esoc

European Space Operations Centre
Robert-Bosch-Strasse 5
D-64293 Darmstadt
Germany
T +49 (0)6151 900
F +49 (0)6151 90495
www.esa.int

TECHNICAL NOTE

Programmatic access to SWE data within the SSA SWE network using HAPI

Prepared by	**S2P Data Systems and Space Weather Teams (author: Kamill Panitzek)**
Reference	**SSA-SWE-HAPI-TN-0001**
Issue/Revision	**1.0**
Date of Issue	**15/03/2021**
Status	**Approved**

European Space Agency
Agence spatiale européenne

APPROVAL

Title Programmatic access to SWE data within the SSA SWE network using HAPI	
Issue Number 1	**Revision Number** 0
Author S2P Data Systems and Space Weather Teams	**Date** 15/03/2021
Approved By	**Date of Approval**
(ESA Data Systems Team) J. Klug	Johannes Klug Digitally signed by Johannes Klug Date: 2021.03.23 08:57:42 +01'00'
(ESA SWE Service Coordinator) A. Glover	Alexi Glover Digitally signed by Alexi Glover Date: 2021.03.23 10:22:43 +01'00'

CHANGE LOG

Reason for change	Issue Nr.	Revision Number	Date
Initial version	1	0	15/03/2021

CHANGE RECORD

Issue Number 1	Revision Number 0		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
Initial version	15/03/2021	all	all

DISTRIBUTION

Name/Organisational Unit

Table of contents:

1 INTRODUCTION

1.1 Purpose and Scope

ESA's SSA Space Weather (SWE) Web Portal [RD-SWE] provides access to Space Weather (SWE) data programmatically through an Application Programming Interface (API). This API is called Heliophysics Application Programming Interface (HAPI) [RD-HAPI] and access is granted to registered users only and hence requires a user account at the Space Weather Portal. This document will guide you through the process of establishing a connection and a session with the SWE HAPI to then access information and data from it.

1.2 Reference Documents

Ref.	Document Title	Reference
[RD-SWE]	SSA Space Weather Portal	https://swe.ssa.esa.int/
[RD-HAPI]	HAPI (main page)	http://hapi-server.org/
[RD-HAPI-SPEC]	HAPI Data Access Specification version 2.1.0	https://github.com/hapi-server/data-specification/blob/master/hapi-2.1.0/HAPI-data-access-spec-2.1.0.md
[RD-HAPI-PDF]	HAPI Data Access Specification version 2.1.0 in PDF format	https://github.com/hapi-server/data-specification/blob/master/hapi-2.1.0/HAPI-data-access-spec-2.1.0.pdf
[RD-CURL]	Command line tool and library for transferring data with URLs	https://curl.se/
[RD-JQ]	Lightweight and flexible command-line JSON processor	https://stedolan.github.io/jq/
[RD-PYTHON]	Python (main page)	https://www.python.org
[RD-PYTHON-REQUESTS]	Requests: HTTP for Humans™ - an elegant and simple HTTP library for Python, built for human beings.	https://requests.readthedocs.io

1.3 Acronyms

API	Application Programming Interface
cURL	Client URL
DS	Data Systems
DST	Destination
EDRS-C	European Data Relay System satellite C
ESA	European Space Agency
GEO	Geostationary Earth Orbit
GK2A	Geo-Kompsat-2A satellite
GP	Ground Processor

HAPI	Heliophysics Application Programming Interface
JSON	JavaScript Object Notation
NGRM	Next Generation Radiation Monitor
PDF	Portable Document Format
REST	Representational State Transfer
S2P	(ESA's) Space Safety Programme
SOSMAG	Service Oriented Spacecraft Magnetometer
SSA	Space Situational Awareness Programme
SSO	Single Sign-On
SWE	Space Weather Element / Space Weather Segment
URL	Uniform Resource Locator
UTC	Universal Time Coordinated

2 HAPI

The SWE HAPI is based on the HAPI specification [RD-HAPI] and implements the HAPI Data Access Specification in its version 2.1.0. The definition of this specification can be found on Github [RD-HAPI-SPEC]. A PDF version is also available [RD-HAPI-PDF].

