IDYN

Object Manager Advanced 9.03

User's Guide for NAV2013 (R2)

Published: October 2014

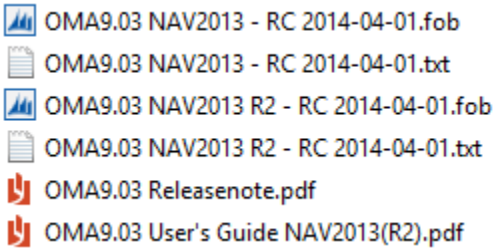
Table of Contents

1 INSTALLATION

This chapter explains how to install Object Manager Advanced 9.03 for NAV2013 (R2). This chapter also explains how to remove Object Manager Advanced with all its data from the database.

1.1 INSTALLING OBJECT MANAGER ADVANCED

Object Manager Advanced installation distributive contains the following files and folders:

OMA9.03 NAV2013 – RC 2014-04-01.fob NAV2013 Objects in .fob format. ***OMA9.03 NAV2013 – RC 2014-04-01.txt*** NAV2013 Objects in .txt format. ***OMA9.03 NAV2013 R2 – RC 2014-04-01.fob*** NAV2013 R2 Objects in .fob format. ***OMA9.03 NAV2013 R2 – RC 2014-04-01.txt*** NAV2013 R2 Objects in .txt format. ***OMA9.03 Releasenote.pdf*** ***OMA9.03 User's Guide NAV2013(R2).pdf*** (This document)	

Perform the following steps to install Object Manager Advanced:

Import Object Manager Advanced .fob file from the installation distributive using the development environment of Microsoft Dynamics NAV:

1) Run Object Designer in **Microsoft Dynamics NAV Development Environment**.
2) Click **File > Import...** and choose the .fob file from the installation distributive.
3) A custom menusuite can be imported if necessary.

1.2 ADDING SQL TRIGGERS

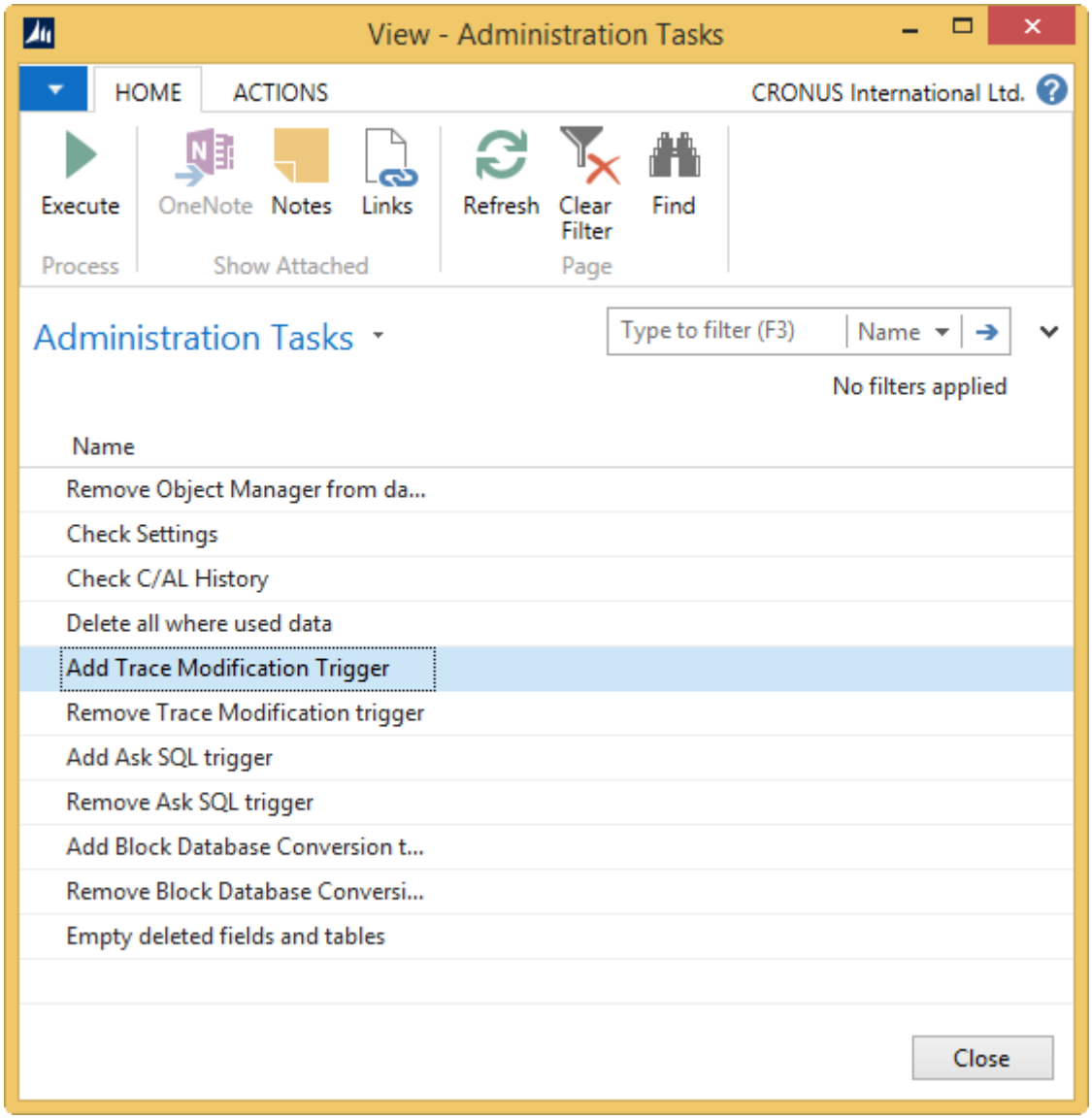
You need to add certain SQL triggers to the Object table of the database in order to use the corresponding features of Object Manager Advanced:

- Trace Modification Trigger enables tracing modifications in database objects.
- Ask SQL Trigger.

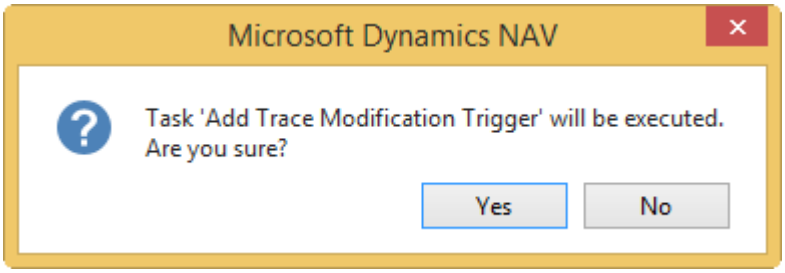
Perform the following steps in Microsoft Dynamics NAV to add the necessary triggers:

1. Go to **Departments > Object Manager > Administration** and then click **Administration Tasks**.
2. In the **Administration Tasks** window, select the task that corresponds to the trigger that you want to add (steps 2-5 should be repeated for each trigger):

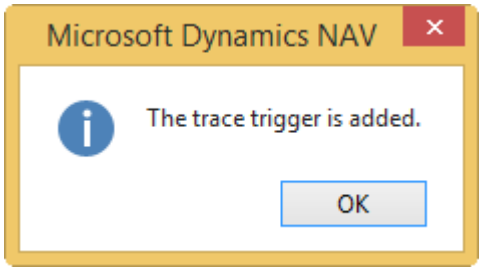
- *Add Trace Modification trigger*
- *Add Ask SQL trigger*
- *Add Block Database Conversion trigger*

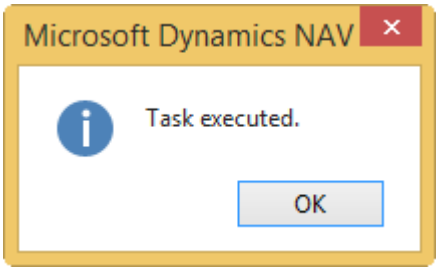


3. Click **Execute** in the ribbon.



4. Confirm the action in the dialog that appears.
   After the trigger is added, the respective messages will be displayed.





You can remove the triggers using the appropriately named administration tasks.

1.3 Removing Object Manager Advanced

To remove Object Manager Advanced from the database, do as follows:

1. Go to **Object Manager > Administration > Administration Tasks**.
2. In the **Administration Tasks** window, select *Remove Object Manager from database* and click **Execute**.
   This action is time consuming.

2 SETUP

This chapter explains how to set up Object Manager Advanced in order to effectively use its features.

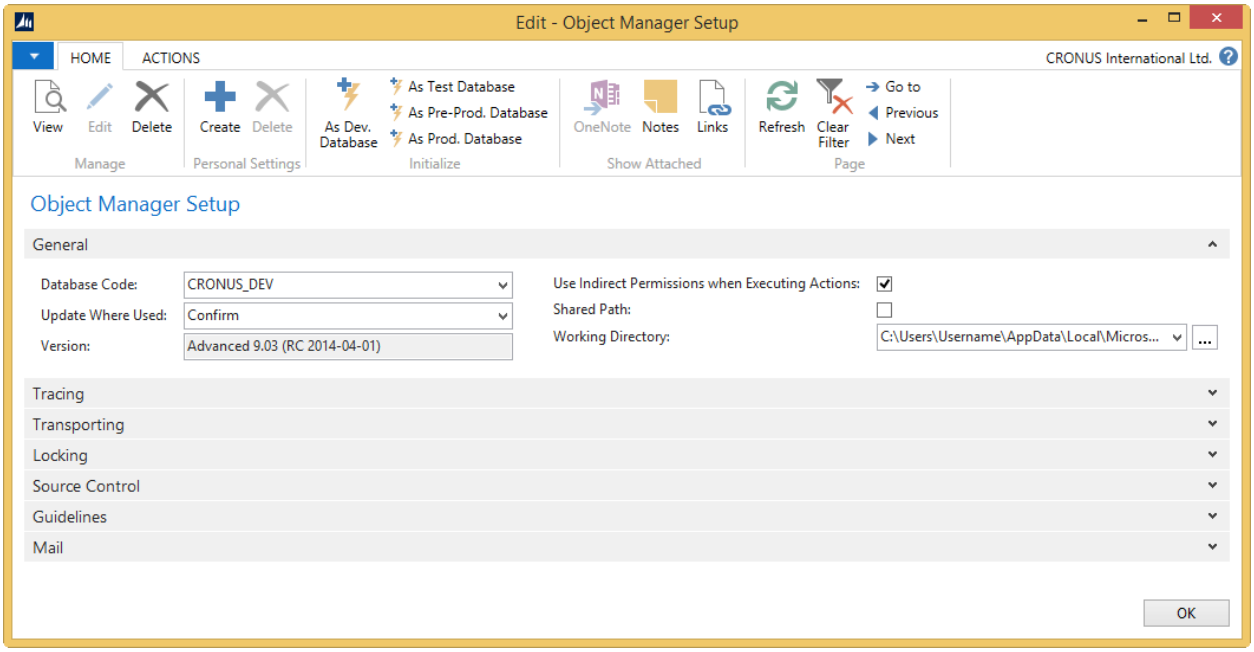
2.1 OBJECT MANGER SETUP

The key settings that define behavior of Object Manager Advanced are located in Object Manager Setup.

