Nasdaq Member Portal™

Short Code Management Application Programmer's Interface Manual

Nordic

Nasdaq Member Portal™ Version:	1.0.1000
Document Revision:	NasdaqMemberPortal_a10
Publish Date:	14 November 2017

Table of Contents

1 Summary of Changes

Changes between this version and version a8 (1.0.1000).

No	Changes	Comment
1	In Chapter "Overview," minor updates of subchapter Throttling on page 7.	
2	Subchapter "Suffix" rewritten and renamed URL Prefix on page 7.	
3	In Chapter Endpoints on page 8, changed all occurrences of **`identifiers/`** to **`/identifiers`**.	
4	In Chapter "Endpoints," in subchapter "GET /identifiers," updated table in subchapter Parameters on page 8.	
5	In Chapter "Endpoints," removed spaces for **`"alias"`** in subchapter Response on page 10.	
6	In Chapter "Endpoints," in subchapter "GET /identifiers/missing," updated subchapter URL Suffix on page 12.	

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 are 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 1:

Membership	Trading System	API MPID Value	API Exchange Value
Cash Equities and Iceland Fixed Income (all exchanges)	INET	XYZ	INET
Commodities	Genium INET	XYZ	NC
Derivatives	Genium INET	XYZ	SE
Cash Fixed Income Stockholm	Genium INET	XYZ	ST
Cash Fixed Income Copenhagen	Genium INET	XYZ	CO
Cash Fixed Income Helsinki	Genium INET	XYZ	HE
Cash Fixed Income Tallinn	Genium INET	XYZ	TA
Cash Fixed Income Riga	Genium INET	XYZ	RI
Cash Fixed Income Vilnius	Genium INET	XYZ	VI

2.3 Paging

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

- **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.
- **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.
- **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: **[response.total] > [request.offset] + [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 2:

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

> **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 3:

Scenario	POST	PUT
Not earlier mapped code	204	204
Mapped earlier, identical data	409	204
Mapped earlier, changed (allowed) values	409	204
In conflict with existing SC (beginDate)	409	409

> **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.

```
{
  "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:

Name	Description/Value	Mandatory
exchange	Alphanumeric value for exchange identification	Y
mpid	Alphanumeric value for member ID	Y
codeType	Client-Person Client-Entity InvestorDecisionMaker-Person InvestorDecisionMaker-Algo ExecutionDecisionMaker-Person ExecutionDecisionMaker-Algo	N
shortcode	Short code numeric value Absence means wildcard	N
longcode	Longcode value Absence means wildcard	N
date	Optional filter parameter, filters mappings active only the given date Absence means wildcard	N
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:


```
{
  "total": 3,
  "result": [
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType":"Client-Entity",
      "shortcode": 110100,
      "beginDate": "2018-01-02",
      "endDate": "2018-12-26",
      "longcode": "549300L8X1Q78ERXFXXX",
      "alias": "Alias1",
      "comment": "Comment 1"
    },
    {
      "exchange": "ST",
      "mpid": "XYZ",
      "codeType":"Client-Person",
      "shortcode": 110101,
      "beginDate": "2018-03-26",
      "endDate": "2018-12-01",
      "longcode": "SE197012111111",
      "alias": "Alias2",
      "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": "Alias3",
      "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:

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

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:


```
{
  "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:


```
[{
 "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.