CONSUMER PRODUCT SAFETY COMMISSION

United States
Consumer Product Safety Commission

International Trade Data System/Risk Assessment Methodology (ITDS/RAM)

HTS Management Application

Application Programming Interface (API) Documentation

July 13, 2018

SENSITIVE SECURITY INFORMATION

THE ATTACHED MATERIALS CONTAIN CPSC INFORMATION THAT IS SENSITIVE, FOR OFFICIAL USE ONLY, OR OTHER TYPES OF SENSITIVE BUT UNCLASSIFIED INFORMATION REQUIRING PROTECTION AGAINST UNAUTHORIZED DISCLOSURE. THE ATTACHED MATERIALS MUST BE HANDLED AND SAFEGUARDED IN ACCORDANCE WITH CPSC MANAGEMENT DIRECTIVES OR POLICIES GOVERNING PROTECTION AND DISSEMINATION OF SUCH INFORMATION.

Document Information

Document Version and History			
Version	Date	Author	Comments
0.1	7/13/2018	Dennis Wallick	First Draft for Review

Contents

Introduction

The HTS Management Application Programming Interface (API) is provided to software developers who access the Harmonized Tariff Schedule (HTS) codes to identify, track, and update the HTS code changes that are maintained by the International Trade Commission (ITC). The API provides the ability to:

- make changes to targeting (i.e., entry filter) and jurisdiction of codes;
- override the ITC descriptions and provide for agency-specific descriptions to codes if needed;
- run various reports of HTS codes, and
- publish the approved targeted and jurisdictional codes to a database.

This document provides details of how to use the API through calls to RESTful endpoints. Complete documentation of each endpoint is provided, along with sequence diagrams of typical logical paths to use the API.

Several data source are used by this API:

Current HTS Data. The data that currently resides in a database while the API is not in use. The API will compare data in the Current HTS Data with data that has been modified by the API through either individual record edits or new ITC data. Current HTS data is only changed when the *finalize* endpoint is called.

Scratch Table. A table in the database that used by the API to store the changes to the HTS data prior to finalization. These changes can come from single record edits, changes from the ITC data, or both.

ITC Data File. A CSV file from the USITC website that follows the format and standards outlined in the ITC document "hts_external_guide," section 1.4, "RESTful API," available on the ITC website. The API provides the ability to upload this file, compare it to the Current HTS Data, and identify all differences.

Application States and State Transition

The application can exists in one of four global states.

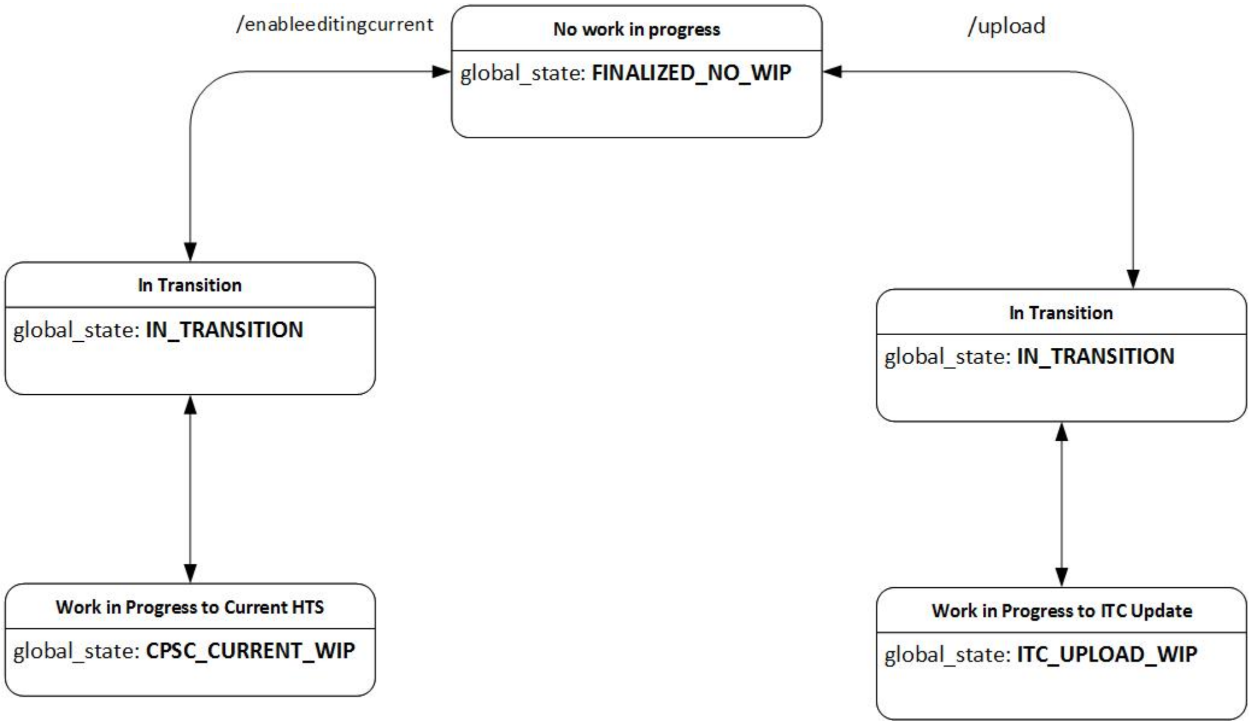
1) **No work in progress.** This state exists only when the user opens the application and no ITC data file has been uploaded and the user has not enabled editing. The API identifies this state as FINALIZED_NO_WIP.

2) **Work in Progress to Current HTS.** This is the state that allows edits to data but an ITC data file has not been uploaded. The API identifies this state as CPSC_CURRENT_WIP.

3) **Work in Progress to ITC Update.** This is the state that allows edits to data and an ITC data file has been uploaded. The API identifies this state as ITC_UPLOAD_WIP.

4) **Transition.** This is a state that exists while transitioning between states. The user cannot invoke this state. The API identifies this state as IN_TRANSITION.

The diagram below details the interactions between these states. See Global State and Endpoints for complete details on allowed endpoints during different states.