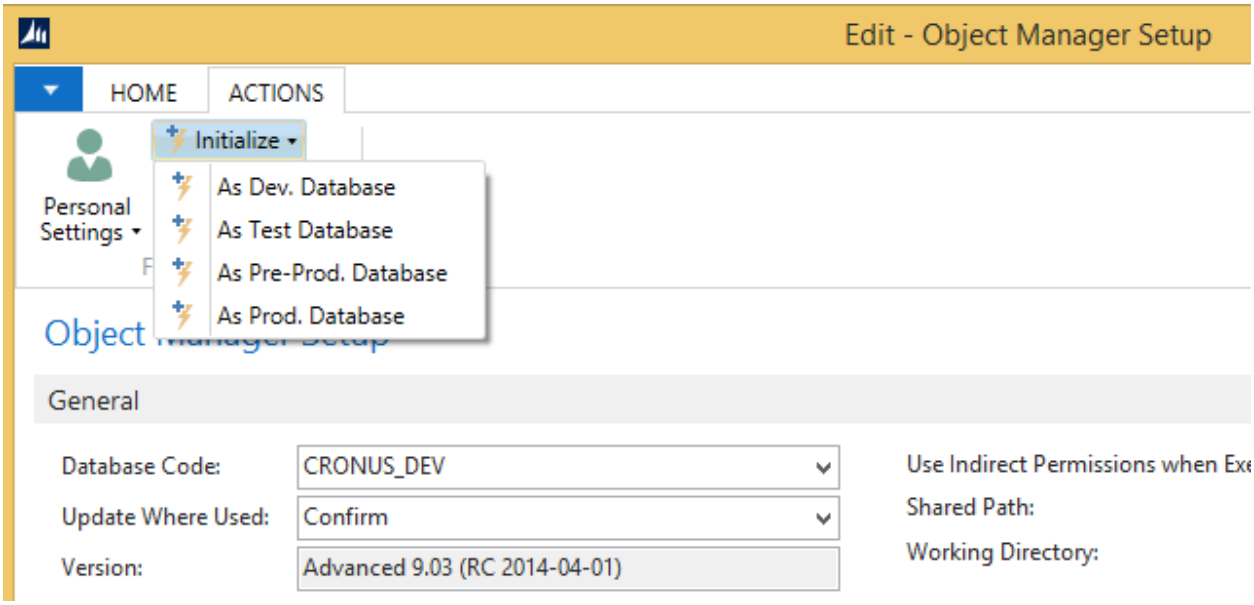
2.1.1 Setup Initialization

To begin using Object Manager Advanced, you need to initialize its setup, that is, provide default settings (statuses, project flows, etc.). Depending on the type of database where you installed Object Manager Advanced, the setup can be initialized with a different collection of settings.

To initialize the setup, open the **Object Manager Setup** window: go to **Departments > Object Manager > Setup** and then click **Object Manager Setup**.



By default, Object Manager Advanced treats the database where it is installed as a development database. If you installed Object Manager Advanced in a test, pre-production, or production database, you can initialize the setup accordingly by using the appropriate button in the **Initialize** group on the ribbon.



Depending on the initialization method, settings in the **Object Manager Setup** window are initialized as follows:

Field Name	Value			
	Development DB	**Test DB**	**Pre-production DB**	**Production DB**
Transporting FastTab				
At Importing Transport (group of fields)				
Confirm Changes	No	Yes	Yes	Yes
Reset Project Status	No	Yes	No	No
Block Project	Blank	No	Yes	Yes
Block Transport	Blank	Yes	No	Yes
Locking FastTab				
Lock Object at Design	Yes	No	No	No
Source Control FastTab				
Save C/AL at (group of fields)				
Modification	If Changed	No	No	No
Assigning	If Changed	No	No	No
Locking	If Changed	No	No	No
Unlocking	If Changed	No	No	No
Transporting	If Changed	No	No	No
Before Import Transport	If Changed	Yes	Yes	Yes
After Import Transport	If Changed	Yes	Yes	Yes

The following setups are initialized in the same way for all database types:

- Number formats and descriptions for projects, transports (**Transporting** FastTab in Object Manager Setup), and branches (**Source Control** FastTab in Object Manager Setup)
- CONSULTANT and DEVELOPER user roles
- DEFAULT project and transport types
- Sample project and transport statuses
- Sample project and transport flows
- Sample comment types

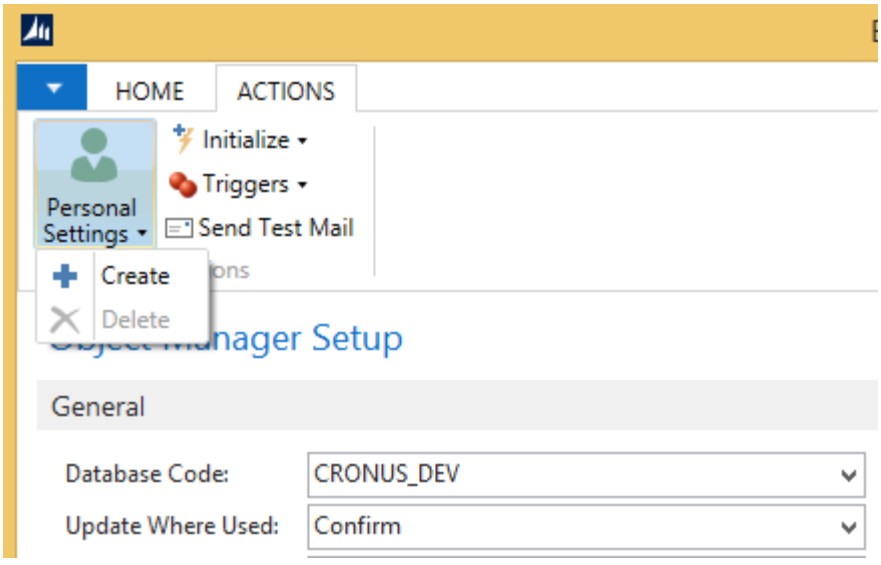
2.1.2 Setup

After you initialize Object Manager Advanced setup, you can proceed to provide settings in the **Object Manager Setup** window.

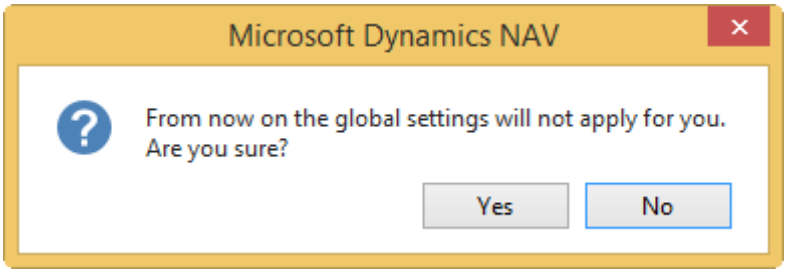
IMPORTANT

WHEN DEVELOPING WITH MULTIPLE DEVELOPERS IN ONE DATABASE, WHERE DIFFERENT MODIFICATION ANALYSIS TOOLS ARE USED SIDE BY SIDE, IT CAN BE NECESSARY TO HAVE PERSONAL SETTINGS THAT WOULD NOT AFFECT OTHER USERS.

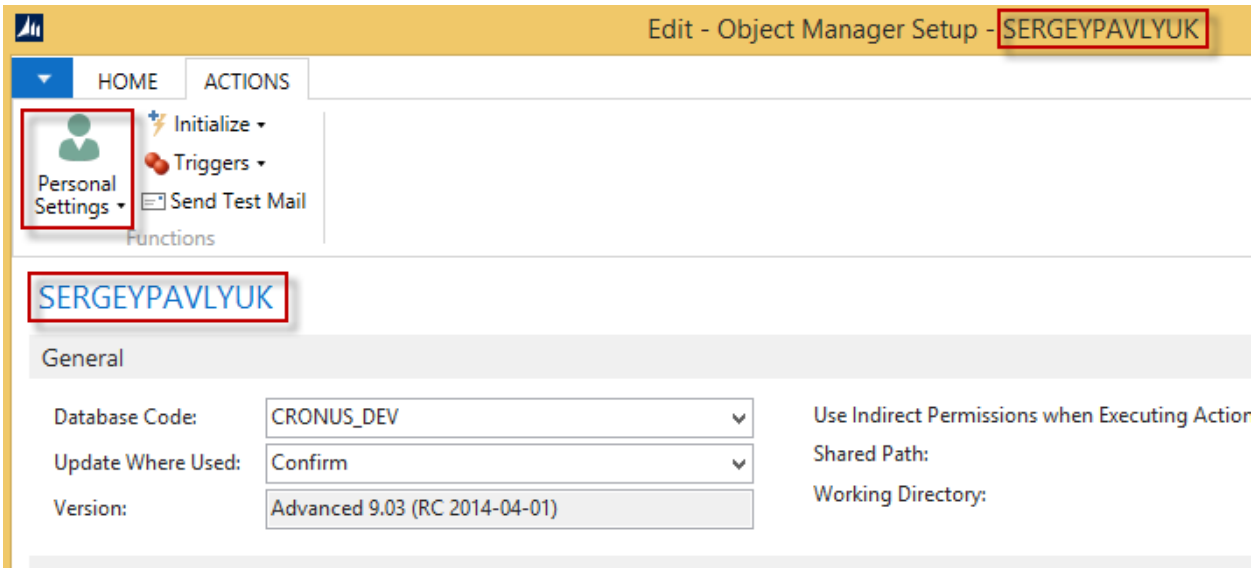
In order to provide separate personal settings for Object Manager Advanced, in **Object Manager Setup** locate **Personal Settings** in the ribbon and click **Create**.



A confirmation message will be displayed.



When you click **Yes**, the title of the **Object Manager Setup** window is updated with your user code, indicating that the settings you provide in this window will only take effect when you are the current Microsoft Dynamics NAV user.



To disable your personal settings, click **Delete** in the **Personal Settings** group in the ribbon.

Field	Description
General FastTab	
Database Code	(Optional) This field can be used to select the database for Object Manager Advanced to work with. By default, this field contains the name of the database where you installed Object Manager Advanced. Before you can select a database, you need to set up the database list.
Update Where Used	Choose how the program must update the Where Used data for Object Manager. Possible options are: *Confirm* – the user will be prompted to update data when needed. *Always* – the data will be updated automatically. *Never* – the data will not be updated by the program.
Version	This field displays the version of Object Manager Advanced that you installed.
Use Indirect Permissions	Select this checkbox to enable modification of data in tables that have indirect permissions.
Shared Path	Object Manager Advanced would assume that client and server are located on the same machine. This setting is used to skip three-tier routine when working with files. Select this checkbox to use the working folder as shared folder, instead of the TMPDIR from NAV.
Working Directory	Enter a path to the directory where Object Manager Advanced will store temporary files.
Transporting FastTab	
Transport (group of fields)	Use these fields to define the number series format, description, default type for transports and paths to import and archive folders.
Project (group of fields)	Use these fields to define the number series format, description, and default type for projects.

Roles (group of fields)	Use these fields to define role shortcuts.
At Importing Transport (group of fields)	Define actions that the program will perform when importing transports by placing checkboxes in the respective fields: **Confirm Changes** – the program will display a confirmation in case objects in transport conflict with objects in the target database. **Reset Project Status**, **Reset Transport Status** – the program will set the project/transport status to the initial status in the project/transport flow. **Compile Objects** – the program will compile objects in the transport. Use the **Block Project** and **Block Transport** option fields to influence project and transport blocking when importing transports: *Blank* – the project/transport will be the same as in the imported transport. *No* – the project/transport will be unblocked. *Yes* – the project/transport will be blocked.
Locking FastTab	
Database Locked	Select this checkbox to lock all database objects.
Lock Object at Design	Select this checkbox to have the program lock an object when you design it without OMA.
Lock Marking	Select this checkbox to warn user that the marking functionality is used by other user at that moment.
At Modifying Object (group of checkboxes)	Place check marks in these fields to make the program perform the corresponding action when you modify an object: Display an error if the object is not locked Display an error if the object is locked by another user Lock the object When you select the **Error if Not Locked** checkbox, the **Error if Locked by Other** checkbox is selected as well and cannot be cleared.
NOTE *FOR MORE INFORMATION ABOUT MARKING AND LOCKING, REFER TO THE MARKING AND LOCKING SECTION OF THE OBJECT EXPLORER CHAPTER LATER IN THE GUIDE.*	
Source Control FastTab	
Executables(group of fields)	Use these fields to provide full paths to the following files: • Compare tool and external C/AL editor executables • Client and server versions of the *finsql* Microsoft Dynamics NAV executable • Zup file ID
Branches (group of fields)	Provide a number series format and a description for branches.
Save C/AL at (group of fields)	Use these fields to set up how the program will save the code of an object to C/AL history when you perform various actions: • Modify the object • Assign modifications to a project • Lock or unlock the object • Transport the object • Before and after you import a transport For each action, you can select three saving options: *If Changed* – code will only be saved if the object was changed (date or version list differ).

