Distr.: General
19 November 2020

Original: English only

Economic Commission for Europe

Executive Committee

Centre for Trade Facilitation and Electronic Business

UN/LOCODE Advisory Group
Fourth annual meeting
Geneva (online), 27 November 2020
Item 6 of the provisional agenda
UN/LOCODE Re-engineering Project

Application Programming Interface (API) Specification for UN/LOCODE Re-engineering Project (URP)

Summary

The United Nations Code for Trade and Transport Locations (UN/LOCODE) is an international standard to identify locations for international trade and transport in information exchange between stakeholders as an alternative to the full name of the location spelt in different languages.

The main objective of the UN/LOCODE Re-engineering Project is to develop a new ICT system to meet new business needs with modern technologies. Application Programming Interface (API) is in the scope of this project because it allows the exchange of data in real-time between the UN/LOCODE system and the business systems of external stakeholders.

This document provides the API specification between the new UN/LOCODE system and the stakeholders' business systems.

Contents

I. Introduction

A. Project background

1. The United Nations Code for Trade and Transport Locations (UN/LOCODE) is a five-character code system that provides a coded representation for the names of transport-related locations which are used for the movement of goods for trade. It is specified in UNECE Recommendation 16. Considering that the resulting webpages and subsequent downloads account for over 80% of the total visits on the UNECE website, UN/LOCODE is a "flagship" product of UNECE.

2. Following the decision made at the second annual meeting of the UN/LOCODE Advisory Group in October 2018 in Hangzhou, China, the UN/LOCODE Re-engineering Project was kicked off in July 2019 with the support of the China National Institute of Standardization (CNIS).

3. The Business Requirement Analysis report was presented and approved at the third annual meeting of the UN/LOCODE Advisory Group in November 2019 in Xiamen, China. The project team was requested to move forward to develop the new system.

4. Based on the decision made at the third annual meeting of the UN/LOCODE Advisory Group, the secretariat shared the first draft of the UN/LOCODE API specification prepared by the International Maritime Organization (IMO) with the UN/CEFACT experts, especially the project leader of two UN/CEFACT projects: RDM2API and API Town Plan, for comments. An online meeting was held at the end of April 2020 to discuss the draft, which enlightened the secretariat to clarify the specification to guide the API development of the new UN/LOCODE system.

B. Document purpose

5. Application Programming Interface (API) is a software intermediary to allow two applications to talk to each other. API is in the scope of the new UN/LOCODE system development. APIs are the key to building new digital channels between the new UN/LOCODE system and other business systems used by the UN/LOCODE business community to facilitate internal and external data exchange.

6. The document aims to place the understanding of the UN/LOCODE API on the same page and to facilitate the developers to implement the API between their business system and the new UN/LOCODE system.

C. Audience

7. The project team, members of the Steering Committee of the project, developers of the UN/LOCODE stakeholders' in-house business systems which want to exchange real-time data with the new UN/LOCODE system.

II. API strategy

A. Current situation analysis

8. If we need to answer the following questions:

- Why do we need an API?

- Why would our users want to use our API?
- What benefit does it offer them?

We have to analyse the current situation on the use of UN/LOCODE in the business community and the best practice in the industry now.

9. Currently, the UNECE secretariat publishes the UN/LOCODE directory free of charge on the UNECE website twice per year. Traditionally, many UN/LOCODE users download the directory in their preferred format and update their UN/LOCODE data in their business systems. On the one hand, this update is done usually in a manual way and time-consuming. On the other hand, the update is not in a real-time mode because approval of a UN/LOCODE Data Maintenance Request (DMR) is not shared with the business community until the release of the UN/LOCODE directory on the UNECE website.

B. Business objective

10. The UN/LOCODE business community expects the API data services provided by the new UN/LOCODE system, which enable them to query UN/LOCODE data in a real-time mode instead of waiting for the bi-annual release. In this way, we could reduce the cost and improve the data quality of the UN/LOCODE data synchronization. The new system will support APIs which allow the external business system to exchange data in real-time. The business community would align their data to any changes from UN and pointed at the UN official release rather than hosting their own.

