

Dubocalc 6.1

Overview and documentation

Function	Engineering
Applicable Department	Coding, Documentation
Document number	1000001
Document author/owner	Marijan Ribarić
Document maintainer	Sandi Uran
Approved by	Marcel Sminia
Approval date	27.9.2021.

Revision history

Revision	Author	Date	Status & description of the change
1	Marijan Ribarić	24.9.2021	
2	Sandi Uran	27.9.2021	
3			

UNRESTRICTED

Version 1
Publish Date 27.9.2021.

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. Key Concepts	3
1.1. Library.....	3
1.2. Project.....	3
2. User Interface	4
2.1. Library.....	4
2.2. Project management UI.....	6
2.3. Single Project UI.....	7
2.3.1. Main Project Screen	7
2.3.2. Priority analysis.....	8
2.3.3. Dashboard.....	9
2.3.4. Reports	10
3. Technical implementation	11
3.1. First run	11
3.2. Database.....	11
3.3. Project and product loader.....	14
3.3.1. Product loader.....	14
3.3.2. Product caching.....	14
3.3.3. Project loader.....	14
3.4. Project Calculations.....	15
3.4.1. NMD2.3 scaling type 2.....	15
3.4.2. NMD2.3 scaling type 3.....	16
3.4.3. NMD3.0 scaling.....	16
3.5. Product Calculations.....	16
3.6. Charts	17
3.7. Import from NMD	18

1. Key Concepts

1.1. Library

The Dubocalc Library contains products that users can use when creating projects. Products (LibProduct) consist of profile sets which contain profiles. A profile (LibProfile) is a link between a profile set and a phase. Environmental categories (LibProfileEnvironmentalCategory) are types of impacts the product has on environment. For example, global warming or ozone depletion. A phase (LibPhase) signifies time in the lifecycle of a product where environmental effect is created. In Dubocalc, following stages exist: Product stage (A1, A2, A3), construction process stage (A4, A5), use stage (B1, B2, B3, B4, B5), end of life stage (C1, C2, C3, C4, D). Profile environmental effect (LibProfileEnvironmentalEffect) is a value for a specific environmental category for a specific profile.

Some products and profile sets can be scaled along one or two dimensions. NMD2.3 scaling refers to product scaling, while NMD3.0 scaling refers to profile set scaling.

To speed up calculations, Dubocalc implements intermediate stage: LibProfileSetResult which contains LibProfileSets without quantity and replacements factored in, CO2 value of a profile set, MKIs separated by environmental categories.

Products can have child products. In that case (both in UI and calculations) profiles sets of all children are taken. Products are grouped in folders called library elements (LibElement).

1.2. Project

In Dubocalc, projects have hierarchical structure. Variants represent different versions of one project that can be compared. User can add elements or element items to a variant. Element is a folder that allows grouping of objects in the project tree. Element item represents a product from the library that was added to the project. Element can have element items or other elements as children. Element items can have only element item sets as children. Element item sets are not shown in the project tree. Users can't add or remove element item sets since they represent profile sets for a product but can modify scaling value if a specific profile set supports it.

2. User Interface

2.1. Library

Library contains the products that are available for use in the project. The library contains both products from the NMD and user created products. Besides products, project elements, along with their subtree, can also be saved in the library to allow for reuse in other projects.

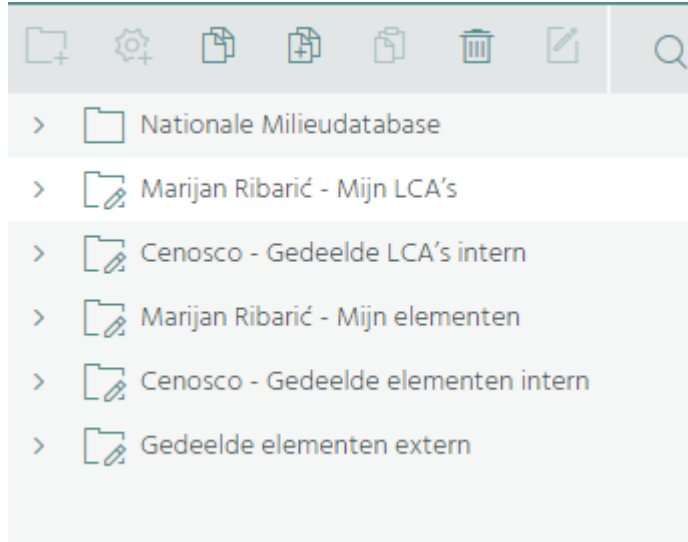


Figure 1 - Library folders

The library contains following folders:

- Products from NMD – grouped by LibElements
- Custom products – personal
- Company custom products
- Personal saved project elements
- Company project elements
- Externally shared project elements