/enableeditingcurrent
No work in progress
global_state: FINALIZED_NO_WIP
/upload
In Transition
global_state: IN_TRANSITION
In Transition
global_state: IN_TRANSITION
Work in Progress to Current HTS
global_state: CPSC_CURRENT_WIP
Work in Progress to ITC Update
global_state: ITC_UPLOAD_WIP

Rest Endpoints

The fourteen endpoints in the Application are summarized here with technical details following:

Endpoint	Purpose
State	Returns the current state of the data as a JSON structure.
Counts	Provides counts of HTS codes by type as a JSON structure.
Get	Returns data based on the parameters supplied. Returns JSON structures or CSV files
EnableEditingCurrent	Changes the state to allow editing of records in Current mode. Returns a JSON structure reporting the new state.
Upload	Performs an upload an ITC file. Returns a JSON structure reporting the new state.
UploadProgress	After an ITC file upload is underway, provides information on the progress and prevents other actions from occurring until the upload process completes. Returns a JSON structure reporting the status of the upload.
Save	Saves changes to an individual record to the scratch table. The record change does not persists to the production HTS data table until *finalize*. Returns a JSON structure reporting the results of the endpoint call.
Finalize	Publishes all changes in the scratch table to the production HTS table. Returns a JSON structure reporting the results of the endpoint call.
Revert	Cancels all changes made through *Save* and *Upload* endpoints. Returns a JSON structure reporting the results of the endpoint call.
Reset	Restores the data to a consistent state after encountering a non-recoverable error. Returns a JSON structure reporting the results of the endpoint call.
AppendixBSummary	Returns the summary data for the Appendix B report as a CSV file.
AppendixBJurisdiction	Returns the detailed data for the Appendix B report as a CSV file.
CBP-CPSC-EntryFilter	Returns the Entry Filter HTS Codes for CBP as a CSV file.
CBP-CPSC-Jurisdiction	Returns the Jurisdiction HTS Codes for CBP as a CSV file.
Login	Returns SSO information (if enabled) or challenges user with login/password
Login/sso	Returns the username of the logged in SSO user
Login/ssoenabled	Returns true if user is logged in to SSO, false if not.
session	Returns a session key defining the logged in user session.

Global State and Endpoints

This table documents which endpoints may be called based on the application state. Calling an invalid endpoint will generate an error. The four CSV-generating endpoints behave the same as the *get* endpoint.

GlobalState	Valid Endpoints	Invalid Endpoints
FINALIZE_NO_WIP	/enableeditingcurrent	/save
	/counts	/finalize
	/get	/revert
	/upload	/uploadprogress
	/state	
CPSC_CURRENT_WIP	/save	/upload
	/finalize	/uploadprogress
	/revert	/enableeditingcurrent
	/counts	
	/get	
	/state	
ITC_UPLOAD_WIP	/save	/upload
	/finalize	/uploadprogress
	/revert	/enableeditingcurrent
	/counts	
	/get	
	/state	
IN_TRANSITION	/uploadprogress	/save
	/state	/finalize
		/revert
		/reports
		/counts
		/get
		/enableeditingcurrent
		/upload

State

Returns which of the four states the API is in.

- **URL**

  /state

- **Method:**

  GET

- **JSON Structure:**

Name	Possible Values and Use
username	The name written to the USERNAME column of the scratch table. It has 4 possible values, each corresponding to 1 of the 4 global states. For any one global state, there can be only one username. The administrator can override these values at runtime. Their defaults are: • For global state CPSC_CURRENT_WIP: TEST_CURR • For global state ITC_UPLOAD_WIP: TEST_UPLD • For global state FINALIZED_NO_WIP: NO_FI_USR • For global state IN_TRANSITION: NO_TR_USR Example: "username": "NO_FI_USR"
globalState	Which of the 4 possible global states the HTS Services application is currently in. These 4 values cannot be overridden. Their meanings are: • CPSC_CURRENT_WIP: There is a work in progress, and it originated from editing the current production codes. • ITC_UPLOAD_WIP: There is a work in progress, and it originated from an upload of a USITC spreadsheet. • FINALIZED_NO_WIP: There is no work in progress. • IN_TRANSITION: Some endpoints of the HTS Services application are momentarily unavailable while the system transitions from one global state to another. Example: "globalState": "FINALIZED_NO_WIP"
stateTransition	Provides the most precise status into the internal state of the HTS Services application. There are 12 possible values, which cannot be overridden: ENABLE_START ENABLE_END UPLOAD_START UPLOAD_END FINALIZE_START FINALIZE_END

	REVERT_START REVERT_END SAVE_START SAVE_END
transitioningNow	Boolean indicating if state transition is underway. True = transition underway; false = transition not underway.
persistenceStatus	This key-value pair is specific to tracking the progress of an ongoing upload. It has 4 possible values, which cannot be overridden. Their meanings are: • INITIALIZED: No upload has occurred (or been in progress) since the last time the state endpoint was hit. • SCANNING: An upload is underway, and it is in the first of its 2 stages. In this stage, no data has yet been written to the scratch table. • UNDERWAY: An upload is underway, and it is in the second of its 2 stages. In this stage, data is being written to the scratch table. • SUCCEEDED: An upload has successfully completed since the last time the state endpoint was hit. Example: "persistenceStatus": "INITIALIZED"
persistenceProgressRemark	This key-value pair is specific to tracking the progress of an ongoing upload. Its value is a human-readable String explaining the corresponding persistenceStatus.
percentage	This key-value pair is specific to tracking the progress of an ongoing upload. Its value is an integer from 0 to 100 (inclusive) showing what percentage of the HTS codes in the uploaded USITC spreadsheet have been written to the scratch table. Its value remains at 0 during the SCANNING stage (shown by "persistenceStatus": "SCANNING"), and later is reset to 0 at the same time that the persistenceStatus is reset to INITIALIZED. (Therefore it is useful only duringthe UNDERWAY stage.) Example: "percentage": 42
progressInRecordsPersistedToScratchTable	This key-value pair is specific to tracking the progress of an ongoing upload. Its value is the numerator when calculating the percentage value. When percentage is 0 or 100, the special value "not applicable" is shown; otherwise, it is the number of records that have been inserted into the scratch table so far. Example: "progressInRecordsPersistedToScratchTable": "not applicable"
progressGoalInRecords	This key-value pair is specific to tracking the progress of an ongoing upload. Its value is the denominator when calculating the percentage value. When percentage is 0 or 100, the special value "not applicable" is shown; otherwise, it is the total number of records that must be inserted into the scratch table during this upload. Example: "progressGoalInRecords": "not applicable"
progressLastCalculated	This key-value pair is specific to tracking the progress of an ongoing upload. Its value is the time when the most recent upload process underwent its last progress calculation. In the special case where no

