Validation Service API

Technical Specification *Validation Service API*

Author:	***R. Schmitz / T. Hoffmann / D. Bialke***
Department:	***Global Product Delivery***
Version:	***1.3.4***
Last Update:	***2021-10-05***

Terms of use, disclaimer

Imprint:

atrify GmbH
Maarweg 165, 50825 Cologne
T +49 221 93373 0 **F** +49 221 93373 199 info@atrify.com
Represented by Jochen Moll

Responsible for the content: Lars Schickner, atrify GmbH, Maarweg 165, 50825 Köln
Commercial register: Cologne Local Court HRB 45457
Sales tax ID: Sales tax identification number according to §27a sales tax law: DE22 4602462
Regulatory Affairs: Local Court Cologne

Table of Contents

1 General Information

1.1 Goal of Document

This document serves as a technical description of the atrify Validation Service API, a RESTful Webservice API used to validate GDSN Trade Items against the current set of atrify validation rules.

1.2 API design

The validation service will take any nested CIN hierarchy, validating it against the desired ruleset, and returning a validation status result and a list of error messages.

The API follows an asynchronous design, i.e. the CIN hierarchy will be sent in one POST request, but the result will only contain an id, which then can be used in further GET requests to obtain the status and (once the request has been processed) the validation result itself. After retrieving the result, it will be removed.

The result of a validation will be a code “valid” or “invalid”, as well as an array of messages. The messages array will be empty if the validation result is “valid”.

Each message will contain a “code” and “description” field. The description will contain the validation message in English language only.

A processed validation result will be available for at least 24 hours. Validation results older than 24 hours will become unavailable, i.e. they will be removed.

Communication happens via encrypted https protocol.

1.3 General workflow

1. Call validation service with a POST request, containing a well-formed GDSN CatalogueItemNotificationMessage as XML payload with one transaction and exactly one nested hierarchy. Retrieve the validation id as a result of your POST request.
2. Call validation service with a GET request using the validation id as path parameter. Look for the status code to become “done”.
3. Retrieve the validation result code (“valid”/”invalid”) and messages.

1.4 Restrictions of the Validation Service API

Please note that receiving a valid result from the Validation Service API does not guarantee that the validation result from the Data Sync Engine (Data Pool) will be valid as well. Main reason is that the API is not connected to the production database and has no possibility to check against existing items and/or the appropriate hierarchy tree. E.g. deviation checks with regard to the GTIN allocation rules (measures and weights), changes of core hierarchy related data (Case to Base GTIN swap) and TPD is not supported.

Further the optional scope parameter “FMCG” is allowed for target market Germany (Code: 276) only.

2 Authentication

2.1 API-Key

Authentication will be done using pre-shared API keys. Without API-Key, access to the service will not be allowed (HTTP 401).
API-Keys have a UUID format and must be placed in every request to the API using a “key” request parameter. They do not expire.

2.1.1 Request

Method	URL
`GET / POST`	`api/v1/validate&key=<your API key>`

Type	Params	Values
`REQUEST`	`key`	`uuid`

2.1.2 Response

Status	Response
`2xx`	**Response will be positive, only if a known and valid API key is used and the permission to the resource is granted for this API key. Response details depend on the request.**
`401`	`{"error":"Invalid API key."} // API-Key missing or invalid.`
`403`	`{"error":"Unauthorized."} // API-Key is valid, but the access is otherwise not allowed.`
`405`	`{"error":"Method not allowed."} // Only GET and POST are accepted request methods.`
`500`	`{"error":"Internal error."}`

3 Validation request

3.1 Create a validation request (POST)

Create a new validation request by uploading a nested CIN message.

3.1.1 Request

Method	URL
`POST`	`api/v1/validate/`

Type	Params	Values

`REQUEST`
`REQUEST`
`POST`
	`key`
	`scope`
	`payload`
		`uuid`
		`string`
		`application/xml`

key
The mandatory API `key` to identify the requesting service.

`scope`
Defines the set of validation rules to be executed against the provided item hierarchy. Possible values are “GDSN” and “FMCG”.
“GDSN” refers to the standard GDSN set of validation rules used also for DSE M2M messaging.
An alternative ruleset is “FMCG”, which includes community validations for fast moving consumer goods on top of the “GDSN” rules, i.e. “FMCG” includes “GDSN”. The “FMCG” ruleset is the same as used in WS|Publishing. Please note that FMCG will be supported with items having target market Germany (Code: 276) only. If you post a message with FMCG scope having any other target market the service will throw an error.

`payload`
The `payload` must contain a well-formed GDSN CatalogueItemNotificationMessage containing one single transaction, with one single, nested hierarchy.

