind hierbei optional. D.h. ein Projekt muss nicht an einem Kunden hängen und eine Unterprojektebene ist nicht verpflichtend. Es ist jedoch auch nur maximal eine Unterprojektebene möglich.

z.B.



NDdGRDAwMS1EQzc4LTQ0MzUtQTQ0NS0xRjRGQzVGRjQ4REZ9Iiwicm9sZXMiOiIiLCJ4dG1haW53b3JrZ3JvdXBfbnIiOiJ7Njg1N
DU5NkMtMjU2Mi00QUEyLTg5NjMtMDA5ODA2RjhDQTVFfSj9.jKt0hmEI2BoxxbBXijHiG5qtCDAbf5IYd05eRVv-
9ayS_e4AtaLpShHNhSVRzB9vjbAKqSZFbKakyUKJhquW-w" }

Wenn man sich diesen dekodiert ansieht, z.B. über <https://www.jsonwebtoken.io>
dann erhält man folgende Datenfelder:

```
{
  "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 Datenobjekte

Datenobjekte werden in der Regel über Standard-REST-Aufrufe abgefragt oder bearbeitet. Datensätze können über REST gelesen, erstellt, geändert oder gelöscht werden. Die Datensätze werden im JSON-Format verarbeitet. Folgende Endpoints stehen dafür zur Verfügung:

5.1 Users (Mitarbeiter, Benutzer)

Ein Benutzer ist ein Mitarbeiter in der Datenbank. Er wird i.d.R. mit seiner eindeutigen `user_nr` referenziert. Der Ersteller eines Datensatzes wird i.d.R. über `creator_nr` angesprochen.

5.1.1 Datenstruktur

```
export class User {
  user_nr: string;           // Eindeutige GUID           [Pflichtfeld]
  firstname: string;        // Vorname           [Pflichtfeld]
  lastname: string;         // Nachname          [Pflichtfeld]

  loginname: string;       // Loginname für Benutzeranmeldung [Pflichtfeld]
  persnr: number;          // Personalnummer

  title: string;           // Titel
  nickname: string;        // Spitzname

  active: boolean;         // Ist der User aktiv? d.h. Sichtbar
  admin: boolean;          // Ist der User Administrator?
  changepwonlogin: boolean; // Passwort bei nächster Anmeldung ändern?

  pricegroup nr: string;   // Standard Preisgruppe
  mainworkgroup nr: string; // Welcher Hauptarbeitsgruppe wurde der User zugeor.?
  costcenter: string;     // Kostenstelle
  securitylevel: number;   // Sicherheitsstufe
  email: string;           // E-Mail Adresse
  hoursday: number;
  daysweek: number;

  loginmode: number;       // Einloggen über Desktop und/oder Web

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

  date_created: Date;      // Erstellungsdatum        [Read-Only]
  lastchange: Date;        // Letzte Änderung         [Read-Only]
  creator nr: string;      // Ersteller                [Read-Only]

  usergroups: string[];    // Zugeordnete Rechtegruppen
}
```

5.1.2 Auflisten

Liest eine Liste von Userdatensätzen ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
------------	--------------------------------

pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit

project_nr	Zugeordnet zu Projekt
not_in_project_nr	Nicht zugeordnet zu Projekt
username	Suchbegriff

Beispiel:

```

endpoint = "users"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
          'pagenumber': 1,
          'pagesize': 10
        }

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

5.1.3 Einlesen (GET)

Liest einen Userdatensatz

Aufrufart: REST, GET

Parameter:

user_nr	GUID des Users
---------	----------------

Beispiel:

```

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 Erstellen (CREATE)

Erstellt einen neuen Userdatensatz. Der neue User bekommt automatisch die Rechtegruppe „Mitarbeiter“ zugewiesen. Das Passwort entspricht dem Loginnamen und muss beim ersten Login geändert werden.

Aufrufart: REST, POST

Parameter:

keine

Beispiel:

```

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 Ändern (UPDATE)

Erstellt einen neuen Userdatensatz. Dabei wird das gesamte Userdatensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

user_nr	GUID des Users
---------	----------------

Beispiel:

```

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 Löschen (DELETE)

Löscht einen Userdatensatz mit allen untergeordneten Datenobjekten (z.B. Zeitstempel oder Aufgaben)

Aufrufart: REST, DELETE

Parameter:

user_nr	GUID des Users
---------	----------------

Beispiel:

```

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 (Kunden)

Die Kundenebene ist für die Zeiterfassung nicht erforderlich, erleichtert jedoch die Auftragsverwaltung erheblich. Ein Kunde wird über die `client_nr` referenziert.