<table>
<tr><td></td><td>upload has occurred since the time when the HTS Services application was restarted, the special value "Progress not yet calculated." is shown. Example: "progressLastCalculated": "2018-07-13 20:18:24"</td></tr>
<tr><td>timestamp</td><td>The time when this request for state information was fulfilled.</td></tr>
<tr><td>lastUpload</td><td>This key has a value consisting of nested key-value pairs, all of which are specific to the outcome of the most recent upload attempt. Their values are taken from the record in table CPSC_HTS_MGMT_HIST that was written at the conclusion of that attempt (regardless of whether it succeeded or failed).
<table>
<tr><td>timestamp</td><td>The time when the most recent upload attempt concluded.</td></tr>
<tr><td>result</td><td>Two possible values: "success" or "failure". Result is "success" IFF incidentCode is "OK".</td></tr>
<tr><td>incidentCode</td><td>Selected from the enumerated type CpscHtsMgmtCodeEnum, but only a subset of those values are used in reporting upload outcomes. The values currently in use are
• OK
• EXCEPTION_PARSING_LINE
• EXCEPTION_PARSING_INDENT
• EXCEPTION_INSERTING_TO_SCRATCH_TABLE
• SAW_RESET_FLAG
• EXCEPTION_READING_FILE
• EXCEPTION_WRITING_TO_FILE_TABLE
OK represents success; all others are accompanied by the sibling key-value pair "result": "failure"</td></tr>
<tr><td>username</td><td>The end-user login name.</td></tr>
<tr><td>notes</td><td>A human-readable String explaining the corresponding incidentCode.</td></tr>
</table></td></tr>
</table>

- **JSON Response Example:**

```
{
"username": "NO_FI_USR",
"globalState": "FINALIZED_NO_WIP",
"stateTransition": "REVERT_END",
"transitioningNow": "false",
"persistenceStatus": "INITIALIZED",
"persistenceProgressRemark": "No persistence has occurred since last system restart.",
"percentage": 0,
"progressInRecordsPersistedToScratchTable": "not applicable",
"progressGoalInRecords": "not applicable",
"progressLastCalculated": "Progress not yet calculated.",
"timestamp": "2018-07-16 15:53:38",
"lastUpload":
        {
        "timestamp": "2018-07-13 15:48:37.113",
        "result": "failure",
        "incidentCode": "EXCEPTION_INSERTING_TO_SCRATCH_TABLE",
        "username": "TBD",
        "notes": "repository error: 1000 records, range 01 - 0304530015"
        }
}
```

Counts

Provides count of all code, jurisdiction codes, and targeted codes. Counts are reported based on the state; in current state, counts will be based on the production data. For work in progress, counts will include changes from the upload (if applicable) and changes made through save.

- **URL**

  /counts

- **Method:**

  `GET`

- **JSON Response Structure:**

Name	Possible Values and Use
refActiveTotal	Zero, or positive integer representing the total number of ten-digit HTS codes in the data.
refActiveTotalJurisdictionTrue	Zero, or positive integer representing the total number of ten-digit HTS codes in the data that have jurisdiction set to true.
refActiveTotalTargetedTrue	Zero, or positive integer representing the total number of ten-digit HTS codes in the data that have targeted set to true.

- **JSON Response Example:**

```
{
"refActiveTotal": "39565",
"refActiveTotalJurisdictionTrue": "9695",
"refActiveTotalTargetedTrue": "294"
}
```

Get

Returns data to the user based on the parameters provided.

- **URL**

  /get

- **Method:**

  GET

- **URL Params**

The Get endpoint uses ten different URL parameters to retrieve information from the data and return it to the UI.

Endpoint Parameter	Use
page	Used for pagination size min=1;
pagesize	Used for pagination size min=0 (disable). Indicates number of records in a page.
root	Defaults to empty string; Indicates root of the entire tree. Can identify root of data to be returned.
mincodelength	Defaults to 2, which is the shortest code length, can be up to 10;
maxdepth	Defaults to 5, which is the depth of 10-digit codes
includelongdesc	Defaults to false. Allows Concatenated description to be returned.
source	FINALIZED_NO_WIP CPSC_CURRENT_WIP ITC_UPLOAD_WIP DIFFS_FROM_UPLOAD CURRENT
filter	Defaults to "no_filter" If included, defines the filter term.
searchterm	If included, applies a search term to the get.
format	Csvflat – for flatted (ten-digit) report exports as CSV Csvtree for CSV exports of all codes Json –For all other data

- **JSON Response Structure:**

Name	Possible Values and Use
Header:	
username	System name chosen from enumerated type in code.
globalState	Will return one of the four global states: • ITC_UPLOAD_WIP (Upload Work is in progress) • CPSC_CURRENT_WIP (CPSC work underway, but no ITC upload) • FINALIZED_NO_WIP (no uploads, no changes) • IN_TRANSITION (application transitioning between states)
source	5 possible values: • ITC_UPLOAD_WIP • CPSC_CURRENT_WIP • FINALIZED_NO_WIP • CURRENT -- Allows pull of FINALIZED_NO_WIP data if that is not the global_state • DIFFS_FROM_UPLOAD – defines differences in the ITC_UPLOAD_WIP from FINALIZED_NO_WIP data
filter	no_filter (default); jurisdiction; targeted
searchterm	If the data returned came from a search, returns the term.
recordCount	Total number of records returned
htsView	Indicates the start of the HTS data.
Data:	
Children	Indicates the start of an array of HTS data
htsCode	The HTS code
htsCodeType	Returns the position in the HTS structure for the code 2,4,6,8 or 10
htsDescription	Short description of the code's description. Will return the CPSC description if one exists, otherwise returns the ITC description.
isItcDescription	True indicates the htsDescription is the ITC description; false indicates htsDescription is the CPSC descrption.
cpscDescription	The full, concatenated description of the code's description. Only returned if includelongdesc is set to true in URL parameter.
jurisdiction	True or false, indicating if the code is within jurisdiction.
targeted	True or false, indicating if the code is targeted.
modified	True or false, indicating if the code has been modified.
sunset	True or false, indicating if the code no longer exists in the ITC data, but had previously been under jurisdiction and targeted within 100 days of the code's elimination.
changeStatus	None, add, remove
notes	Notes added to record through save endpoint

