

Dubocalc

NMD API implementation

Function	Engineering
Applicable Department	Documentation
Document number	1000001
Document author/owner	Marijan Ribarić
Document maintainer	Marijan Ribarić
Approved by	Marcel Sminia
Approval date	16.11.2021.

Revision history

Revision	Author	Date	Status & description of the change
1	Marijan Ribarić	22.11.2021	
2			
3			

UNRESTRICTED

Version 1
Publish Date 22.11.2021.

DISCLAIMER

This document is classified as UNRESTRICTED. Access and distribution is allowed to dedicated and assigned stakeholders by Cenosco only.

Table of Content

Table of Content	2
1 Running the importer	3
2 Importer logic	3
2.1 Deactivating items	3
2.2 Adding generic items	3
2.3 Adding elements	4
2.4 Adding products	4
2.4.1 NMD2.3 Scaling	4
2.4.2 Adding profile sets	4
2.4.3 Adding profiles and profile environmental effects	5
2.5 Final steps	5

1 Running the importer

The nightly importer is implemented as a workflow (scheduled task that is started at specified times). To run the importer WFSchedulerStarter project needs to be running, as a Windows service on production and UAT or started in Visual Studio when debugging. The workflow for the importer is started in intervals of a one hour.

Import of NMD data starts in method UpdateNMDLibraryDataFromWF in the class NMDLibraryDataWFMethods. The process starts only if the next scheduled update is before current time. This information is stored ApplicationSettings table, in a row with name value of NextUpdateLibrarySchedule.

The updater is then run for every revision date since the last one. Information about last revision stored ApplicationSettings table, in a row with name value of RevisionDate. If there are no revision dates to process (the import already successfully for current date), the importer exits.

2 Importer logic

The code for the updating mechanism is in the NMDLibraryDataUpdater.cs.

The importer runs the update for every revision date separately. The data is fetched from the NMD API on the getUpdatesByDay_4 endpoint. The parameter ZoekDatum is set to the current revision date. The API response is deserialized in to NMDUpdateApiResponse object. If no data is available for the revision date, the object will have noDataAvailableFTS message and the process is over for that revision date.

2.1 Deactivating items

The importer deactivates the items first. Once items are in the database, they're not updated. If an item is changed, a new revision is created and becomes available on the API. The item that is no longer active because a newer version exist is deactivated. The items that need to be deactivated are specified in Gedeactiveerde_Entiteiten field in the response JSON and are deserialized into NMDUpdateApiDeactivatedResponse object.

The following items are deactivated in this step: Units, map categories, phases, environmental categories, scaling formulas, application areas, products, elements. These items are deactivated by setting DeactivatedDate on the item to the value provided by the API or the current revision date if the value is missing from the API response.

2.2 Adding generic items

The new items are in the Nieuwe_Publicaties field in the response JSON and which is deserialized to NMDUpdateApiResponse object. The items are then converted to the database objects and saved (see table 1 for reference). These items include units, map categories, phases, environmental categories, scaling formulas and application areas. Custom data that is used in the UI (such as ordering and visibility flags) is taken from the previous revision of the same item.

Table 1 - Relations between field in API response and application data models

Field in API response	Deserialization model	Database model
NMD Eenheid Versies	NMDEenheidVersiesModel	LibUnits
NMD Type Kaart Versies	NMDTypeKaartVersiesModel	LibMapCategory
NMD Milieucategorie Versies	NMDFasenVersiesModel	LibPhase
NMD ProfielMilieueffect Versies	NMDMilieucategorieVersiesModel	LibEnvironmentalCategory
NMD Schalingsformules Versies	NMDSchalingsformulesVersiesModel	LibScalingFormula
NMD ToepassingsGebieden Versies	NMDToepassingsGebiedenVersiesModel	LibApplicationArea
NMD Product Versies	NMDProductVersiesModel	LibProduct
NMD Element Versies	NMDElementVersiesModel	LibElement

2.3 Adding elements

New items from NMD_Element_Versies are converted to LibElement. Additionally, PureElementCode is extracted from ElementCode. The new elements are then saved and reloaded from the database. Child elements are then linked to the parent elements and saved again. The relations between parent and child elements are found in the NMD_Element_Element_Versies field of API response.

2.4 Adding products

For every product in NMD_Product_Versies an LibProduct object is created. If a product is standalone (has no children and is not a parent) or it's a parent object, it is linked with an element (using NMD_Product_Element_ProfielSet_Versies field in the response). The product is also linked with unit of measure and application area. If the product is a child product, it is linked with its parent through NMD product id.

The parent products are then reloaded from database to get their real database IDs. The child products are linked to their parents with real database IDs. The unit of measure is also set to the unit of measure of the parent.

If there's something wrong with the product data, product is invalidated. The product can be invalidated in following situations:

- If the product is a parent product but has no children
- If it's standalone or a child product, but it has no profile sets.
- NMD_ProfielSets_Versies or NMD_Product_Element_ProfielSet_Versies data is missing from API response
- All profile sets that belong to the product are invalidated

When product is invalidated, the field "invalid" is set to 1. If the product has children, they are also invalidated. If the invalidated product is a child product, the parent is also invalidated, but only if all other children are also invalid.

2.4.1 NMD2.3 Scaling

If the product has NMD2.3 scaling, the information about the scaling is retrieved from NMD23_Type2_SchalingInfoBijProduct in case of type 2 scaling and NMD23_MVT_infoBijProduct in case of type 3 scaling. This data is deserialized into LibNMD23ProfileSetScaling object. The scaling data is then linked to units of measure data from the database and with the product it belongs to. NMD23ScalingType is set to the appropriate scaling type.

For the scaling type 3 information about variants of the product from MvtVariants field in the JSON are saved to the LibNMD23ProfileSetScalingVariant table.

2.4.2 Adding profile sets

Profile sets are then added for each product. Information about profile sets is found in the NMD_ProfielSets_Versies field in the API response. This information is converted to LibProfileSet object and saved to the database.

Part of the information from NMD_ProfielSets_Versies that is related to scaling is instead saved to LibProfileSetScaling object. If scaling data exists for a profile set, IsScalable field is set to true for the product (and its parent if it exists) that the current profile set belongs to.

If the NMD_Profiel_Versies field is missing in the response, or there are no profiles linked to the profile set, the profile set is invalidated. In this case, the "invalid" field is set to 1 on the profile set. The profile set is also invalidated if all profiles linked to it are invalidated.

2.4.3 Adding profiles and profile environmental effects

After each profile set is added, the profiles linked to it are added. Profiles from NMD_Profiel_Versies field in the API are converted to the LibProfile objects and saved to the database. The profiles are then reloaded from the database to get their unique primary keys. The environmental effects from the NMD_ProfielMilieueffect_Versies field in the response are converted to LibProfileEnvironmentalEffect object and saved to the database and linked to the previously fetched profiles.

If NMD_ProfielMilieueffect_Versies is missing in the response, or there are no profile environmental effects linked to the profile, the profile is invalidated. This is done by setting the “invalid” field to 1.

2.5 Final steps

When everything from the API response is imported, additional steps are performed. Firstly, LibProfileSetResult records are generated and added. LibProfileSetResults are intermediate stage records used to speed up calculation.

Then the MKI calculations of the newly imported products are started using DCCalculationHelper class. When calculations finish the import process for the current revision date is done and the RevisionDate row in the ApplicationSettings column is updated by setting NValue to the processed revision date.

When all revision dates are processed, the date and time for the next scheduled import is set and the process is complete.