11. From the user perspective, it can be supported in

- Pull mode

The secretariat will provide a live service to business systems to query UN/LOCODE. The use case is that “I want to query the current code list”. This is historically not a service provided by the secretariat. A big challenge is caused by the service level commitment (24/7, 99.99% uptime, average response within 2 seconds, etc), which means a kind of high availability service supported by a new operating model and considerable resources.

- Push mode

The user case is that “tell me when something has changed”, for example, it is approved to add a new entry or to modify an existing entry in the UN/LOCODE directory. By using an event subscription model, a subscriber can be informed of any changes triggered by the Data Maintenance Request (DMR) validation.

12. As stated in the approved Business Requirement Analysis report of the UN/LOCODE Re-engineering Project, the secretariat continues to support the current free-of-charge services to the UN/LOCODE business community with the new system. We assume that some stakeholders continue to maintain their own data sets in their business system in a traditional way so that we continue to release the UN/LOCODE directory twice per year and the release number identified in each directory is needed for reference. Besides the directory published in different formats (.html, .txt, .mdb and .cvs) on the UNECE website, some stakeholders practised to publish the UN/LOCODE directory on the GitHub repository. In this way, updates can be pushed out to subscribers automatically and they will be notified to automate the updating of their own UN/LOCODE application.

13. The design of the UN/LOCODE API is driven by the customer demands to automatically update their internal system through electronic data exchange with the new UN/LOCODE system rather than a calendar reminder to check the official

publication on the website twice a year, especially after submitting a DMR for a new UN/LOCODE and seeing which status it is up to in the DMR life cycle.

C. Service model

14. The UN policy is to make UN/LOCODE accessible as freely as possible. The UNECE secretariat has limits to balance the demands and the IT infrastructure resources required to provide these services. Therefore, we plan to design and develop the UN/LOCODE API in an incremental way to implement web services for essential queries based on the following criteria:

- how many business systems will access the API?
- how often the API is called?
- What is the workload on the UN server-side to respond to the API?
- how crucial is the service for the stakeholder?
- 24*7?
- Up to 99% availability?

III. API design

A. Design principles

15. The development of the UN/LOCODE API is driven by a predefined specification that has been carefully tested and evaluated by potential API users.

16. As mentioned above, two ways to implement APIs are:

- Publishing the code list in a structured JSON format
- Opening an API for general consumption, i.e., offering a service that application builders will embed into their application for a dynamic query.

17. After thinking about the balance point between demands and recourses, equipment and performance, and the costs for the IT infrastructure, we will support Representational State Transfer (REST) APIs. The API users send a request to a resource stored on the server and the server responds with the requested information in JSON structure.

18. The design of the API aims to have:

- Consistent and expandable data structure
- Clear design and documentation
- High performance data interface
- Secure authentication system

B. Web service type

19. The main use case is that “I want to query the current UN/LOCODE data”. In the new system, the UN/LOCODE data is structured from two perspectives:

- UN/LOCODE directory
- Data Maintenance Request (DMR)

20. Many data elements are overlapped in the directory and DMR, such as metadata: location name, subdivision, function and coordinates. We need to view special data elements from a distinct perspective. For example, ‘Status’ in the directory means the UN/LOCODE creditability, in other words, who approved this UN/LOCODE. ‘Status’ in DMR means its status in the DMR life cycle, like “Drafted”, “Submitted”, “FP Review”, “MT Review”, “Approved” and “Rejected”.

21. Therefore, we can classify the queries as follows:

- Directory query service
- DMR status service

22. Version 1 (v.1) of the API will only support the queries mentioned above and other web services might be added in the next API version via the procedure agreed in this specification.

C. API versioning

23. Version 1 (v.1) of the API only provides GET functions based on the supported use case. For example, the proposal from IMO is to support:

- Get the port details by a UN/LOCODE
- Get a list of ports by a country/territory
- Get a change history of a single port
- Query a validation status of a DMR

It includes versioning in the URL, such as https://unlocode.unece.org/api/v1/resource, to version the API.

D. Method

24. Base URL: https://unlocode.unece.org/api

- GET /

To get a list of methods available by the API, including the description and the example of the returned data of each method.