- **JSON Response Example:**

```
{
"username": "NO_FI_USR",
"globalState": "FINALIZED_NO_WIP",
"source": "CURRENT",
 "filter": "no_filter",
"searchterm": "0601101500",
"recordCount": "5",
"htsView":
[ { "data" : {
"htsCode": "06",
"htsCodeType": "2",
"htsDescription":
"Live trees and other plants; bulbs, roots and the like; cut flowers and
ornamental foliage",
"isItcDescription": "true",
"jurisdiction": "false",
"targeted": "false",
"modified": "false",
"sunset": "false",
"changeStatus": "none",
"notes":
""},
"children": ...
"children": [
{ "data" :
{ "htsCode": "0601101500",
"htsCodeType": "10",
"htsDescription": "Tulip bulbs",
"isItcDescription": "true",
"jurisdiction": "false",
"targeted": "false",
"modified": "false",
"sunset": "false",
"changeStatus":
"none", "notes":
""
}
```

- **Notes**

Also see these endpoints: CBP-CPSC-EntryFilter; CBP-CPSC-Jursidiction ;AppendixBJurisdiction, and AppendixBSummary.

These endpoints detail uses by the Consumer Product Safety Commission (CPSC) of the get endpoint for specializes reporting purposes. These reports may be useful to other agencies.

Upload

Takes the uploaded ITC file and loads to into the scratch table. The file must follow the specifications outlined in the ITC document “hts_external_guide,” section 1.4, “RESTful API,” available on the ITC website.

- **URL**

  / upload

- **Method:**

  ```
  POST
  ```

- **Payload Structure:**

  Multipart/mime file

  ITC File Structure

Parameter	Use
HTS Number	The ITC HTS code
Indent	1,2,3,4 or 5, indicating the indentation and location of the code within its parent structure
Description	The ITC HTS code’s description
Unit of Quantity	Not used by the API.
General Rate of Duty	Not used by the API.
Special Rate of Duty	Not used by the API.
Column 2 Rate of Duty	Not used by the API.

- **JSON Response Example:**

  ```
  {"result": "Upload and import now underway"}
  ```

Upload Progress

After upload, shows progress of the processing of the uploaded file. When completed, the *state* endpoint will return ITC_UPLOAD_WIP. State also reports this information during *uploadprogress*.

- **URL**

  /uploadprogress

- **Method:**

  `GET`

- **JSON Response Structure:**

Name	Possible Values and Use

persistenceStatus	INITIALIZED=Upload has not started yet
	SCANNING=Row scanning is underway
	UNDERWAY=Upload is underway
	SUCCEEDED=Upload has completed

persistenceProgressRemark	Summary of information of status.
percentage	% completion.
ProgressInRecordsPersistedToScratchTable	#records persisted to scratch table.
progressGoalInRecords	Total number of #records in upload
progressLastCalculated	Datetime stamp structure was returned

- **JSON Response Example:**

```
{
"persistenceStatus": "SCANNING",
"persistenceProgressRemark": "Row scanning is underway. Rows scanned:
33384. Persistence will begin once all rows are scanned.",
"percentage": 0,
"progressInRecordsPersistedToScratchTable": "not applicable",
"progressGoalInRecords": "not applicable",
"progressLastCalculated": "not applicable",
" progressLastCalculated ": "2018-07-09 17:16:30"
}
```

Enable Editing Current

Allows the user to edit and save changes from either Upload or in Current Mode. When completed, the *state* endpoint will return CPSC_CURRENT_WIP

- **URL**

  /enableeditingcurrent

- **Method:**

  ```
  GET
  ```

- **JSON Response Example:**

```
{
"editingCurrentIsEnabled": "true",
"globalState": "CPSC_CURRENT_WIP"
}
```

Save

Saves a single record as an edit to the data.   Saved changes persist in a scratch table until *finalize* or *revert*.

- **URL**

  /save

- **Method:**

  ```
  POST
  ```

- **Payload Structure:**

Parameter	Use
htsCode	An existing HTS record being edited. Must already be in database with a change status of “add” or “none.”
description	The description to save for the code.
jurisdiction	True or false if the code is in jurisdiction.
targeted	True or false if the code is targetd.
notes	Any user notes
changeStatus	Can be “add” or “none” Cannot alter a record with change status of “remove.”

- **Payload Example**

```
{
"htsCode":"2203000090",
"description":"In containers each holding over 4 liters or quarts",
"jurisdiction":true,
"targeted":false,
"notes":"This is a note",
"changeStatus":"none"
}
```

- **JSON Response Example:**

```
{"count": 1}
```

Finalize

When user clicks “Finalize” button, finalizes all records from the ITC upload or current edit with the user changes applied to the production table which persists after finalize.

- **URL**

  /finalize

- **Method:**

  ```
  GET
  ```

- **JSON Response Example:**

  ```
  {
  "remark": "Finalization into lookup table is complete.",
  "recordCount": "39086"
  }
  ```

Revert

Reverts the application back to current state. All user changes through *Upload* or *Save* are lost.

