Nasdaq Member Portal™

Short Code Management Application Programmer’s Interface Manual

Nordic
All content in this document is owned, or licensed, by Nasdaq, Inc. or its affiliates (‘Nasdaq’). Unauthorized use is prohibited without written permission of Nasdaq. While reasonable efforts have been made to ensure that the contents of this document are accurate, the document is provided strictly “as is”, and no warranties of accuracy are given concerning the contents of the information contained in this document, including any warranty that the document will be kept up to date. Nasdaq reserves the right to change details in this document without notice. To the extent permitted by law no liability (including liability to any person by reason of negligence) will be accepted by Nasdaq or its employees for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document.

Copyright © 2019 Nasdaq, Inc. All rights reserved.
Table of Contents

1  Summary of Changes ....................................................... 4
2  Overview ........................................................................ 5
   2.1  API Fundamentals ...................................................... 5
   2.1.1  Format .................................................................. 5
   2.2  Record Uniqueness ...................................................... 5
   2.3  Paging ...................................................................... 6
   2.4  Throttling ................................................................. 7
   2.5  Authorization ............................................................... 7
   2.6  URL Suffix ................................................................. 7
3  Endpoints ........................................................................ 8
   3.1  Identifiers ................................................................. 8
       3.1.1  POST & PUT /identifiers/ ....................................... 8
       3.1.2  GET /identifiers/ ................................................... 9
       3.1.3  GET /identifiers/missing ........................................ 11
   3.2  Response Codes and Error Messages ............................ 13
## Summary of Changes

Changes between this version and version a8 (1.0.1000).

<table>
<thead>
<tr>
<th>No</th>
<th>Changes</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Added Iceland Fixed Income exchange for Genium INET, table in section 2.2</td>
<td></td>
</tr>
</tbody>
</table>
2 Overview

The purpose of this manual is to describe the functionality of the Application Programmer’s Interface towards exchange members, customers, and Independent Software Vendors (ISV) for the Nasdaq short code management service, which is to be used for facilitating MiFID II Order Record Keeping.

2.1 API Fundamentals

The fundamentals of the Short Code Management API is described in this chapter.

2.1.1 Format

All data is in JSON format and the character encoding is in UTF-8 format. The client must support gzip and this must be indicated in all server requests: AcceptEncoding: gzip.

2.2 Record Uniqueness

If nothing else is stated, all services are keyed upon participant identities and defined as Exchange + Participant ID (MPID), hereafter referred to as participant. An API user can access short code information for one or several participants. The administration, both write and read, is done by defining the participant in a service call. Both the Exchange and Participant ID values are case sensitive and should be entered with all capital letters. To simplify the integration and on-boarding, all exposed objects contain all key values that define a short code and therefore compression is required, see chapter Format on page 5.

Note

The exchange code in the system is for Genium INET. Genium INET has a 2-letter Exchange code.

The constant INET is the Exchange code for INET Nordic, the Nasdaq trading platform for Nordic and Baltic equities trading.
Example 1:
Each row in the table below represents an autonomous short code space.

<table>
<thead>
<tr>
<th>Membership</th>
<th>Trading System</th>
<th>API MPID Value</th>
<th>API Exchange Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash Equities (all exchanges)</td>
<td>INET</td>
<td>XYZ</td>
<td>INET</td>
</tr>
<tr>
<td>Commodities</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>NC</td>
</tr>
<tr>
<td>Derivatives</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>SE</td>
</tr>
<tr>
<td>Cash Fixed Income Stockholm</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>ST</td>
</tr>
<tr>
<td>Cash Fixed Income Copenhagen</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>CO</td>
</tr>
<tr>
<td>Cash Fixed Income Helsinki</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>HE</td>
</tr>
<tr>
<td>Cash Fixed Income Tallinn</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>TA</td>
</tr>
<tr>
<td>Cash Fixed Income Riga</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>RI</td>
</tr>
<tr>
<td>Cash Fixed Income Vilnius</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>VI</td>
</tr>
<tr>
<td>Cash Fixed Income Iceland</td>
<td>Genium INET</td>
<td>XYZ</td>
<td>IS</td>
</tr>
</tbody>
</table>

2.3 Paging

The following attributes control the paging, segmentation, and mechanism for data sets and queries, where paging is available:

2.3.1 Offset is the first item received in an answer, where 0 is the first item in the set. If a larger offset than available items is supplied, an empty answer will be received.

2.3.2 Limit is the maximum number of items received in an answer. If unset, the server will use the system default. If the client sends a value larger than the max value supported by the system, the server may limit the number of items to a system default value.

2.3.3 Total attribute in the response provides the actual number of records available for the current search. For example, to determine if more values are available. The following logic should be applied: \[ \text{response.total} > \text{request.offset} + \text{number of actual records in response} \].

