ScoutNet.de APIs Documentation

Release 0.2

Jan Brohl

09.06.2018

Inhalt

Es gib im Grunde genommen 5 APIs:

- den Webservice der laut Doku „*RPC* im *REST*-Stil mit *JSON* als Datenaustauschformat implementiert“ - im weiteren Text *SN-REST* genannt
- den anderen Webservice den man per *JSON-RPC* (1.0) über HTTP(S) POST ansprechen kann - im weiteren Text *SN-RPC* genannt
- den CalDAV-Webservice der Lese- und Schreib-Zugriff auf Kalender nach dem *CalDAV*-Protokoll (mit Abweichungen) über HTTP(S) ermöglicht - im weitern Text *SN-CalDAV* genannt
- das Pfadi-Archiv-Template-System
- das Kalender-Template-System von show.php

KAPITEL 1

SN-RPC

Falls Du hier etwas noch nicht findest kannst du in den Quälcode des offiziellen Clients auf https://github.com/scoutnet/plugins.sn_webservice schauen...

1.1 API

Alle Aufrufe gehen per JSON-RPC (1.0) per HTTP-POST an die selbe URL:

`POST https://www.scoutnet.de/jsonrpc/server.php`

Die Doku zu den Funktionen ist fast direkt kopiert aus dem Server-Code.

`get_data_by_global_id`(*global_id*[, *query*])

Return Data

Parameter

- **`global_id`** (*`array|string`*) – Global id for which we want elements
- **`query`** (*`object`*) – *Query* to filter the elements

Rückgabe Elemente als Attribute eines Objects - die Namen sind zusammengesetzt aus *Typen* und IDs

`checkPermission`(*type*, *global_id*, *username*, *auth*)

Checks if user has permission for object

Parameter

- **`type`** (*`string`*) – type of the record
- **`global_id`** (*`integer`*) – global Id of the object to check for
- **`username`** (*`string`*) – username of the User
- **`auth`** (*`string`*) – JSON-serialized array containing the a timestamp and two hashes(sha1 and md5) of the three other parameters which is encrypted with AES-CBC with the Users Api key and IV=1234567890123456

Rückgabe 0 if object is stored, 1 if user is not allow, 2 if an other error occured

requestPermission(*type*, *global_id*, *username*, *auth*)
Request permission for object

Parameter

- **type** (*string*) – Type of Record
- **global_id** (*integer*) – Global id of Object to request for
- **username** (*string*) – Username
- **auth** (*string*) – JSON-serialized array containing the a timestamp and two hashes(sha1 and md5) of the three other parameters which is encrypted with AES-CBC with the Users Api key and IV=1234567890123456

Rückgabe 0 if object is stored

setData(type,id,object,username,auth()
Writes Data to scoutnet

Parameter

- **type** (*string*) – this is the type of the record
- **id** (*integer*) – the unique id of the object (if this is -1 the object is created)
- **object** (*object*) – the object itself
- **username** (*string*) – the username of the user
- **auth** (*string*) – JSON-serialized array containing the a timestamp and two hashes(sha1 and md5) of the four other parameters which is encrypted with AES-CBC with the Users Api key and IV=1234567890123456

Rückgabe 0 if object is stored, 1 if user is not allow, 2 if an other error occured

deleteObject(*type*, *global_id*, *id*, *username*, *auth*)
Deletes object from scoutnet

Parameter

- **type** (*string*) – this is the type of the record
- **global_id** (*integer*) – the global id for the object
- **id** (*integer*) – the unique id of the object
- **username** (*string*) – the username of the user
- **auth** (*string*) – JSON-serialized array containing the a timestamp and two hashes(sha1 and md5) of the three other parameters which is encrypted with AES-CBC with the Users Api key and IV=1234567890123456

Rückgabe 0 if object is stored, 1 if user is not allow, 2 if an other error occured

Die Erzeugung des Auth-Strings ist etwas kompliziert aber sie sollte funktionieren wie in *sn_rpc.py*

1.2 Schemas

Hier sind *JSON-Schemas* zu *SN-RPC*

1.2.1 Query

Bisher ist dies weitgehend spekulativ.