2.1 General Overview

The SWE HAPI is served as a web-based REST API. The API is accessible through URLs of the respective API endpoints, as defined in the HAPI Data Specification [RD-HAPI-SPEC]. These API endpoints can be accessed via a browser and also programmatically by using tools such as cURL [RD-CURL] or using Python 3 requests [RD-PYTHON]. The description of the HAPI endpoints is outside of the scope of this document, please refer to the HAPI Data Specification [RD-HAPI-SPEC] for further details. Furthermore, please refer to Section 3 for limitations of the implementation of the SWE HAPI.
All endpoints require the user to be authenticated and authorised before access is provided. If the user does not have access, he/she will be forwarded to an OpenAM login page for authentication. After successful authentication, the user is forwarded to the respective endpoint. When accessing one of the HAPI endpoints for the first time through the browser, the following basic steps are automatically executed by the browser (manual steps in brackets):

1. [User tries to access HAPI endpoint at https://swe.ssa.esa.int/hapi/<endpoint>]
2. HAPI creates a session for the user and redirects him/her to OpenAM for authentication
3. The browser follows the redirect to OpenAM
4. [The user logs in with her user credentials]
5. OpenAM redirects the user after successful authentication back to HAPI
6. HAPI grants access to the user on the specified endpoint
7. All future requests to HAPI endpoints are granted as long as the session is valid

For a command-line or machine-to-machine access to endpoints, the S2P DS Team has created a Bash script as well as a Python script to obtain the necessary authentication and authorization tokens imitating the browser behaviour. In future releases more methods will be provided to obtain authorisation to access the SWE HAPI.

2.2 Programmatic Access using Bash and cURL

In order to access the API programmatically using cURL, the following steps need to be performed:

1. Establish a session with HAPI by executing the `get_hapi_session_cookies.sh` script
2. Use the obtained session cookies to access a certain resource / HAPI endpoint

These steps as well as prerequisites are described in more detail below.

2.2.1 Prerequisites

Before a session with the SWE HAPI can be established using the mentioned Bash script there are three prerequisites that must be fulfilled:

1. Valid user credentials with the SWE Portal
   (can be obtained through registration at [RD-SWE])
2. Availability of cURL on the executing system
   (can be obtained from [RD-CURL] or through the system's package manager)
3. Availability of jq on the executing system
   (can be obtained from [RD-JQ] or through the system's package manager)

As the SWE HAPI responds with json responses, the jq program is quite useful to parse the output of any subsequent requests made to HAPI.

2.2.2 Get HAPI Session Cookies

The S2P DS Team has developed a script to authenticate a user with OpenAM and establish a session with HAPI, ultimately authorising the user to access data and information through the HAPI interface. The script is written as a UNIX/LINUX-based script using the Bash shell and can be downloaded from the SWE Portal at:
https://swe.ssa.esa.int/documents/20182/165822/get_hapi_session_cookies.sh.

To use the script, a help page was implemented. The content is provided when executing the script with the `-h` flag. It is also provided below for convenience.

```
Usage: ./get_hapi_session_cookies.sh [OPTION]

Prerequisites:
This script requires cURL and jq to be installed on the system.

This script will establish a connection with the SWE Portal using provided user
credentials to access HAPI. For this, first the user will be authenticated with
the portal and a session will be established. Then session tokens are obtained
to access the HAPI.

The output of this script will be the access tokens for which a session has
been established. The access tokens can be passed on as cookies for subsequent
requests to the HAPI.

Available options:
  -h              Display this help page and exit.
  -v              Enable verbose mode.
  -s              Silent mode. If successful, the script will only output one
                  line with the session cookies. This mode is best suitable for
                  use in programmatic access and with options -u and -p.
  -u <USERNAME>   Set the username to access HAPI. If not provided, the user
                  will be prompted to enter a username.
  -p <PASSWORD>   Set the user password to access HAPI. If not provided, the
                  will be prompted to enter a password.
  -c <PORTAL-URL> Set a custom SWE Portal URL where HAPI is hosted. This will
                  override the default value: https://swe.ssa.esa.int/
  -t              If provided, the script will access the HAPI capabilities
                  endpoint at <PORTAL-URL>/hapi/capabilities with cURL verbose
                  mode to test the obtained session tokens one more time.
```

