



Bulk API Manual - Automated reporting to the TNS

Updated: Jan 2025 (adapted to TNS2.0.)

Table of Contents

1. General.....	2
2. Bots	3
3. The <i>SANDBOX</i> Environment and the “ <i>Bulk report</i> ” API test page	3
4. File Uploads.....	6
5. AT and Classification Reports.....	6
6. TSV submission (deprecated).....	13
7. Feedback messages	15

1. General

AT reports (AT reps) and Classification reports (Class reps) can be submitted to the TNS by either using the interactive forms on the TNS website, or via the "Bulk" API method - submission of JSON-formatted data or TSV (Tab-Separated-Values) lists in an automated way from defined machines ("Bots") of e.g. the various contributing surveys.

The Bulk reports are being processed as promptly as possible in an asynchronous mode, dependent on the actual incoming queue of reports and the server's available resources. (Usually the processing of the reports is immediate.)

- Each report, sent to the TNS by the Bulk API (JSON/TSV), receives a sequential report-id that is provided to the sender of the report (serving as a "receipt").
- The sending machine should query the API (every second or a few) for the reply of the corresponding report-id.
- The report is processed in its turn, and once a reply is obtained for the given report-id, the sending machine can relate to it and parse the resulting feedbacks (e.g. if a new event was created or if it already exists on the system for the reported coordinates, the newly designated or existing name, etc...).

We have set up on our "sandbox" site an API TEST webpage where it is possible to experiment with submission of reports using a dedicated form. This is described in *Section 3*.

Important – Please do not commence sending real Bulk reports to the production site before verifying on the sandbox environment that all your codes and scripts work flawlessly and that the JSON/TSV are submitted and processed in a correct manner!

Python sample codes are provided for assisting with the development of the relevant scripts, explained in *Section 8*.

The JSON format for preparation of the AT and Class Reps are described in *Section 4*.

Do not hesitate to [contact us](#) for any assistance and/or clarifications.

2. Bots

1.1. General

Registered users can define Bots that are required for sending automated reports using the Bulk API. This is done by clicking the “+Add bot” button on the BOTS page: <https://www.wis-tns.org/bots>

A Bot, e.g. “PESSTO_Bot0”, is associated with a survey/group and includes a definition of an *api_key*, which serves for authentication of any data submitted to the TNS from the given Bot. It is possible to define multiple Bots per given survey/affiliation.

For associating a Bot with specific group/s, click to edit the Bot on the BOTS page and then select the required group/s, for the group owner/s to approve. (The users and bots associated with each group are displayed on the GROUPS page.)

1.2. Obtaining an API Key

Upon creation of a Bot, an *api_key* is provided to the relevant person/s handling the automatic reports from the survey/group. This *api_key* should be kept and used for the submission of any data entry.

If there’s a need for a new *api_key*, an owner of the Bot may click to edit the Bot (on the BOTS page) and there check the “Create new API Key” checkbox and save. The newly created *api_key* will be displayed.

3. The *SANDBOX* Environment and the “Bulk report” API test page

3.1. General

For experimentation and development of the sending of reports via the bulk API (and the additional available API’s) we’ve set up the following sandbox site: <https://sandbox.wis-tns.org>

This site is a replica of the production site but such that the data can be freely experimented with. Once a week, on Sunday UT 04, the data on the sandbox site is reset with data from the real site, so note that any data you have submitted for experimentation will be gone after the reset.

In this respect, note that also the Bots that have been defined in the sandbox env. will be overridden by the Bots as they are defined on the production site. Therefore, best to have your Bots defined on the production site also during the experimentation period so that their definitions hold and remain on the sandbox.