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
event_ids	object	Beschreibung fehlt	
event_ids.afterDate	string	Beschreibung fehlt	
event_ids.before	string	Beschreibung fehlt	
event_ids.beforeDate	string	Beschreibung fehlt	
event_ids.limit	integer	Beschreibung fehlt	
event_ids.uid	string	Beschreibung fehlt	
events	object	Beschreibung fehlt	
events.afterDate	string	Beschreibung fehlt	
events.before	string	Beschreibung fehlt	
events.beforeDate	string	Beschreibung fehlt	
events.limit	integer	Beschreibung fehlt	
events.uid	string	Beschreibung fehlt	
index	object	Beschreibung fehlt	
index.deeps	integer	Beschreibung fehlt	
index.limit	integer	Beschreibung fehlt	
index.uid	string	Beschreibung fehlt	
kalenders	object	Beschreibung fehlt	
kalenders.limit	integer	Beschreibung fehlt	
kalenders.uid	string	Beschreibung fehlt	

1.2.2 Typen

Typen für die Attribute des Antwortobjekts von *`get_data_by_global_id()`*

Event

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
content	object	Beschreibung fehlt	
content.All_Day	boolean	Beschreibung fehlt	
content.Created_At	string	Beschreibung fehlt	
content.Created_By	string	Beschreibung fehlt	
content.Description	string	Beschreibung fehlt	
content.End	string	Beschreibung fehlt	
content.ID	integer	Beschreibung fehlt	
content.Kalender	string	Beschreibung fehlt	
content.Keywords	object	Beschreibung fehlt	
content.Last_Modified_At	string	Beschreibung fehlt	
content.Last_Modified_By	string	Beschreibung fehlt	
content.Location	string	Beschreibung fehlt	
content.Organizer	string	Beschreibung fehlt	
content.SSID	string	Beschreibung fehlt	
content.Start	string	Beschreibung fehlt	
content.Stufen[]	array	Beschreibung fehlt	
content.Target_Group	string	Beschreibung fehlt	
content.Title	string	Beschreibung fehlt	
content.UID	integer	Beschreibung fehlt	
content.URL	string	Beschreibung fehlt	
content.URL_Text	string	Beschreibung fehlt	
content.ZIP	string	Beschreibung fehlt	
type	string	Beschreibung fehlt	• It must be equal to one of the elements in [„event“]

Kalender

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
content	object	Beschreibung fehlt	
content.Ebene	string	Beschreibung fehlt	
content.Ebene_Id	integer	Beschreibung fehlt	
content.Forced_Kategories	object	Beschreibung fehlt	
content.ID	string	Beschreibung fehlt	
content.Ident	string	Beschreibung fehlt	
content.Name	string	Beschreibung fehlt	
content.Used_Kategories	object	Beschreibung fehlt	
content.Verband	string	Beschreibung fehlt	
type	string	Beschreibung fehlt	• It must be equal to one of the elements in [„kalender“]

Index

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
content	object	Beschreibung fehlt	
content.ebene	string	Beschreibung fehlt	
content.id	string	Beschreibung fehlt	
content.latitude	string	Beschreibung fehlt	
content.longitude	string	Beschreibung fehlt	
content.name	string	Beschreibung fehlt	
content.number	string	Beschreibung fehlt	
content.ort	string	Beschreibung fehlt	
content.parent_id	string	Beschreibung fehlt	
content.plz	string	Beschreibung fehlt	
content.url	string	Beschreibung fehlt	
type	string	Beschreibung fehlt	• It must be equal to one of the elements in [„index“]

1.3 Beispiele

Teilweise getestet mit *Python* 3.5, benötigt *Bibliothek requests* und *Bibliothek pycrypto*

1.3.1 Beispielbenutzung

Aufrufe der Funktionen von *sn_rpc.py*

Events für Gruppe 4 abfragen:

```
get_data_by_global_id("4",{"events":{}})
```

Events für Gruppe 4 und 3 abfragen:

```
get_data_by_global_id(["4","3"],{"events":{}})
```

Events für Gruppe 4 abfragen die vor dem 12.01.2012 liegen:

```
get_data_by_global_id("4",{"events":{"before":"12.01.2012"}})
```