	Yes – code will always be saved for the action. *No* – code will not be saved for the action.
Mail FastTab	
Mail Type	Choose whether Object Manager Advanced must send email notifications through Outlook or SMTP.
SMTP (group of fields)	If you select the *SMTP* mail type, provide the appropriate credentials.
NOTE *TO TEST THE MAIL SETUP, CLICK SEND TEST MAIL ON THE ACTIONS TAB OF THE RIBBON. THE PROGRAM WILL SEND A TEST EMAIL TO THE CURRENT MICROSOFT DYNAMICS NAV USER.*	

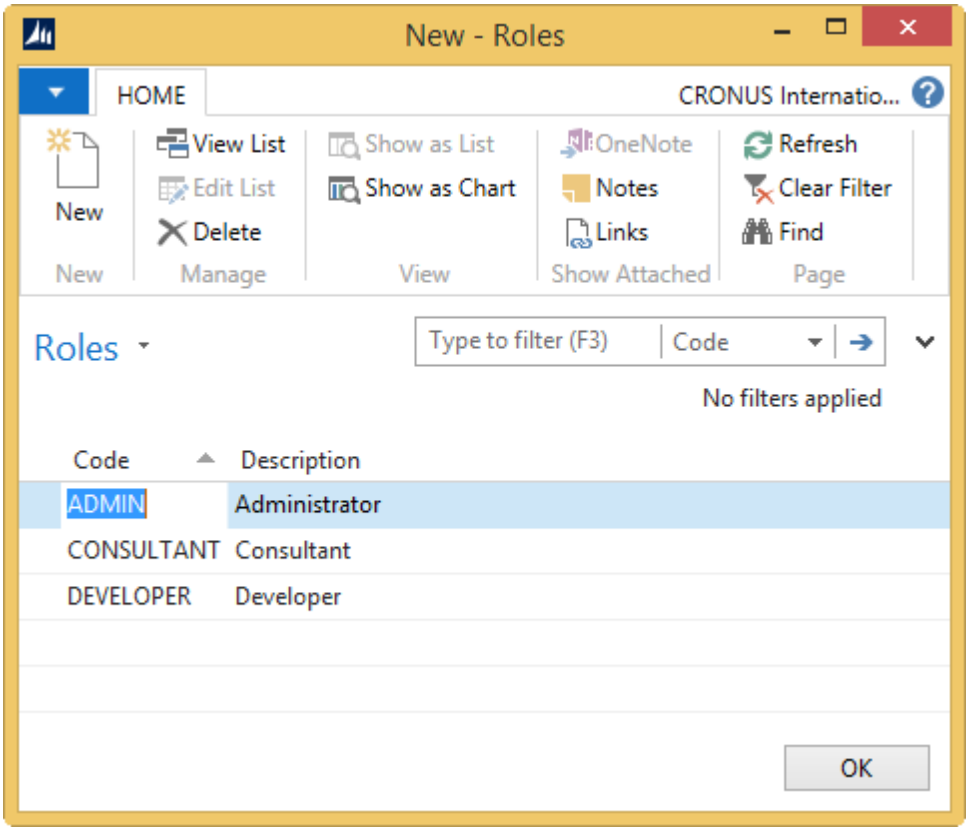
2.2 Users and Roles

Object Manager Advanced requires that you specify the roles that users can perform in the add-on and provide the list of these users.

2.2.1 Roles

To define user roles, go to **Departments > Object Manager >Setup > Roles**. In the window that appears, enter the appropriate roles and their descriptions.

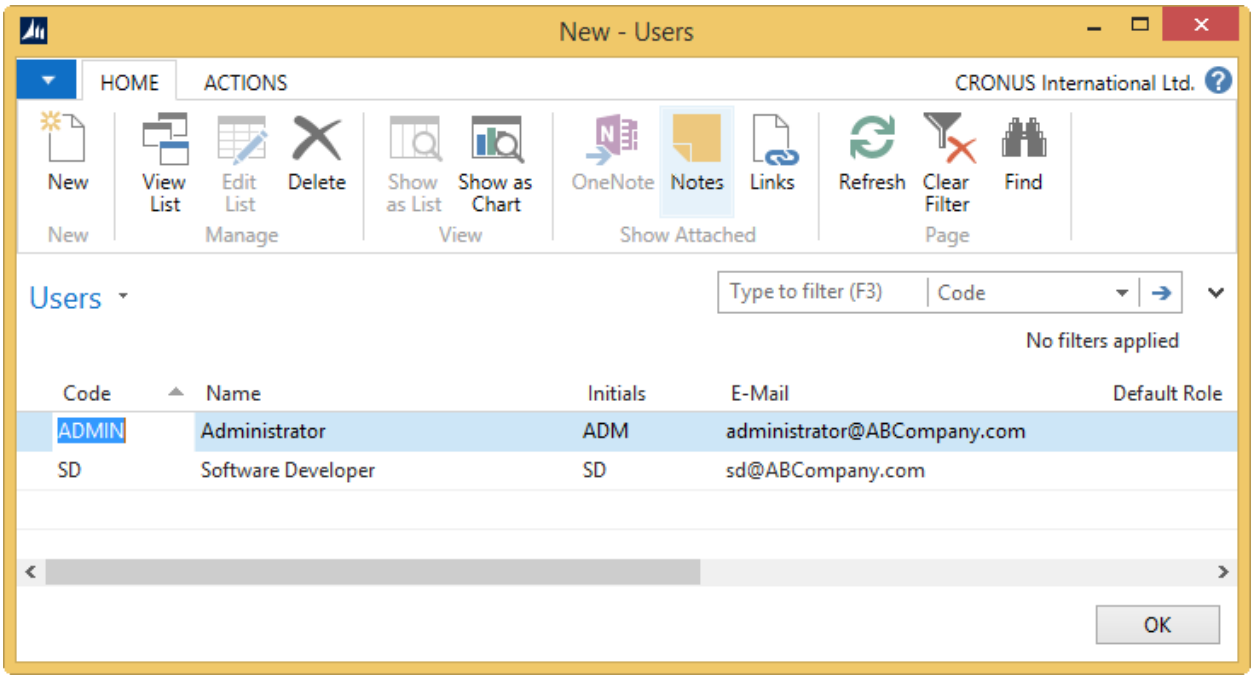
Object Manager Advanced includes the Consultant and Developer sample roles.



2.2.2 Users

To set up users, go to **Departments > Object Manager > Setup > Users**. In the window that appears, fill in the fields as follows:

Field	Description
Code	Enter a code for the user.
Name **Initials** **E-mail**	Enter the user's name, initials, and e-mail, respectively.
Default Role	A user can have a default role. This is used when you insert a new project. The code of the active user is placed in the corresponding role on the project card.
My Objects Version List Filter	Users can specify filters that identify a specific set of objects.
Short User ID **Capitalized User ID**	Enter the Microsoft Dynamics NAV user ID that corresponds to the current user. The capitalized user ID will is used in the 'Locked by' field in the object table.

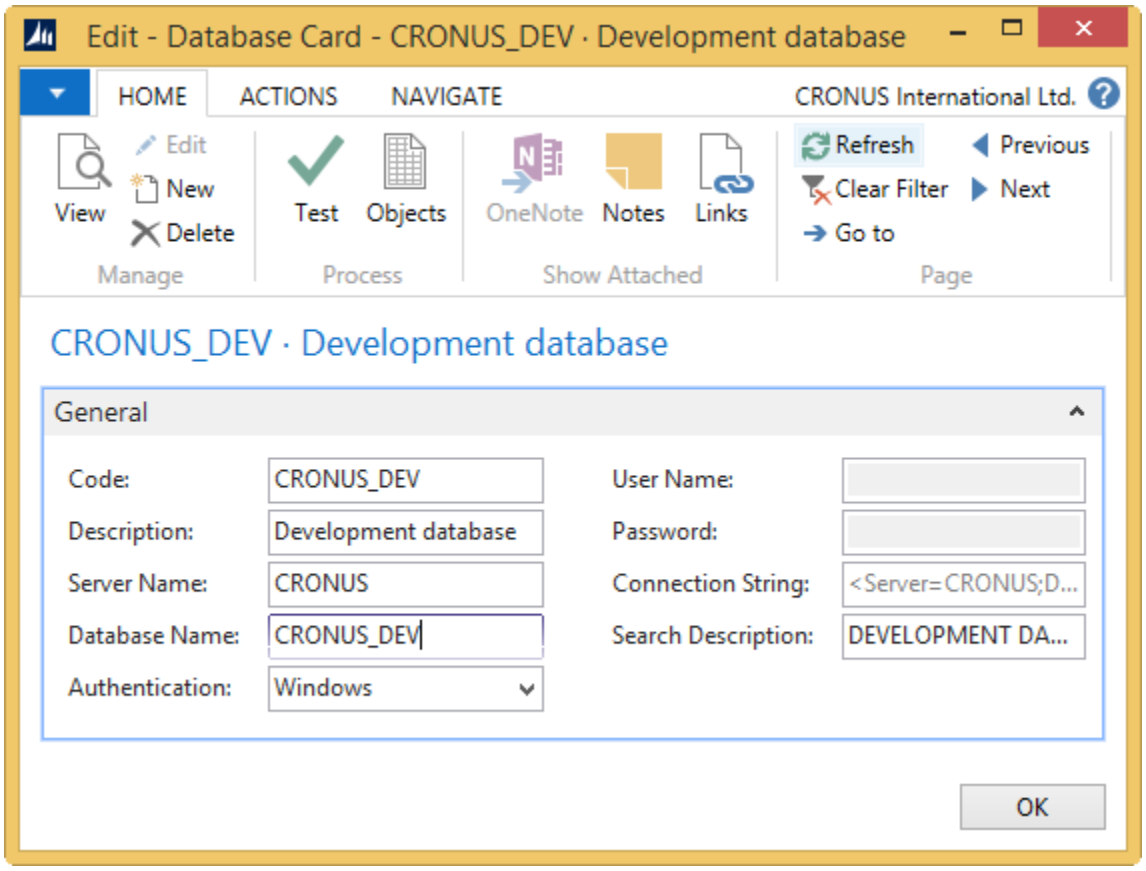


2.3 DATABASES

To be able to work with several databases in Object Manager Advanced, you need to define the list of databases with the appropriate connection credentials. To do this, perform the following steps:

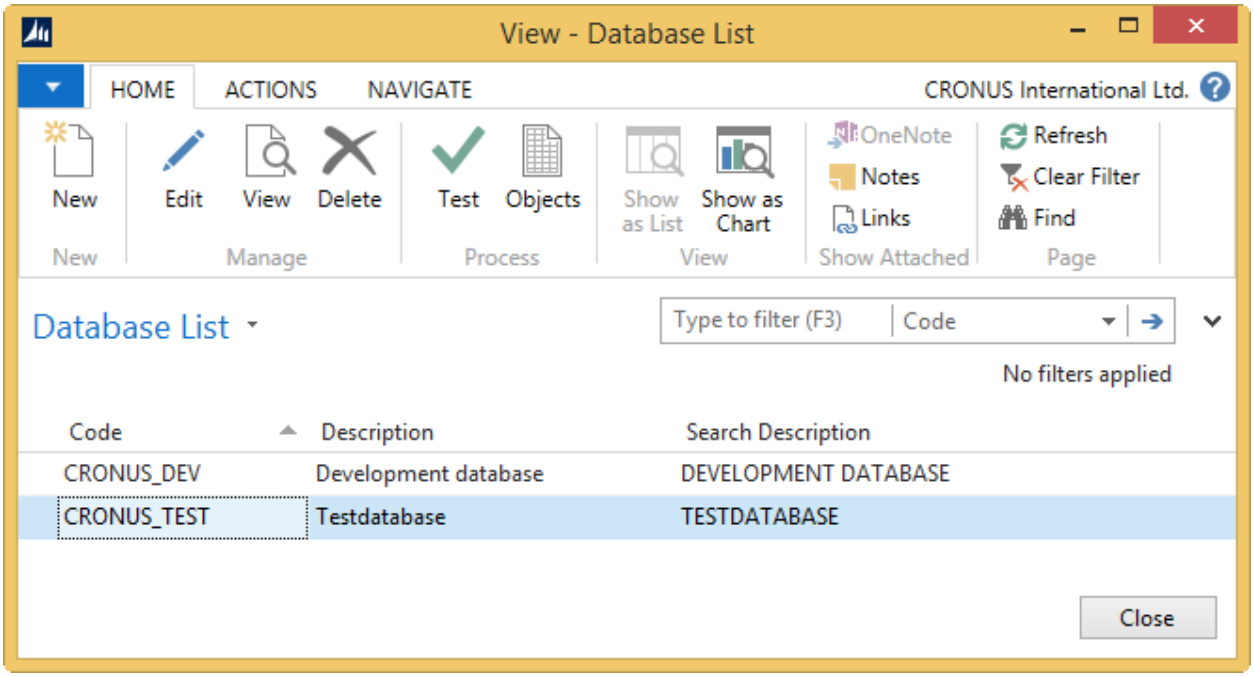
1. Go to **Departments > Object Manager > Setup** and then click **Databases**.
2. Click **New** in the ribbon to create a record.
3. Fill in the fields on the database card as follows:

Field	Description
Code Description **Search Description**	Enter a code and a description for the database, respectively. The search description will be updated accordingly.
Server Name **Database Name** **Connection String**	Enter the name of the server and database. The connection string will be updated accordingly.
Authentication	Select the authentication type of the database. Possible options are *Windows* and *User Name*.
User Name **Password**	If the current database's authentication type is *User Name*, enter the corresponding user credentials in these fields.



- To verify the connection to the database, click **Test** in the ribbon.
- To view the list of objects in the database, click **Objects** in the ribbon.

4. Return to the database list and create all the appropriate databases.



NOTE: *THE DATABASES THAT YOU ADD TO THE LIST CAN ONLY BE USED FOR DATABASE COMPARISON. OBJECT MANAGER ADVANCED CANNOT FULLY WORK WITH DATABASES WHERE IT IS NOT INSTALLED.*

2.4 Project and Transport-Related Setup

To effectively manage projects and transports in Object Manager Advanced, you need to set up the following entities:

- Statuses
- Flows
- Types
- Comment types

Note

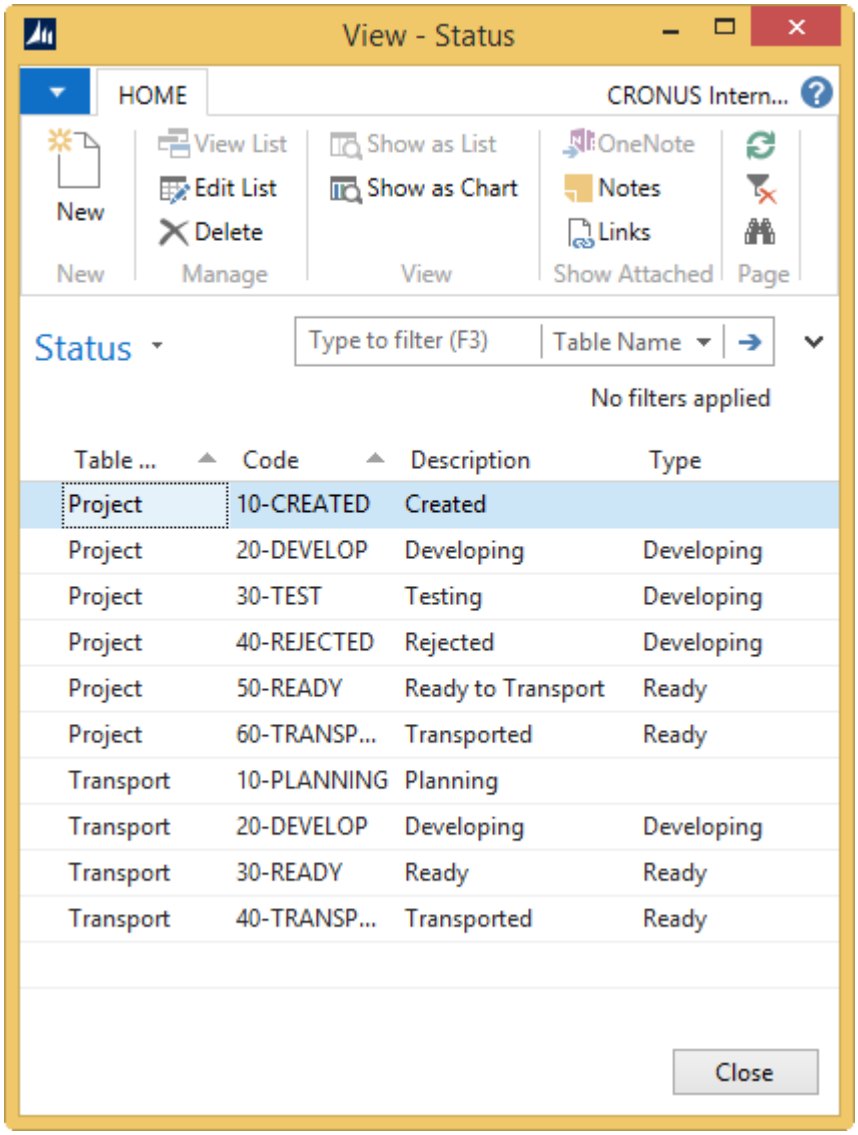
The use of projects and transports will be described in the corresponding subsequent chapters of this guide.

2.4.1 Statuses

Project and transport statuses help you track the state of projects and transports.

To define statuses, go to **Departments > Object Manager > Setup > Status Setup**.

Object Manager Advanced includes the following sample statuses:



You can use the statuses provided, adjust them as necessary, or enter lines for new project and transport statuses by filling in the fields in the window as follows:

Field	Description
Table Name	Choose whether the current status will apply to projects or transports.
Code **Description**	Enter a code and a description for the current status, respectively.
Type	Select the type of status. The type corresponds to the state of the project or transport. Possible options are: • *Blank* – for new projects and transports. • *Developing* – for projects or transports in development. • *Ready* – for completed projects and transports.

Note

You need to set up a number of statuses that can be used in a sequence.

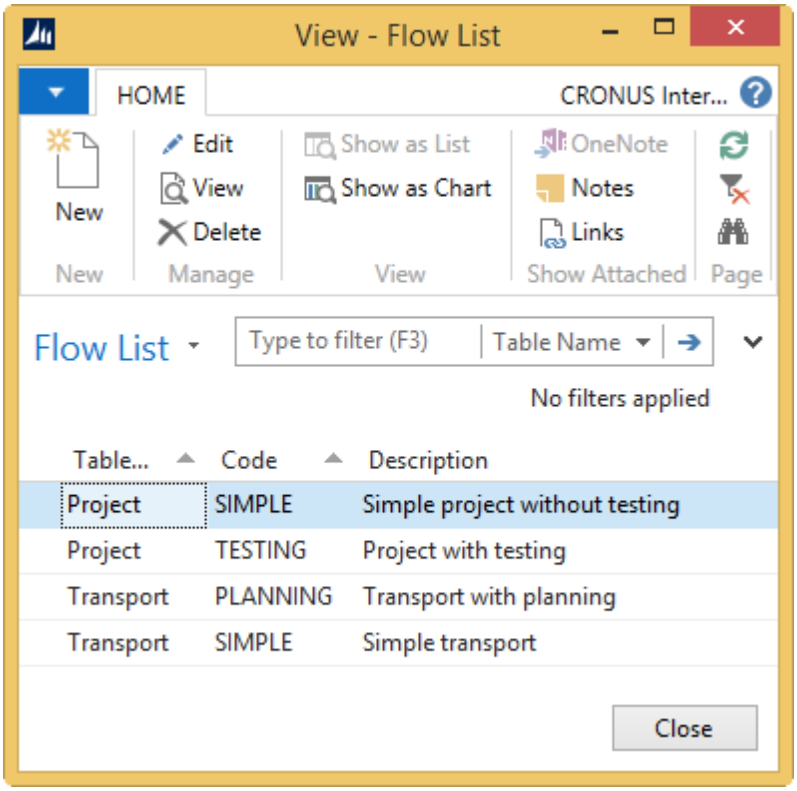
After you set up statuses, you will be able to use them to set up project and transport flows.

2.4.2 Flows

Project and transport flows help you set up the process of working with projects and transports. Using flows, you can define the sequence of statuses that a project or a transport passes, assign user roles to the appropriate statuses, and set up e-mail notifications for when status is changed.

To set up project and transport flows, go to **Departments > Object Manager > Setup > Flows**.

Object Manager Advanced includes the following sample flows:



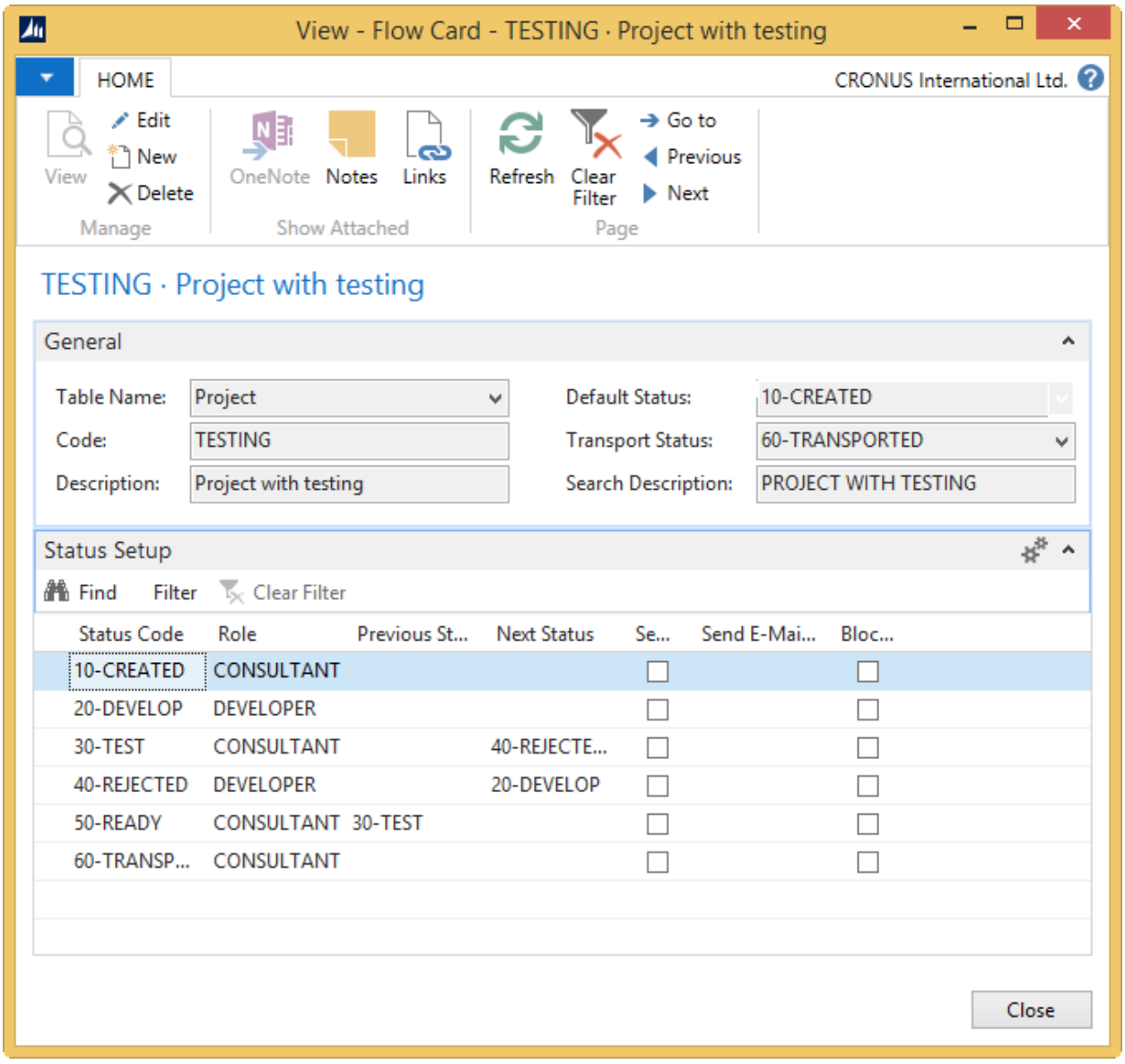
You can use these flows, modify them, or define new ones.