3.2. The “Bulk report” API test page

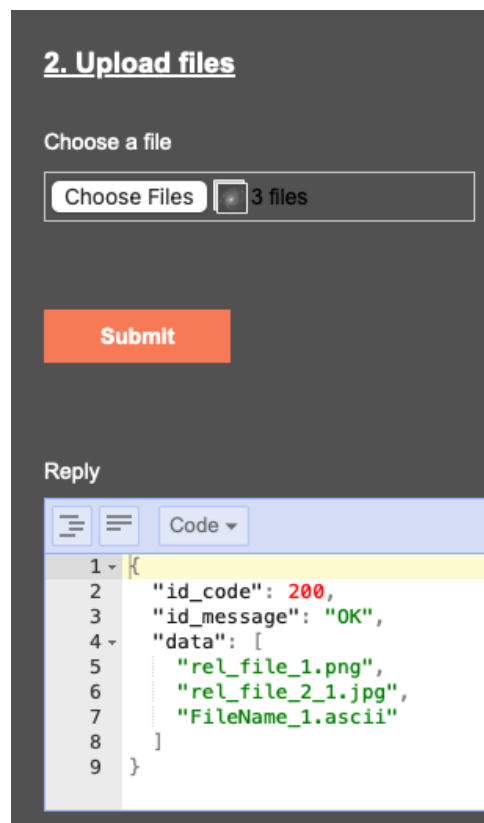
The API TEST FORM page is active only on the sandbox site, and its aim is to serve for experimentation with the various APIs, including the sending of JSON/TSV reports. The form on the *Bulk report* tab mimics the stages that should be implemented by the scripts for the automated submissions to the TNS.

<https://sandbox.wis-tns.org/api/test>

The Bulk report form consists of the following **4 stages**:

- ❖ Insertion of the `api_key` and the bot's `tns_marker` credentials (must be specified for authentication).
- ❖ Uploading of the relevant files (e.g. related files, the spectrum files of a classification report etc...).

After clicking the Submit button, a reply is displayed in the JSON editor below to approve that the files were uploaded successfully (or not), with the exact temporary names as they should be specified in the report itself. E.g.:



- ❖ Submission of the report itself by either uploading a TSV file or by specifying the JSON-formatted data in the JSON editor window.

Clicking the "Get AT rep example" or "Get Classification example" (below the "Send data" button of the JSON editor) loads to the JSON editor an example template of the given report.

After submission of a JSON report (by clicking the "Send data" button), a confirmation for the submitted report appears in the JSON editor, with specification of the obtained report-id. E.g.:

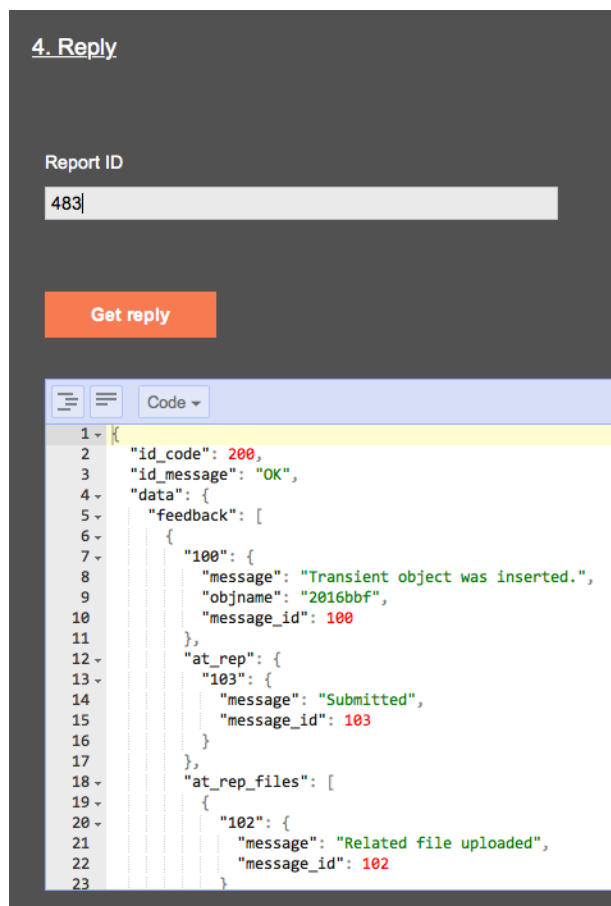


```
1 {
2   "id_code": 200,
3   "id_message": "OK",
4   "data": {
5     "time_recieved": 1459080823,
6     "report_id": 483
7   }
8 }
```

The report is then automatically processed.

❖ Checking for the reply of the processed report.

