Instructions for using the Update Endpoint of the NMD3.0

Author: R.H. Dijkstra, RBit Informatica BV, for SBK / NMD

Date: 22 juni 2020

Version: 1.0

Contents

Introduction.

The modus operandi of the NMD3.0 is that new “instrumentholders” (companies that have a license with SBK/NMD to develop calculational tools using the NMD environmental data) start by being handed a “data delivery file”. This contains a back up of those tables and data in the underlying database of NMD that the instrumentholders need to make their computations according to the Rules of Calculation of NMD3.0.

This data delivery file contains all necessary data as of the day at which it is produced.

The NMD, however, is a “historic” database. This means that all data are time-based.

This has several consequences:

1. MKI calculations must always be made for a specific date (because the number can be different from one date to another)
2. The people making the product cards are continuously working to make new cards and improve and update older ones. Such new information is then “published” (which means: made available to the instrumentholders for a certain day)
3. The result of 1. and 2. is that instrumentholders must keep their database up to date daily to prevent making calculations based on obsolete data.

This is done through the Update Endpoint, for which this document contains the instructions

The mechanism of the Update Endpoint, and how it is linked to the publication mechanism

Publication through versioning tables

In NMD3.0 all products and other content tables are "published" through the use of a "versioning table" that is linked to each base table. This versioning table always has the name of the base table it belongs to, but with the postfix "_Veries" added at the end.

> *In principle, the Update Endpoint simply sends you the new records for each versioning table, as they are published for the specified ZoekDatum (Search Date).*

The update endpoint works with a search date.

Although the update endpoint is meant to be used daily, it is still necessary to pass the date as a parameter to the Update Endpoint. This enables the software of the user to retrieve updates of multiple days (f.i. when for some reason updates for some dates have been missed). The date for which the data are retrieved we call the ZoekDatum (= Search date)

> *The **ZoekDatum** that you pass along with the request is - by definition - also the date to be used for DatumActief when you add a record and for DatumInActief when you deactivate a record*

The Time slot for retrieving the updates.

The time-slot for retrieving new updates is every morning between 2 am and 6 am (CET)

*The importance of retrieving Updates in the **correct order**.*

As we will see below, the mechanism of updating and versioning is based on an unbroken chain of active and deactivated versions for each data entity. Together they determine the period for which each version is valid. To keep this chain of DatumActief and DatumInactief consistent (and thus to keep the time structure correct) you must always download and integrate the updates **in the right order**: If you or your software have missed a number of days of updates, you must always retrieve the updates for these dates in the right order, before your return to normal operations!

What if you missed one or more updates?

So for example when you have had some kind of problem, and the updates for (say) 16 to 18 of may 2020 have been missed, and you now want to restart your system, you (or more to the point: your software) must first run the Update Endpoint with ZoekDatum=20200516, then run it with ZoekDatum=20200517, then run it with ZoekDatum=20200518 before the normal updating routine can be resumed!

Technical maintenance of the server and in the datacenter

The same thing goes for the availability of the server. It does not happen very often, but every few months there will be a temporary shutdown of the server at night (normally only for a few minutes), because of maintenance on the technology platform or in the datacenter where the server is hosted. This too can lead to a missed update. Therefore it is necessary to write and install your retrieval software in such a way that if it finds the server unreachable, it tries again one hour later or so, and maybe repeat that a few times). If, however, your software missed the update for a specific day, the same as the above goes for catching up the next night.

Adding updated information to a previously added Product (or other entity)

When an update in the Update Endpoint pertains to a previously published product there will already be a record in the base table and (at least) one record in the Versioning table that goes with it. In this

case, only a new versioning record needs to be added, and the DatumInActief of the previously Active records needs to be set.

The currently active versioning record can be recognized by a date value in the field “DatumActief” (Date of Activity) that is <= ZoekDatum and a DatumInActief (date of deactivation) that is NULL[1].