Info über Gruppe 4 abfragen:

```
get_data_by_global_id("4",{"index":{}})
```

Übergeordnete Gruppe zu Gruppe 4 suchen:

```
get_data_by_global_id("4",{"index":{}})
```

1.3.2 Beispielclient

Quellcode 1: sn_rpc.py

```
import os
import json
import hashlib
import time
import base64
import requests
import Crypto.Cipher.AES


SN_RPC_URL = "https://www.scoutnet.de/jsonrpc/server.php"

SN_RPC_IV = b'1234567890123456'


class RPCError(Exception):
    pass


def rpc(url, method, *params):
    """
    JSON-RPC 1.0 über HTTP(S) POST
    """
    call_id = os.urandom(16).hex()
    json_data = {
        "method": method,
        "params": params,
        "id": call_id
    }
    got = requests.post(
        url,
        json=json_data
    ).json()
    e = got["error"]
    if e is not None:
        raise RPCError(e)
    return got["result"]


def get_data_by_global_id(global_id, query={}):
    return rpc(
        SN_RPC_URL,
        "get_data_by_global_id",
        global_id,
        query
    )


def checkPermission(type, global_id, username, auth):
    return rpc(
        SN_RPC_URL,
        "checkPermission",
        type,
        global_id,
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        username,
        auth
    )


def requestPermission(type, global_id, username, auth):
    return rpc(
        SN_RPC_URL,
        "requestPermission",
        type,
        global_id,
        username,
        auth
    )


def setData(type, id, object, username, auth):
    return rpc(
        SN_RPC_URL,
        "requestPermission",
        type,
        id,
        object,
        username,
        auth
    )


def deleteObject(type, global_id, id, username, auth):
    return rpc(
        SN_RPC_URL,
        "requestPermission",
        type,
        global_id,
        id,
        username,
        auth
    )


def generate_auth(api_key, check_value):
    auth = {
        "sha1": hashlib.sha1(check_value).hexdigest(),
        "md5": hashlib.md5(check_value).hexdigest(),
        "time": time.time()
    }
    auth = json.dumps(auth).encode("ascii")
    first_block = os.urandom(16)
    aes = Crypto.Cipher.AES.new(api_key, Crypto.Cipher.AES.MODE_CBC, SN_RPC_IV)
    blocks = first_block + auth
    blocks = blocks + bytes(16 - len(blocks) % 16)  # padding
    out = aes.encrypt(blocks)
    out = base64.b64encode(out)
    for a, b in ((b'+', b'-'), (b'/', b'_'), (b'=', b'~')):
        out = out.replace(a, b)
    return out

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
111
112 def generate_check_value(*args):
113     return ("".join(str(p) for p in args)).encode("ascii")
```

KAPITEL 2

SN-REST

Was Du hier etwas noch nicht findest, steht vielleicht in der offiziellen Doku auf https://www.scoutnet.de/api-info/webservice.html

2.1 API

2.1.1 Gruppen

GET https://www.scoutnet.de/api/0.2/group/
Eine Gruppe (*Group*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET https://www.scoutnet.de/api/0.2/groups/
Eine Sammlung (*Collection*) von Gruppen (*Group*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET https://www.scoutnet.de/api/0.2/group/(*group_id*)**/**
Eine Gruppe (*Group*) abrufen.

Parameters

- **group_id** – ID der Gruppe

GET https://www.scoutnet.de/api/0.2/group/(*group_id*)**/events/**
Sammlung (*Collection*) von Events (*Event*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/children/**
Sammlung (*Collection*) von untergeordneten Gruppen (*Group*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/parent/**
Direkt übergeordnete Gruppe (*Group*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/parent/***layer***/** Übergeordnete Gruppe (*Group*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe
- **layer** – Ebene der übergeordneten Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/parents/**
Sammlung (*Collection*) übergeordneter Gruppen (*Group*) zu einer Gruppe abrufen.

Zu tun: was bedeutet der options-parameter?

Parameters

- **group_id** – ID der Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/urls/**
Sammlung (*Collection*) von URLs (*URL*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe

GET **https://www.scoutnet.de/api/0.2/group/**(*group_id*)**/sections/**
Sammlung (*Collection*) von Stufen (*Section*) zu einer Gruppe abrufen.