Upon inserting the given report-id and clicking on the “Get reply” button, the reply JSON of the report is displayed in the JSON editor, with the relevant feedback indications and messages. E.g.:



4. Reply

Report ID

483

Get reply

```
1 {
2   "id_code": 200,
3   "id_message": "OK",
4   "data": {
5     "feedback": [
6       {
7         "100": {
8           "message": "Transient object was inserted.",
9           "objname": "2016bbf",
10          "message_id": 100
11        }
12      },
13      "at_rep": {
14        "103": {
15          "message": "Submitted",
16          "message_id": 103
17        }
18      },
19      "at_rep_files": [
20        {
21          "102": {
22            "message": "Related file uploaded",
23            "message_id": 102
24          }
25        }
26      ]
27    }
28   }
29 }
```

Note that in the JSON editor windows, the JSON data can be displayed in both “Code” and “Tree” modes. (For copy/pasting lines use the “Code” mode, clearly.)

4. File Uploads

4.1. General

Any files accompanying a report, e.g. a related file, spectrum etc..., should be uploaded to the system BEFORE the submission of the report itself. The report should include the file name exactly as it was given by the feedback to the upload, see 5.4, otherwise the system will not be able to locate it.

4.2. URL: <https://www.wis-tns.org/api/set/file-upload>

Sandbox URL: <https://sandbox.wis-tns.org/api/set/file-upload>

4.3. Data description

- Type: POST
- Mandatory parameters:
 - `api_key` – the bot's API Key
 - `User-Agent` – the bot's `tns_marker`
 - `files[i]` – array of files to upload; $i=0\dots N$ (number of uploaded files)
- See the `tns_2_api_bulk_report.py` sample code, upload files section.

4.4. Upon a successful upload, the system will reply back with a name for each of the uploaded files (the names of the files may be changed by the system).

5. AT and Classification Reports

5.1. General

Submission of reports by JSON-formatted data allows for the maximal flexibility. A single or multiple entries (of an AT or Classification report/s) can be specified, and for specific items within a report (e.g. photometry entries in the AT report or spectra entries in a classification report) up to 10 entries can be specified, exactly like in the interactive dedicated forms (“AT/Classification Report Form”).

The amount of entries to be listed within a single AT or Classification submission is limited to 100. So, for instance, if your survey's mode of operation is defined to send discovery reports in a summarized way (at the end of the night or every several hours) rather than for each discovered candidate transient separately, and the list surpasses 100 entries, it should be split to several reports.

Sections 5.5 and 5.6 below list the complete possible key-value pairs for the AT and Classification Reports. Note that these values, as well as the allowed formats and mandatory vs. non-mandatory parameters are in complete agreement with the interactive forms webpages. Section 5.7 lists the FRB Report JSON keywords.

For obtaining the necessary id's of the groups, instruments, filters, units, object-types, spectrum-types etc..., check on the Bulk report page the “Get AUX tables values id's” in the JSON submit section (below the JSON editor).

5.2. URL: <https://www.wis-tns.org/api/set/bulk-report>
Sandbox URL: <https://sandbox.wis-tns.org/api/set/bulk-report>

5.3. Data description

- Type: POST
- Mandatory parameters:
 - `api_key` – the bot's API Key
 - User-Agent – the bot's `tns_marker`
 - `data` – holds the JSON report
- See the `bulk_report.py` sample code, AT/Class report sections.

5.4. Upon a successful submission, a reply JSON is sent back with the "report_id" identifying the report that was just submitted. Use this report in order to request back the submission reply (see section 8) Note: we expect the reply to be ready immediately, however, this depends on the server's load and may take a little longer. Therefore, we suggest to have a mechanism in place quering the server for a reply every several seconds, until a reply appears. In case there is a problem and no reply is obtained, please contact us by email and provide the report-id.

5.5. AT Report JSON key-value list

- Specified in **red** are mandatory keys; Not part of the JSON: in **green** are the previous keyword names that should be replaced with the new names, and clarifications in **blue**.
- Internal name: It is possible to either provide the exact internal name of the object as it exists in the reporting survey, or instead, if the internal name is chosen to be created according to the name provided by the TNS, to provide instructions for the construction of the internal name by specifying the *internal_name_format* values. E.g. if specifying the following *internal_name_format* values:

```
"internal_name_format": {  
  "prefix": "iPTF",  
  "year_format": "YY",  
  "postfix": ""  
},
```