5.2.1 Datenstruktur

```
export class Client {
  client_nr: string;           // Eindeutige GUID           [Read-Only]
  clientid: number;           // Kundennummer           [Pflichtfeld]
  clientname: string;         // Kundenname             [Pflichtfeld]
  info: string;               // Kommentar zum Kunden
  active: boolean;            // Aktiv/Inaktiv
  securitylevel: number;      // Sicherheitsstufe
  color: number;              // Farbe
  hasvat: boolean;            // Umsatzsteuerpflichtig
  clientreference: string;    // Referenznummer
  vatid: string;              // Umsatzsteuer-ID

  maincontact_nr: string;     // GUID des Hauptkontakts
  pricetable_id_nr: string;   // GUID der Preistabelle
  created_workgroup_nr: string; // GUID der Hauptarbeitsgruppe

  userdefined1: string;       // Frei belegbare Benutzerfelder 1-10
  userdefined2: string;
  userdefined3: string;
  userdefined4: string;
  userdefined5: string;
  userdefined6: string;
  userdefined7: string;
  userdefined8: string;
  userdefined9: string;
  userdefined10: string;

  date created: Date;         // Erstellt am           [Read-Only]
  lastchange: Date;           // Letzte Änderung       [Read-Only]
  creator_nr: string;         // GUID Ersteller        [Read-Only]
}
```

5.2.2 Auflisten

Liest eine Liste von Kundendatensätzen ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit
user_nr	Mitarbeiter

state	0 = Alle, 1 = Nur aktive, 2 = Nur mit zugeordneten Projekten
favoritesonly	Nur wenn ein Kundenprojekt als Favorit markiert wurde
clientname	Suchbegriff

Beispiel:

```

endpoint = "clients"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
          'pagenumber': 1,
          'pagesize': 10
        }

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

5.2.3 Einlesen (GET)

Liest einen Kundendatensatz

Aufrufart: REST, GET

Parameter:

client_nr	GUID des Kunden
-----------	-----------------

Beispiel:

```

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 Erstellen (CREATE)

Erstellt einen neuen Kundendatensatz

Aufrufart: REST, POST

Parameter:

keine

Beispiel:

```

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 Ändern (UPDATE)

Erstellt einen neuen Kundendatensatz. Dabei wird das gesamte Datensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

client_nr	GUID des Kunden
-----------	-----------------

Beispiel:

```

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

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

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

5.2.6 Löschen (DELETE)

Löscht einen Kundendatensatz mit allen untergeordneten Datenobjekten (z.B. Projekten, Zeitstempel oder Aufgaben)

Aufrufart: REST, DELETE

Parameter:

client_nr	GUID des Kunden
-----------	-----------------

Beispiel:

```

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 (Kundenadressen, Kontakte)

Die Kundenadressen (Kontakte) werden verwendet um verschiedene Ansprechpartner in einem Unternehmen verwalten zu können. Eine Kundenadresse gehört immer fest zu einem Kunden und wird über die `contact_nr` referenziert.

5.3.1 Datenstruktur

```
export class Contact {
  contact_nr: string;           // Eindeutige GUID           [Read-Only]
  client_nr: string;           // GUID des Kunden           [Pflichtfeld]
  firstname: string;           // Vorname
  name: string;                 // Nachname                   [Pflichtfeld]
  description: string;         // Beschreibung der Funktion im Unternehmen
  comment: string;             // Kommentar

  phone1: string;              // Telefon 1
  phone2: string;              // Telefon 2
  fax: string;                 // Fax
  mobile: string;              // Mobil
  email1: string;              // Email 1
  email2: string;              // Email 2
  website: string;             // Webseite
  address1: string;            // Adresse 1
  address2: string;            // Adresse 2
  zip: string;                 // PLZ
  city: string;                // Ort
  state: string;               // Bundesland
  country: string;             // Land

  title: string;               // Titel
  salutation: number;          // Anrede
  birthday: Date;              // Geburtstag
  contacttype: number;         // Art des Kontakts (z.B. Rechnungsanschrift,
                               // Firmenadresse, usw.)

  textualaddress: string;      // Freie Briefanschrift
  clientnameaddress: string;   // Kundenname der auf der Briefanschrift stehen soll

  active: boolean;             // Aktiv/Inaktiv

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

  date_created: Date;          // Erstellt am               [Read-Only]
  lastchange: Date;            // Letzte Änderung           [Read-Only]
  creator_nr: string;          // GUID des Erstellers       [Read-Only]
}
```

5.3.2 Auflisten

Liest eine Liste von Kontaktdatenätzen ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit
user_nr	Mitarbeiter
client_nr	Übergeordneter Kunde
contactname	Suchbegriff

Beispiel:

```

endpoint = "contacts"
URL = apiurl + endpoint
client_nr = "{12345678-...}"

PARAMS = {'sortdirection': 'asc',
          'client_nr': client_nr,
          'pagenumber': 1,
          'pagesize': 10
          }

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

5.3.3 Einlesen (GET)

Liest einen Kontaktdatenatz.