When executed with valid credentials, the script will provide the required session cookies. Below an example output is provided when used without any arguments:

```
$ ./get_hapi_session_cookies.sh
Please enter username: portal_user
Please enter password:

Session with HAPI successfully established!
Authentication cookie that was obtained for establishing HAPI session:
esa-ssa-sso-cookie=AQIC5wM2LY4SfcwMbTOMfUo_pdw_UOixzOgeXBRiynawzvs.*AAJTSQACMDEAAlNL
ABQtMTk4NDIxMTQwMzcwOTMyNzg0Nw..*

Session cookies to be used for subsequent HAPI calls:
XSRF-TOKEN=a038b9da-dfcd-4407-918b-c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D
6089A26
```

In the last line, the session cookies called `XSRF-TOKEN` and `JSESSIONID` can be seen. All subsequent HAPI calls can be executed providing these two cookies. Please refer to Section 2.2.3 for examples.

As described in the above help text, the script can be called from within another script to assign the session cookies to a variable using the `-s` flag. An example of such an assignment can be seen below:

```
SESSION_COOKIES=$(./get_hapi_session_cookies.sh –s –u $USER –p $PASSWORD)
```

2.2.3 Using session cookies on subsequent calls to HAPI endpoints

In the above section, it was shown how to obtain the necessary session cookies by using the provided script. In this section two examples are provided on how to use the session cookies to request information or data from HAPI.

1. Use the obtained session cookies to access the HAPI capabilities (note: we are using the `-s` flag on cURL because we are only interested in the json response line):

   ```
   $ curl -s --cookie "XSRF-TOKEN=a038b9da-dfcd-4407-918b-
   c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D6089A26"
   https://swe.ssa.esa.int/hapi/capabilities
   ```

   In the above request, we use the session cookies from the example output listed in Section 2.2.2. The output provided by HAPI will look as follows:

   ```
   {"version":"2.1","status":{"code":1200,"message":"OK"},"outputFormats":["csv",
   "json"]}
   ```

   If used within a script as described in the previous Section, we can use the assigned variable `$SESSION_COOKIES` instead:

```
$ curl -s --cookie "$SESSION_COOKIES"
https://swe.ssa.esa.int/hapi/capabilities
```

2. Use the obtained session cookies to access SOSMAG GP data from HAPI:

   ```
   $ curl -s --cookie "XSRF-TOKEN=a038b9da-dfcd-4407-918b-
   c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D6089A26"
   "https://swe.ssa.esa.int/hapi/data?id=spase://SSA/NumericalData/GEO-KOMPSAT-
   2A/esa_gk2a_sosmag_recalib&time.min=2021-01-21Z&time.max=2021-01-
   22Z&format=json"
   ```

   In the above request, we use the session cookies from the example output listed in Section 2.2.2 to request recalibrated SOSMAG GP data from a single day on 21/01/2021. The output format is set to be json. The output provided by HAPI will look as follows:

   ```
   {"HAPI":"2.1.0","status":{"code":1200,"message":"OK"},"format":"json","paramet
   ers":[{"name":"utc","type":"isotime","length":null,"size":null,"units":"UTC","
   fill":null,"description":"Timestamp in coordinated Universal Time
   (UTC)","label":null},{"name":"version","type":"double","length":null,"size":nu
   ll,"units":null,"fill":null,"description":"Data version
   identifier","label":null},{"name":"b_gse_x","type":"double","length":null,"siz
   e":null,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE
   coordinates (X
   component)","label":null},{"name":"b_gse_y","type":"double","length":null,"siz
   e":null,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE
   coordinates (Y
   component)","label":null},{"name":"b_gse_z","type":"double","length":null,"siz
   e":null,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE
   coordinates (Z component)","label":null}...
   ```

   Note: the output is naturally very long (roughly 10MB in size) and hence truncated here. A full output of requesting just ten seconds of SOSMAG data can be found in Appendix. To store the (large) output of the above command into a file called `sosmag.json` use the `-o` flag or append the command with `> sosmag.json`:

   ```
   $ curl -s --cookie "XSRF-TOKEN=a038b9da-dfcd-4407-918b-
   c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D6089A26" –o sosmag.json
   "https://swe.ssa.esa.int/hapi/data?id=spase://SSA/NumericalData/GEO-KOMPSAT-
   2A/esa_gk2a_sosmag_recalib&time.min=2021-01-21Z&time.max=2021-01-
   22Z&format=json"

   $ curl -s --cookie "XSRF-TOKEN=a038b9da-dfcd-4407-918b-
   c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D6089A26"
   "https://swe.ssa.esa.int/hapi/data?id=spase://SSA/NumericalData/GEO-KOMPSAT-
   2A/esa_gk2a_sosmag_recalib&time.min=2021-01-21Z&time.max=2021-01-
   22Z&format=json" > sosmag.json
   ```

More endpoints provided by HAPI can be looked up at the HAPI Data Access Specification [RD-HAPI-SPEC].

2.3 Programmatic Access using Python 3

In order to access the API programmatically using Python 3, the following steps need to be performed:

1. Establish a session with HAPI by executing the `get_hapi_session_cookies.py` script
2. Use the obtained session cookies to access a certain resource / HAPI endpoint

These steps as well as prerequisites are described in more detail below.

2.3.1 *Prerequisites*

Before a session with the SWE HAPI can be established using the Python-based script there are three prerequisites that must be fulfilled:

1. Valid user credentials with the SWE Portal
   (can be obtained through registration at [RD-SWE])
2. Availability of Python 3 on the executing system
   (can be obtained from [RD-PYTHON])
3. Availability of Python `requests` library on the executing system
   (can be obtained from [RD-PYTHON-REQUESTS])

Besides the requests library, the `json` library is also needed. However, this should be included with the Python 3 installation.

2.3.2 *Get HAPI Session Cookies*

The S2P DS Team has developed a Python-based script to authenticate a user with OpenAM and establish a session with HAPI, ultimately authorising the user to access data and information through the HAPI interface. The script provides functions that can be used within another Python program. It can be downloaded from the SWE Portal at:
https://swe.ssa.esa.int/documents/20182/165822/get_hapi_session_cookies.py

Details on how to use the script can be obtained from the comments in the script header. The info is also included here for convenience:

```
Purpose:
This Python script/module establishes a session with HAPI at a given URL and outputs
the session cookies that can be used for subsequent calls to HAPI. It provides three
functions that can be used in other Python-based programs to access HAPI and retrieve,
for instance, data:
- get_auth_cookie(username, password)
      This only authenticates the user at the SSA SSO server and return the auth
      cookie.
- get_session_cookies(auth cookie)
      Establishes a session with HAPI using an auth cookie and return the session
      cookies.
- establish_hapi_session(username, password)
      Directly establishes a session with HAPI and returns the session cookies.
      Internally, an auth cookie is obtained first and then a session with HAPI is
      established.
```

```
- get_hapi_capabilities(jsession_id, xsrf_token)
      A test function that requests the HAPI/capabilities. It also serves as an HAPI
      example.

The main method can be executed to test the entire script.

Prerequisites:
Python 3 and Python libraries requests and json installed on the system

Usage:
Execute the script with Python 3 to test its functionality. Make sure to adjust the
username and password mentioned in the main function. Use the methods described above
within another script to establish a session with HAPI and to use the obtained session
cookies for subsequent calls to HAPI (e.g. to request data).
```

To be able to execute the script, valid user credentials need to be provided. For this, the main method of the script needs to be modified to replace the placeholders with proper user credentials:

```
[...]
def main():
   username = '<USERNAME>'
   password = '<PASSWORD>'
   [...]
```

After this change, the script can be executed to test its functionality. Below, an example output is provided:

```
$ python3 get_hapi_session_cookies.py
Authentication successful. Obtained authentication cookies:
  esa-ssa-sso-cookie:
AQIC5wM2LY4Sfcy_HJwKGRnWNscH9ZkiE7g_akyseW3u2Kw.*AAJTSQACMDEAAlNLABQtNzUxMzEwODIxODk
1MjE5Mjg4MQ..*
Session successfully established. Obtained session cookies:
  JSESSIONID: 8D9203F2D9B385FA45BC26B7A6C76847
  XSRF-TOKEN: 4ea659a7-bc02-47da-ba6b-470894a90a9d
Testing session cookies on hapi/capabilities:
(True, {'version': '2.1.0', 'status': {'code': 1200, 'message': 'OK'},
'outputFormats': ['csv', 'json']})
```

The session cookies called `XSRF-TOKEN` and `JSESSIONID` can be seen in the output. All subsequent HAPI calls can be executed providing these two cookies.