- GET /v1/locodes/detail/{UNLOCODE}

To get the details of a UN/LOCODE entry in the UN/LOCODE directory.

- GET /v1/locodes/list/{COUNTRY_CODE}

To get a list of UN/LOCODEs for a requested two-alphabet code of country or territory, based on 3166-1.

- GET /v1/locodes/history/{UNLOCODE}

To get a modification history of a UN/LOCODE entry.

- Get /v1/dmr/{DMR_REQUESTID}

To get the current status of a DMR by a Request ID.

E. JSON data format

25. JSON data format of the result returned by each method is as follows:

- GET/

URL: https://unlocode.unece.org/api/

Parameter: None

Returned result:

Field	Type	Description
code	Integer	Status identifier code returned by the API. Please refer to Annex II.
message	String	Message returned by the API.
data	Object	Data object returned by the API, as defined as follows:
list	Object Array	Fields in the data object, which is a list of methods available by the API.
name	String	A field in the list: the name of the method.
url	String	A field in the list: the URL of the method.
description		A field in the list: the description of the method.
returnDataExample	String	A field in the list: the example of JSON data object returned by the method.

Example result returned by the method：

```
{
   code:200,
   message:”success”,
   data:{
         list:[
               {
                name:”Get port details by a UN/LOCODE”,
                url:”https://unlocode.unece.org/api/v1/locodes/detail/{UNLOCODE}”,
                description:”Get port details by a UN/LOCODE”,
                returnDataExample:”{country:’’,unlocode:’’,name:’’,functions:’’}”
              },
                {
                 name:”Get a list of ports by a country/territory”,
                 url:”https://unlocode.unece.org/api/v1/locodes/list/{COUNTRY_CODE}”,
                 description:”Get a list of ports by a country/territory”,
                 returnDataExample:”[{country:’’,unlocode:’’,name:’’,functions:’’}]”
                }
                ......
         ]
   }
}
```

- GET /v1/locodes/detail/{UNLOCODE}

URL: https://unlocode.unece.org/api/v1/locodes/detail/{UNLOCODE}

Parameter: {UNLOCODE}, a five-character UN/LOCODE

Returned result:

Field	Type	Description
code	Integer	Status identifier code returned by the API.
message	String	Message returned by the API.

Field	Type	Description
data	Object	Data object returned by the API, as defined as follows, in line with the UN/LOCODE directory:
country	String	A field of the data object: the country/territory name based on ISO 3166-1
unlocode	String	A field of the data object: the five-character UN/LOCODE
subdivision	String	A field of the data object: the code of the subdivision
name	String	A field of the data object: the location name
namewodiacritics	String	A field of the data object: the location name without diacritics
cordinatesLat	String	A field of the data object: the latitude in a format of ddmmN/S
cordinatesLng	String	A field of the data object: the longitude in a format of dddmmE/W
functions	String	A field in the data object: the function(s) of the location
status	String	A field of the data object: the status of the UN/LOCODE entry to show its creditability
date	String	A field of the data object: the publishing data
remarks	String	A field of the data object: the remarks

Example result returned:

```
{
   code:200,
   message:”success”,
   data:{
         country:”China”,
        unlocode:”CNPEK”,
        subdivision:”10”,
        name:”Beijing”,
        cordinatesLat:”3925N”
        cordinatesLng:”13607E”
        functions:”1------”,
        status:”AI”,
        date:”9601”,
        remarks:””
   }
}
```

- GET /v1/locodes/list/{COUNTRY_CODE}

  URL: https://unlocode.unece.org/api/v1/locodes/list/{COUNTRY_CODE}

  Parameter: {COUNTRY_CODE}, two-alphabet code based on ISO3166-1

  Returned result:

Field	Type	Description
code	Integer	Status identifier code returned by the API.
message	String	Message returned by the API.
data	Object	Data object returned by the API, as defined as follows:
count	Integer	A field of the data object: the number of UN/LOCODE entries returned by the API
list	Object Array	A field of the data object: the list of UN/LOCODE entries
country	String	A field of the list: the country/territory name based on ISO 3166-1
unlocode	String	A field of the list: the five-character UN/LOCODE
subdivision	String	A field of the list: the code of the subdivision based on ISO 3166-2
name	String	A field of the list: the location name
namewodiacritics	String	A field of the list: the location name without diacritics
cordinatesLat	String	A field of the list: the latitude in a format of ddmmN/S
cordinatesLng	String	A field of the list: the longitude in a format of dddmmE/W
functions	String	A field of the list: the function(s) of the location
status	String	A field of the list: the status of the UN/LOCODE entry to show its creditability
date	String	A field of the list: the publishing date
remarks	String	A field of the list: the remarks

Example result:

```
{
   code:200,
   message:”success”,
   data:{
         count:2000,
         List:[
         {
          country:”China”,
         unlocode:”CNPEK”,
         subdivision:”10”,
         name:”Beijing”,
         cordinatesLat:”3925N”,
         cordinatesLng:”13607E”,
         functions:”1------”,
         status:”AI”,
         date:”9601”,
         remarks:””
```

},
{
country:”China”,
unlocode:”CNLPO”,
subdivision:”10”,
name:”Lingbo”,
cordinatesLat:”3925N”,
cordinatesLng:”13608E”,
functions:”1------”,
status:”AI”,
date:”9601”,
remarks:””
}
.....
]
}
}

- GET /v1/locodes/history/{UNLOCODE}

URL: https://unlocode.unece.org/api/v1/locodes/history/{UNLOCODE}

Parameter: {UNLOCODE}, a five-character UN/LCOODE

Returned result:

Field	**Type**	**Description**
code	Integer	Status identifier code returned by the API.
message	String	Message returned by the API.
data	Object	Data object returned by the API, as defined as follows:
count	Integer	A field in the data object: the amount of the DMRs related to the requested UN/LOCODE
list	Object Array	A field in the data object, the list of the DMRs related to the requested UN/LOCODE
country	String	A field in the list: the country/territory name based on ISO 3166-2
subdivision	String	A field in the list: the subdivision code based on ISO 3166-2
unlocode	String	A field in the list: the five-character UN/LOCODE
name	String	A field in the list: the location name
cordinatesLat	String	A field in the list: the latitude in a format of ddmmN/S
cordinatesLng	String	A field in the list: the longitude in a format of dddmmE/W
functions	String	A field in the list: the function(s) of the location
status	String	A field in the list: the status of the UN/LOCODE entry to show its creditability

Field	Type	Description
date	String	A field of the list: the publishing date
remarks	String	A field of the list: the remarks
submitter	String	submitter of the DMR
DMRtype	String	A field in the list: the type of DMR, "+": a DMR for a new UN/LOCODE "I": a DMR to change other attributes of the UN/LOCODE than the location name "#": a DMR to change the location name and other attributes (optional) "X": a DMR to delete the UN/LOCODE entry

Example result:


```
{
   code:200,
   message:"success",
   data:{
        count:2000,
        List:[
        {
         country:"China",
        unlocode:"CNPEK",
        subdivision:"10",
        name:"Beijing",
        cordinatesLat: "3925N"
        cordinatesLng: "13607E",
        functions:"1234---",
        status:"AI",
        date:"9601",
        remarks:"",
        submitter:"abc"
        DMRtype: "|"
        },
        {
         country:"China",
        unlocode:"CNPEK",
        subdivision:"10",
        name:"Beijing",
        cordinatesLng: "3925N",
        cordinatesLat:"13607E",
        functions:"1------",
        status:"AI",
        date:"9601",
        remarks:"",
        submitter: "bca"
        DMRtype:"+"
        }
        .....
        ]
   }
}
```


- Get /v1/dmr/{DMR_REQUESTID}

URL:https://unlocode.unece.org/api/v1/dmr/{DMR_REQUESTID}

Parameter: {DMR_REQUESTID}, unique key of DMR

Example result:

Field	Type	Description
code	Integer	The status identifier code returned by the API.
message	String	Message returned by the API.
data	Object	Data object returned by the API, as defined as follows:
requestId	String	A field in the data object: the Request ID of the DMR
country	String	A field in the data object: the two-alphabet country code based on ISO 3166-2
subdivision	String	A field in the data object: the subdivision code based on ISO 3166-2
unlocode	String	A field in the data object: the five-character UN/LOCODE
name	String	A field in the data object: the location name
functions	String	A field in the data object: the function(s) of the location
submitDate	Date	A field in the data object: the date to submit the DMR
submitter	String	A field in the data object: the submitter of the DMR, whose name is registered in the UN/LOCODE system
DMRstatus	String	A field of the data object: the status of the DMR validation, such as submitted, FP Review, MT Review, Rejected, Approved.