Aufrufart: REST, GET

Parameter:

contact_nr	GUID des Kontakts
------------	-------------------

Beispiel:

```

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 Erstellen (CREATE)

Erstellt einen neuen Kontaktdatenatz.

Aufrufart: REST, POST

Parameter:

keine

Beispiel:

```

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 Ändern (UPDATE)

Erstellt einen neuen Kontaktdatensatz. Dabei wird das gesamte Datensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

contact_nr	GUID des Kontakts
------------	-------------------

Beispiel:

```

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

DATA = {
    'name': 'Muster',
    'firstname': 'Max',
    ...
}

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

5.3.6 Löschen (DELETE)

Löscht einen Kontaktdatensatz.

Aufrufart: REST, DELETE

Parameter:

contact_nr	GUID des Kontakts
------------	-------------------

Beispiel:

```

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 (Projekte)

Die Projekte sind der Kernbestandteil des Xpert-Timer und deshalb ist für nahezu alle Aktionen eine `project_nr` notwendig.

5.4.1 Datenstruktur

```
export class Project {
  project nr: string;           // Eindeutige GUID [Pflichtfeld]
  parent project nr: string;   // Eindeutige GUID des Hauptprojekts
  client nr: string;           // Eindeutige GUID des Kunden

  projectname: string;         // Name des Projekts [Pflichtfeld]
  projectnumber: string;       // Alphanummerische Projektnummer

  comment: string;             // Projektbeschreibung
  securitylevel: number;       // Sicherheitsstufe
  color: number;               // Farbe

  projecttype nr: string;      // GUID des Projekttyps
  minutesneeded: number;       // Benötigte Zeit in Minuten. Diese Feld ist READ-ONLY
                                // und wird aus den Zeitstempeln berechnet
  minutesestimated: number;

  state: number;               // Projektstatus: 0=psRunning, 1=psPaused, 2=psCancelled,
                                // 3=psFinished, 4=psInactive, 5=psAccounted
                                // 6=psToBeAccounted, 100-109=psUserdefined1-10

  progress: number;            // Fortschritt in Prozent
  priority: number;            // Priorität
  timeaccount: boolean;        // Zeitkonto Ja/Nein
  accountmode: number;         // Art des Abrechnungsmodus
  teamproject: boolean;        // Teamprojekt. Ist das Projekt für alle sichtbar?
  subprojectorder: number;     // Sortierreihenfolge der Unterprojekte

  getpricefrom: number;        // Abrechnungssatz
  flatrateprice: number;       // Pauschalpreis
  billingunit: number;         // Abrechnungseinheit
  priceperbillingunit: number; // Preis pro Einheit

  addtimestampmode: number;

  startdate: Moment;           // Geplanter Projektbeginn
  enddate: Moment;             // Geplantes Projektende
  datestarted: Moment;         // Tatsächlicher Projektbeginn
  datefinished: Moment;        // Tatsächliches Projektende

  maincontact nr: string;      // GUID des Hauptkontakts
  user nr: string;              // Verantwortlicher
  secondincharge_nr: string;    // 2.Verantwortlicher
  pricetable_id_nr: string;     // Preistabelle
  created workgroup nr: string; // Hauptarbeitsgruppe

  userdefined1: string;         // Benutzerfeld 1-10
  userdefined2: string;
  userdefined3: string;
  userdefined4: string;
  userdefined5: string;
  userdefined6: string;
  userdefined7: string;
}
```

```

userdefined8: string;
userdefined9: string;
userdefined10: string;

lastchange: Moment;           // Letzte Änderung [Read-Only]
creator nr: string;           // Ersteller [Read-Only]
date created: Moment;         // Erstellt am [Read-Only]

// Read-Only-Bereich
clientname: string;           // Name des Kunden (Read-Only)
clientid: number;             // Kundennummer (Read-Only)
mainprojectname: string;     // Name des Hauptprojekts (Read-Only)
mainprojectnumber: string;    // Nummer des Hauptprojekts (Read-Only)
}

```

5.4.2 Auflisten

Liest eine Liste von Projektdatensätzen ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit
user_nr	Mitarbeiter

client_nr	Nur Projekte eines Kunden anzeigen
parent_project_nr	Nur Unterprojekte eines Hauptprojekts anzeigen
projectname	Suchbegriff
mainprojectonly	Nur Hauptprojekte BOOL
timeaccountonly	Nur mit Zeitkonto BOOL
teamprojectonly	Nur Teamprojekte BOOL
assignedtouser	Nur wenn zugewiesen zu user (user_nr) BOOL
favoritesonly	Nur Favoriten BOOL
projecttype_nr	Projekttyp GUID
projectstate	Projektstatus INTEGER
accountmode	Abrechnungsmodus INTEGER
priority	Priorität INTEGER

Beispiel:

```

endpoint = "projects"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
          'pagenumber': 1,
          'pagesize': 10
        }

```

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

5.4.3 Einlesen (GET)

Liest einen Projektdatensatz.