The proposed implementation pattern for integration is to query page by page, starting from offset 0 and until all items has been received as indicated in the total attribute. The total attribute may change in each response if the data set changes.

Note

It is strongly recommended to not map short codes so that the data set queried for is reduced while a paged query for missing short codes is ongoing.

All data sets returned by paged services are sorted according to a defined sort order, typically internal creation sequence or time stamp with purpose to keep distributed pages invariant, and will add new data in end of the sequence.
2.4 Throttling

All endpoints may have a limitation of how many requests that can be handled within a certain time frame. The limitations can be different depending on type of endpoint. Systems may have limitations per user, operation type, or globally. A client implementation should be able to handle the HTTP 429 error code, "Too many requests", for all endpoints. In such a case, the request is not handled and the client is advised to wait and try again later.

2.5 Authorization

All requests to the API must contain a valid authorization token associated with a Member Portal API user. Members can request an API user associated with their memberships in Member Portal.

The token is supplied on each request to the API as a standard HTTP authorization header of type Bearer:

Authorization: Bearer <token>

2.6 URL Prefix

URL Prefix for Production: https://memberportal.nasdaq.com/scregistry/api

URL Prefix for Test: https://memberportal-ext-test.nasdaq.com/scregistry/api
3 Endpoints

3.1 Identifiers

The management of the members’ short code repository is available in the endpoint `/identifiers`.

3.1.1 POST & PUT /identifiers

The endpoint `/identifiers` is used to upload or modify a single item. Both http POST and PUT is allowed.

HTTP verb POST is typically used to insert a short code mapping for the first time. PUT is used to update a previously registered mapping and it is also allowed for initial insert. The response 2xx indicates successful update and others responses, such as 4xx and 5xx, indicates that the item was not updated in the registry.

3.1.1.1 URL Suffix

/identifiers

3.1.1.2 Parameters

The item contains the following fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Format/Values</th>
<th>Mandatory</th>
<th>Modifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td>exchange</td>
<td>Alphanumeric value for Exchange identification</td>
<td>2 or 4 alphanumeric</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>mpid</td>
<td>Alphanumeric value for MPID</td>
<td>1..5 alphanumeric</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>codeType</td>
<td>Short code series</td>
<td>Client-Person Client-Entity InvestorDecisionMaker-Person InvestorDecisionMaker-Algo ExecutionDecisionMaker-Person ExecutionDecisionMaker-Algo</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>shortcode</td>
<td>Short code numeric value</td>
<td>Integer value 4 ... 4 294 967 295</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>beginDate</td>
<td>Code effective from date</td>
<td>YYYY-MM-DD</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>endDate</td>
<td>Absent means unset, valid forever</td>
<td>YYYY-MM-DD (≥ beginDate)</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>longcode</td>
<td>The private identifier value</td>
<td>If Codetype equals Client-Entity then exactly 20 characters are allowed. All other code types have a maximum of 50 characters.</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>alias</td>
<td>Convenience field for member arbitrary use</td>
<td>1 to 32 alphanumeric May only contain letters A-Ö (case insensitive) and digits and must start with a letter. Space is not allowed.</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>comment</td>
<td>Convenience field for member arbitrary use</td>
<td>1 to 32 alphanumeric</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
Note
The short code values 0-3 are pre-reserved.

3.1.1.3 Response
A response of a successful update or insert will not contain any body.

The endpoint handles POST and PUT verbs differently, where POST is recommended for initial upload and PUT for changes of already registered short code mappings.

The table below illustrates examples of return codes for different logical scenarios, specific for identifiers upload.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>POST</th>
<th>PUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not earlier mapped code</td>
<td>204</td>
<td>204</td>
</tr>
<tr>
<td>Mapped earlier, identical data</td>
<td>409</td>
<td>204</td>
</tr>
<tr>
<td>Mapped earlier, changed (allowed) values</td>
<td>409</td>
<td>204</td>
</tr>
<tr>
<td>In conflict with existing SC (beginDate)</td>
<td>409</td>
<td>409</td>
</tr>
</tbody>
</table>

Note
A response of a successful update or insert will not contain a body (for example, http 204)

A 409 (conflict) response will contain the key values of the object for which it is conflicting with (might be the same record supplied or another, similar, record with another beginDate). Please note that a response may or may not contain the complete object and attributes such as longcode, alias, or comments may be omitted.

Other http error codes may occur and for more generic errors such as format validations the body may contain an error message, see chapter Response Codes and Error Messages on page 13.