Parameters

- **group_id** – ID der Gruppe

2.1.2 Events

GET **https://www.scoutnet.de/api/0.2/event/**
Event (*Event*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/events/**
Eine Sammlung (*Collection*) von Events (*Event*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/event/**(*event_id*)**/**
Event (*Event*) abrufen.

Parameters

- **event_id** – ID des Events

GET **https://www.scoutnet.de/api/0.2/event/**(*event_id*)**/group/**
Gruppe (*Group*) zu Event abrufen.

Parameters

- **event_id** – ID des Events

2.1.3 URLs

GET **https://www.scoutnet.de/api/0.2/url/**
URL (*URL*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/urls/**
Eine Sammlung (*Collection*) von URLs (*URL*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/url/**(*url_id*)**/**
URL (*URL*) abrufen.

Parameters

- **url_id** – ID der URL

2.1.4 Stufen

GET **https://www.scoutnet.de/api/0.2/section/**
Stufe (*Section*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/sections/**
Eine Sammlung (*Collection*) von Stufen (*Section*) abrufen.

Query Parameters

- **json** (*JSON-Array*) – *PfadiQL*-Anfrage

GET **https://www.scoutnet.de/api/0.2/section/**(*section_id*)**/**
Stufe (*Section*) abrufen.

Parameters

- **section_id** – ID der Stufe

2.2 Schemas

Hier sind *JSON-Schemas* zu *SN-REST*

Noch ist das nur alles automatisch erzeugt und gibt nur eine Übersicht welche felder es überhaupt gibt.

2.2.1 Event

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
description	string	Beschreibung fehlt	
end_date	string	Beschreibung fehlt	
end_time	[string, null]	Beschreibung fehlt	
group_id	string	Beschreibung fehlt	
id	string	Beschreibung fehlt	
keywords	object	Beschreibung fehlt	
keywords.elements	object	Beschreibung fehlt	
keywords.kind	string	Beschreibung fehlt	
kind	string	Beschreibung fehlt	• It must be equal to one of the elements in [„event“]
last_modified_by	string	Beschreibung fehlt	
last_modified_on	string	Beschreibung fehlt	• It must be formatted as date-time
location	string	Beschreibung fehlt	
organizer	string	Beschreibung fehlt	
start_date	string	Beschreibung fehlt	
start_time	[string, null]	Beschreibung fehlt	
target_group	string	Beschreibung fehlt	
title	string	Beschreibung fehlt	
uid	string	Beschreibung fehlt	
url	string	Beschreibung fehlt	• It must be formatted as uri
url_text	string	Beschreibung fehlt	
zip	string	Beschreibung fehlt	

2.2.2 URL

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
group_id	string	Beschreibung fehlt	
id	string	Beschreibung fehlt	
kind	string	Beschreibung fehlt	• It must be equal to one of the elements in [„url“]
text	string	Beschreibung fehlt	
url	string	Beschreibung fehlt	• It must be formatted as uri

2.2.3 Section

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
color	string	Beschreibung fehlt	
end_age	string	Beschreibung fehlt	
id	string	Beschreibung fehlt	
kind	string	Beschreibung fehlt	• It must be equal to one of the elements in [„section“]
name	string	Beschreibung fehlt	
start_age	string	Beschreibung fehlt	

2.2.4 Group

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
city	string	Beschreibung fehlt	
country	string	Beschreibung fehlt	
district	string	Beschreibung fehlt	
federal_state	string	Beschreibung fehlt	
global_id	string	Beschreibung fehlt	
internal_id	string	Beschreibung fehlt	
kind	string	Beschreibung fehlt	• It must be equal to one of the elements in [„group“]
layer	string	Beschreibung fehlt	
name	string	Beschreibung fehlt	
zip	string	Beschreibung fehlt	

2.2.5 Collection

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
element_kind	string	Beschreibung fehlt	• It must be equal to one of the elements in [„event“, „group“, „section“, „url“]
elements[]	array	Beschreibung fehlt	
kind	string	Beschreibung fehlt	

2.3 PfadiQL

Hier gibt es noch nichts zu sehen aber es gibt schon eine Doku auf https://www.scoutnet.de/api-info/pfadiql.html