Aufrufart: REST, GET

Parameter:

project_nr	GUID des Projekts
------------	-------------------

Beispiel:

```
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 Erstellen (CREATE)

Erstellt einen neuen Projektdatensatz.

Aufrufart: REST, POST

Parameter:

client_nr	GUID des Kunden für den das Projekt angelegt wird
parent_project_nr	GUID des Hauptprojekts für welches das Projekt angelegt wird

Beispiel:

```
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 Ändern (UPDATE)

Erstellt einen neuen Projektdatensatz. Dabei wird das gesamte Datensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

project_nr	GUID des Projekts
------------	-------------------

Beispiel:

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

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

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

5.4.6 Löschen (DELETE)

Löscht einen Projektdatensatz mit allen untergeordneten Datenobjekten (z.B. Projekten, Zeitstempel oder Aufgaben)

Aufrufart: REST, DELETE

Parameter:

project_nr	GUID des Projekts
------------	-------------------

Beispiel:

```
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 (Aufgaben)

Über die Aufgaben können den Projekten und Benutzern spezifische Tätigkeiten zugeordnet werden, welche zu einem Stichtag erledigt werden sollten. Sie werden über die `todo_nr` referenziert.

5.5.1 Datenstruktur

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

  project_nr: string;       // GUID des Projekts [Pflichtfeld]
  user_nr: string;         // GUID des Bearbeiters (Wenn leer, dann Aufgabenpool)
  creator_nr: string;      // GUID des Erstellers [Pflichtfeld]
  todocategory_nr: string; // GUID der Aufgabenkategorie

  subject: string;        // Betreff [Pflichtfeld]
  body: string;          // Textkörper
  itemdate: Moment;
  donedate: Moment;      // Erledigt am
  readdate: Moment;      // Gelesen am

  tobetodonefrom: Moment; // Zu erledigen ab
  tobetodonetill: Moment; // Zu erledigen bis

  progress: number;       // Fortschritt in %
  priority: number;      // 5 Stufige Priorität (0=Low;...;5=Highest)

  tasknumber: number;     // Laufende Aufgabennummer [Read-Only]

  notecount: number;      // Anzahl Notizen [Read-Only]
  attachmentcount: number; // Anzahl Dokumente [Read-Only]

  lastchanged: Moment;    // Letzte Änderung [Read-Only]
  date created: Moment;   // Erstellt am [Read-Only]

  minutesneeded: number;  // Zeit benötigt in Minuten (Wird aus Zeitstempeln berechnet)
  minutesestimated: number; // Zeit geschätzt in Minuten
  taskstate: number;     // Aufgabenstatus: 0=tsNotStarted, 1=tsChecking,
                        // 2=tsInProgress, 3=tsPaused, 4=tsWaiting,
                        // 5=tsDeclined, 6=tsDone
                        // 7=tsTesting, 8=tsTestFailed, 9=TestOk
                        // 10=tsProblem, 11 = tsReady
                        // 100-104=tsUserdefined1-5

  taskstatecomment: string; // Kommentar zum Status

  textformat: number;     // Textart des Textkörpers (xttdtfRVF=0, xttdtfRTF=1,
                        // xttdtfText=2, xttdtfHtml=3);

  // Read-Only-Bereich
  client_nr: string;      // GUID der Kunden (Read-Only)
  parent_project_nr: string; // GUID des Hauptprojekts (Read-Only)
  clientname: string;     // Kundenname (Read-Only)
  clientid: number;       // Kundennummer (Read-Only)
  mainprojectname: string; // Hauptprojektname (Read-Only)
  mainprojectnumber: string; // Hauptprojektnummer (Read-Only)
  projectname: string;    // Projektname (Read-Only)
  projectnumber: number;  // Projektnummer (Read-Only)
  firstname: string;     // Vorname des Bearbeiters (Read-Only)
  lastname: string;      // Nachname des Bearbeiters (Read-Only)
  creatorfirstname: string; // Vorname des Erstellers (Read-Only)
}
```

```

creatorlastname: string; // Nachname des Erstellers (Read-Only)

projectstate: number; // Projektstatus (Read-Only)
hastimeaccount: boolean; // Hat das Projekt ein Zeitkonto? (Read-Only)
}

```

5.5.2 Auflisten

Liest eine Liste von Aufgabendatensätzen ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit
user_nr	Mitarbeiter

client_nr	Nur Aufgaben eines Kunden anzeigen
project_nr	Nur Aufgaben eines Projekts anzeigen
parent_project_nr	Nur Aufgaben eines Hauptprojekts anzeigen
comment	Suchbegriff
taskstate	Aufgabenstatus INTEGER
todocategory_nr	Aufgabenkategorie GUID
priority	Priorität INTEGER
taskpoolmode	0 = Alles, 1 = Ohne Pool, 2 = Nur Pool
favoritesonly	Nur von Projektfavoriten
daterange	Datumsfilter 0 = Alles, 1 = Heute, 2 = Gestern, 3 = Diese Woche, 4 = Letzte Woche, 5 = Dieser Monat, 6 = Letzter Monat 7 = Dieses Jahr, 8 = Letztes Jahr, 9 = Freie Eingabe 10 = Dieses Quartal, 11 = Letztes Quartal, 12 = Letzte 3 Tage 13 = Letzte 7 Tage, 14 = Letzte 14 Tage, 15 = Letzte 21 Tage 16 = Letzte 30 Tage
daterangefrom	Datumsbereich Von (Nur wenn daterange = 9) ISODATESTRING
daterangetill	Datumsbereich Bis (Nur wenn daterange = 9) ISODATESTRING

Beispiel:

```

endpoint = "tasks"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
          'pagenumber': 1,
          'pagesize': 10
        }

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

```

5.5.3 Einlesen (GET)

Liest einen Aufgabendatensatz.