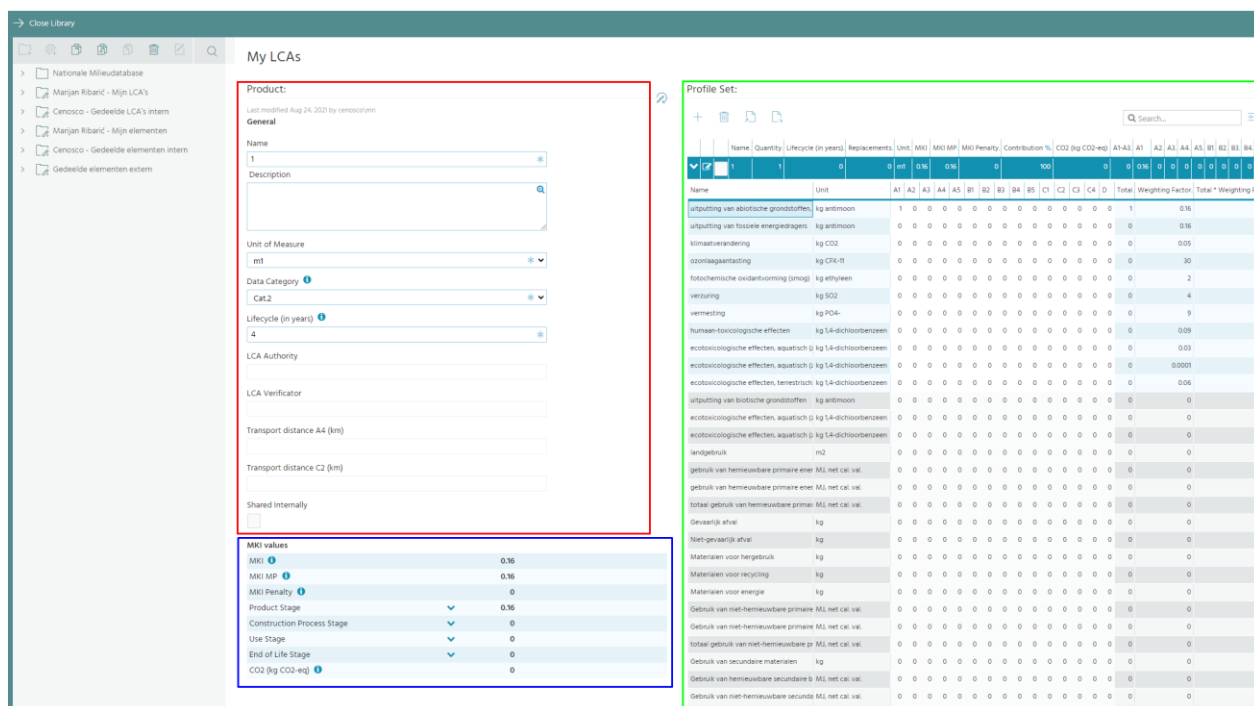


Figure 2 – Library product details

Selecting a product in the library, or adding a new product, opens the product details page. This page consists of three parts: Form for changing the information about the product (red outline), form with MKI values (blue outline), and form with profile sets and profiles (green outline).

The information form (red outline) provides a way to change basic information about the product. Some of this information is used when calculating MKI. Only a subset of information is displayed if NMD product is selected (and the information is read only).

Form with MKI values (blue outline) shows a set of MKI values for the product. The following MKI values are shown: Total MKI value, MKI value without penalty, MKI penalty. Also, MKI values separated by stages are shown, along with CO2 values.

The grid with green outline shows detailed information about profile sets that belong to the product. Each profile set row can be expanded to show profiles that profile set consists off. The rows of this grid are environmental effects and columns are phases. Additional columns are related to an environmental effect: sum of values for all phases, weighting factor of the environmental effect, and final value (sum multiplied by weighting factor).

In the toolbar above the grid a user has options to add, delete, import, or export a profile set.

2.2. Project management UI

The first screen after login is the project management screen.

Duurzaam Bouwen Calculator

Start a new project or select one from the list.

New Project

Import

Nationale Milieudatabase

Manage your own LCAs.

Library

My Projects							Deleted Projects	
Project name	Start Date	Lifecycle (in years)	Use data with publication date up to	Owner	Users			
Calculation test	Jun 17, 2021	15	Jun 17, 2021	Marijan Ribarić	1			
Lifecycle test	Aug 13, 2021	150	Aug 13, 2021	Marijan Ribarić	1			
Project Calculations	Jul 19, 2021	250	Jul 19, 2021	Marijan Ribarić	1			
Test project	Jun 2, 2021	50	Jun 2, 2021	Marijan Ribarić	1			
TreeView Test	Aug 31, 2021	142	Aug 31, 2021	Marijan Ribarić	1			

Figure 3 - Project management screen

All projects that current user has access to are shown in the main grid on this page, along with the basic information about them.

The action bar above the grid allows the following actions to be executed against the selected product.



Figure 4 - Available project actions

1. Open project – shows single project UI
2. Copy project – duplicates the project
3. Export – exports the project to a file
4. Generate report – generates one of the available reports
5. Delete – deletes the project (sets flag to deleted, does not permanently remove)
6. Invite user – allows adding and removing users from the project
7. Event history – shows grid with changes to the selected project

Besides the actions on the grid, the user can also add a new project, or import previously exported project.

Selecting “deleted project” tab user can see projects that were deleted using the previously described menu. When in that tab, user can see the same grid. The toolbar of the grid has options to restore the project and to permanently delete the project.

2.3. Single Project UI

User interface for project creation consists of three pages: Project, priorities analysis and dashboard.

2.3.1. Main Project Screen

In the main project screen user can manipulate with all the objects in the project structure. This screen consists of the tree view on the left side and the details form on the right side.