- To edit a flow, double-click the needed line or click **Edit** in the ribbon.
- To create a flow, click **New** in the ribbon.

Fill in the fields on the flow card as follows:

Field	Description
General FastTab	
Table Name	Choose whether the current flow will apply to projects or transports.
Code Description **Search Description**	Enter a code and a description for the current flow, respectively. The search description will be updated accordingly.
Default Status	Define the default status for the current flow.
Transport Status	Define the final (transporting) status for the current flow.
Status Setup FastTab	

Enter lines for project or transport statuses that you want to include in the current flow.	
Status Code	Enter the status code.
Role	Specify the role that is allowed to see the project or transport statuses
Previous Status **Next Status**	Enter the codes of statuses that precede and follow the current status, respectively.
Send E-Mail to Active User	Select this checkbox to send an email notification to the project/transport active user when the current status is reached.
Send E-Mail to Roles	Specify additional roles to whom an email notification will be sent when the current status is reached.
Block Project/Block Transport	Select this checkbox to block the project or transport when the current status is reached.



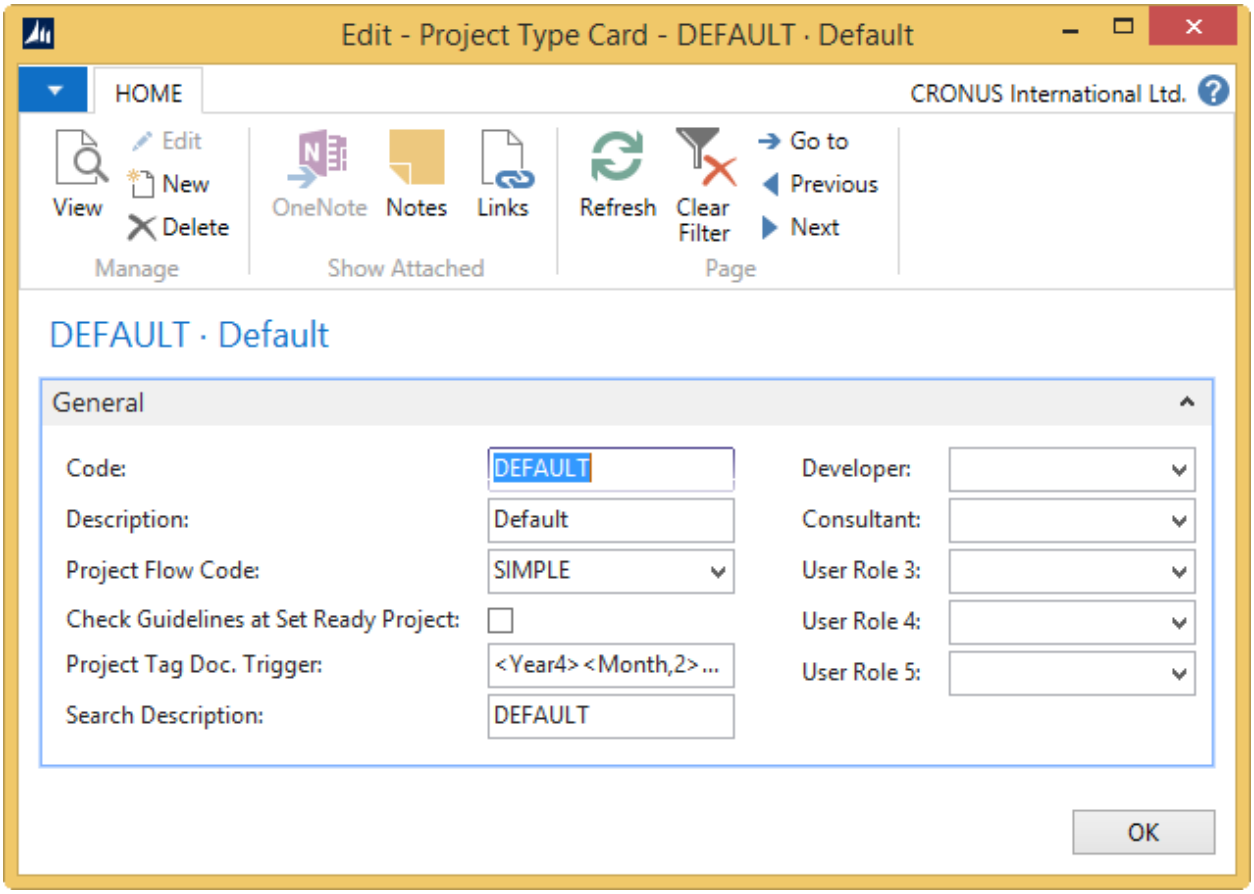
2.4.3 Types

Project and transport types allow you to distinguish between different kinds of projects and transports, as well as define various general project or transport settings by type.

Project Types

To define project types, go to **Departments > Object Manager > Setup > Project Types**. Create a record and fill in the fields as follows:

Field	Description
Code Description **Search Description**	Enter a code and a description for the project type. The search description will be updated accordingly.
Project Flow Code	Specify a project flow code for the project type.
Check Guidelines at Set Ready Project	Select this checkbox to have the program perform guideline checks when projects of the current type reach a status of type *Ready*.
Project Tag Doc. Trigger	Define the project tag that will be added to the Documentation trigger of objects assigned to projects of the current type.You can use date expressions and the following placeholders: • %1 for project number • %2 for active user's initials • %3 for description of the modification
User Role 1..User Role 5	Specify users that will typically have roles in projects of the current type.



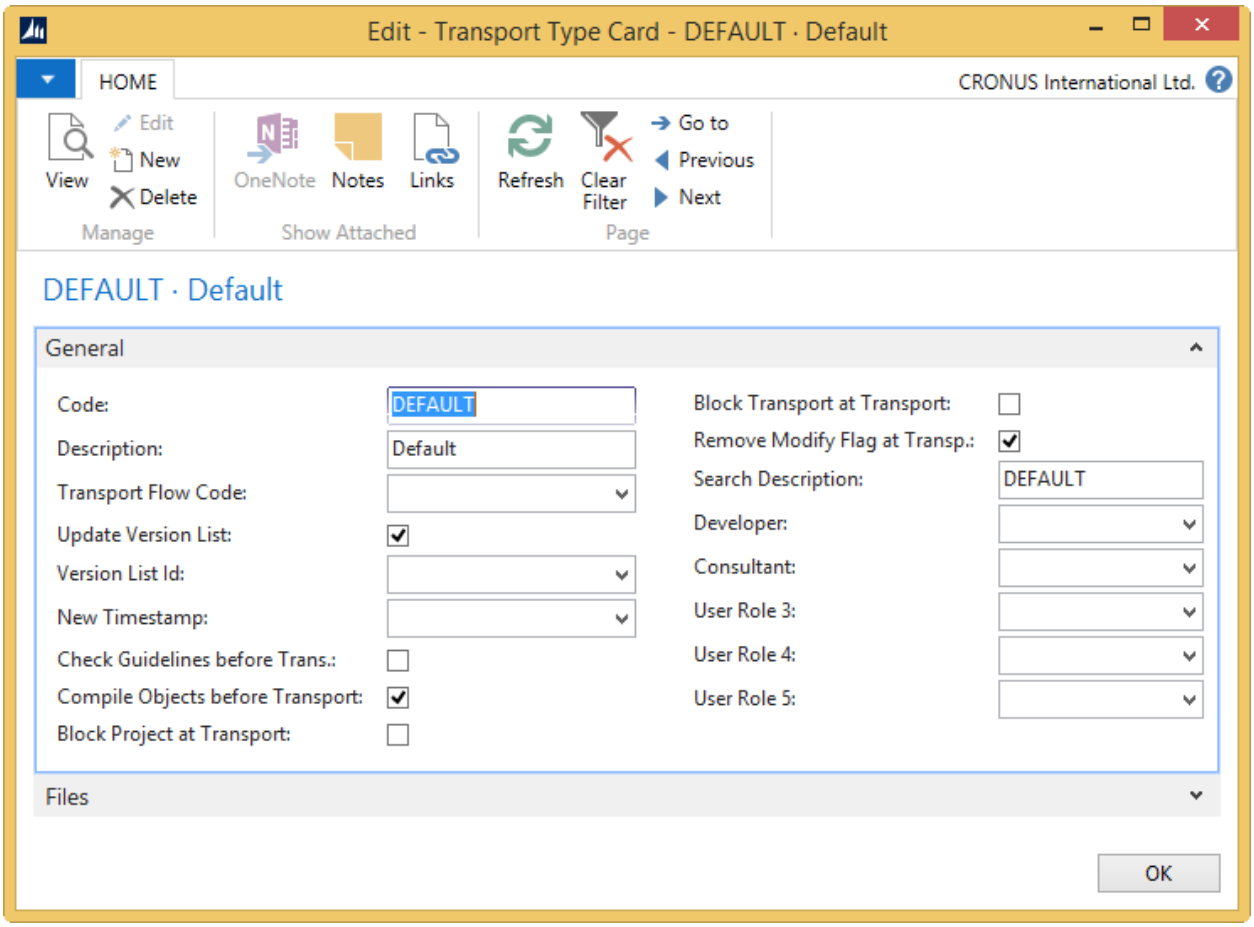
Create as many project types as needed.

Transport Types

To define transport types, go to **Departments > Object Manager > Setup > Transport Types**. In the window that appears, create a record and fill in the fields as follows:

Field	Description
General Tab	
Code **Description** **Search Description**	Enter a code and a description for the transport type. The search description will be updated accordingly.
Transport Flow Code	Specify a transport flow code for the transport type.
Update Version List	Select this checkbox to have the program update the version list of objects in transports of the current type with an ID that you specify.
Version List ID	Select a version list ID for objects in transports of the current type. Use the AssistButton next to the field to browse version list IDs in the database. Alternatively, enter a string that will be used as the version list ID.
New timestamp	Select the definition of timestamp as a date of transport or moment of transport. Use the AssistButton next to the field to browse through the options available. Alternatively, enter a string that will be used a template.
Check Guidelines before Trans. **Compile Objects before Transport** **Block Project at Transport** **Block Transport at Transport** **Remove Modify Flag at Transp.**	Place check marks in these fields to have the program perform the corresponding action when performing transports of the current type. **Note** The program can only check guidelines if there are no unknown comments. If you choose to block projects or block the transport at transporting, you will not be able to unblock projects or transport after transport is completed.
User Role 1..User Role 5	Specify users that will typically have roles in transports of the current type.
Files Tab	
Export Path	Enter a path to the folder where transport files will be saved.
Subfolder for Each Transport **Subfolder Name**	Select this checkbox to have the program create subfolders for each transport of the current type. Otherwise, all transports will be saved in a single folder. If you choose to create subfolders, enter a subfolder name using date expressions and the following placeholders: • %1 for transport number • %2 for version list ID • %3 for version list number
Include TXT-Objects in Transport	
Create HTML File **Create FIB File** **Create OBJ File** **Create TXT File** **Create FAB Files**	Place check marks in these fields to have the program create the corresponding transport files: • HTML – a report with transport information. • FIB – all transport and project data, actions, document and objects in transport. • OBJ – objects. • TXT – objects in text format. • FAB – all project and transport data, actions to be executed before and after transporting.