Aufrufart: REST, GET

Parameter:

todo_nr	GUID der Aufgabes
---------	-------------------

Beispiel:

```

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 Erstellen (CREATE)

Erstellt einen neuen Aufgabendatensatz.

Aufrufart: REST, POST

Parameter:

project_nr	GUID des Projekts für welches die Aufgabe
------------	---

Beispiel:

```

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 Ändern (UPDATE)

Erstellt einen neuen Aufgabendatensatz. Dabei wird das gesamte Datensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

todo_nr	GUID der Aufgabes
---------	-------------------

Beispiel:

```

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 Löschen (DELETE)

Löscht einen Aufgabendatensatz mit allen untergeordneten Datenobjekten (z.B. Zeitstempel oder Notizen)

Aufrufart: REST, DELETE

Parameter:

todo_nr	GUID der Aufgabe
---------	------------------

Beispiel:

```
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 (Zeitstempel)

Ein Zeitstempel enthält einen einzelnen Erfassungsdatensatz, welcher immer einem Benutzer und einem Projekt zugeordnet ist.

5.6.1 Datenstruktur

```
export class Timestamp {
    times_nr: string;           // Eindeutige GUID [Read-Only]
    project_nr: string;         // GUID des Projekts [Pflichtfeld]
    todo_nr: string;           // GUID der Aufgabe (optional)
    user_nr: string;           // Bearbeiter des Zeitstempels [Pflichtfeld]
    itemdate: Moment;          // Datum des Zeitstempels (ohne Uhrzeit)
    fromtime: Moment;          // Datum+Zeit Anfang [Pflichtfeld]
    tilltime: Moment;          // Datum+zeit Ende [Pflichtfeld]
    pausedsince: Moment;       // Zeitpunkt seit dem die Pause läuft
    comment: string;           // Kommentar
    minutesneeded: number;     // Summe der Minuten (wird automatisch berechnet)
    minutespause: number;      // Pause in Minuten
    manualentry: boolean;      // Ist der Eintrag ein Nachtrag?
    state: number;             // Status: 0=xttssAccountNormal, 1=xttssDontAccount,
                                // 2=xttssIsAccounted
    creator_nr: string;        // GUID des Erstellers [Read-Only]

    fulltimestampmode: number; // Wurde der Zeitstempel gerundet?
    tilltimebeforerounding: Moment; // Nicht gerundetes Ende des Zeitstempels

    timezonename: string;      // Name der Zeitzone

    location: string;          // Ort
    location_lat: number;      // GPS-Koordinate Latitude
    location_lng: number;      // GPS-Koordinate Longitude

    pricegroup_nr: string;     // GUID der Preisgruppe
    activity_nr: string;        // GUID der Tätigkeit

    // Read-Only-Bereich
    clientname: string;        // Kundenname (Read-Only)
    clientid: number;          // Kundennummer (Read-Only)
    mainprojectname: string;   // Hauptprojektname (Read-Only)
    mainprojectnumber: string; // Hauptprojektnummer (Read-Only)
    projectname: string;       // Projektname (Read-Only)
    projectnumber: number;     // Projektnummer (Read-Only)

    taskname: string;          // Betreff der Aufgabe (Read-Only)
    tasknumber: number;        // Nummer der Aufgabe (Read-Only)
    isteamproject: boolean;     // Ist das Projekt ein Teamprojekt (Read-Only)
    projectstate: number;      // Projektstatus (Read-Only)

    firstname: string;         // Vorname des Bearbeiters
    lastname: string;          // Nachname des Bearbeiters
    persnr: string;            // Personalnummer des Bearbeiters
    client_nr: string;          // GUID des Kunden (Read-Only)
    parent project nr: string; // GUID des Hauptprojekts (Read-Only)

    activityname: string;       // Bezeichnung der Tätigkeit (Read-Only)
    activityshortcut: string;   // Kurzbezeichnung der Tätigkeit (Read-Only)
    activitycolor: number;      // Farbe der Tätigkeit (Read-Only)
    pricegroupname: string;     // Bezeichnung der Preisgruppe (Read-Only)
}
```

}

5.6.2 Auflisten

Liest eine Liste von Zeitstempeln ein.

