TNS APIs Manual

Updated: Aug 2016

Table of Contents

1. General

This manual serves for explanation of the additional APIs maintained by the TNS, for changing proprietary period end date and retrieving object lists and object details by various criteria. The *Bulk report* API is described in the corresponding *Bulk report manual*.

As opposed to the Asynchronous *Bulk report* mechanism, the APIs described here are synchronous; i.e. a reply JSON is obtained immediately, without having to check for a reply for a given report-id.

We have set up on our *sandbox* site an API TEST FORM webpage where it is possible to experiment with the various APIs. The URL of the API TEST FORM page is:

https://sandbox-tns.weizmann.ac.il/api

where three forms are available for experimenting with the following APIs:

- *Bulk report* (as mentioned, refer to the *Bulk report manual*)
- *Change prop period*
- Search, which includes the *Search*, *Get object*, and *Get file* APIs.

An api_key should always be provided (defining the sending Bot). The input JSON-formatted data is placed in the relevant section's JSON editor, and upon submission, the reply JSON is displayed on the same editor.

Further explanations about the sandbox can be found in *Sec. 2* of the *Bulk report manual*.

Explanations about the Bots and the API-Keys can be found in *Sec. 3* of the *Bulk report manual*.

A PHP sample code is provided for assisting with the development of the *Change prop. period end date* and the *Get object* APIs, as examples, and this is explained in *Sec. 7*.

Do not hesitate to contact us for any assistance and/or clarifications.

2. Change proprietary period end date

2.1. General

This API serves for updating the prop. period end date of reported entries, for switching an AT report and/or classification report and/or spectra to public, either on the same day or some future date (back-dating is not possible). Only a bot that is associated with the group that submitted the report (AT/Classification/spectra) can request to perform such a change to the prop. period end date. The sample code as described in Sec. 8 provides an example for this API.

2.2. URL: https://wis-tns.weizmann.ac.il/api/set/prop-period

Sandbox URL: https://sandbox-tns.weizmann.ac.il/api/set/prop-period

2.3. Data description

- Type: POST
- Mandatory parameters:
  - api_key – the bot's API Key
  - data – holds the JSON report

2.4. JSON key-value list

- Specified in red are mandatory keys; in green example values, and remarks (not part of the JSON) in blue.
- Fields containing preset values (e.g. groupid) must include the appropriate value id. A list (in JSON format) of possible values can be viewed on the Bulk report page or downloaded at: https://wis-tns.weizmann.ac.il/api/values
- It is possible to change the end prop. period date for any of the following reported entries (by toggling the '0' / '1' values): AT report (together with the associated photometry points), Classification report, and the uploaded spectra.

```
{
 "objname": "2016abc",
 "groupid": "2",
 "end_prop_period_date": "2016-12-31",
 "at": "1",                                    // "1" or "0" for updating or not the relevant
 "classification": "0",                        // ATs/classifications/spectra.
 "spectra": "0"
}
```

2.5. If no errors, the reply feedback contains counters of the updated entries, e.g.:

```
  "feedback": {
   "affected_rows": {
    "at": {
     "at_reps_count": 1,
     "obj_count": 1,
     "phot_count": 2
    },
    "classification": 1,
    "spectra": 1
   }
  }
```

Relevant possible errors are e.g. {0,6,7}, as specified in the feedback messages table in Sec. 6.

3. Search objects

3.1. General

This API serves for listing the object names (objname & prefix) satisfying given search criteria. The search criteria are (applying "AND" operator for whichever combination of parameters):

- Cone search for given RA/DEC, radius & units.
  If RA/DEC are specified without values for radius/units, the default values used are 3 arcsec.
- Object name (like)
- Internal name (like)

Access permissions are taken into account. If an object is proprietary (not public), only a Bot that is associated with its reporting (source) group will have access to it. Likewise, for the following *Get object* API.