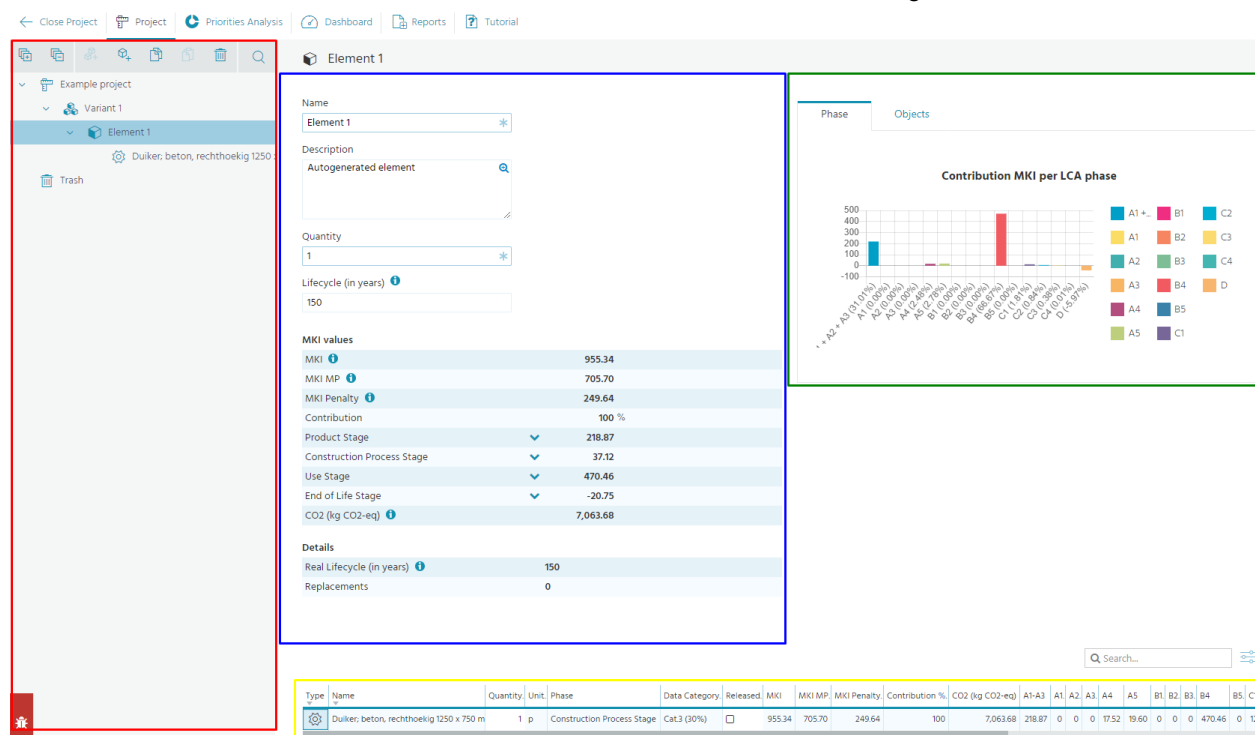


Figure 5 - Main project screen

Tree view (red outline) shows the project hierarchy and allows navigation through project objects. There are two root nodes in the tree: project node and trash node.

Project node contains objects that are in the projects but are not removed. Only variants can be direct children of the project node. This node's children can be drag and dropped to move the node to another parent, to change the order or to move the node to trash (delete item).

Items that are deleted from the project hierarchy, but not completely removed from the project are shown as children of trash node. Items that are currently in the trash can be restored or permanently deleted. Clicking on an item in the tree view selects the item and displays its details in the right part of the screen.

On the details screen there are following elements: form with detailed information about the current item (blue outline), charts that visualize MKI contribution per phase or per object (green outline) and grid with detailed information about current node's children (yellow outline). If an object is deleted only a read-only details form is shown.

Details form shows information about current project node. Depending on the type of object fields shown in the form differ. In general, name and description of the objects are shown, along with the MKI values for the project. The following MKI values are shown: Total MKI value, MKI value without penalty, MKI penalty. Also, MKI values separated by stages are shown, along with CO2 values.

Two charts are available on project screen: a bar chart that shows total MKI separated by phases and a pie chart that shows total MKI contribution of each direct child of the current object.

Grid with direct children (yellow outline) contains detailed information about direct children of the current object. Objects in the grid can be edited in place.

2.3.2. Priority analysis

Priority analysis page also contains tree view (although in read-only mode), form for choosing the type of priority analysis, and various visualizations of the data.

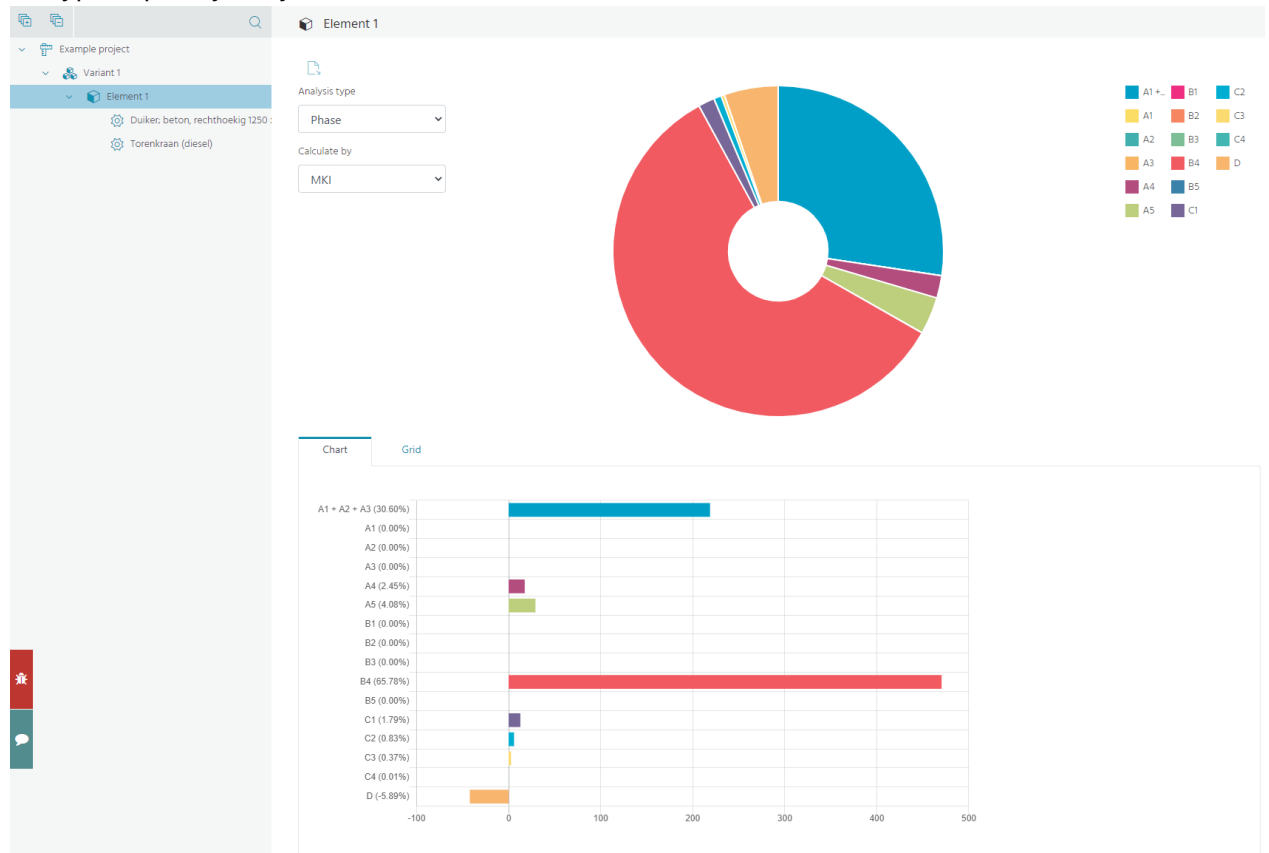


Figure 6 - Priority analysis screen

Data is primarily displayed in a pie chart on the top of the screen, but the same data is also shown as a bar chart or in a grid format depending on which tab is selected. The data can be calculated based on the MKI of an object selected in the tree view or based on the specified environmental impact of that item.