Aufrufart: REST, GET

Parameter:

pageNumber	Nummer der angeforderten Seite
pageSize	Anzahl Datensätze pro Abfrage (Seitengröße)
sortfield	Sortierspalte
sortdirection	Sortierreihenfolge
visibility	Sichtbarkeit

client_nr	Nur Zeiten eines Kunden anzeigen
parent_project_nr	Nur Zeiten eines Hauptprojekts anzeigen
project_nr	Nur Zeiten eines Projekts anzeigen
todo_nr	Nur Zeiten einer Aufgabe anzeigen
user_nr	Nur Zeiten eines Users anzeigen
search	Suchbegriff für Kommentar
projectname	Suchbegriff für Projektname
projecttype_nr	Projekttyp GUID
projectstate	Projektstatus INTEGER
timestampstate	Zeitstempelstatus INTEGER
accountmode	Abrechnungsmodus INTEGER
hasLocation	Mit Ortsinformationen BOOL
daterange	Datumsfilter 0 = Alles, 1 = Heute, 2 = Gestern, 3 = Diese Woche, 4 = Letzte Woche, 5 = Dieser Monat, 6 = Letzter Monat 7 = Dieses Jahr, 8 = Letztes Jahr, 9 = Freie Eingabe 10 = Dieses Quartal, 11 = Letztes Quartal, 12 = Letzte 3 Tage 13 = Letzte 7 Tage, 14 = Letzte 14 Tage, 15 = Letzte 21 Tage 16 = Letzte 30 Tage
daterangefrom	Datumsbereich Von (Nur wenn daterange = 9) ISODATESTRING
daterangetill	Datumsbereich Bis (Nur wenn daterange = 9) ISODATESTRING

Beispiel:

```

endpoint = "times"
URL = apiurl + endpoint

PARAMS = {'sortdirection': 'asc',
          'pagenumber': 1,
          'pagesize': 10
        }

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

5.6.3 Einlesen (GET)

Liest einen Zeitstempeldatensatz.

Aufrufart: REST, GET

Parameter:

times_nr	GUID des Zeitstempels
----------	-----------------------

Beispiel:

```

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 Erstellen (CREATE)

Erstellt einen neuen Zeitstempel.

Aufrufart: REST, POST

Parameter:

project_nr	GUID des Projekts für welches der Zeitstempel erstellt werden soll
todo_nr	GUID des Aufgabe für welches der Zeitstempel erstellt werden soll
user_nr	GUID des Mitarbeiter für welchen der Zeitstempel erstellt werden soll

Beispiel:

```

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 Ändern (UPDATE)

Erstellt einen neuen Zeitstempel. Dabei wird das gesamte Datensatz mit den angegebenen Werte überschrieben. Ein auslassen von Feldern führt zu Datenverlust.

Aufrufart: REST, PUT

Parameter:

times_nr	GUID des Zeitstempels
----------	-----------------------

Beispiel:

```

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 Löschen (DELETE)

Löscht einen Zeitstempel.

Aufrufart: REST, DELETE

Parameter:

times_nr	GUID des Zeitstempels
----------	-----------------------

Beispiel:

```

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 (Serverkonfiguration)

Über die Systemdaten können diverse Konfigurationsparameter ausgelesen werden, die in der weiteren Verarbeitung notwendig bzw. informativ sein könnten. Der Systemdatensatz kann über das XTREST-API nicht verändert werden.

5.7.1 Datenstruktur

```
export class System {
  system nr: string;

  database_nr: string;           // GUID der Datenbank
  databasename: string;         // Name der Datenbank
  databasetype: string;         // Art/Hersteller der Datenbank

  patchleveldb: number;         // Patchlevel der Datenbank
  patchlevelserver: number;     // Patchlevel des XTCloudServers
  restapilevel: number;
  workgroupmode: number;        // Arbeitsgruppenmodus

  licence nr: string;           // GUID der Lizenz
  licenceholder: string;        // Name des Lizenznehmers

  currentusercount: number;     // Anzahl aktiver Benutzer in DB
  licenceusercountuniversal: number; // Anzahl Universaluser
  licenceusercountdesktoponly: number; // Anzahl Nur-Desktop-User
  licenceusercountwebonly: number; // Anzahl Nur-Web-user
  licenceusercountmobileonly: number; // Anzahl Nur-Mobile
  licenceusercountnologin: number; // Anzahl No-Login-User
  licencextcloudserverdateend: Date;
  licencextcloudserverdateendmessage: string;
  licencesubscriptiondateend: Date;
  licencesubscriptiondateendmessage: string;

  version: string;              // Derzeitige Version des Servers
  versiondate: string;          // Derzeitiges Versionsdatum des Servers

  versionavailable: string;     // Im Internet vorhandene Version des Servers
  versiondateavailable: string; // Versionsdatum des Servers
  updateavailable: boolean;     // Ist laut Server ein Update vorhanden?