the internal name created for a TNS name **2016xyz** will be **iPTF16xyz**.
- All fields containing preset values (id's; e.g. `reporting_groupid`, `at_type`, `instrumentid`...) must include the appropriate value id. A list (in JSON format) of possible values can be viewed on the Bulk page (clicking the "Get AUX tables values id's") or downloaded at: <https://www.wis-tns.org/api/get/values>
- You can add to your report json an "internal_ids" section for providing up to 5 keyword-value pairs, that will be returned with the reply json. This can help you match between the data you sent (e.g. which object) and the reply. The limitations being up to 30 characters, alphanumeric and `'_' | '-' | '#'`, for both the keyword names and the values. Those values are NOT saved in the TNS's database but are only meant to be returned exactly as they were provided within the reply json.

```

{
  "at_report": {
    "0": {
      "ra": {
        "value": "10:20:30.04",
        "error": "0.5",
        "units": "arcsec"
      },
      "dec": {
        "value": "+20:30:40.05",
        "error": "0.5",
        "units": "arcsec"
      },
      "reporting_groupid": "2",
      "data_source_groupid": "2",
      "reporter": "J. Smith, on behalf of SurveyName...",
      "discovery_datetime": "2016-03-01.234",
      "at_type": "1",
      "host_name": "NGC 1234",
      "host_redshift": "",
      "internal_name": "",
      "internal_name_format": {
        "prefix": "prefixStr",
        "year_format": "YY",
        "postfix": "postfixStr"
      },
      "auto_class": [
        {
          "algorithm": "algo1",
          "version": "1.1",
          "objtypeid": 3,
          "prob": 0.8,
          "additional_params": []
        },
        {
          "algorithm": "algo1",
          "version": "1.1",
          "objtypeid": 5,
          "prob": 0.2,
          "additional_params": []
        }
      ],
      "remarks": "",
      "end_prop_period": "",
      "prop_period_groups": [
        "1",
        "2"
      ],
      "non_detection": {
        "obsdate": "2016-02-28.123",
        "limiting_flux": "21.5",
        "flux_unitid": "1",
        "filterid": "50",
        "instrumentid": "103",

```

// "0" is always the first entry; if multiple entries
// add "1", "2" etc...

// reporting_group_id
// discovery_data_source_id

// Possible to specify this if not specifying a
// specific internal_name.
// Valid values: 'YY' / 'YYYY'

// A new section for specifying classification
// probabilities, as obtained by specific tools

// The end proprietary period date
// Groups that should have access to the data
// if prop period specified. No need to specify the
// reporting group again

// Like in the form – if there is no last
// non-detection photometry measurement in
// hand, archival info should be specified.
// flux_units
// filter_value
// instrument_value


```

    "exptime": "60",
    "observer": "Robot",
    "comments": "",
    "archiveid": "", // archival info should include an id (0 if no specific
    "archival_remarks": "" // catalog) and a remark
  },
  "photometry": {
    "0": {
      "obsdate": "2016-03-01.234",
      "flux": "19.5",
      "flux_error": "0.2",
      "limiting_flux": "",
      "flux_unitid": "1", // flux_units
      "filterid": "50", // filter_value
      "instrumentid": "103", // instrument_value
      "exptime": "60",
      "observer": "Robot",
      "comments": ""
    },
    "1": { // If multiple photometry entries, add "1", "2" etc
      "obsdate": "",
      "flux": "",
      "flux_error": "",
      "limiting_flux": "",
      "flux_unitid": "",
      "filterid": "",
      "instrumentid": "",
      "exptime": "",
      "observer": "",
      "comments": ""
    }
  },
  "related_files": {
    "0": {
      "related_file_name": "FC.png",
      "related_file_comments": "Finding Chart..."
    },
    "1": {
      "related_file_name": "DisclImage.jpg",
      "related_file_comments": "Discovery image..."
    }
  },
  "internal_ids": { // Up to 5 customized keyword-value pairs,
    "internal_name": "Obj_internal_name", // max 30 chars for both keyword & value
    "internal_objid": "Obj_internal_id"
  }
}