Example 2:
This example shows an example of a 409 body.

```json
{
    "exchange": "ST",
    "mpid": "XYZ",
    "codeType":"Client-Person",
    "shortcode": 110100,
    "beginDate": "2017-01-02"
}
```

3.1.2 GET /identifiers
The GET identifiers service provides functionality to retrieve identifiers for previously stored values.
3.1.2.1 URL suffix
/identifiers

3.1.2.2 Parameters
The following fields can be used to retrieve identifiers for previously stored values:

Table 4:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description/Value</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>exchange</td>
<td>Alphanumeric value for exchange identification</td>
<td>Y</td>
</tr>
<tr>
<td>mpid</td>
<td>Alphanumeric value for member ID</td>
<td>Y</td>
</tr>
</tbody>
</table>
| codeType| Client-Person
          Client-Entity
          InvestorDecisionMaker-Person
          InvestorDecisionMaker-Algo
          ExecutionDecisionMaker-Person
          ExecutionDecisionMaker-Algo | N         |
| shortcode| Short code numeric value                        | N         |
|         | Absence means wildcard                          |           |
| longcode| Longcode value                                  | N         |
|         | Absence means wildcard                          |           |
| date    | Optional filter parameter, filters mappings active only the given date| N |
|         | Absence means wildcard                          |           |
| offset  | Page offset, default is 0 if no value given     | N         |
| limit   | Page limit, defaults to system-defined value if not present | N |

Example 3:

https://.../identifiers?exchange=ST&mpid=XYZ&offset=0&limit=100

3.1.2.3 Response
The response is a structured object containing the total number of available items and the list itself.
### Example 4:

```json
{
  "total": 3,
  "result": [
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "Client-Entity",
      "shortcode": 110100,
      "beginDate": "2018-01-02",
      "endDate": "2018-12-26",
      "longcode": "549300L8X1Q78ERXFXXX",
      "alias": "Alias 1",
      "comment": "Comment 1"
    },
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "Client-Person",
      "shortcode": 110101,
      "beginDate": "2018-03-26",
      "endDate": "2018-12-01",
      "longcode": "SE197012111111",
      "alias": "Alias 2",
      "comment": "Comment 2"
    },
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "InvestorDecisionMaker-Algo",
      "shortcode": 110102,
      "beginDate": "2017-02-24",
      "endDate": "2017-07-11",
      "longcode": "ALGO BETA 2",
      "alias": "Alias 3",
      "comment": "Comment 3"
    }
  ]
}
```

#### 3.1.3 GET /identifiers/missing

The short codes and the reconciliation outcome for all members is available under the endpoint `identifiers/missing`.

This method will look up in any short code, used in the trading systems, for which long codes are currently missing. The data provided is updated close to real-time.

**Note**

Additional order activities may occur during the processing of the query. These order activities will not be included in the response.
3.1.3.1 URL suffix
identifiers/missing

3.1.3.2 Parameters
The following fields can be used to retrieve information:

Table 5:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description/Value</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>exchange</td>
<td>Alphanumeric value for exchange identification</td>
<td>Y</td>
</tr>
<tr>
<td>mpid</td>
<td>Alphanumeric value for member MPID</td>
<td>Y</td>
</tr>
<tr>
<td>codeType</td>
<td>Client-Person, Client-Entity, InvestorDecisionMaker-Person, InvestorDecisionMaker-Algo, ExecutionDecisionMaker-Person, ExecutionDecisionMaker-AlgoClient or DecisionMaker</td>
<td>N</td>
</tr>
<tr>
<td>offset</td>
<td>Default is 0 if no value given.</td>
<td>N</td>
</tr>
<tr>
<td>limit</td>
<td>Default is system defined limit if no value given.</td>
<td>N</td>
</tr>
</tbody>
</table>

Example 5:

https://.../identifiers/missing?exchange=ST&mpid=XYZ&offset=0&limit=100

3.1.3.3 Response
The response is a structured object containing the total number of available items and the list itself.
Example 6:

```json
{
  "total": 3,
  "result": [
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "-Person",
      "shortcode": 100123,
      "usedDate": "2017-02-17"
    },
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "InvestorDecisionMaker-Person",
      "shortcode": 110123,
      "usedDate": "2018-02-05"
    },
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType": "InvestorDecisionMaker-Person",
      "shortcode": 110124,
      "usedDate": "2018-02-06"
    }
  ]
}
```

Note

The attribute `usedDate` refers to the date on which the short code integer was observed in an order event. If a given short code is used on several different dates, each date will produce a new entry.

### 3.2 Response Codes and Error Messages

When a request returns a 4xx or 5xx HTTP status response code, the body may contain a collection of error objects. The object consists of an optional field property describing what object property the message refers to. The message property is a string describing the error that has occurred.

Example 7:

```json
[[
  "field": "longcode",
  "message": "Illegal format"
],
[
  "field": "comment",
  "message": "Value too long, max length 32"
]]
```
For certain conflicts, for example as described in chapter POST & PUT /identifiers/ on page 8, the body may instead contain an object of which item it is in conflict with.