Following types of priority analyses exist (the terms data and value in following explanation refer to total MKI of the object or MKI of specific environmental impact, depending on selection):

- Phase – Value is calculated by phase
- Object per phase – Values in a specified phase of the direct children
- Top 15 items per phase – The data consists of element items with the highest value for the specific phase
- Top 15 categories per phase – The data consists of library elements with the highest value for the specific phase
- Object all phases – Values of the direct children
- Top 15 items all phases – The data consists of element items with the highest value
- Top 15 categories all phases – The data consists of library elements with the highest value

2.3.3. Dashboard

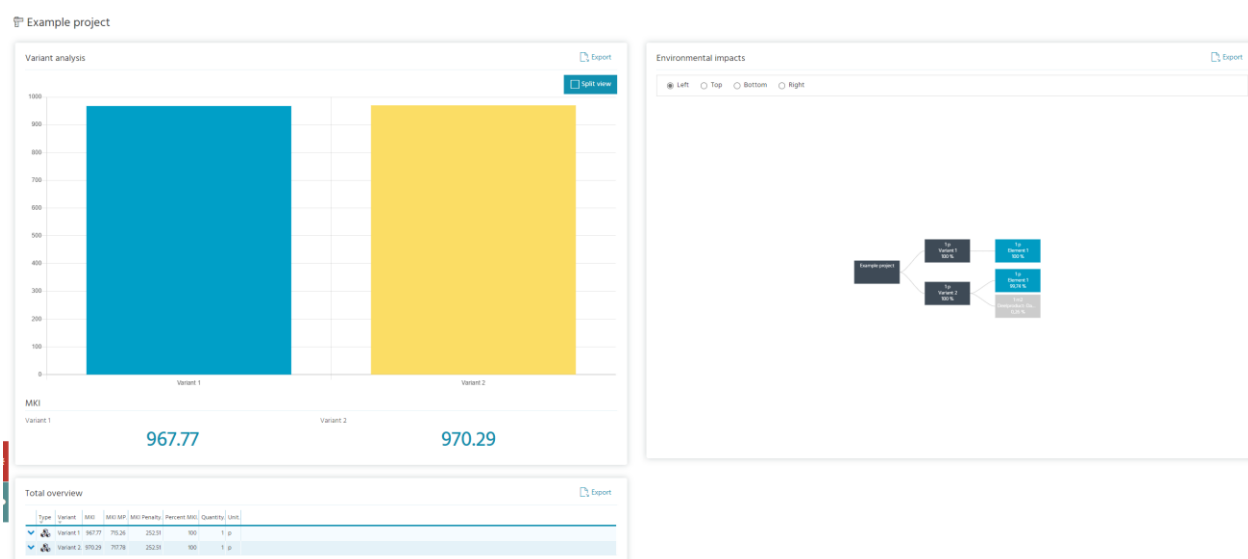


Figure 7 – Dashboard

The dashboard screen provides an overview of the project. The section titled “Variant analysis” allows for comparison of different variants in the project in the form of bar graph by default, and stacked bar graph (where the bar is divided by child objects) when split view is selected. The environmental impacts section presents interactive hierarchical project structure in form of a space tree graph. Clicking on a node on the graph further displays child nodes of that graph. The final section is titled total overview. This section contains a grid that shows the detailed information of all objects in the project. The grid initially shows variants that project contains. Clicking on the arrows in the first column shows the direct children of the selected object. The objects can be expanded up to element item sets.

2.3.4. Reports

Clicking on the report menu a window opens where a user can select a report to generate.

Generate Report ✕

Report name
Standaardrapport
Beknopt Overzicht
Uitgebreid Overzicht
Totaal Overzicht

Items per page: 25 ▼

 Generate as Word
 Generate as PDF

Figure 8 - Available reports

Following reports are available:

Standard report (Standaardrapport) – This report starts with a section that displays a table with basic project properties (name, description, quantity, lifetime). Next section provides a variant analysis chart which compares project variants by their MKI in a form of bar graph. After that, for each variant a table with variant properties is shown along with following subsections:

- Child object breakdown in a form of a bar graph
- Element properties – basic element properties. This section further has subsections:
 - Phase analysis – Shows pie chart for element MKI separated by product phase group
 - Item properties – Table with basic information about element items (repeats for each element item that belong to the element)

This report can be generated as PDF or DOCX.

Brief overview (Beknopt Overzicht) – This report shows brief overview of the project. It displays a table with basic information for each object. The report has a hierarchical structure. This report can be generated as PDF or DOCX.

Extensive overview (Uitgebreid Overzicht) – Same as brief? Needs to be checked.

Total overview (Totaal Overzicht) – Contains two tables. First table contains basic information about the project. Second table shows information about the element items. This table contains following data: name of the parent variant, the name of the element for the current element item, LibElement and LibMapCategory which the product belongs to, unit of measure, quantity, CO2 emission value, MKI values (total MKI, penalty MKI, MKI without penalty, MKI for each phase). The data is also aggregated for every variant and for the project.

This report can be generated as XLSX.

Endpoint for first three reports is `api/Report/ExecuteReport`. The endpoint for the last report (total overview) is `api/Project/TotaalOverzichtExport`.

3. Technical implementation

3.1. First run

In the development of Dubocalc Visual Studio 2019 was used. The workloads that need to be installed are shown in the figure below.