```

5.6. Classification Report JSON key-value list

```
{
  "classification_report": {
    "0": {
      "name": "2016abc",
      "classifier": "K. Smith, on behalf of SurveyName",
      "objtypeid": "3",
      "redshift": "0.123",
      "groupid": "2",
      "end_prop_period": "", // Proprietary period for the classification itself.
      "prop_period_groups": [ // prop_period_groups are relevant only
        "1" // when specifying a proprietary period.
      ],
      "remarks": "Another Ia",
      "spectra": {
        "0": {
          "obsdate": "2016-03-07.89",
          "instrumentid": "1",
          "exptime": "600",
          "observer": "J. Smith",
          "reducer": "K. Smith",
          "spectypeid": "1",
          "ascii_file": "FileName.ascii",
          "fits_file": "FileName.fits",
          "remarks": "",
          "end_prop_period": "" // Proprietary period for the provided spectrum.
        },
        "1": { // If multiple spectra entries, add "1","2" etc
          "obsdate": "",
          "instrumentid": "",
          "exptime": "",
          "observer": "",
          "reducer": "",
          "spectypeid": "",
          "ascii_file": "",
          "fits_file": "",
          "remarks": "",
          "end_prop_period": ""
        }
      }
    },
    "related_files": {
      "0": {
        "related_file_name": "Classification_plot.png",
        "related_file_comments": "SNID Classification plot..."
      },
      "1": {
        "related_file_name": "",
        "related_file_comments": ""
      }
    },
    "internal_ids": { // Up to 5 customized keyword-value pairs,
      "internal_name": "Obj_internal_name", // max 30 chars for both keyword & value
      "internal_objid": "Obj_internal_id"
    }
  }
}
```

5.7. FRB Report JSON key-value list

```
{
  "frb_report": {
    "0": {
      "ra": {
        "value": "10:11:12.34",
        "error": "20",
        "units": "arcsec"
      },
      "dec": {
        "value": "-21:22:23.45",
        "error": "1.5",
        "units": "arcmin"
      },
      "reporting_groupid": "86",
      "data_source_groupid": "86",
      "internal_name": "FRB20250101A",
      "at_type": "5",
      "reporter": "J. Smith, on behalf of ...",
      "discovery_datetime": "2025-01-01 10:11:12.345",
      "barycentric_event_time": "",
      "end_prop_period": "",
      "prop_period_groups": [],
      "transient_redshift": "0.012",
      "host_name": "NGC 1234",
      "host_redshift": "",
      "repeater_of_objid": "",
      "public_webpage": "https://www.chime-experiment.ca",
      "region_ellipse": "[[11.1, -22.2, 20, 15, 30.0, 2.0], [11.2, -22.3, 16, 11, 30.0, 3.0]]",
      "region_ellipse_unitid": "28",
      "region_polygon": "[[[1,2],[3,4],[5,6],[7,8]]]",
      "region_filename": "frb_region.fits",
      "dm": "440",
      "dm_err": "4.4",
      "dm_unitid": "25",
      "galactic_max_dm": "23.5",
      "galactic_max_dm_model": "NE2001",
      "remarks": "",
      "photometry": {
        "0": {
          "obsdate": "2025-01-01 10:11:12.345",
          "flux": "20",
          "flux_error": "2",
          "limiting_flux": "22",
          "flux_units": "8",
          "filter_value": "1",
          "instrument_value": "222",
          "snr": "11",
          "fluence": "234",
          "fluence_err": "23.4",
          "fluence_unitid": "22",
          "exptime": "90",
          "observer": "Robot",
```

```

    "burst_width": "2.3",
    "burst_width_err": "0.23",
    "burst_width_unitid": "23",
    "burst_bandwidth": "234",
    "burst_bandwidth_err": "23.4",
    "burst_bandwidth_unitid": "24",
    "scattering_time": "3",
    "scattering_time_err": "0.3",
    "scattering_time_unitid": "23",
    "dm_struct": "24",
    "dm_struct_err": "2.4",
    "dm_struct_unitid": "25",
    "rm": "22",
    "rm_err": "2.2",
    "rm_unitid": "26",
    "frac_lin_pol": "0.1",
    "frac_lin_pol_err": "0.01",
    "frac_circ_pol": "0.2",
    "frac_circ_pol_err": "0.02",
    "ref_freq": "600",
    "ref_freq_unitid": "24",
    "inst_bandwidth": "400",
    "inst_bandwidth_unitid": "24",
    "channels_no": "4",
    "sampling_time": "10",
    "sampling_time_unitid": "23",
    "comments": ""
  }
},
"related_files": {
  "0": {
    "related_file_name": "",
    "related_file_comments": ""
  }
},
"internal_ids": { // Up to 5 customized keyword-value pairs,
  "internal_name": "Obj_internal_name", // max 30 chars for both keyword & value
  "internal_objid": "Obj_internal_id"
}
}
}
}