2.4 Beispiele

Getestet mit *Python* 3.5, benötigt *Bibliothek requests*

2.4.1 Beispielbenutzung

Aufrufe der Funktionen von *sn_rest.py*

Events für Gruppe 4 abfragen:

```
sn_rest("group/4/events/")
```

Events für Gruppe 4 und 3 abfragen:

```
sn_rest("/events/","group_id=? or group_id=?",["4","3"])
```

Events für Gruppe 4 abfragen die vor dem 12.01.2012 liegen:

```
sn_rest("group/4/events/", "end_date < ?",["2012-01-12"])
```

Info über Gruppe 4 abfragen:

```
sn_rest("group/4/")
```

Übergeordnete Gruppe zu Gruppe 4 suchen:

```
sn_rest("group/4/parent/")
```

2.4.2 Beispielclient

Quellcode 1: sn_rest.py

```
import json
import requests


SN_REST_BASE_URL = "https://www.scoutnet.de/api/0.2/"


def sn_rest(path, *args):
    if args:
        params = {
            "json": json.dumps(args)
        }
    else:
        params = {}
    return requests.get(
        SN_REST_BASE_URL + path,
        params=params
    ).json()
```

KAPITEL 3

SN-CalDAV

Eigentlich gibt es hier nicht viel zu sagen weil die sich Scoutnet.de hier weitgehend an die Standards hält.

3.1 API

Die API entspricht bis auf eine Ausnahme der den offiziellen Standards die auf mehrere Dokumente verteilt sind - die Wikipedia-Seite ist hier ein guter Start.

Es gibt einen wichtigen Unterschied zu CalDAV: ScoutNet.de benutzt das Feld Description für ein *JSON*-Objekt in dem zusätzlich zur Beschreibung andere Daten gespeichert werden mit denen manche CalDAV-Clients nichts anfangen können - genaueres in den zugehörigen *Schemas*.

3.2 Schemas

Hier ist das *JSON-Schemas* für das Description-Feld *SN-CalDAV*

Original: `JSON-Schema`

Lesbarer:

Name	Type	Description	Validations
Info	string	Beschreibung fehlt	
Kategorie[]	array	Beschreibung fehlt	
Link_Text	string	Beschreibung fehlt	
Veranstalter	string	Beschreibung fehlt	
Zielgruppe	string	Beschreibung fehlt	

3.3 Beispiele

Ausprobiert mit *Python* 3.5, benötigt *Bibliothek iCalendar* und *Bibliothek caldav*

Funktioniert aber leider nur in sehr einfachen Fällen - Sonderzeichen im *JSON*-String im Description-Feld machen Probleme. Über Korrekturvorschläge würde ich mich freuen.

Demo für *sn_caldav.py*

```
import getpass
user = input("User: ")
pw = getpass.getpass()
group_id = input("Group-ID: ")

print()
print("Searching calendar...")
c = get_cal(user=user, pw=pw, group_id=group_id)
print("found", c)
print()
print("New Event:")
e = LocalEvent(BERLIN.localize(datetime.datetime(2100, 1, 1)),
               BERLIN.localize(datetime.datetime(2100, 1, 2)),
               "Testevent")
print(e)
print()
print("Adding Event...")
print("saved as", e.add_to_cal(c))
print()
input("Press enter to delete the event")
e.delete_caldav()
print("deleted")
```