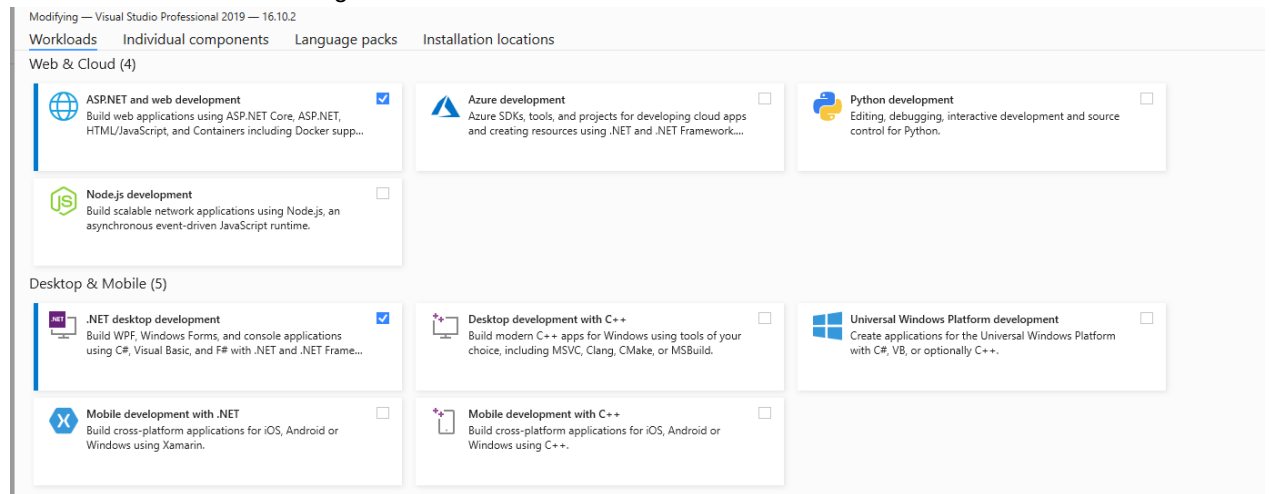


Figure 9 - VS2019 Workflows needed to run DuboCalc

Beside this workflow make sure to install SQL Server Data Tools component and Windows Workflow Foundation component.

Also, Visual C++ Redistributable Package needs to be installed. Installer located in the project "{ProjectRoot}\Ref\vc_redist_x86.exe" can be used to install the package.

When running Dubocalc for the first time startup projects need to be set up. For Service solution startup project should be DataService. For Client solution startup project should be Application. For WFSchedulerStarter startup project should be WFSchedulerService.

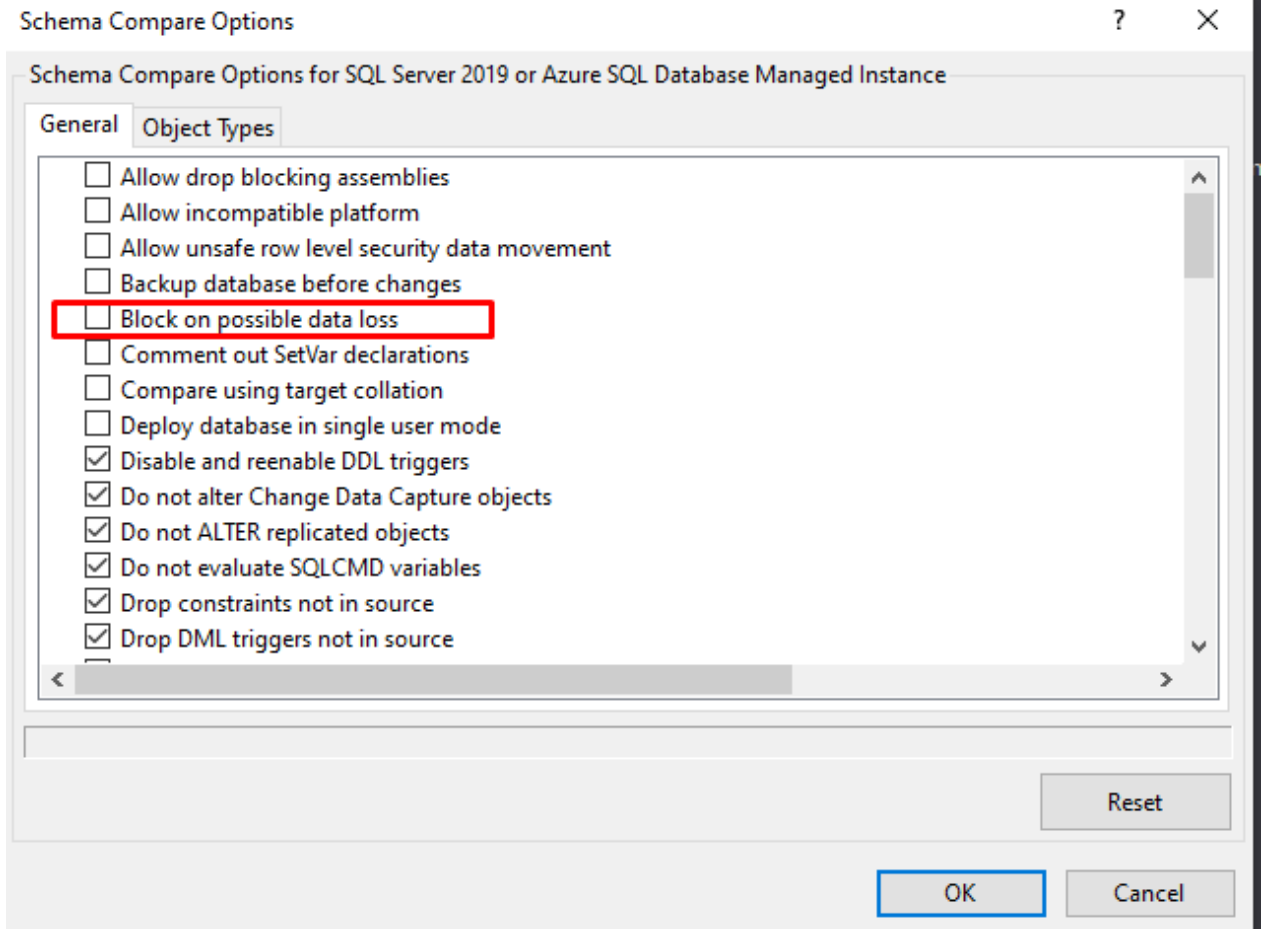
3.2. Database

When making a change to the database, the application data model should also be updated.