ID	ProductID	ProductNaam	DatumActief	DatumInActief
11632	15907	Deelproduct: Verhardingen in kg, Asfalt...	2019-08-21 00:00:00.000	*NULL*
11633	15907	Deelproduct: Verhardingen in kg, Asfalt...	2019-08-19 00:00:00.000	2019-08-21 00:00:00.000
11630	15907	Deelproduct: Verhardingen in kg, Asfalt...	2019-08-19 00:00:00.000	2019-08-19 00:00:00.000
11631	15907	Deelproduct: Verhardingen in kg, Asfalt...	2019-08-19 00:00:00.000	2019-08-19 00:00:00.000
11616	15907	Deelproduct: Verhardingen in kg, Asfalt...	2019-08-17 00:00:00.000	2019-08-19 00:00:00.000

The actual addition of the update information to the database is done in two steps:

1. Simply add the record with the newly published data to the Versioning table (where the DatumActief must, of course, be equal to the date for which the Updates are being retrieved)
2. The Datum**In**Actief of the currently active version (where DatumInActief = NULL) must also be set **to the same date** as the DatumActief of the update record that we just added under bullet 1. In this way, the currently active version is deactivated at the same moment that the new version is added that now becomes the active version.

This completes the update of the data a product or other data entity in NMD3.0

Adding a new product to the database or cache.

When you maintain the base table in your system or cache (theoretically the data structure is consistent even without the actual base tables) your software also must add a new base table record when the update contains the first version of a completely new product (or other entity).

The logic for this has no specific complications except one: how to know if a record in the Update Endpoint belongs to a completely new entity?

To be new or not to be new?

To answer this question you can write code that does a simple test whether a record with the specific {Entity}ID already exists in the base table. If this is the case it is an update, if not it is a new record

Deactivating a record:

A third function of the Update Endpoint is relaying the information that certain entities have been deactivated by a certain date. This is done by setting the DatumInActief of the currently active record to the specified date without at the same time adding a new versioning record. (how this works with the Update Endpoint will be explained below.

For the new Category 3 product cards phases, 6,7 and 8 are not filled with data

A few weeks ago we completed the “Categorie 3” data entry application. In this application the phases B3, B4, and B5 with PhaseID 6,7 and 8 are never used, therefore they also not stored.

[1] Actually, in the data of the NMD3.0 a record can also be deactivated for a date > today, but since the Update Endpoint does work for dates > today and does not give future information, that can not happen in your data.

More in general it is not a rule that all phases are always filled with data.

Implementation in the Update Endpoint

The above is implemented in a very simple way in JSON of the Update Endpoint:

For Updates and New Entities:

1. The Update needs to be retrieved every day
2. When the request is sent to the Endpoint the specific day is passed along as a parameter (ZoekDatum = SearchDate)
3. For Each {tableName}_Versies table in the NMD3.0, for all records where DatumActief= SearchDate the fields that are also in the DataDelivery Field will be represented in a JSON object as a simple flat list of field names and values.
4. You can thus process each of those objects just as if they were records in the Datadelivery file and add them to your database or cache in the same way that you processed the Datadelivery file.
5. Only two extra things need to be done:
   - (very important!) the Datum**In**Actief of the existing Active Record (DatumActief > 0 and Datum**In**Actief = NULL) with the same Entity ID must be set to ZoekDatum.
   - Check if the Object represents a new entity (as explained above). If so add the corresponding record to the corresponding base table (If you use those in your data structure)

Of course: if you have built a proprietary cache structure with a different architecture than that of the DataDevilery file it is up to you to figure out the corresponding transformations that you need to get the data from the JSON of the Update Endpoint into your cache (or whatever structure you use).

Deactivation of an entity.

The deactivation of an entity (which means that as of the date of deactivation it can not be used in calculations anymore) is done by setting its Datum**In**Actief to the specific date where EntityIDs match and the current Datum**In**Actief is NULL, **without** at the same time adding a new active record. The result of this operation is, of course, that as of the day of deactivation there will no longer be an active record, so for search dates after that date, the entity will no longer be available for computations.

BEWARE: you must **NOT** set the DatumInActief for all version records with the specified Entity ID, because by doing so you would destroy the historic chain of available version in the past.

Thus you must **only** set the DatumInActief where it is currently NULL

The update Endpoint does not look into the future

One limitation of the structure of the Update Endpoint is that it does not look into the future. You can specify a ZoekDatum in the past, but not in the future. When a data worker adds a publication or deactivation date further into the future, this will not be shown in the Update Endpoint, until that day has actually arrived.

If there is a popular demand for this we could make a separate Endpoint that produces all future publications or deactivations from a certain date.

Sample JSON:

In the small sample JSON below we first show the output for a ZoekDatum where only some records for the table NMD_ToepassingsGebieden_Versies have been published (Nieuwe_Publicaties).