```

6. TSV submission (deprecated)

*** In TNS2.0 we have removed the option for reporting via TSVs.

*** If this causes any concern, please feel free to contact us.

6.1. General

Submission of reports by constructing of TSV (Tab-Separated-Values) files is less recommended than the JSON-formatted data and serves as a kind of a “shortcut”, if willing to submit multiple entries (up to the 100 limit) of an AT or Classification report in a less flexible way.

Why less flexible? Because in a TSV list, each entry spans only a single row, so e.g. only one photometry point, that of the discovery, can be specified for an AT report, or only a single spectrum for a classification report.

Also note that a TSV report can hold an AT report OR classification report, but not both.

Sections 5.2 and 5.3 below show the columns of the TSV lists; example files are available for download on the Bulk report page as mentioned in Section 3.2 (stage 3 of the bulk page).

6.2. URL: <https://www.wis-tns.org/api/csv-report>

Sandbox URL: <https://sandbox.wis-tns.org/api/csv-report>

6.3. Data description

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

6.4. Upon a successful submission, a reply JSON is sent back with the "report_id" identifying the report that was just submitted. All details described in section 6.4 hold here as well.

6.5. AT Report TSV format

- Columns for which specifying a value is mandatory are marked with ‘*’.
- All fields containing preset values (e.g. groupid, at_type, instrument_id...) must include the appropriate value id. See section 6.5 for details.

A	B	C	D	E	F	G	H	I	J	K
report type: at										
RA*	DEC*	reporting_group_id*	discovery_data_source_id*	reporter/s*	discovery_datetime*	at_type*	host_name	host_redshif	internal_nan	remarks
			(groupid of the disc. data source)		(=photom. Obsdate)	(1=PSN)	(if known)	(if known)		
20:30.0	+20:30:40.05	2	2	J. Smith, on behalf of SurveyName...	2020-03-01.234	1	NGC 1234	NULL	NULL	NULL

L	M	N	O	P	Q	R	S	T	U	V	W
proprietary_period			Last non-detection				Non-detection archival info				
groups (comma delim.)	value	units	obsdate	lim_flux	flux_units	filter_id	instrument_id	exptime	observer	archive_id	archival_info_remarks
1,2	7	days	2020-02-28.	21.5	1	50	103	60	Robot	NULL	NULL

X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
photometry (discovery)							rel-file1	rel-file2			
obsdate*	flux*	flux_units*	filter_id*	instrument_i	exptime	observer	name	comments	name	comments	
2020-03-01.7	19.5	1	50	103	60	Robot	rel_file_1.pn	Finding Char	rel_file_2.jpg	Discovery image...	

6.6. Classification Report TSV format

A	B	C	D	E	F	G	H	I
report type: class								
name*	classifier*	objtypeid*	redshift	groupid*	associated_groups	remarks	class_proprietary_period	
				(source group)	(comma delim.)		value	units
2020aayt	K. Smith, on behalf of SurveyName	3	0.123	2	1,2	Another la	0	days

J	K	L	M	N	O	P	Q
spectrum							
obsdate*	instrument_i	exptime	observer*	reducer	ascii_file*	fits_file	remarks
2016-03-07.8	1	600	J. Smith	K. Smith	spectra_exar	spectra_exar	NULL

T	U	V	W
rel-file1		rel-file2	
name	comments	name	comments
rel_file_1.pn	SNID Classification plot...	NULL	NULL

7. Feedback messages

7.1. General

The feedback messages obtained from the Bulk API are similar to those obtained on the interactive AT and Classification report forms. The 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.

7.2. URL: <https://www.wis-tns.org/api/get/bulk-report-reply>
Sandbox URL: <https://sandbox.wis-tns.org/api/get/bulk-report-reply>

7.3. Data description

- Type: POST
- Mandatory parameters:
 - `api_key` – the bot's API Key
 - `User-Agent` – the bot's `tns_marker`
 - `report_id` – holds the report-id provided from the report submission interface.
- See the `bulk_report.py` sample code, `reply` function.

7.4. 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, <code>internal_name</code>) 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