3.3.1 Beispielclient

Quellcode 1: sn_caldav.py

```
import icalendar
import caldav
import json
import datetime
import pytz

__version__ = "0.1"


DESCRIPTION_MAP = {"info": "Info", "kategorien": "Kategorie", "veranstalter":
↪"Veranstalter",
                   "zielgruppe": "Zielgruppe", "link_text": "Link_Text"}


BERLIN = pytz.timezone("Europe/Berlin")


def decode_description(s):
    return {k.lower(): v for k, v in json.loads(s.replace('\\"', '"')).items()}

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

def encode_description(obj):
    return json.dumps({v: obj[k] for k, v in DESCRIPTION_MAP.items()}).replace('"', '\\"')


def get_cal(user, pw, group_id):
    url = "https://{user}:{pw}@webcal.scoutnet.de/webcal/{group_id}/".format(
        user=user, pw=pw, group_id=group_id)
    client = caldav.DAVClient(url)
    principal = client.principal()
    calendars = principal.calendars()
    return calendars[0]


class LocalEvent:
    EVENT_MAP = {"summary": "summary", "dtstart": "start", "dtend": "end",
                 "url": "url", "uid": "uid", "created": "created", "last-modified": "modified"}

    def __init__(self, start, end, summary, url="", info="", kategorien=None, veranstalter="", zielgruppe="", link_text="", uid=None, created=None, modified=None, caldav_event=None):
        self.start = start
        self.end = end
        self.summary = summary
        self.url = url
        self.info = info
        self.kategorien = [] if kategorien is None else kategorien
        self.veranstalter = veranstalter
        self.zielgruppe = zielgruppe
        self.link_text = link_text
        self.uid = uid
        self.created = created
        self.modified = modified
        self.caldav_event = caldav_event

    def to_ical(self):
        now = datetime.datetime.now(pytz.UTC)
        cal = icalendar.Calendar()
        cal.add('prodid', '-//Brohlsoft//ScoutNet.de Python-CalDAV//')
        cal.add('version', __version__)
        event = icalendar.Event()
        event.add("url", self.url)
        event.add('summary', self.summary)
        event.add('dtstart', self.start)
        event.add('dtend', self.end)
        event.add('created', self.created or now)
        event.add('dtstamp', self.modified or now)
        event.add('last-modified', self.modified or now)
        event.add('description', encode_description(dict(info=self.info, kategorien=self.kategorien,
                                                          veranstalter=self.veranstalter,
                                                          zielgruppe=self.zielgruppe,
                                                          link_text=self.link_text)))
        event.add('uid', self.uid or caldav.uuid.uuid4())
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        cal.add_component(event)
        return cal.to_ical()

    @classmethod
    def from_ical(cls, s):
        cal = icalendar.Calendar.from_ical(s)
        event = cal.subcomponents[0]
        d = decode_description(event["description"])
        for k, v in cls.EVENT_MAP.items():
            d[v] = event[k]
        return cls(**d)

    @classmethod
    def load_caldav(cls, caldav_event):
        obj = cls.from_ical(caldav_event.data)
        obj.caldav_event = caldav_event
        return obj

    def save_caldav(self):
        s = self.to_ical()
        self.caldav_event.data = s
        self.caldav_event.save()

    def delete_caldav(self):
        self.caldav_event.delete()

    def add_to_cal(self, cal):
        self.caldav_event = cal.add_event(self.to_ical())
        return self.caldav_event

    def __repr__(self):
        return ("LocalEvent(start={start!r}, end={end!r}, summary={summary!r}, url=
↪{url!r}, info={info!r}, "
                "kategorien={kategorien!r}, veranstalter={veranstalter!r}, zielgruppe=
↪{zielgruppe!r}, "
                "link_text={link_text!r}, uid={uid!r}, created={created!r}, modified=
↪{modified!r}, "
                "caldav_event={caldav_event!r})").format(start=self.start, end=self.
↪end, summary=self.summary,
                                                         url=self.url, info=self.info,
↪ kategorien=self.kategorien,
                                                         veranstalter=self.
↪veranstalter, zielgruppe=self.zielgruppe,
                                                         link_text=self.link_text,␣
↪uid=self.uid,
                                                         created=self.created,␣
↪modified=self.modified,
                                                         caldav_event=self.caldav_
↪event)


```

KAPITEL 4

Pfadi-Archiv-Template-System

Allgemeine Info zum Pfadi-Archiv gibt es auf https://archiv.scoutnet.de/

Gute Info zur Benutzung des zugehörigen Template-Systems liegt auf https://archiv.scoutnet.de/howto.php

KAPITEL 5

Show.php Template-Export

Über ein PHP-Script können Kalenderdaten mit Smarty-Templates exportiert werden.

Die Dokumentation dazu ist auf https://github.com/scoutnet/public/wiki/show.php zu finden.

KAPITEL 6

Glossar

interpreter Ein Interpreter ist ein Programm das Programme ausführt.

python Python ist eine Programmiersprache, meint manchmal auch den Standard-Interpreter von https://www.python.org/ .

pip Werkzeug zum installieren von Python-Bibliotheken.