3.2. URL: https://wis-tns.weizmann.ac.il/api/get/search
Sandbox URL: https://sandbox-tns.weizmann.ac.il/api/get/search

3.3. Data description

- Type: POST
- Mandatory parameters:
  - api_key – the bot's API Key
  - data – holds the JSON report

3.4. JSON key-value list

e.g. A cone search, including a given internal name:

```
{
  "ra": "05:24:18.0",
  "dec": "+09:10:37.0",
  "radius": "5",
  "units": "arcsec",
  "objname": "",
  "internal_name": "Gaia16azh"
}
```

3.5. The JSON reply array contains a list of object (+prefix) names; e.g.:

```
"reply": [
  {
    "objname": "2016evp",
    "prefix": "AT"
  }
]
```

4. Get object (details/entries)

4.1. General

This API retrieves object details per given object name, including its associated photometry and/or spectra if requested.

The object name should be exact (e.g. the 'objname' value as obtained by the previous *Search objects* API), and without prefix; e.g. '2016abc'.

If the object's spectra are requested, the ascii/FITS files of the spectrum/a will be specified in the JSON key-value reply list. To download the actual file/s use the following *Get file* API.

4.2. URL: https://wis-tns.weizmann.ac.il/api/get/object

Sandbox URL: https://sandbox-tns.weizmann.ac.il/api/get/object

4.3. Data description

- Type: POST
- Mandatory parameters:
  - api_key – the bot's API Key
  - data – holds the JSON report

4.4. JSON key-value list

```
{
  "objname": "2016G",
  "photometry": "1",          // “1” or “0” for retrieving or not the associated photometry/spectra
  "spectra": "1"
}
```

4.5. The reply JSON contains the object details. A spectrum section within the reply list may look like, e.g.:

```
"spectra": [
  {
    "obsdate": "2016-01-10 14:42:31",
    "jd": 2457400,
    "public": "1",
    "end_prop_period": null,
    "exptime": "2100",
    "groupid": "0",
    "observer": "Jujia Zhang",
    "reducer": "Jujia Zhang",
    "asciifile":        "https://wis-tns.weizmann.ac.il/system/files/uploaded/None/None_2016G_2016-01-
10_14:42:31_LJT_YFOSC.dat",
    "fitsfile": "",
    "remarks": "",
    "source_group_name": null,
    "instrument": {
      "id": "107",
      "name": "YFOSC"
    },
    "telescope": {
      "id": "77",
      "name": "LJT"
    }
  }
], ...
```

5. Get file

5.1. General

This API serves for downloading a given file via providing the appropriate URL, as e.g. obtained from the *Get object* API.

Clearly, a check is performed that the requesting Bot (via the supplied api_key) has access permission to the required URL.

Files on the public domain (not proprietary) can be accessed without providing an api_key.

5.2. URL: The file's URL; e.g. from the 'asciifile' key above:

https://wis-tns.weizmann.ac.il/system/files/uploaded/None/None_2016G_2016-01-10_14:42:31_LJT_YFOSC.dat

5.3. Data description

- Type: POST
- Mandatory parameters:
    - api_key – the bot's API Key

6. Feedback messages

6.1. General

The feedback messages can be easily identified by their message id's, allowing for efficient parsing of the resulting feedbacks. We list here the mapping between the message id's and their corresponding texts. (The majority of the listed messages are relevant for the Bulk report API; messages 7,10,110 are revelant for the APIs of this manual.)

6.2. Mapping of the messages

Message id	Message	Provided values
General		
200	OK	
201	Created	
400	Bad request	
401	Unauthorized	
403	Forbidden	
404	Not Found	
Blocking errors		
0	Invalid	
1	Last non-detection should precede the Discovery Datetime	
2	At least one Photometry point - that of the discovery - should be filled	
3	Required field	
4	Proprietary period cannot extend more than 100 years	
5	An identical AT report (sender, RA/DEC, discovery date) already exists	
6	Last non-detection or archival info must be filled	
7	Proprietary period cannot be backdated	
10	Unauthorized	
20	ASCII file or FITS file must be uploaded	
21	At least one Spectra should be filled	
22	An ASCII file must be uploaded	
Feedbacks		
100	Transient object was inserted	objname (eg '2016abc')
101	Transient object exists	objname (eg '2016abc'), prefix (eg 'AT'), type, RA, DEC, separation
102	Related file uploaded	
103	Submitted	
110	No results found	
120	New object type set	new_object_type
121	Object name prefix has changed	new_object_name
122	Object redshift was set	new_redshift
123	Object redshift changed	new_redshift

7. Sample codes

7.1. General

A set of functions written in ***PHP*** is provided to serve as examples for developing the necessary codes for the APIs listed in this manual. The sample code (zipped) files can be downloaded from the provided links on the TNS help page.