Example result:

```
{
   code:200,
   message:”success”,
   data:{
         requestId:”UN-2020-00102”,
        country:”CN”,
        subdivision:”10”,
        unlocode:”CNPEK”,
        name:”Beijing”,
        functions:”1-3----”,
        submitDate:”10/10/2020”,
        submitter:”duanxh”,
        DMRstatus:”MT Review”,
   }
}
```

IV. API management

A. API authentication

26. The API can be used not only for acquiring, but also submission. However, the audience for Put, Post and Delete is small, while the audience for Get is huge. As mentioned above, version 1 of the API only supports Get.

27. An API key is a unique identifier used to identify the calling application for usage and potential billing purposes. The REST Authentication is implemented via HTTP BASIC Authentication; When calling the API, the authentication data is put in Header, i.e., Basic base64(apiKey).

28. An API Key management interface will be provided to manage API keys. It also allows the API users to view their access/ error log.

29. If necessary, the billing model could be available to calculate based on usage. For example, it is free for the first 100 queries while it will be charged for queries beyond the threshold. The billing model is out of the scope of this specification and subject to the approval of the UN/LOCODE Advisory Group.

B. API security

30. All the published API methods could be managed and configured for security, such as available status, limits on calls, a quota of free queries for a single user, and limit on queries per day for a single user.

31. The API calls need to be recorded for further analysis. Abnormality and failure of all API calls should be analysed to correct the API and improve services.

32. We also need to monitor the performance of all API calls, record the response time of each call, analyse requests with a long response time, and make corresponding improvements.

C. New services

33. If necessary, the stakeholders in the UN/LOCODE business community should submit a use case based on the template in Annex I to request a new service through API. The use case will be analysed upon submission to the secretariat. If it is approved by the UN/LOCODE Advisory Group, the new service will be implemented in the next API version.

V. Further development

34. Version 2 (v.2) of the API might provide more services, such as other methods to facilitate DMR submission through other business systems.

35. What is more, we could study feasibility of using GraphQL which allows more complex queries. For example, when we want to get a list of all new seaports created in the past two years, we can get it instead of filtering the qualified seaports from a complete list returned by the API.

Annex I. API use case template

API Use Case Name
<Brief description. Usually, a paragraph or less.>

Industry
<What industry does this use case service?>

Actors
<A list of the Actors who communicate with this Use Case>

Priority
<What version will this API be implemented?>

Pre-Conditions
<A list of conditions that must be true before the Use Case starts>

Post-Conditions
<A list of conditions that must be true when the Use Case ends, no matter which Scenario is executed.>

Extension Points
<If the Use Case has extension points, list them here.>

“Used” Use Cases
<If the Use Case uses other Use Cases, list them here.>

Activity Diagram
<An activity diagram of the flow of events, or some significant or complex part of the flow of events.>

Scenarios
<They should at least be listed here but may also include a brief description.>

Sequence Diagrams
<If you do not have separate documents for Scenarios, you might include sequence diagrams for them here.>

Subordinate Use Cases
<If the Use Case has subordinate Use Cases, show them here. Or you could include a Use Case diagram for the subordinate Use Cases. Or both. Also tell what subsystem is responsible for this subordinate Use Case.>

Annex II. API Status Identifier Code

Code	Message
200	OK
400	Bad Request
401	Unauthorized
404	Not Found
500	Internal Server Error / Program Exception
800	API key does not exist / disabled
801	API key reaches the maximum number of requests today
802	Request data parsing failed
803	Request parameter format error
804	The requested data was not found