  isdemodb: boolean;           // Ist die Datenbank eine Demodatenbank?
  dblanguageid: number;        // Sprache der Datenbank
  timezonebias: number;
  timezone: string;            // Name der Serverzeitzone
  currentservertime: Date;     // Aktuelle Serverzeit auf dem Webserver

  projectssetprojectnumbertomp: boolean; // Projektnummer eines neuen Projekts auf die
                                          // Nummer der Hauptprojekt setzen
  projectsdeleteaskforcode: boolean;     // Löschcode abfragen vor dem Löschen von
                                          // Kunde oder Projekt
  projectsautoincnumber: boolean;        // Projektnummer automatisch hochzählen
  manualentryrangeactive: boolean;       // Zeitstempel nachträglich hinzufügen aktiv?
  manualentryrange: number;              // Zeitstempel nachträglich hinzufügen Tage
  edittimesrangeactive: boolean;         // Zeitstempel nachträglich bearbeiten aktiv?
  edittimesrange: number;                // Zeitstempel nachträglich bearbeiten Tage

  vatpercentage: number;                // Standard Mwst-Satz
}
```

```
passwordminimumlength: number;           // Minimale Passwortlänge
passwordcomplexity: number;              // Sicherheitsanforderungen an Passwort

pricetable_id_nr: string;                // Basispreistabelle

moduleactiveclients: boolean;           // Modul Kundenverwaltung aktiv?
moduleactivetasks: boolean;             // Modul Aufgaben aktiv?
moduleactivehistory: boolean;           // Modul Historie aktiv?
moduleactivenotes: boolean;             // Modul Notizen aktiv?
moduleactivedms: boolean;               // Modul Dokumente aktiv?
moduleactiveinvoice: boolean;           // Modul Faktura aktiv?
moduleactiveprojectitems: boolean;      // Modul Leistungen aktiv?
moduleactivereminders: boolean;         // Modul Erinnerungen aktiv?
moduleactivepricegroups: boolean;       // Modul Preisgruppen aktiv?
moduleactiveworkgroups: boolean;        // Modul Arbeitsgruppen aktiv?
moduleactivebookingarchive: boolean;    // Modul Buchungsarchive aktiv?

haswebcam: boolean;
}
```

5.7.2 Einlesen (GET)

Liest einen Systemdatensatz.

Aufrufart: REST, GET

Parameter:

keine

Beispiel:

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

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

6 Suchfunktionen

6.1 GetUserByPersnr

Benutzer anhand seiner Personalnummer finden.

Aufrufart: JRPC, POST

Parameter:

persnr	Personalnummer des gesuchten Mitarbeiters INTEGER
--------	---

Rückgabe:

GUID des Benutzers (user_nr)

Beispiel:

```
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

Benutzer anhand seines Loginnamens finden.

Aufrufart: JRPC, POST

Parameter:

loginname	Loginname des gesuchten Mitarbeiters
-----------	--------------------------------------

Rückgabe:

GUID des Benutzers (user_nr)

Beispiel:

```
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

Kunde anhand seiner Kundennummer finden.

Aufrufart: JRPC, POST

Parameter:

clientid	Kundennummer des gesuchten Kunden INTEGER
----------	---

Rückgabe:

GUID des Kunden (client_nr)

Beispiel:

```

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

Kunde anhand seines Namens finden.

Aufrufart: JRPC, POST

Parameter:

clientname	Name des gesuchten Kunden
------------	---------------------------

Rückgabe:

GUID des Kunden (client_nr)

Beispiel:

```

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

Projekt anhand seiner Projektnummer finden.

Aufrufart: JRPC, POST

Parameter:

projectnumber	Alphanummerische Projektnummer des gesuchten Projekts STRING
---------------	--

Rückgabe:

GUID des Projekts (project_nr)

Beispiel:

```
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

Aufgabe anhand seiner Aufgabennummer finden.

Aufrufart: JRPC, POST

Parameter:

tasknumber	Aufgabennummer der gesuchten Aufgabe INTEGER
------------	--

Rückgabe:

GUID der Aufgabe (todo_nr)

Beispiel:

```
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

Kunde anhand eines benutzerdefinierten Feldes finden.

Aufrufart: JRPC, POST

Parameter:

fieldnumber	Nummer des benutzerdefinierten Feldes (1-10) INTEGER
search	Suchtext STRING

Rückgabe:

GUID des Kunden (`client_nr`)

Beispiel:

```
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

Projekt anhand eines benutzerdefinierten Feldes finden.

Aufrufart: JRPC, POST

Parameter:

fieldnumber	Nummer des benutzerdefinierten Feldes (1-10) INTEGER
search	Suchtext STRING

Rückgabe:

GUID des Projekts (`project_nr`)

Beispiel:

```
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

Aufgabe anhand eines benutzerdefinierten Feldes finden.

Aufrufart: JRPC, POST

Parameter:

fieldnumber	Nummer des benutzerdefinierten Feldes (1-5) INTEGER
search	Suchtext STRING

Rückgabe:

GUID der Aufgabe (todo_nr)

Beispiel:

```
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 Hilfsfunktionen

7.1 GetClientname

Den Namen eines Kunden über dessen client_nr abfragen.