2.3.3 Using session cookies on subsequent calls to HAPI endpoints

In the above section, it was shown how to obtain the necessary session cookies by using the provided Python-based script. In this section an example is provided on how to use the session cookies to request information or data from HAPI within a Python-based program. For this, it is assumed that the provided script is being imported into the Python program:

```
import get_hapi_session_cookies

def main():
    username = '<USERNAME>'
    password = '<PASSWORD>'
    session_established, jsession_id, xsrf_token = get_hapi_session_cookies.
establish_hapi_session(username, password)

    if session_established:
          data = get_data(jsession_id, xsrf_token)
          # do something with the /data/ dict

if __name__ == “__main__”:
    main()
```

The mentioned `get_data` function in the above example needs to be implemented. For this, the provided script also includes an example that retrieves the capabilities from HAPI. Using this skeleton and the description from Section 2.2.3 the following function to retrieve SOSMAG data can be written to retrieve json-formatted data. (As seen in the output of the *capabilities* endpoint, HAPI can return `json` and `csv` formats at the moment):

```
import json
import requests

def get_date(jsession_id, xsrf_token):
    response = requests.get(
"https://swe.ssa.esa.int/hapi/data?id=spase://SSA/NumericalData/GEO-KOMPSAT-
2A/esa_gk2a_sosmag_recalib&time.min=2021-01-21Z&time.max=2021-01-22Z&format=json",
                cookies = {
                      'JSESSIONID': jsession_id,
                      'XSRF_TOKEN': xsrf_token,
                })
    data = json.loads(response.content)
    return data
```

This function is just an example and a flexible version needs to be implemented. However, it already showcases all necessary parts to request data from HAPI using the *data* endpoint. The last two lines in the above example translate the response into a Python dictionary using the `json` Python module before returning the data. This is so the response can be easily accessed from the rest of the program in the typical Python way. However, changing these two lines to convert and return the response using different data formats is possible as well.

More endpoints provided by HAPI can be looked up at the HAPI Data Access Specification [RD-HAPI-SPEC].

3 LIMITATIONS

Currently, the access to data through the SWE HAPI is restricted to data sets from SOSMAG on GK2A only with data sets from NGRM on EDRS-C coming soon.
Also, as described in Section 2.1 authorisation to HAPI currently only works reliably using the provided script to imitate the browser behaviour and to establish a session with HAPI. In future releases it is foreseen to introduce further authorisation methods.

Please be aware that by retrieving the data you agree to not further distribute the data nor to make it accessible for third parties.