Ist bei der Standardinstallation von Python inzwischen dabei aber kann auch direkt von https://pypi.python.org/pypi/pip geladen werden.

rpc RPC steht für Remote Procedure Call.

https://en.wikipedia.org/wiki/Remote_procedure_call

rest REST steht für Representational State Transfer.

https://en.wikipedia.org/wiki/Representational_state_transfer

json JSON steht für JavaScript Object Notation.

https://en.wikipedia.org/wiki/JSON

json-rpc JSON-RPC steht für *RPC* mit *JSON*.

https://en.wikipedia.org/wiki/JSON-RPC

icalendar iCalendar ist ein Dateiformat zum Austausch non Kalenderdaten.

https://de.wikipedia.org/wiki/ICalendar

caldav CalDAV (Calender Distributed Authoring and Versioning) ist ein Protokoll zum Bearbeiten und Abgleichen von Kalendern.

https://de.wikipedia.org/wiki/CalDAV

json-schema Ein JSON-Schema beschreibt wie ein JSON-Dokument aufgebaut sein darf um dem Schema zu entsprechen.

Im Grunde genommen ist es eine Art formal überprüfpbare Dokumentation siehe dazu http://json-schema.org/

sn-rest SN-REST bezeichnet hier den Webservice der laut Doku „*RPC* im *REST*-Stil mit *JSON* als Datenaustauschformat implementiert“.

sn-rpc SN-RPC bezeichnet hier den Webservice den man per *JSON-RPC* (1.0) über HTTP(S) POST ansprechen kann.

sn-caldav SN-CalDAV bezeihnet hier den Webservice den man per modifiziertem *CalDAV* (+ *JSON*) ansprechen kann.

bibliothek requests Requests ist eine Python-Bibliothek für einfache Benutzung von HTTP(S).

Kann man mit *pip* laden oder direkt von https://pypi.python.org/pypi/requests

bibliothek icalendar iCalendar ist eine Python-Bibliothek zum Arbeiten mit *iCalendar*-Datenstrukturen.

Kann man mit *pip* laden oder direkt von https://pypi.python.org/pypi/icalendar

bibliothek caldav Caldav ist eine Python-Bibliothek zum Bearbeiten von *CalDAV*-Kalendern.

Kann man mit *pip* laden oder direkt von https://pypi.python.org/pypi/caldav

bibliothek pycrypto Pycrypto ist eine Python-Bibliothek mit kryptografischen Funktionen.

Kann man mit *pip* laden oder direkt von https://pypi.python.org/pypi/pycrypto

HTTP Routing Table

/https:

GET https://www.scoutnet.de/api/0.2/event/, 12
GET https://www.scoutnet.de/api/0.2/event/(event_id)/, 12
GET https://www.scoutnet.de/api/0.2/event/(event_id)/group/, 13
GET https://www.scoutnet.de/api/0.2/events/, 12
GET https://www.scoutnet.de/api/0.2/group/, 11
GET https://www.scoutnet.de/api/0.2/group/(group_id)/, 11
GET https://www.scoutnet.de/api/0.2/group/(group_id)/children/, 11
GET https://www.scoutnet.de/api/0.2/group/(group_id)/events/, 11
GET https://www.scoutnet.de/api/0.2/group/(group_id)/parent/, 12
GET https://www.scoutnet.de/api/0.2/group/(group_id)/parent/(layer)/, 12
GET https://www.scoutnet.de/api/0.2/group/(group_id)/parents/, 12
GET https://www.scoutnet.de/api/0.2/group/(group_id)/sections/, 12
GET https://www.scoutnet.de/api/0.2/group/(group_id)/urls/, 12
GET https://www.scoutnet.de/api/0.2/groups/, 11
GET https://www.scoutnet.de/api/0.2/section/, 13
GET https://www.scoutnet.de/api/0.2/section/(section_id)/, 13
GET https://www.scoutnet.de/api/0.2/sections/, 13
GET https://www.scoutnet.de/api/0.2/url/, 13
GET https://www.scoutnet.de/api/0.2/url/(url_id)/, 13
GET https://www.scoutnet.de/api/0.2/urls/, 13
POST https://www.scoutnet.de/jsonrpc/server.php, 3

Stichwortverzeichnis