- **URL**

  /revert

- **Method:**

  ```
  GET
  ```

- **JSON Response Example:**

  ```
  {
  "remark": "Reversion to previous finalization is complete. Work in
  progress has been discarded.",
  "recordCount": "39527"
  }
  ```

Reset

Allow a *finalize* to be terminated

- **URL**

  /reset

- **Method:**

  `GET`

- **JSON Response Example:**

  ```
  {
  "resetOccurred": "true",
  "recordsDeletedFromScratch": "41221",
  "haltForReset": "false",
  "timestamp": "2018-07-09 17:20:48"
  }
  ```

AppendixBSummary

Returns a CSV file consisting of three lines: 1) a header; 2) Added HTS codes, and 3) removed HTS codes

- **URL**

  /appendixbsummary

- **Method:**

  GET

- **URL Params**

  SOURCE

- **Data Params**

  CURRENT – returns the data from the production data (ignores Work in Progress if any)
  ITC_UPLOAD_WIP or CPSC_CURRENT_WIP – returns data from Work in Progress,

- **Response:**

  CSV file.

- **Example CSV File:**

```
HTSReport_AppendixB_Summary
Added HTS Codes "8415820110"
Removed HTS Codes,"8502110000","8504230041",
```

AppendixBJurisdiction

Returns a CSV file consisting of all codes within jurisdiction.

- **URL**

  /appendixbjurisdiction

  - **Method:**

    GET

  - **URL Params**

    SOURCE

  - **Data Params**

    CURRENT – returns the data from the production data (ignores Work in Progress if any)
    ITC_UPLOAD_WIP or CPSC_CURRENT_WIP – returns data from Work in Progress, or error message if state is CURRENT.

  - **Response:**

    CSV file

  - **Example CSV File:**

```
HTS Code
"1704903550"
"2207106010"
"2207106090"
"2207200010"
"2207200090"
"2712100000"
```

CBP CPSC Entry Filter Report

Returns a CSV file consisting of all codes within the entry filter.

- **URL**

  / CBP-CPSC-EntryFilter

  - **Method:**

    GET

  - **URL Params**

    SOURCE

  - **Data Params**

  - CURRENT – returns the data from the production data (ignores Work in Progress if any)
  - ITC_UPLOAD_WIP or CPSC_CURRENT_WIP – returns data from Work in Progress, or error message if state is CURRENT.

  - **Response:**

    CSV file

  - **Example CSV File:**

```
HTS Code
"1704903550"
"2207106010"
"2207106090"
"2207200010"
"2207200090"
```

CBP CPSC Jurisdiction

Provides the detailed HTS codes for the CBP Jurisdiction as a CSV file. Each HTS codes is returned as a single line of data. CBP CPSC Jurisdiction differs from the AppendixBJursidiction end point in that it will include codes which have a "sunset" entry filter provision of an end date in the future (calculated at 100 days from the date of finalization). Sunset codes are calculated automatically by an ITC Upload for a targeted code is has been eliminated by ITC.

- **URL**

  / CBP-CPSC-Jurisdiction

  - **Method:**

    GET

  - **URL Params**

    SOURCE

  - **Data Params**

    CURRENT – returns the data from the production data (ignores Work in Progress if any)
    ITC_UPLOAD_WIP or CPSC_CURRENT_WIP – returns data from Work in Progress, or error message if state is CURRENT.

  - **Response:**

    CSV file

  - **Example CSV File:**

```
HTS Code
"1704903550"
"2207106010"
"2207106090"
```

Login

Redirects user to CPSC SSO. If the user is logged in via SSO, returns authorization information. If the user is not logged in via SSO, challenges user with login/password dialog from SSO.

- **URL**

  /login

- **Method:**

  `GET`

- **JSON Response Example:**

```
{

      "details":
            {
            "remoteAddress":"IPAddress",
            "sessionId":"fWz_pE8_U0txTxYqPlZXfDahr6n12347PLHMhauC"},
            "authorities":
            [     {"authority":" ROLE_VALUE_2"},
                  {"authority":"ROLE_VALUE_2"},
                  {"authority":"ROLE_VALUE_N"}
            ],
            "accountNonExpired":true,
            "accountNonLocked":true,
            "credentialsNonExpired":true,
            "enabled":true},
            "credentials":null,
            "name":"username"
}
```

Login/ssoenabled

Returns true if user is logged in to SSO, false if not

- **URL**

  /loginssoenabled

- **Method:**

  ```
  GET
  ```

- **Response Example:**

  ```
  true
  ```

Login/sso

Returns the username of the logged in SSO user.

- **URL**

  /loginsso

- **Method:**

  ```
  GET
  ```

- **Response Example:**

  ```
  "yourusername"
  ```

session

Returns a session key defining the logged in user session

- **URL**

  /session

- **Method:**

  `GET`

- **JSON Response Example:**

```
{"session": "YZb412342pwvqnYPLa0RjZssE3qK5nIgMtUnb9i8",
"username": username}
```

Sequence Diagrams

The four sequence diagrams below show scenarios of how the API can be used with an application. The sequences display a hypothetical user application calling the endpoints in a logical fashion.

In the first two examples, the process begins with opening the application and proceed to the user changing the global state, changing data and returning data. The process concludes with Finalization.

The last two examples show how an application can be used for 1) revert, where the user has decided to abandon changes made through /save and /upload, and 2) in a view only scenario, where the global state remains FINALIZED_NO_WIP.