Filename HTML File **Filename FIB File** **Filename OBJ File** **Filename TXT File** **Filename FAB Definition File** **Filename FAB Before File** **Filename FAB After File**	Enter names for transport files that you choose to create. Similarly to filling in the **Subfolder Name** field, you can use date expressions and placeholders.



Create as many transport types as needed.

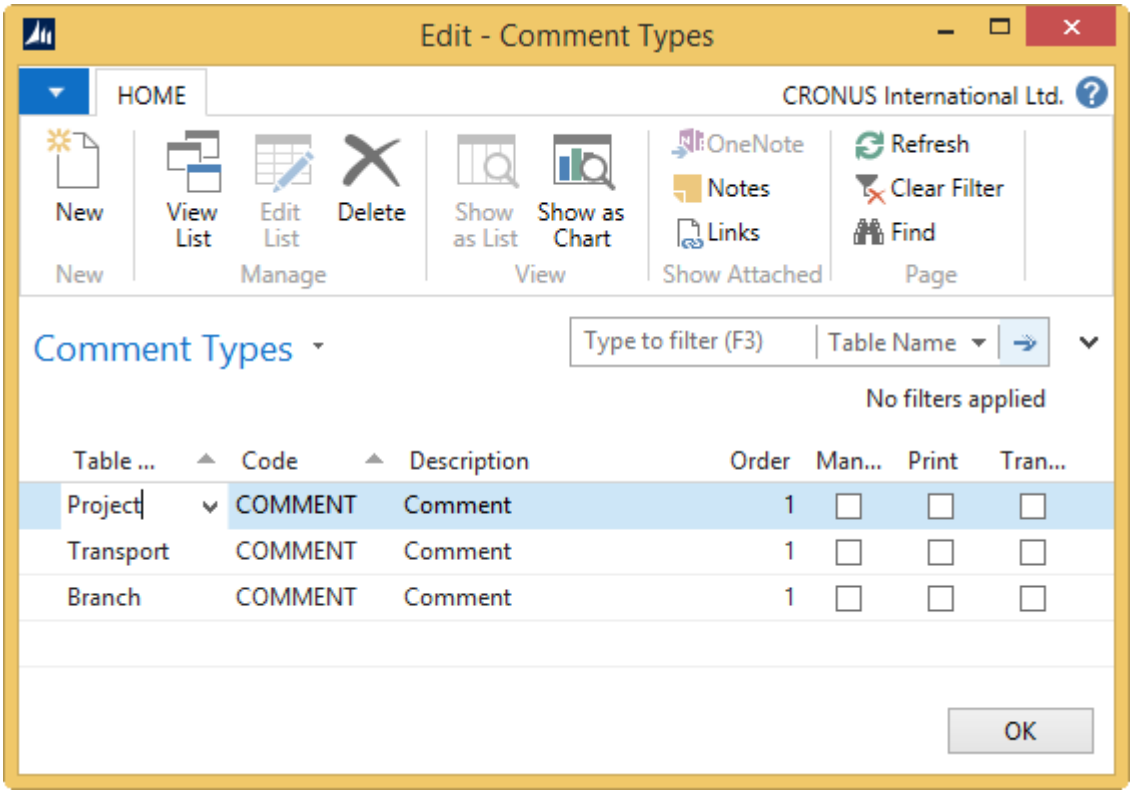
2.4.4 Comment Types

Comment types help you organize comments for projects, transports, and branches.

To define comment types, go to **Object Manager > Setup > Comment Types**. In the window that appears, enter lines for comment types by filling in the fields as follows:

Field	Description
Table Name	Choose whether the current comment type will apply to projects, transports, or branches.
Code **Description**	Enter a code and a description for the comment type.
Order	Use this field to define the order of comment types if you set up several comment types for the same entity (project, transport, and branch).

Mandatory	Select this checkbox to indicate that users must leave comments of the current type before a project can be transported / transport performed.
Print	Select this checkbox to include comments of the current type in the project or transport report.
Transport	Select this checkbox to include comments of the current type in transport files.

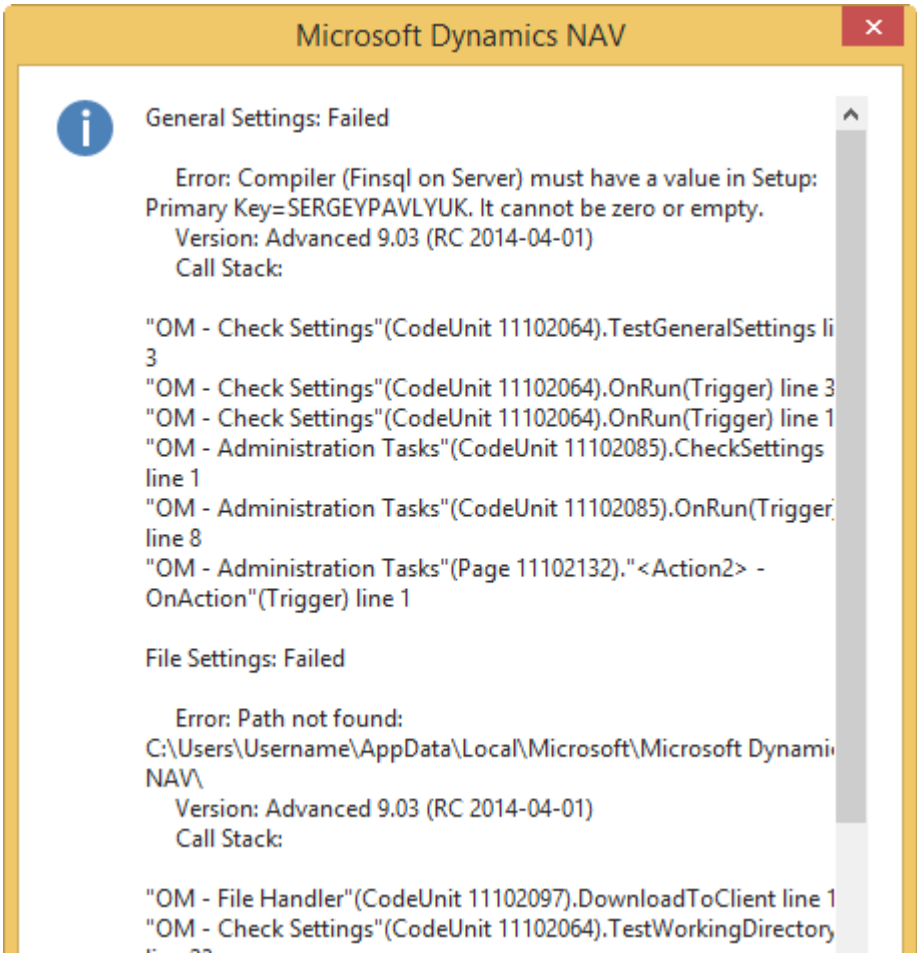


2.5 Setup Check

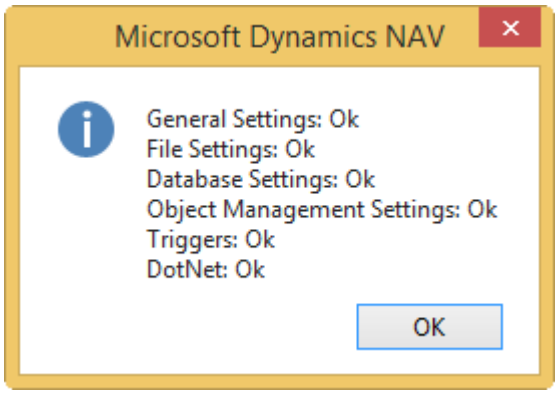
To ensure that you have provided all the settings necessary for Object Manager Advanced to work, perform the following actions:

1. Go to **Departments > Object Manager > Administration** and then click **Administration Tasks**.
2. Select *Check Settings* and then click **Execute** in the ribbon. Confirm the action in the dialog that appears.

In case there are errors in Object Manager Advanced setup, a message with error details will be displayed.



If there are no errors, the following message will be displayed:



3 OBJECT EXPLORER

The object explorer is the main tool of Object Manager Advanced. It combines the functionality of the classic Object Designer available in the Microsoft Dynamics NAV development environment with various additional functions that help you work on objects. The object explorer also provides shortcuts to analysis tools of Object Manager Advanced and offers a summary of work that you perform over objects in the database.

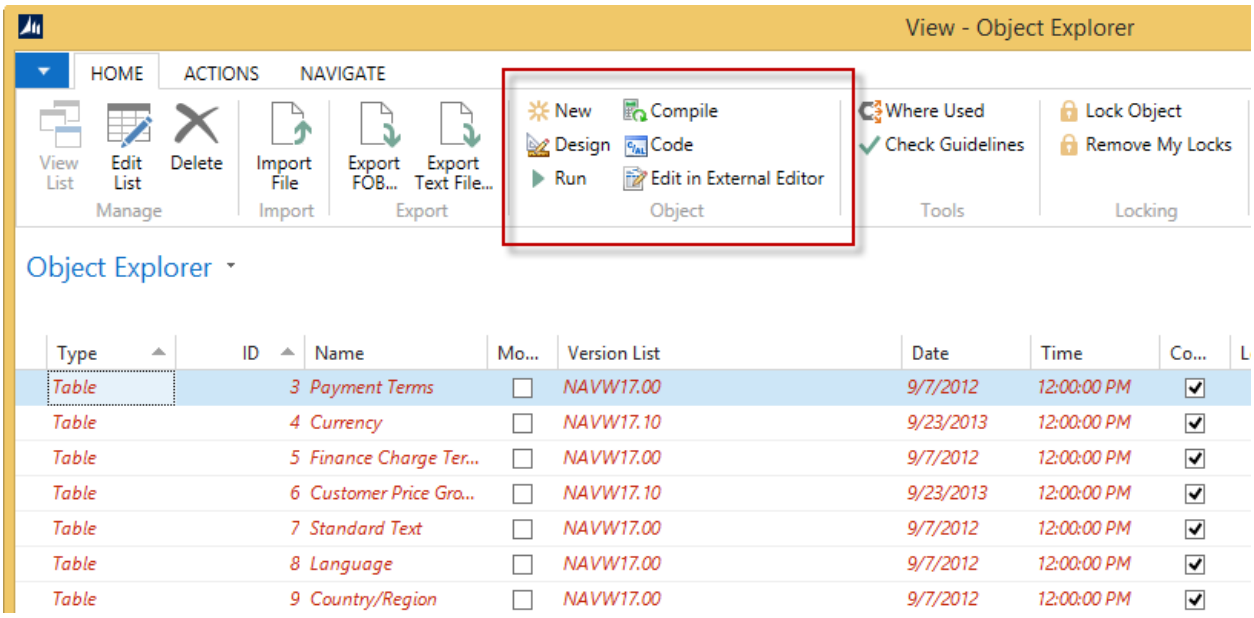
This chapter describes the basic functions that you can perform over objects in the object explorer and explains how to use the object explorer for export and import. The chapter also introduces you to object marking and locking, and describes the object modification tools available in the object explorer.

The use of analysis tools and standalone features available in Object Manager Advanced will be described in subsequent chapters.

> **!! IMPORTANT**
> Before you perform any actions in Object Manager Advanced, you need to update C/AL history and Where Used data of objects in the database. The program will prompt you to do this when needed, or you can update the data manually.