A larger JSON output can be found in the new Update Endpoint where for the ZoekDatum of 20200622 there is an update of complete published (test) product.

About the structure of the JSON

In the Object “Nieuwe_Publicaties” (new Publications) you see that for most tables there are no newly published records for that day

In the Object “Gedeactiveerde_Entiteiten” (Deactivated Entities) you find the information about deactivated Entity records for each tabel

BEWARE: in the Update section the JSON will present ALL updated records for the complete tree of a product: ((parent)Product -> Product_Product -> (child) Product -> Product_element_ProfielSet -> Profielset -> Profile -> ProfielMilieuEffect) . We do this because each of the corresponding records in these tables can and probably will contain specific new information.

With the deactivations, however, we only give the ID of the deactivated product, because in NMD3.0 the rule is that a **product is always activated or deactivated as a whole**. Therefore, for the deactivated products it is a matter of simple logic that if the top-level Product record is deactivated, then all corresponding records in the following steps in the hierarchy as described above must also all be deactivated (in the same way as the top-level product is deactivated: by setting DatumInActief in the _Versies record where DatumInActief = NULL to the ZoekDatum

Note: if for a specific date there are no updates for a specific table, it will say: “na”.


```
{
 "Resource": "getUpdatesByDay",
 "zoekDatum": "20190309",
 "Message": "noDataAvailableFTS",
 "Nieuwe_Publicaties": {
  "NMD_Product_Versies": "na",
  "NMD_Product_Product_Versies": "na",
  "NMD_Product_Element_ProfielSet_Versies": "na",
  "NMD_ProfielSet_Versies": "na",
  "NMD_Profiel_Versies": "na",
  "NMD_ProfielMilieueffect_Versies": "na",
  "NMD_Eenheid_Versies": "na",
  "NMD_Type_Kaart_Versies": "na",
  "NMD_Milieucategorie_Versies": "na",
```

```
"NMD_Element_Versies": "na",
"NMD_Element_Element_Versies": "na",
"NMD_Fasen_Versies": "na",
"NMD_Schalingsformules_Versies": "na",
"NMD_ToepassingsGebieden_Versies": [
 {
  "ToepassingsGebiedID": 1,
  "Toepassing": "B&U",
  "VersieNR": 1
 },
 {
  "ToepassingsGebiedID": 2,
  "Toepassing": "GWW",
  "VersieNR": 1
 },
 {
  "ToepassingsGebiedID": 3,
  "Toepassing": "B&U en GWW",
  "VersieNR": 1
 },
 {
  "ToepassingsGebiedID": 4,
  "Toepassing": "Compleet Project",
  "VersieNR": 1
 },
 {
  "ToepassingsGebiedID": 5,
  "Toepassing": "Proces",
  "VersieNR": 1
 }
]
```

```
 },
 "Gedeactiveerde_Entiteiten": {
  "NMD_Product_Versies": "na",
  "NMD_Eenheid_Versies": "na",
  "NMD_Type_Kaart_Versies": "na",
  "NMD_Milieucategorie_Versies": "na",
  "NMD_Element_Versies": "na",
  "NMD_Element_Element_Versies": "na",
  "NMD_Fasen_Versies": "na",
  "NMD_Schalingsformules_Versies": "na",
  "NMD_ToepassingsGebieden_Versies": "na"
 }
}
```


Reachability of the Update Endpoint:

You can reach the testing version of the Update Endpoint (to develop your software) at:

https://milieudatabase-datainvoer.nl/NMD_30_API_UPDATE_TEST_v0.8/api/NMD30_Web_API/GetUpdatesByDay?ZoekDatum=20200622

The VERB for this request is GET

(as customary you can add "&includeNULLs=true" to the end of the link if you want to have the fields with NULL values included)

For the credentials for the Oauth2 process: you can use the same process as for all of the API Endpoints and also the same Refresh_Token that you have been issued earlier for the retrieval of Access_Tokens

To view the output of the Endpoint in Postman or another REST Viewer you can use this temporary Access_Token

NPfmpilm8QkZ+qPlaeoq1qgilgnccmeMUKtcj6YAzhg2mdJpruMQOKxByTfIFxOXI7N7Edbf0wZPlIrMIor9/+MZ2lX/tv/4erDEteb7znYrXoymLVmZxkJGVgZq9UbQ