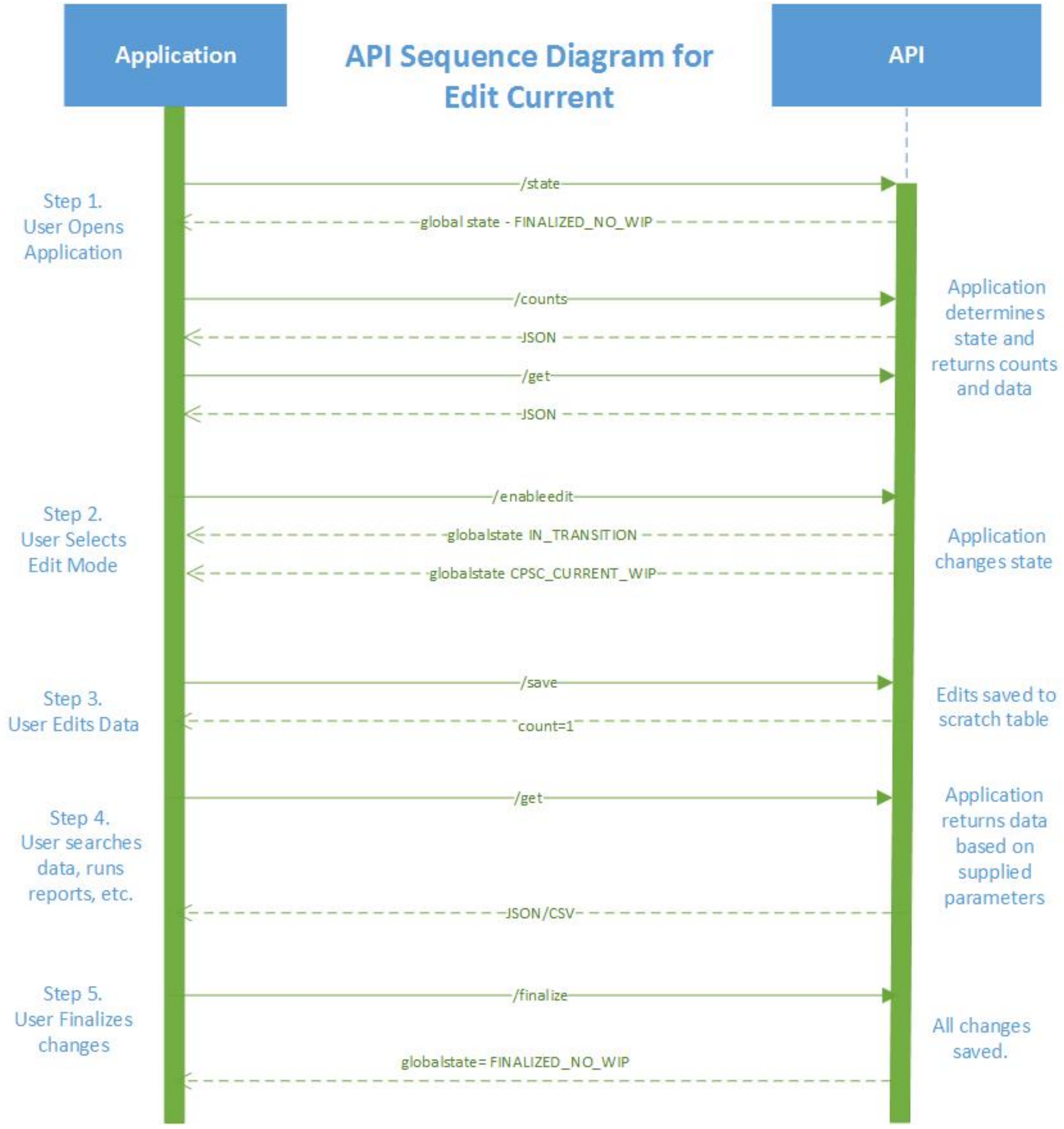
Edit Current

In this scenario, the API is called (step 1) returning state. Data is returned through get and counts.

Data change can begin by calling the *enableedit* endpoint (step 2), which returns the new global state.

Following *enableedit*, the both data edits and data retrieval searches take place with the *save* (Step 3) and *get* (step 4) endpoints. These steps can be performed repeatedly and in any order.

The process completes when the *finalize* endpoint (step 5) is called.