3.1 BASIC FUNCTIONS

The object explorer retains the basic functions of the object designer in the Microsoft Dynamics NAV development environment – creating, designing, compiling and running objects. To perform these functions, go to **Departments > Object Manager > Objects > Object Explorer** use the respective buttons in the **Object** group on the **Home** tab of the ribbon.



- To create an object, select an existing object of the desired type and then click **New**. The development environment will open with the corresponding object type designer window.

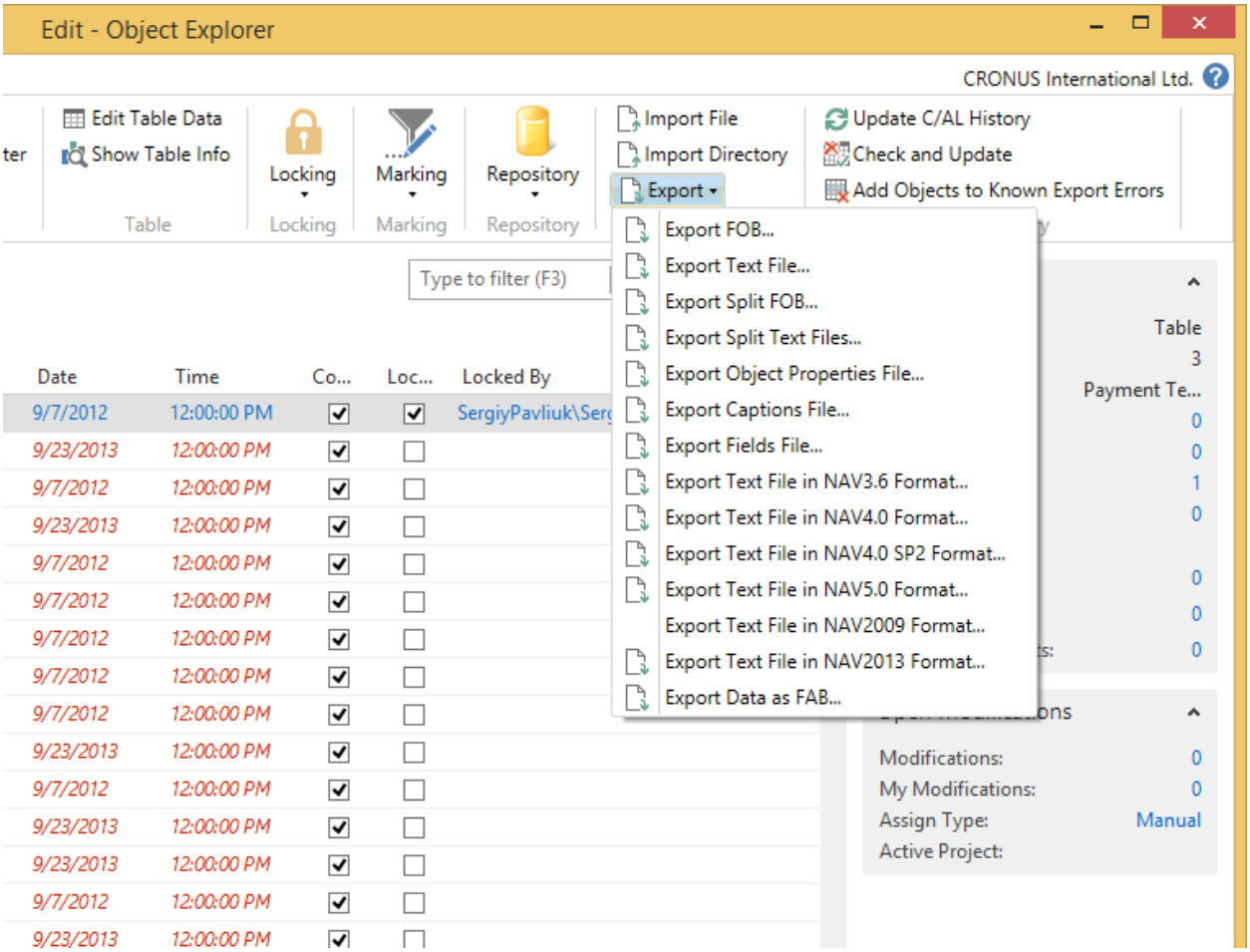
- To design an object, select the needed object and then click **Design**. The object will be opened in design mode in the development environment.
- To compile or run an object, select the needed object and then click **Compile** or **Run**, respectively.

3.2 Export and Import

Using the object explorer, you can perform export and import of objects in various formats.

3.2.1 Export

The object explorer offers the following options for exporting objects:



To use these options, select one or more objects that you want to export and go to the **Actions** tab of the ribbon; in the **Export/Import** group, click **Export** and then click the appropriate item:

Export Option	Description
Export FOB	Export selected object(s) to a .fob file.
Export Text File	Export selected object(s) to a .txt file.
Export SplitFOB	Export each of the selected objects to a separate .fob file.

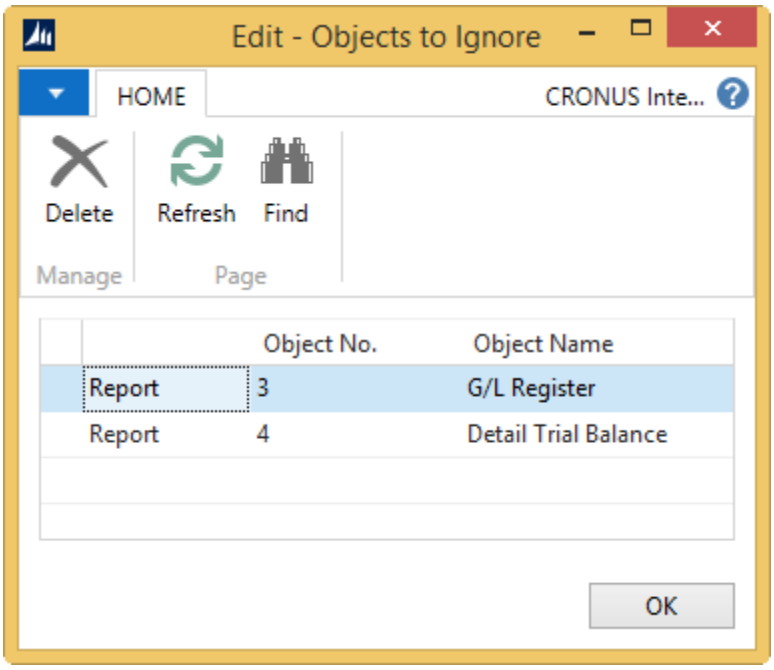
Export Split Text Files	Export each of the selected objects to a separate .txt file.
Export Object Properties File	Export properties of the selected object(s) to a .obp file.
Export Captions File	Export captions contained in the selected object(s) to a .txt file.
Export Fields File	Export fields of the selected table(s) to a .fld file.
Export Text File in NAV 3.6 Format..Export Text File in NAV2013 Format	Export selected objects to a .txt file for the corresponding version of Microsoft Dynamics NAV.
Export Data as FAB	Export selected objects to a .fob file

When you click an item, the program will prompt you to browse to a location and provide a name for the resulting file.

3.2.2 Known Export Errors

In case there are export limitations on some objects in the database, you can add these objects to know export errors to have the program ignore them during export of multiple objects. To do this, select the appropriate objects in the object explorer and then click **Add to Known Export Errors** on the **Actions** tab of the ribbon. The program will prompt you to confirm this action.

You can view the list of objects, as well as add or remove objects to ignore, go to **Departments > Object Manager > Setup** and then clicking **Objects to Ignore**.



3.2.3 Import

From the objects explorer, you can initiate import of objects contained in a single file or in several files stored in a directory. To do this, go to the **Actions** tab of the ribbon and in the **Ex-/Import** group, click **Import File** or **Import Directory**, respectively. The program will prompt you to browse to the needed file or directory;

when you do so, the import objects worksheet will be opened, and you can proceed to perform the import.

3.3 Marking and Locking

The object explorer allows you to enhance your work on objects with marking and locking.

3.3.1 Marking

Marking helps you bring out objects that you intend to work with.

You can mark objects in the object explorer in either of the following ways:

- Select one or more lines and click **Mark Selection** on the **Actions** tab of the ribbon.
- Have the program mark objects contained in a file by clicking **Mark Objects with Import File** on the Actions tab of the ribbon and then browsing to the appropriate object file in FOB, TXT, FIB or OBJ format.

When you mark an object, its version list is updated with # at the beginning.

To remove marks, use the buttons on the **Actions** tab of the ribbon in the object explorer:

- Select one or more marked objects and then click **Remove Marks** to remove marks from the selected objects.
- Click **Remove All Marks** to remove marks from all database objects.

Note
IN CASE MARKING IS LOCKED IN OBJECT MANAGER SETUP, OTHER USERS CANNOT REMOVE MARKS PLACED BY YOU, AND YOU CANNOT REMOVE MARKS PLACED BY OTHER USERS.

You can also mark and unmark objects when using the following Object Manager Advanced tools:

- Project and transport cards
- Import Objects worksheet
- Search String in C/AL Code
- Compare tools
- Check License
- Translation tool
- C/AL History

NOTE
The use of these tools is described in the appropriate sections later in the guide.

3.3.2 Locking

Locking prevents other users from modifying objects that you currently work with.

In Object Manager Advanced, there are several ways to place locks on objects:

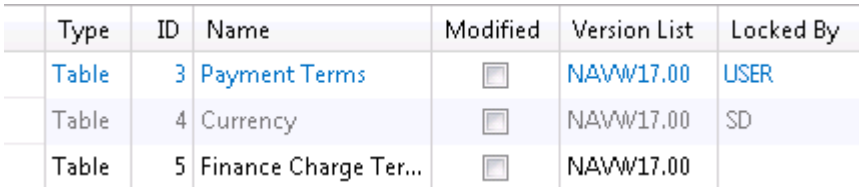
- Lock the entire database in Object Manager Setup.
- Have the program place a lock on an object when you perform any modification over it (design the object in the development environment,

modify code or properties, run object tools) using the appropriate fields on the **Locking** FastTab in Object Manager Setup.

- Place a check mark in the **Locked** field on the object line in the object explorer.
- Select one or more objects in the object explorer and then click **Lock Object** in the ribbon.
- Have the program lock objects contained in a file by clicking **Locking > Lock Objects with Import File** on the Actions tab of the ribbon in the object explorer and then browsing to the appropriate object file in FOB, TXT, FIB or OBJ format.

When you lock an object, other users can only open it or view its properties in read-only mode and cannot make any modifications to it. Similarly, you cannot modify an object that is locked by another user.