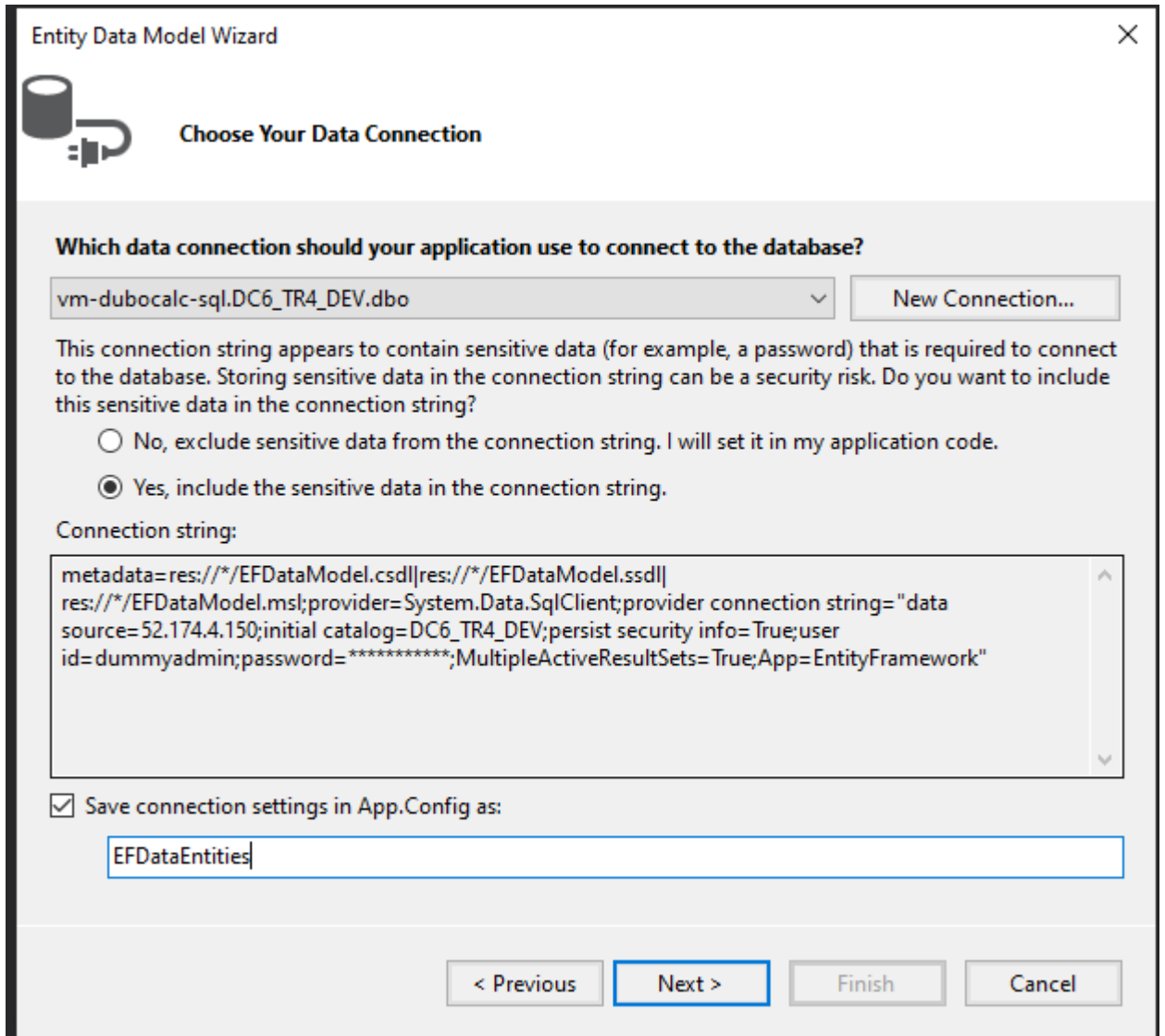
The SQL files that describe the database structure are in DubocalcDB solution. The procedure for updating database and data model is as follows:

1. Update database from DubocalcDB solution

- Update SQL files according to what needs to be changed
- Right click the project in visual studio, select schema compare
- Select the target database
- Deselect "block on possible data loss" (see figure below)
- Click on compare in the toolbox
- If the changes are what you expect click on update in toolbox. This will update the database



2. Open UpgradeProject solution and build it
3. Add ADO.NET Entity Data Model – name it EFDataModel 4
4. Choose EF Designer from database
5. Select database connection, select “Yes, include the sensitive data in the connection string”
6. Set connection settings name as EFDataEntities (see figure)
7. If a screen shows up where Entity Framework 6.X can’t be selected, restore nuget packages and rebuild the project.
8. Select all tables except _RefactorLog, AliasMap, sysdiagrams
9. Select all views except ones starting with vwCRpt
10. Unselect all procedures
11. Make sure pluralization is off
12. Set model namespace to DataModel
13. Open Services solution
14. Open DataModel project
15. Right click on CopyEdmx.tt -> Run custom tool
16. Check changes on EFDataModel.edmx to see if they reflect your changes
17. Expand EFDataModel.edmx and run EFDataModel.tt and EFMetaData.tt
18. If you were adding a table/view expand Classes folder and include created files for your table/view in Gen and UGen folders. If you were deleting a table/view delete files for that table/view from Gen and UGen folders.



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

vm-dubocalc-sql.DC6_TR4_DEV.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☒ Yes, include the sensitive data in the connection string.