APPENDIX

Below is the output when requesting ten seconds of recalibrated SOSMAG data:

```
user@linux:~/$ curl -s --cookie "XSRF-TOKEN=a038b9da-dfcd-4407-918b-
c40a86f773b0;JSESSIONID=676D6897FF5CAF11D3CCE807D6089A26"
"https://swe.ssa.esa.int/hapi/data?id=spase://SSA/NumericalData/GEO-KOMPSAT-
2A/esa_gk2a_sosmag_recalib&time.min=2021-01-21T00:00Z&time.max=2021-01-
21T00:00:10Z&format=json"
```

```
{"HAPI":"2.1.0","status":{"code":1200,"message":"OK"},"format":"json","parameters":[
{"name":"utc","type":"isotime","length":null,"size":null,"units":"UTC","fill":null,"
description":"Timestamp in coordinated Universal Time
(UTC)","label":null},{"name":"version","type":"double","length":null,"size":null,"un
its":null,"fill":null,"description":"Data version
identifier","label":null},{"name":"b_gse_x","type":"double","length":null,"size":nul
l,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE coordinates (X
component)","label":null},{"name":"b_gse_y","type":"double","length":null,"size":nul
l,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE coordinates (Y
component)","label":null},{"name":"b_gse_z","type":"double","length":null,"size":nul
l,"units":"nT","fill":"NaN","description":"Magnetic Field B in GSE coordinates (Z
component)","label":null},{"name":"b_hpen_p","type":"double","length":null,"size":nu
ll,"units":"nT","fill":"NaN","description":"Magnetic Field B in HPEN coordinates (P
component)","label":null},{"name":"b_hpen_e","type":"double","length":null,"size":nu
ll,"units":"nT","fill":"NaN","description":"Magnetic Field B in HPEN coordinates (E
component)","label":null},{"name":"b_hpen_n","type":"double","length":null,"size":nu
ll,"units":"nT","fill":"NaN","description":"Magnetic Field B in HPEN coordinates (N
component)","label":null},{"name":"position_x","type":"double","length":null,"size":
null,"units":"km","fill":"NaN","description":"Spacecraft Position in GSE (X
component)","label":null},{"name":"position_y","type":"double","length":null,"size":
null,"units":"km","fill":"NaN","description":"Spacecraft Position in GSE (Y
component)","label":null},{"name":"position_z","type":"double","length":null,"size":
null,"units":"km","fill":"NaN","description":"Spacecraft Position in GSE (Z
component)","label":null},{"name":"data_flags","type":"double","length":null,"size":
null,"units":null,"fill":null,"description":"Data flags, see
InformationURL","label":null},{"name":"final","type":"double","length":null,"size":n
ull,"units":null,"fill":null,"description":"Calibration status, see
InformationURL","label":null},{"name":"frequency","type":"double","length":null,"siz
e":null,"units":null,"fill":null,"description":"Current sampling
frequency","label":null}],"startDate":"2019-02-28T00:00Z","stopDate":"2050-01-
01T00:00Z","cadence":"PT0.062S","description":"Recalibrated L2 Magnetic Field Data
with 1-16Hz from SOSMAG on GEO-KOMPSAT-2A in geostationary orbit at 128.2E. These
data are available to all registered users of the ESA Space Weather Service Portal.
Please be aware that by downloading the data you agree to not further distribute the
data nor to make it accessible for third
parties.","resourceURL":null,"resourceID":"spase://SSA/NumericalData/GEO-KOMPSAT-
2A/esa_gk2a_sosmag_recalib","data":[["2021-01-21T00:00:00.900Z",0,-16.63,-
3.022,118.199,-36.974,2.387,113.507,32,1,false,-
23148.984,31584.71,15639.606],["2021-01-21T00:00:01.900Z",0,-16.963,-3.006,117.875,-
36.659,2.094,113.328,32,1,false,-23151.321,31582.775,15640.053],["2021-01-
21T00:00:02.900Z",0,-17.026,-2.901,117.663,-36.467,2.08,113.176,32,1,false,-
23153.658,31580.84,15640.5],["2021-01-21T00:00:03.900Z",0,-16.774,-3.025,117.851,-
36.768,2.218,113.237,32,1,false,-23155.996,31578.905,15640.946],["2021-01-
21T00:00:04.900Z",0,-16.619,-3.229,118.183,-37.129,2.252,113.447,32,1,false,-
23158.333,31576.97,15641.393],["2021-01-21T00:00:05.900Z",0,-16.693,-3.195,118.168,-
37.058,2.212,113.465,32,1,false,-23160.67,31575.035,15641.84],["2021-01-
21T00:00:06.900Z",0,-17.049,-3.179,117.865,-36.738,1.905,113.312,32,1,false,-
23163.006,31573.099,15642.287],["2021-01-21T00:00:07.900Z",0,-16.874,-
3.174,117.968,-36.869,2.053,113.348,32,1,false,-
23165.343,31571.164,15642.733],["2021-01-21T00:00:08.900Z",0,-16.758,-
3.243,118.123,-37.042,2.118,113.436,32,1,false,-
23167.679,31569.228,15643.18],["2021-01-21T00:00:09.900Z",0,-16.669,-3.24,118.132,-
37.093,2.186,113.415,32,1,false,-
23170.016,31567.292,15643.626]],"contactID":"spase://SSA/Person/SSCC.Helpdesk"}
```