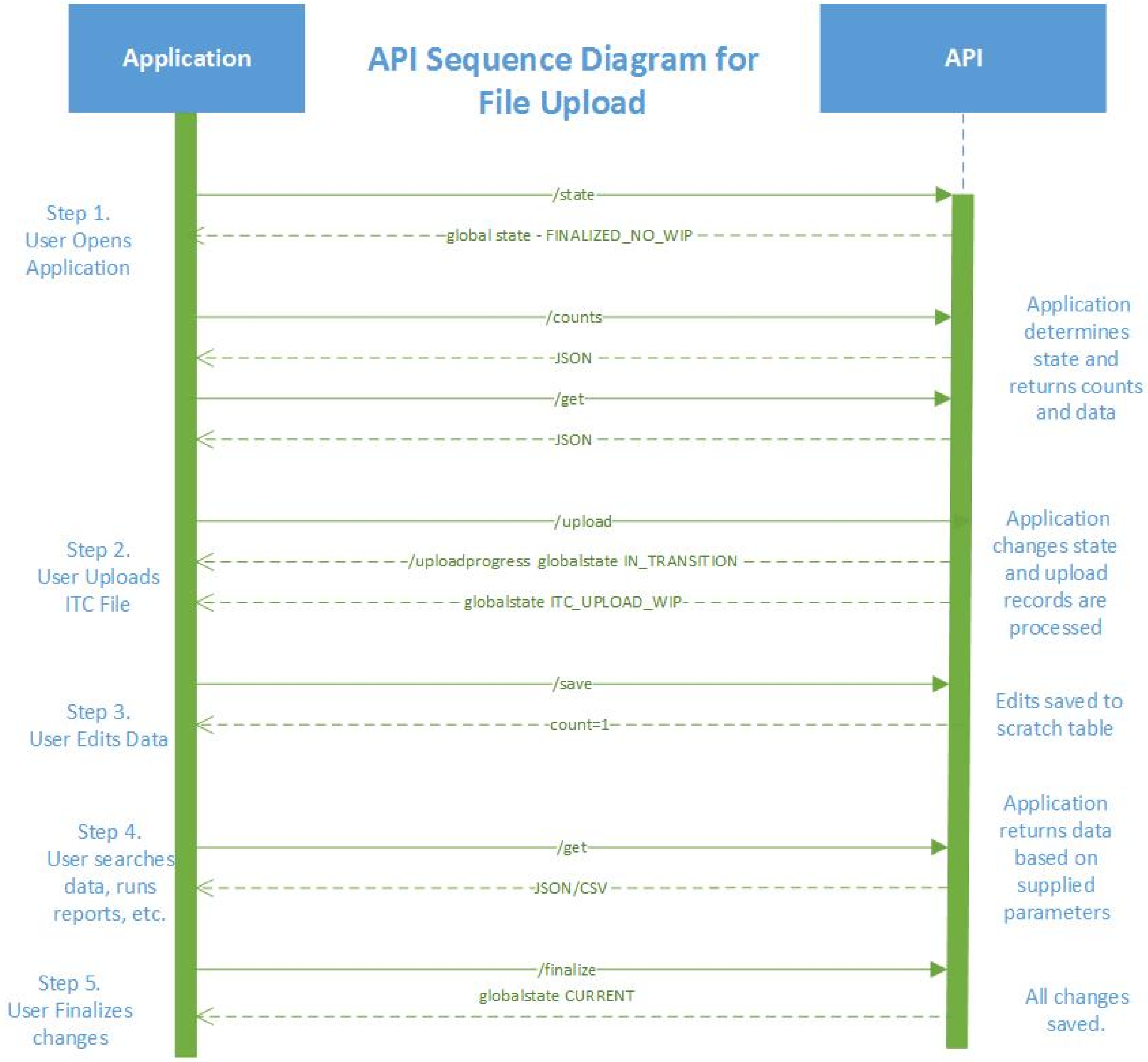

Application
API Sequence Diagram for Edit Current
API
/state
global state - FINALIZED_NO_WIP
Step 1. User Opens Application
/counts
JSON
/get
JSON
Application determines state and returns counts and data
/enableedit
globalstate IN_TRANSITION
globalstate CPSC_CURRENT_WIP
Step 2. User Selects Edit Mode
Application changes state
/save
count=1
Step 3. User Edits Data
Edits saved to scratch table
/get
JSON/CSV
Step 4. User searches data, runs reports, etc.
Application returns data based on supplied parameters
/finalize
globalstate= FINALIZED_NO_WIP
Step 5. User Finalizes changes
All changes saved.

File Upload

This scenario is identical to Edit Current, except that for Step 2 the *upload* endpoint is invoked instead of *enableedit*.

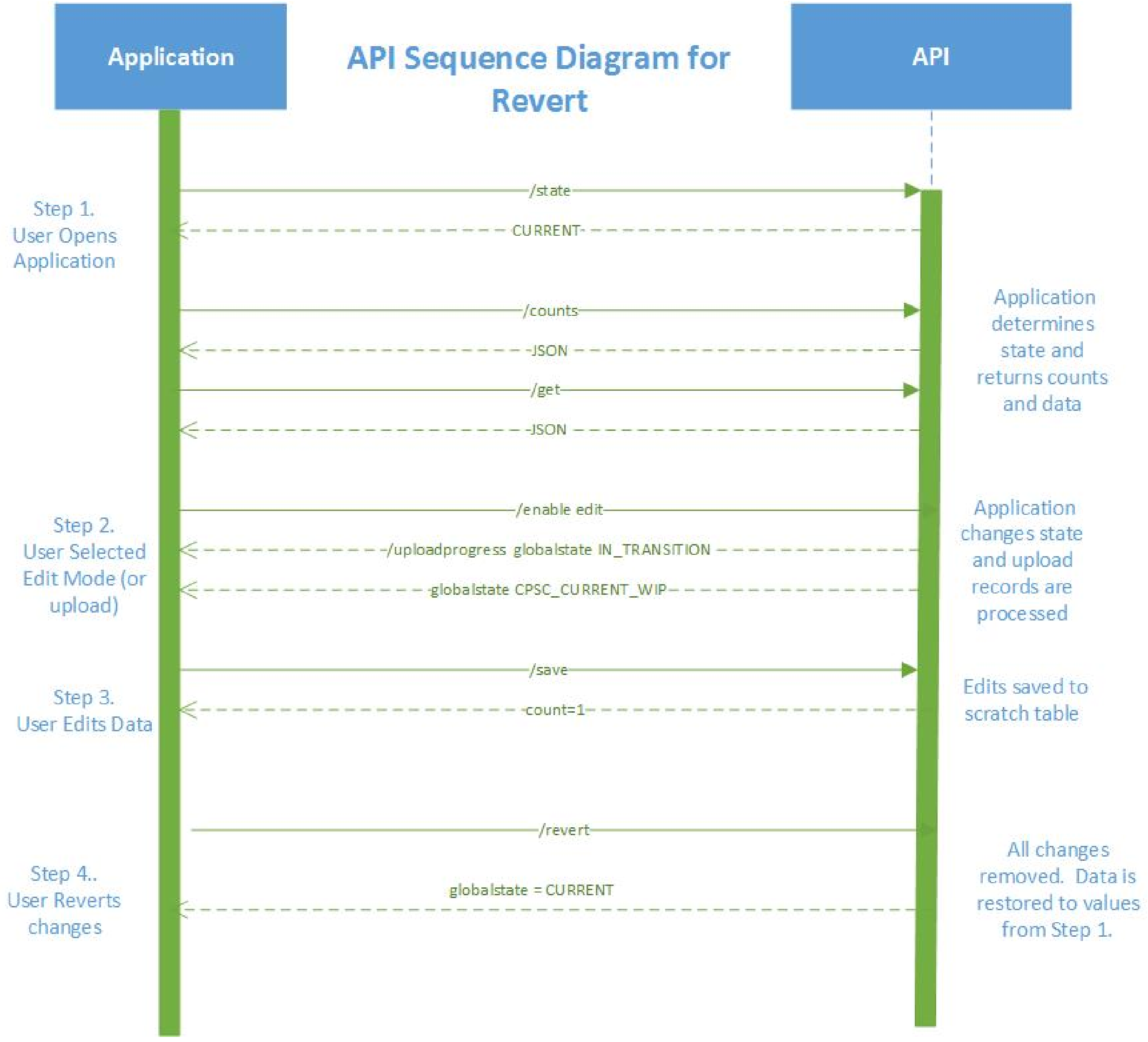
This allows the application to have the API compare a new ITC file with the current data and display the differences.