Connection string:

```
metadata=res://*/EFDataModel.csdl|res://*/EFDataModel.ssdl|
res://*/EFDataModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=52.174.4.150;initial catalog=DC6_TR4_DEV;persist security info=True;user
id=dummyadmin;password=*****;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in App.Config as:

EFDataEntities

< Previous Next > Finish Cancel

Figure 10 - disable block on possible data loss

19. Open Repository project
20. Open EFDataModel.Context.tt
21. Change string domain = "Full"; to string domain = "Raw";
22. Save changes to run the tool
23. Revert changes
24. Save changes again to run the tool
25. Open DataService project
26. If you added views/tables include controllers in Controllers\API\Gen and Controllers\OData\Gen
27. If you removed views/tables remove controllers in Controllers\API\Gen and Controllers\OData\Gen
28. In DataService/Content run Validations.tt
29. Compile projects to verify everything works, revert UpgradeProject

3.3. Project and product loader

Both product and project have a hierarchical structure and Classes ProductTreeLoader and ProjectTreeLoader provide a way to load only things we need from products and projects.

3.3.1. Product loader

An object of class ProductTreeLoaderOptions can be provided to ProductTreeLoader to specify exactly what needs to be loaded. The following parameters can be specified (Boolean unless specified):

- LoadChildrenProducts – load child products
- LoadElement – load LibElements
- LoadProfileSet – load ProfileSets
- LoadProfileSetResults – load ALL ProfileSetResults
- LoadPartialProfileSetResults – load only CO2 and Total MKI ProfileSetResults
- LoadProfile – load LibProfiles
- LoadProfileEnvironmentalEffect – load LoadProfileEnvironmentalEffect
- LoadPhase – load LibPhases
- LoadEnvironmentalCategory – load LoadEnvironmentalCategory
- LoadEnvironmentalCategoryIds – list of ids. Profiles for which environmental category will be loaded
- LoadUnits – load LibUnits
- LoadMapCategory – load LibMapCategory
- LoadScalingData – load both NMD2.3 and NMD3.0 scaling data
- UseProductCache – use cache for LibPhases, LibUnits, LibMapCategory, LibEnvironmentalCategory, LibScalingFormula

3.3.2. Product caching

To speed up calculations 100 of the most used products are cached (class ProductCache). If a product is changed, it is removed from cache. Besides products, cache can be used for other database objects that do not change or change rarely over time. This cache is currently used for LibPhases, LibUnits, LibMapCategory, LibEnvironmentalCategory, LibScalingFormula. This cache is refreshed every 24 hours.

3.3.3. Project loader

An object of class ProductTreeLoaderOptions can be provided to ProjectTreeLoader to specify exactly what needs to be loaded. The following parameters can be specified (Boolean unless specified):

- LoadProduct – load product information
- LoadPhase – load LibPhases
- LoadUnits – load LibUnits
- LoadElementItemSets – load ElementItemSets
- LoadDeletedBit – integer. 1 – only objects that are not deleted from the tree, 2 – only deleted object, 3 – both deleted and not deleted

Besides hierarchical loading, projects can also be loaded as a flat list. In that case database returns list of objects with following fields: id, name, type, parent id, parent type, deleted.

Project loader also provides a method for generating a flat structure of the variant (GetProjectTreeFlatStructure). This method accepts id and type of an object. From that the variant id is determined. If type is not a variant, database function getVariantIdFromElement or getVariantIdFromElementItems is called. With this information, database function getProjectVariantStructureFromVariantId is called. The function returns whole structure as a list of objects. The object contains following fields: id, name, type, parent id, parent type, is object deleted (moved to trash).

3.4. Project Calculations

The project calculation happens in the DCProjectCalculator class. The class provides exposes two public methods: CalculateProjectTree - calculates the whole project tree, CalculateVariantTree - calculates only one project variant.

When calculating total MKI for a variant the project tree is traversed from top to bottom. On each level the children of each element are collected and then traversed. This process continues until the calculation process gets to the bottom of the tree (to the element item sets). At this point CO2 and Total MKI LibProfileSetResults are used to get initial value (when calculating charts based on environmental effect, LibProfileSetResult for a specified environmental effect is loaded instead). Additional modifiers are applied to this value. Some of the modifiers are set to the object higher in the hierarchy but are still applied on element item set level.

The modifiers that can be applied to element item sets are:

- Quantity and replacements of profile set
- Is element item released (set on element item level) – Sets A4 value to C2, A5 value to C1 + C3 + C4 + D. Values for all the phases (besides A4 and A5) are set to 0.
- Transport (set on element item level) – if a product has distanceA4 set, the value user set for distance is divided by default distanceA4 value to get a distance factor. Value of phase A4 is then multiplied by this factor.
- NMD2.3 scaling (set on element item level) – Type 2 and Type 2
- NMD3.0 scaling

The results are then carried from the bottom of the tree to the top. On each tree level MKI is adjusted by quantity and replacements. If lifecycle of a parent element is longer than lifecycle of a child element replacements are added. Number of replacements is calculated by dividing the parent lifecycle by child lifecycle and subtracting 1. Total MKI of a child object is then multiplied by number of replacements and the resulting value is added to B4 phase. Quantity modifier is calculated by simply multiplying values for each phase by quantity value.

To make this process generic, class MKIResult is used which contains all fields relevant to the calculations. This class is used in both project and product calculations. It is also used when calculating data for charts. MKIResult provides AssignValuesFromEntity method to convert IMkiEntity to MKIResult. IMkiEntity is an interface that is implemented by every database class that holds MKI values.

3.4.1. NMD2.3 scaling type 2

The scaling factor for NMD2.3 scaling type 2 is set by dividing the user entered value by default value if scaling is on one dimension. If scaling is on two dimensions, the product of user provided values for the dimensions is divided by product of default values. Value of every phase is then multiplied by the scaling factor.

This logic is implemented in DCProductScaling class. Scaling data is retrieved from the LibNMD23ProfileSetScaling table. For the calculation, important columns from this table are Dimension1 and Dimension2 which specify the default scaling values. Since this type of scaling is bound to ElementItem, the user entered values are saved in the ElementItems table (columns Dimension1Value, Dimension2Value). The calculated scaling factor is also saved in this table (column ScalingFactor).

3.4.2. NMD2.3 scaling type 3

In this case, multiple variants of the same product exist for different scaling values. When calculating the scaling factor, the current product variant needs to be selected. If the value equals one of the variant's default value, that variant is selected. Otherwise, the variant with the higher value is selected. The scaling value must be between the values of lowest and highest variants. If a value is between variants, the MKI is interpolated.