Example
https://vs-api.atrify.com/v1/validate?scope=FMCG&key=cx22ex78-c8x4-42f4-97x0-8a5dxb5x8e0x

3.1.2 Response

Status	Response
`200`	**Response will be an object containing the validation id, status, and timestamp. An example response is:** `{ "id": "ae2697a2-5a1b-4838-adb0-e85abb9b43bf", "status": "in_progress", "timestamp": "2018-03-13T14:55:47+01:00" }`
`401`	`{"error":"Invalid API key."}`
`403`	`{"error":"Unauthorized."}`
`500`	`{"error":"Internal error."}`

3.2 Retrieve the validation result (GET)

After obtaining the validation request id, the current state of the validation can be obtained via GET request. As soon as the validation state is “done”, the validation result will be filled with data:

3.2.1 Request

Method	URL
`GET`	`api/v1/validate/<id>`

Type	Params	Values
`REQUEST`	`key`	`uuid`

key
The mandatory API `key` to identify the requesting service.

Example
https://vs-api.atrify.com/v1/validate/1590677f-ba29-4828-81b7-abd44d120ada?key=cx22ex78-c8x4-42f4-97x0-8a5dxb5x8e0x

3.2.2 Response

Status	Response
`200`	**If the status is “done”, the validation result will contain the complete validation report. A failed (invalid) validation may look like so:**

```
{
  "id": "ae2697a2-5a1b-4838-adb0-e85abb9b43bf",
  "status": "done",
  "timestamp": "2018-03-13T14:55:47+01:00",
  "validation": {
    "result": "invalid",
    "timestamp": "2018-03-13T14:55:49+01:00",
    "scope": "GDSN",
    "messages": [
      {
        "code": "31703",
        "description": "05011835104295/8711997000004/528:
GDSN Numeric Rule ID 1010: \"Is Trade Item A Despatch Unit\"
must be populated for the trade item.",
```

	`"severity": "error"`
	`},`
	`{`
	`"code": "31704",`
	`"description": "05011835104295/8711997000004/528: GDSN Numeric Rule ID 1011: \"Is Trade Item An Invoice Unit\" must be populated for the trade item.",`
	`"severity": "error"`
	`}`
	`]`
	`}`
	`}`

200	**An example for a successful (valid) validation is:**
	`{`
	`"id": "ae2697a2-5a1b-4838-adb0-e85abb9b43bf",`
	`"status": "done",`
	`"timestamp": "2018-03-13T14:55:47+01:00",`
	`"validation": {`
	`"result": "valid",`
	`"timestamp": "2018-03-13T14:55:49+01:00",`
	`"scope": "GDSN",`
	`"messages": []`
	`}`
	`}`

401	`{"error":"Invalid API key."}`
403	`{"error":"Unauthorized."}`
500	`{"error":"Internal error."}`

`id`
The `id` uniquely identifies the validation request and it's result. With it, the validation result and state can be retrieved at any time (for up to 24 hours).

`status`
The validation status will be **in_progress**, as long as the validation request has not been processed yet by the validation engine. It will change to **done**, as soon as it has been finished and the report becomes available.

`timestamp`
The ISO 8601 formatted time, when the request has been created.

`validation`
The validation data structure will contain the validation status and report, only if the overall

`status` is **done**. Until then, the validation structure will be empty, i.e. **null**.

`validation.result`
Can be “**valid**”, or “**invalid**”, depending on whether the validation found any issues with the provided item hierarchy or not. “**Invalid**” result should not be confused with an error condition. It just says that the provided CIN does not correspond to the selected validation ruleset for some reason.

`validation.timestamp`
This will contain the ISO 8601 formatted timestamp, when the validation was done.

`validation.scope`
The `scope`, which has been selected as a request parameter to the POST request. Currently, “GSDN” and “FMCG” are available.

`validation.messages`
If the validation result is “invalid”, the messages array will contain all validation problems found by the engine. If the result is “valid”, this array will be empty.

`validation.messages[i].code`
A validation message always consists of a code and description. This code field is an internal value that is associated with a given validation rule.

`validation.messages[i].description`
The validation error itself, usually contains all necessary information to track down the issue with an item. It will contain item key information, as well as the attribute names involved and a description what was expected, and why the error occurred.

`validation.messages[i].severity`
One of the following:

- fatal - indicates a problem on the level of the XML, such as a missing tag etc.
- error - indicates a validation error that needs to be fixed
- warning - indicates a potential problem that however does not prevent processing

History of Changes

No.	Date	Editor	Changes
1.0	2017	Daniel Bialke	Initial version
1.1	14.11.2018	Daniel Bialke	API-Key
1.2	10.07.2019		Change of company name.
1.3	28.10.2019	Selçuk Övüç	Chapter 1.4 added.
1.3.1	11.12.2019	Selçuk Övüç	Examples for POST and GET requests added in chapter 3.2.1 and 3.2.2. Also a remark added in chapter 1.2 that communication happens via encrypted https protocol.
1.3.2	22.02.2021	Selçuk Övüç	Chapter “Terms of use, disclaimer” added.
1.3.3	29.7.2021	Stefan Besling	Clarification of method usage and return codes.
1.3.4	05.10.2021	Selçuk Övüç	Chapter 1.4 updated regarding TPD exclusion.