The sample codes direct to the sandbox site for experimentation, e.g.:

```
class TNSClient {

     /// TNS's API URL
     protected static $baseAPIUrl = 'https://sandbox-tns.weizmann.ac.il/api/';
```

Change this URL to 'https://wis-tns.weizmann.ac.il/api/' for working against the real site.

7.2. Sample code for the *Change prop period* API

Name: tns_api_change_prop_period_sample_code.php

The section that submits the request and receives the reply is given as follows:

```
define('API_KEY', '1234567890123456789012345678901234567890');

// Change prop period.
$feed_handler = new TNSClient(array('api_key' => API_KEY));
$json = array(
  "objname"        => "2016csv",
  "groupid"        => "2",
  "end_prop_period_date" => "2016-10-10",
  "at"             => "1",
  "classification"  => "0",
  "spectra"        => "0"
);
$reply = $feed_handler->setPropPeriod(json_encode($json));
print_r($reply);
```

Parsing the reply (if in error or not) can be performed as follows:

```
// Check if key is present.
$find = findKey($reply, 'id_code');
$find = intval($find);

if($find > 400) {
     print_r('General error: ' . $reply['id_message']);
}
else if($find == 400) {
     print_r('Bad request. Please check feedback for more information');
}
else {
     // Do something.
}
```

7.3. Sample code for the *Search*, *Get object* & *Get file* APIs

Name: tns_api_search_sample_code.php

The section that submits the query and receives the reply is given as follows (in this example, a 5 arcsec cone search):

```
define('API_KEY', '1234567890123456789012345678901234567890');

// Send search parameters.
$feed_handler = new TNSClient(array('api_key' => API_KEY));
$json = array(
  "ra"      => "05:24:18.0",
  "dec"     => "+09:10:37.0",
  "radius"  => "5",
  "units"   => "arcsec",
  "objname" => "",
  "internal_name" => ""
);
$reply = $feed_handler->getSearch(json_encode($json));
print_r($reply);
```

If no errors, looping over the retrieved objects and for each one obtaining it's details (*Get object* API), including the photometry and spectra entries. For each spectrum, downloading its asciifile using the *Get file* API implementation.

```
// Check if key is present.
$find = findKey($reply, 'id_code');

if($find > 400) {
        print_r('General error: ' . $reply['id_message']);
}
else if($find == 400) {
        print_r('Bad request. Please check feedback for more information');
}
else {
  $obj_request = array(
    "photometry"  => "1",
    "spectra"     => "1"
  );

  // Looping over retrieved objects list
  foreach($reply['data']['reply'] as $delta => $objname) {
    $obj_request["objname"] = $objname['objname'];

        // Retrieving object's data (photometry,spectra)
    $object = $feed_handler->getObject(json_encode($obj_request));
    print_r($object);

    // Check if key is present.
    $obj_find = findKey($reply, 'id_code');

    if($obj_find > 400) {
        print_r('General error: ' . $reply['id_message']);
    }
    else if($obj_find == 400) {
        print_r('Bad request. Please check feedback for more information');
    }
    else {
      foreach($object['data']['reply']['spectra'] as $index => $spectra) {
        if(isset($spectra['asciifile']) && !empty($spectra['asciifile'])) {
          $ascii = $feed_handler->getFile(array('file' => $spectra['asciifile']));
                         $filename = basename($spectra['asciifile']);
                         file_put_contents($filename, $ascii);
        }
      }
    }
  }
}
```

The getObject function is defined as follows:

```
public function getObject($options = array()) {
            $feed_url = $this->buildUrl('get/object');
            $feed_parameters = $this->buildParameters(array('data' => $options));
            return $this->getFeed($feed_url, $feed_parameters);
}
```

Whereas the getFile function is:

```
public function getFile($options = array()) {
            $feed_url = $options['file'];
            $feed_parameters = $this->buildParameters(array());
            return $this->getFeed($feed_url, $feed_parameters);
}
```