This logic is also implemented in DCProductScaling class and uses the same DB tables as does type 2 scaling. The different variants are saved in the LibNMD23ProfileSetScalingVariant table. When different product variant than current is selected, the LibProductId of that variant is linked to the ElementItem.

3.4.3. NMD3.0 scaling

The calculation of a scaling factor for NMD3.0 scaling is done in two steps. Initial scaling factor is calculated in the same way as NMD2.3 scaling. The default scaling values used for calculating the initial scaling factor are found in the LibProfileSetScaling table (columns ScalingSizeX1, ScalingSizeX2). The initial scaling is then *corrected* using a scaling formula (LibScalingFormula). Parameters for the formula are in the LibProfileSetScaling table (ScalingFormulaA1 – A, ScalingFormulaB1 – B, ScalingFormulaC – C). Finally, values for all phases are multiplied by scaling factor. The minimum and maximum values for each scaling dimension are also located in the LibProfileSetScaling table (ScalingMinX1, ScalingMaxX1, ScalingMinX2, ScalingMaxX2).

3.5. Product Calculations

When calculating products, the product tree is also traversed from top to bottom. If product is standalone (has no children) profile sets are traversed. If product has children, child products are traversed. Final nodes in the product tree are LibProfileSetResults.

LibProfileSetResults are calculated from environmental effects (environmental effects are children of a profile, profiles are children of profile sets). Only environmental effects with bitmask 1 are included in total MKI, and only environmental effects with bitmask 4 are calculated for CO2 value (Table – LibEnvironmentalCategory, column – EnvironmentalCategoryBitMask). Two LibProfileSetResults are created, one for total MKI and one for CO2. Beside those two, one LibProfileSetResult is created for each environmental effect with before specified bitmasks.

Table LibProfileSetResults contains column ResultType which contains the type of a LibProfileSetResult: 1 – total result, 2 – CO2 result, 3 – result per environmental category. LibProfileSetResult values are not modified by replacements or quantity but are multiplied by weighting factor (Table – LibEnvironmentalCategory, column – weighting factor) which is specific for each environmental category. Depending on the factor for a map category (table – LibMapCategory, column - StorageUntested) additional penalty is added to MKI. This penalty is saved to MKISumPenalty column as is calculated by multiplying the total MKI without D phase (columns MKISumMP, D) with factor. This result is saved to the MKISumTotal column.

When LibProfileSetResults are calculated, similarly to project calculations, product tree is traversed from bottom to top. Quantity and replacements are added at each step, following the same rules as in project calculation.

3.6. Charts

Endpoints for chart data (that is rendered in the main project screen and in the priority analysis screen) are in ChartDataController.

The actual generation of data is done in two steps. First the full MKI data is gathered. This happens in the ChartDataProvider class when full MKI is selected, and ChartDataProviderEnvImpact when an environmental category is specified.

ChartDataProvider provides following public methods:

- **GetSingleResult** – Returns a result based on the database ID and type of object directly from the database. Returns MKIResult.
- **GetDirectChildResults** – Returns direct children of an item. First, the flat tree of a variant is generated (class ProjectTreeLoader – method GetProjectTreeFlatStructure). From this data direct children are determined and fetched from the database.
- **GetAllChildResults** – Returns all element items that are children of the current objects. This method works like previous method. However, all children of type element item are fetched, instead of direct children regardless of type.
- **GetVariantsWithChildren** – Returns a list of variants for the projects with all its direct child elements (elements, element items) bound.

When an elemental category is selected, ChartDataProviderEnvImpact is used. This class inherits from the ChartDataProvider. Unlike with ChartDataProvider, the data for single environmental category is not directly saved in the database. Instead, the data is recalculated using DCProjectCalculator. GetSingleResult, GetDirectChildResults, GetAllChildResults are implemented and differ only in the filtering part (GetDirectChildResults – only direct children, GetAllChildResults – all element items).

After data generating is done, the data needs to be filtered. This happens in the DCCharts class. All public methods in this class return ChartDataModel which contains data UI can render. DCCharts implements following methods:

- **GetPhasesMKI** – splits the MKIResult into phases, and generated ChartDataModel for each phase.
- **GetObjectsPerPhase** – Creates ChartDataModel for all child objects. The methods accept parameter filterByPhase which signals whether only MKI for a single phase should be used.
- **GetTop15ElementItems** – filters top 15 values that are returned by GetAllChildResults. Additionally, allows filtering by phase.
- **GetTop15Categories** – Similar to previous method. However, instead of using element items, element items are grouped into LibElements they are linked to through LibProduct. This method also allows filtering by phase.
- **VariantChartData** – Creates ChartDataModel for a variant and its direct children using data from GetVariantsWithChildren.

When exporting a chart or generating a report, CefSharp is used to save the image of the generated chart. The path of the executable file of the CefSharp project used for this is "{ProjectRoot}\Ref\CefSharp-master V4\CefSharp.Wpf.Example.exe". Capturing the image of the chart is implemented in the ChartCaptureHelper class.

3.7. Import from NMD

To stay up to date with data available in NMD Dubocalc runs the importer every night to add new data. The logic for this is contained in WFSchedulerStarter solution.

Method UpdateNMDLibraryDataFromWF (class NMDLibraryDataWFMethods) is called by the framework when both DataService and WFSchedulerService are running if following conditions are met:

- Date and time saved in the ApplicationSettings table (column Avalue, row where name equals NextUpdateLibrarySchedule) are before current date and time.
- Date and time saved in the WFNextSchedule (column Next, row where name equals "NMD Library Update") are before current date and time

First date that will be checked for updates is day after the revision date save in the ApplicationSettings table (column Avalue, row with name RevisionDate).

The actual logic for updating the database with data from the NMD API is in NMDLibraryDataUpdater and starts with method NMDLibraryDataUpdate.

Important values in the App.config file in WFSchedulerService project for this functionality are: NMDApiUri - URL for NMD service is set in the t, NMDApiRefreshTokenUri - authentication service URL, NMDApiRefreshToken - refresh token.

If you want to receive emails with results of an update set DebugUpdateMailEnabled to true, and DebugMail to an email address you want to receive update results to.

NMDApiUri, NMDApiRefreshTokenUri and NMDApiRefreshToken need to also be specified in Web.config in the DataService project.