Aufrufart: JRPC, POST

Parameter:

client_nr	GUID des Kunden
-----------	-----------------

Rückgabe:

Name des Kunden

7.2 GetProjectname

Den Namen eines Projekts über dessen project_nr abfragen.

Aufrufart: JRPC, POST

Parameter:

project_nr	GUID des Projekts
------------	-------------------

Rückgabe:

Name des Projekts

7.3 GetTaskname

Den Betreff einer Aufgabe über deren todo_nr abfragen.

Aufrufart: JRPC, POST

Parameter:

todo_nr	GUID der Aufgabe
---------	------------------

Rückgabe:

Name der Aufgabe

7.4 GetUsername

Den Namen eines Mitarbeiters über dessen user_nr abfragen.

Aufrufart: JRPC, POST

Parameter:

user_nr	GUID des Mitarbeiters
---------	-----------------------

Rückgabe:

Name des Benutzers

7.5 ProjectAssignUser

Mitarbeiter zu einem Projekt hinzufügen.

Aufrufart: JRPC, POST

Parameter:

project_nr	GUID des Projekts
user_nr	GUID des Mitarbeiters
assignsubprojects	Auch evtl. vorhandene Unterprojekte hinzufügen?

Rückgabe:

Statuswert 0 = Error, 1 = Ok

7.6 ProjectUnassignUser

Mitarbeiter aus einem Projekt entfernen.

Aufrufart: JRPC, POST

Parameter:

project_nr	GUID des Projekts
user_nr	GUID des Mitarbeiters

Rückgabe:

Statuswert 0 = Error, 1 = Ok

7.7 TaskSetUser

Bearbeiter einer Aufgabe setzen.

Aufrufart: JRPC, POST

Parameter:

todo_nr	GUID der Aufgabe
user_nr	GUID des Mitarbeiters

Rückgabe:

todo_nr und user_nr

7.8 TaskSetState

Aufgabenstatus einer Aufgabe setzen.

Aufrufart: JRPC, POST

Parameter:

todo_nr	GUID der Aufgabe
taskstate	INTEGER Aufgabenstatus (0 – 11, 100 – 104)

Rückgabe:

todo_nr und taskstate

7.9 ProjectSetState

Projektstatus eines Projekts setzen.

Aufrufart: JRPC, POST

Parameter:

project_nr	GUID des Projekts
projectstate	INTEGER Projektstatus (0 – 6, 100 – 109)

Rückgabe:

project_nr und projectstate

7.10 ClientGetMinutesNeeded

Benötigte Zeit eines Kunden in Minuten holen.

Aufrufart: JRPC, POST

Parameter:

client_nr	GUID des Kunden
-----------	-----------------

Rückgabe:

Minuten INTEGER

7.11 ProjectGetMinutesNeeded

Benötigte Zeit eines Projekts in Minuten holen.

Aufrufart: JRPC, POST

Parameter:

project_nr	GUID des Projekts
------------	-------------------

Rückgabe:
Minuten INTEGER

7.12 TaskGetMinutesNeeded

Benötigte Zeit einer Aufgabe in Minuten holen.

Aufrufart: JRPC, POST

Parameter:

todo_nr	GUID der Aufgabe
---------	------------------

Rückgabe:
Minuten INTEGER

7.13 ProjectStart

Startet ein Projekt.

Aufrufart: JRPC, POST

Parameter:

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

Rückgabe:
JSON des Zeitstempels

7.14 TaskStart

Startet eine Aufgabe.

Aufrufart: JRPC, POST

Parameter:

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

Rückgabe:
JSON des Zeitstempels

7.15 TimestampStop

Stoppt einen laufenden Zeitstempel für einen User.

Aufrufart: JRPC, POST

Parameter:

user_nr	GUID des Users
comment	Zeitstempelkommentar (optional)

Rückgabe:

JSON des Zeitstempels

8 Beispiele

Folgende, in Python erstellte, Codebeispiele sollen das Zusammenspiel der einzelnen XT-Funktionen erläutern. Die Beispiele sind als Konsolenanwendungen gestaltet um unnötigen Overhead zu vermeiden.

Weitere Beispiele zu anderen Programmierumgebungen finden Sie im Ordner „\Examples\“.

Falls sie die API-Anbindung in Python vornehmen, sollten Sie am Besten den API-Wrapper aus dem Ordner „\Examples\Python\xtrestapi_wrapper“ verwenden. Beispiele dafür liegen unter „\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))

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

if not response: exit
data = response.json()
# Pick first timestamp from list
minutesneeded = data[0]['minutesneeded']
print('=> minutesneeded: ', minutesneeded)

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

```

9 Dokumenthistorie

Hier werden alle an diesem Dokument vorgenommenen Änderungen chronologisch festgehalten.

Datum	Beschreibung
01.03.2021	Erste Version der Dokumentation