As before, both edit data and data retrieval searches take place with the *save* (Step 3) and *get* (step 4) endpoints, and conclude with *finalize* (step 5).


Application
API Sequence Diagram for File Upload
API
Step 1. User Opens Application
/state
global state - FINALIZED_NO_WIP
/counts
JSON
/get
JSON
Application determines state and returns counts and data
Step 2. User Uploads ITC File
/upload
/uploadprogress globalstate IN_TRANSITION
globalstate ITC_UPLOAD_WIP
Application changes state and upload records are processed
Step 3. User Edits Data
/save
count=1
Edits saved to scratch table
Step 4. User searches data, runs reports, etc.
/get
JSON/CSV
Application returns data based on supplied parameters
Step 5. User Finalizes changes
/finalize
globalstate CURRENT
All changes saved.

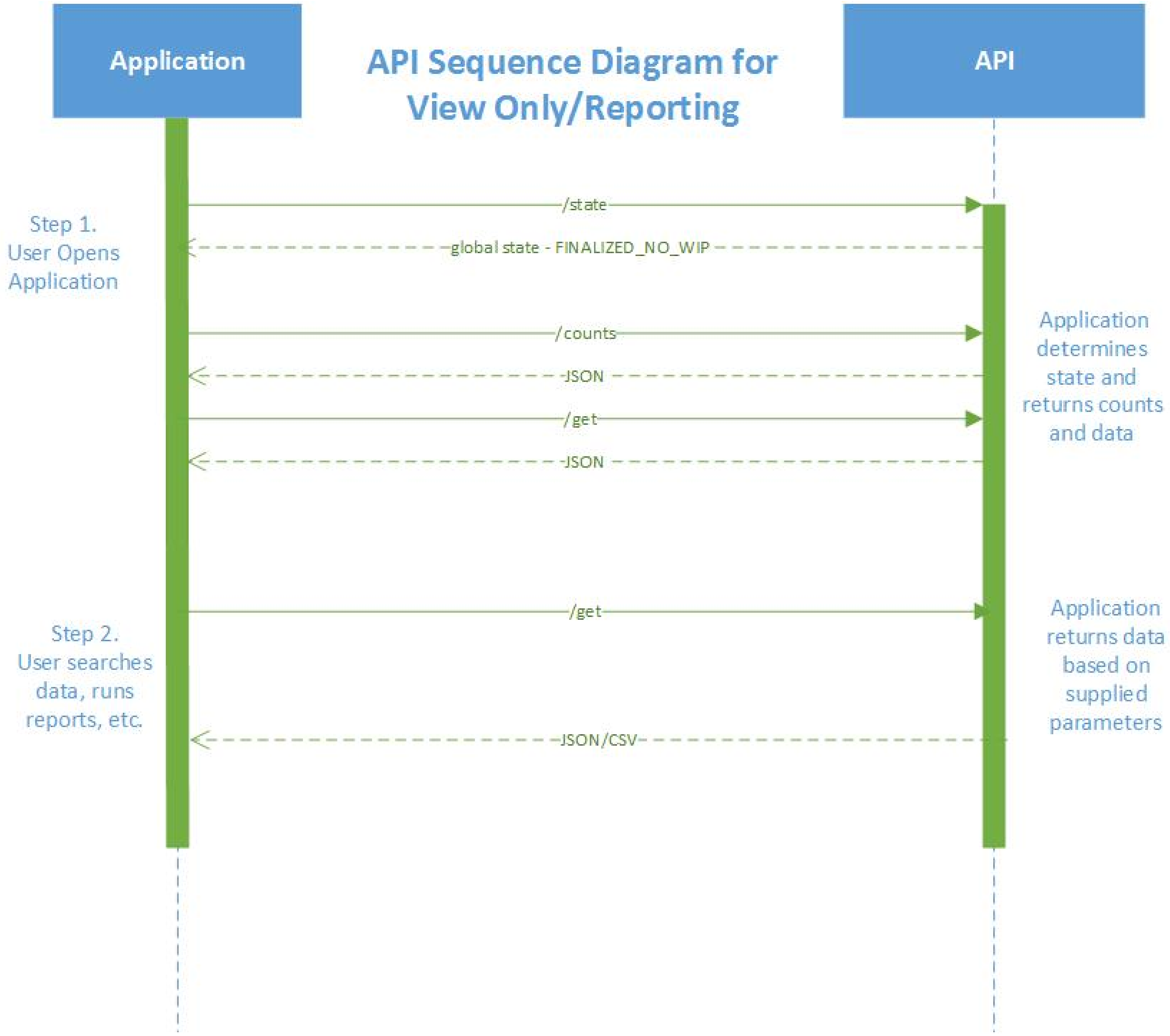
Revert

This scenario is identical to Edit Current, except that for Step 5, instead of calling the *finalize* endpoint, *revert* is invoked. This will return the data to where it was in step 1, regardless of any changes performed through *save*. The *revert* endpoint will also abandon any data modifications that may have been identified through the *upload* process as well.


Application
API Sequence Diagram for Revert
API
/state
CURRENT
Step 1.
User Opens
Application
/counts
JSON
/get
JSON
Application determines state and returns counts and data
/enable edit
/uploadprogress globalstate IN_TRANSITION
globalstate CPSC_CURRENT_WIP
Step 2.
User Selected Edit Mode (or upload)
Application changes state and upload records are processed
/save
count=1
Step 3.
User Edits Data
Edits saved to scratch table
/revert
globalstate = CURRENT
Step 4..
User Reverts changes
All changes removed. Data is restored to values from Step 1.

View Only

In this scenario, the API uses the *get* endpoint for reports and searches. Because neither *upload* or *enableedit* are called, the global state does not change from FINALIZE_NO_WIP and therefore *finalize* or *revert* do not need to be called.


Application
API Sequence Diagram for View Only/Reporting
API
/state
global state - FINALIZED_NO_WIP
Step 1.
User Opens
Application
/counts
JSON
/get
JSON
Application
determines
state and
returns counts
and data
/get
Step 2.
User searches
data, runs
reports, etc.
JSON/CSV
Application
returns data
based on
supplied
parameters