Lines in the object explorer for objects locked by you and other users are displayed in different colors:

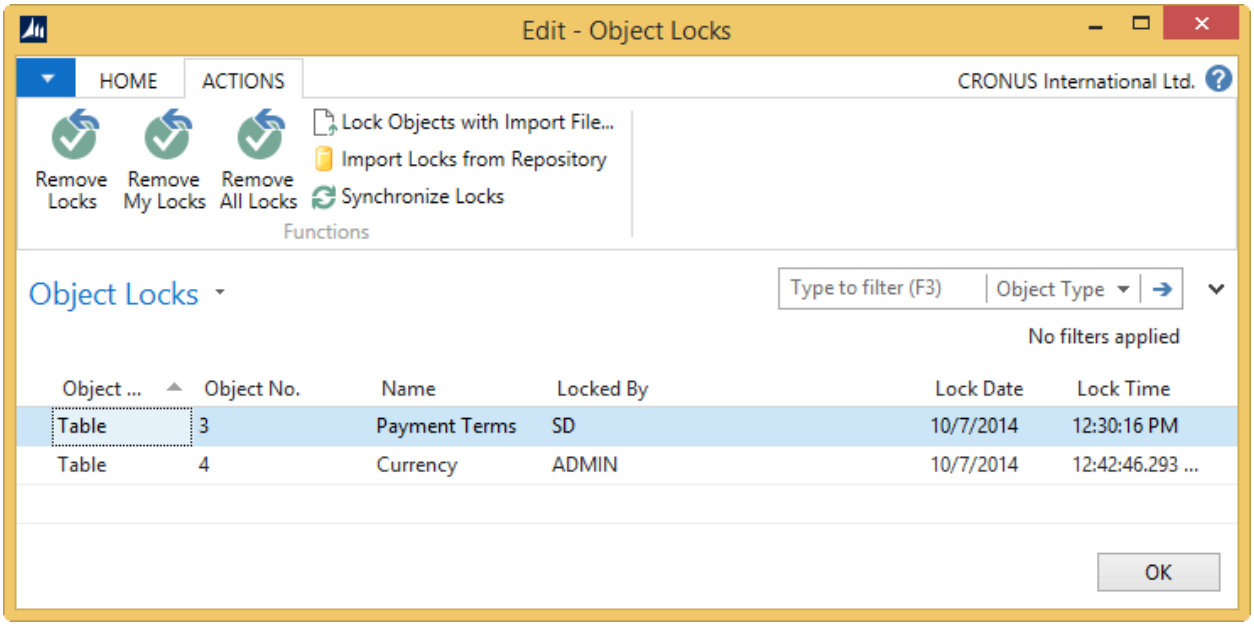
- Blue for objects locked by you
- Gray for objects locked by others
- 

Type	ID	Name	Modified	Version List	Locked By
Table	3	Payment Terms	☐	NAVW17.00	USER
Table	4	Currency	☐	NAVW17.00	SD
Table	5	Finance Charge Ter...	☐	NAVW17.00	

- There are several ways to remove object locks:
- Clear the **Locked** check box on the object line in the object explorer.
- Select the needed object(s) in the object explorer and then click **Locking > Remove Lock** on the **Actions** tab of the ribbon.
  All of the selected objects must be locked by you.
- Click **Locking > Remove My Locks** on the **Actions** tab of the ribbon in the object explorer to remove all locks that you have placed.
- Click **Locking > Remove All Locks** on the **Actions** tab of the ribbon in the object explorer to remove all locks in the database.

Note

YOU CAN ALSO GET AN OVERVIEW OF OBJECT LOCKS IN THE DATABASE, PLACE AND REMOVE OBJECT LOCKS, AS WELL AS LOCK OBJECTS WITH IMPORT FILE USING THE OBJECT LOCKS WINDOW. TO OPEN THE WINDOW, GO TO DEPARTMENTS > OBJECT MANAGER > OBJECTS AND THEN CLICK OBJECT LOCKS.



3.4 Object Modification

Along with basic development functions, the object explorer allows you to edit C/AL code of objects and change their properties; you can also use the object explorer to edit data in tables and view table properties.

3.4.1 Editing C/AL Code

From the object explorer, you can view and edit C/AL code of a selected object, both in Microsoft Dynamics NAV and in an external editor.

Note

TO BE ABLE TO EDIT CODE IN AN EXTERNAL EDITOR, ENSURE THAT YOU PROVIDE A PATH TO THE EXTERNAL EDITOR EXECUTABLE IN OBJECT MANAGER SETUP.

3.4.1.1 Editing Within Microsoft Dynamics NAV

To edit the code of an object using the built-in Microsoft Dynamics NAV editor, select the needed object and then click **Code** in the ribbon. Use the **C/AL History Lines** window that appears as follows:

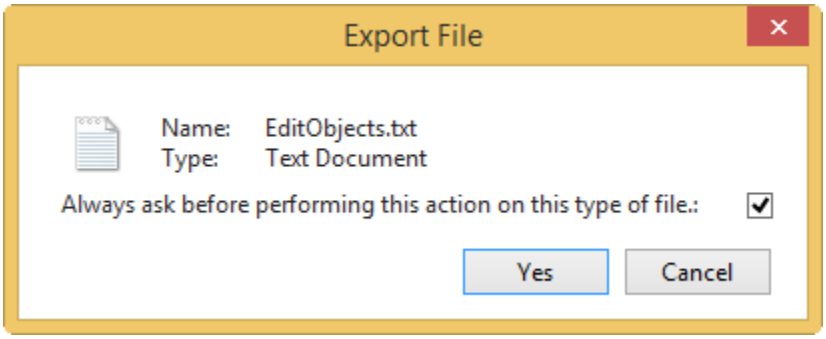
- Click the **Code** field on a line and modify code as needed.
- Click **New Line** on the **Actions** tab of the ribbon to add a line of code above the currently selected line.
- Place check marks in the **Breakpoint** field on one or more lines and then use the **Debugger** group in the ribbon to debug code.
- Use the **Edit** group in the ribbon to save, undo, or refresh modifications.

Additionally, you can switch to editing code in the external editor by clicking **Edit in External Editor** in the ribbon.

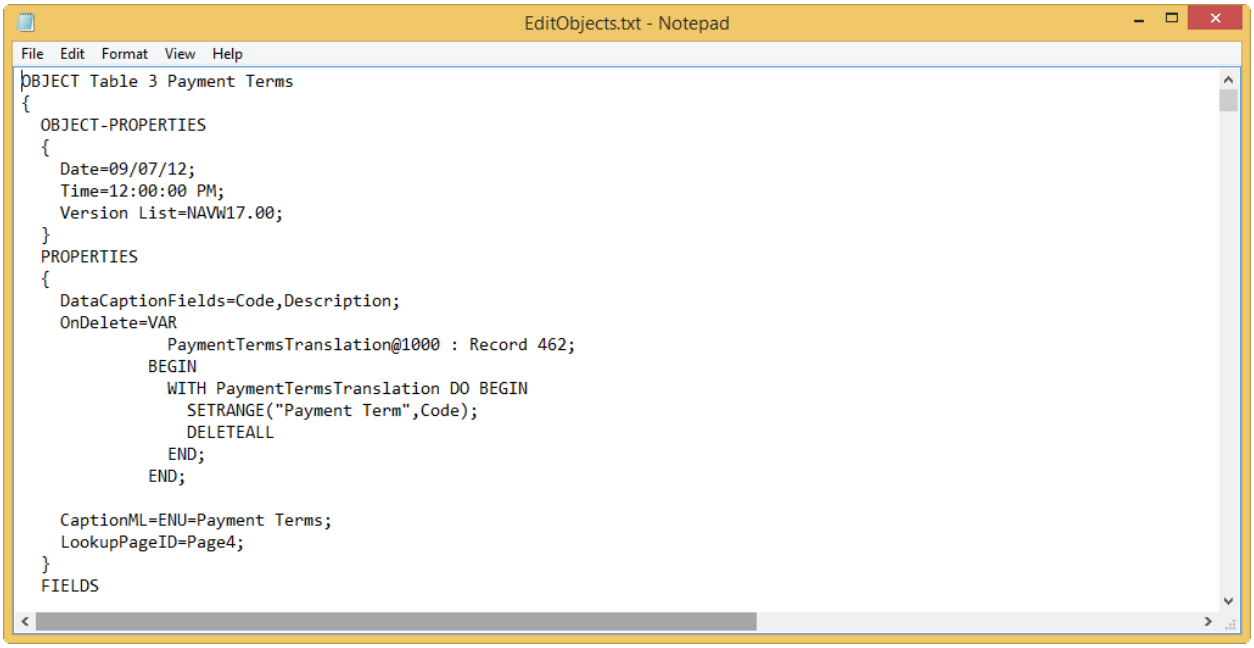
3.4.1.2 Editing in External Editor

To edit the code of one or more objects in an external editor, perform the following steps:

1. Select the object(s) in question and then click **Edit in External Editor** in the ribbon.
   The program will prompt you to confirm exporting the code of the selected object(s) to a text file.

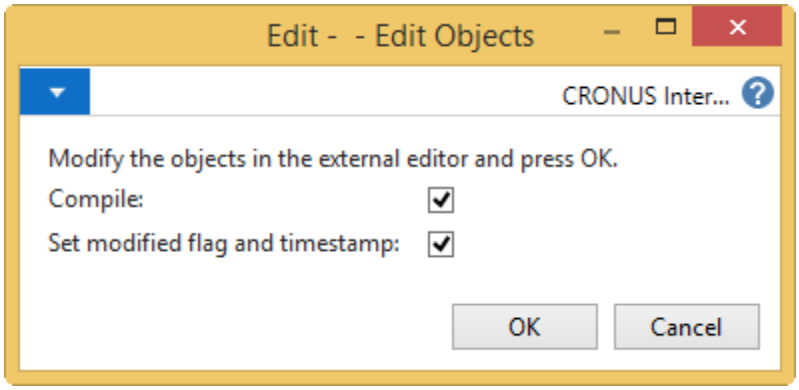


2. Click **Yes**.
   The program will create the text file in the working directory that you define in Object Manager Setup; the file will then be opened in the external editor.



NOTE
To disable the prompt for the next time you use the external editor, clear the check box in the dialog window.

3. When done editing, close the external editor and return to Microsoft Dynamics NAV. Use the **Edit Objects** dialog window to define whether you want to compile the object and/or update its **Modified** flag and date/time.



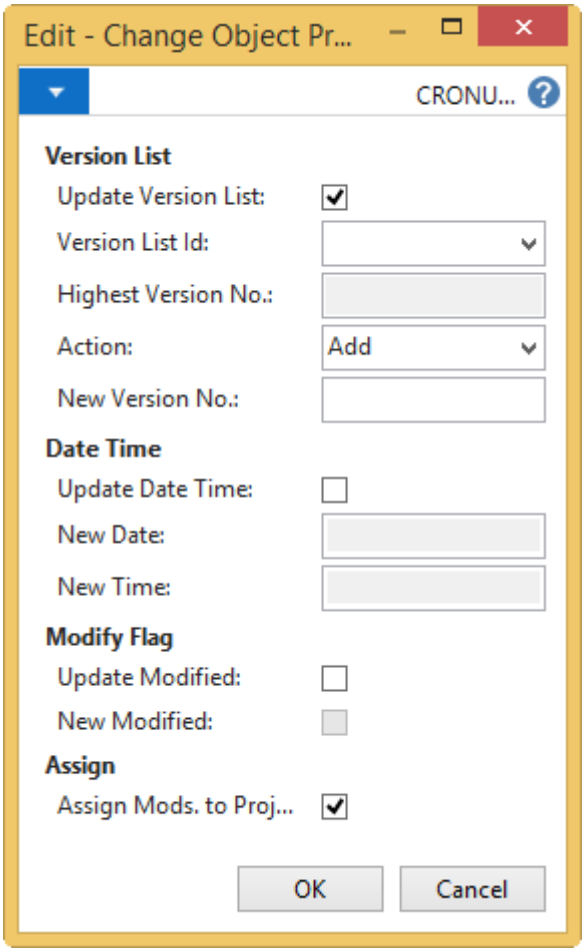
4. Click **OK** to save the modified object and return to the object designer.

3.4.2 Changing Object Properties

Using the object explorer, you can change the version list, modify the date/time, and update the Modified flag of one or more objects.

To change object properties, select the needed object(s) and then click **Change Object Properties**. In the window that appears, fill in the fields as follows:

Field	Description
Version List Group	
Update Version List	Select this check box to update the version list of the object in question.
Version List ID	Select the version list ID that you want to use for the update.
Highest Version No.	This field displays the highest version number in the selected version list.
Action	Choose whether to add or remove the selected version list and number to/from the object in question.
New Version No.	If you choose to add the version list and number to the object in question, provide the new version number. If you choose to remove the version list and number, leave the field blank.
Date/Time Group	
Update Date/Time	Select this check box to update the date/time of the object in question.
New Date **New Time**	Enter the new date and time for the object, respectively.
Modified Flag Group	
Update Modified	Select this check box to update the Modified flag of the object in question.
New Modified	Use this field to define the state of the Modified flag of the object in question after the update.
Assign Group	
Assign Mods. to Project	Select this check box to have the program record the current changes as modifications and prompt you to assign them to a project.



Click **OK** to update object properties.

3.4.3 Working with Tables

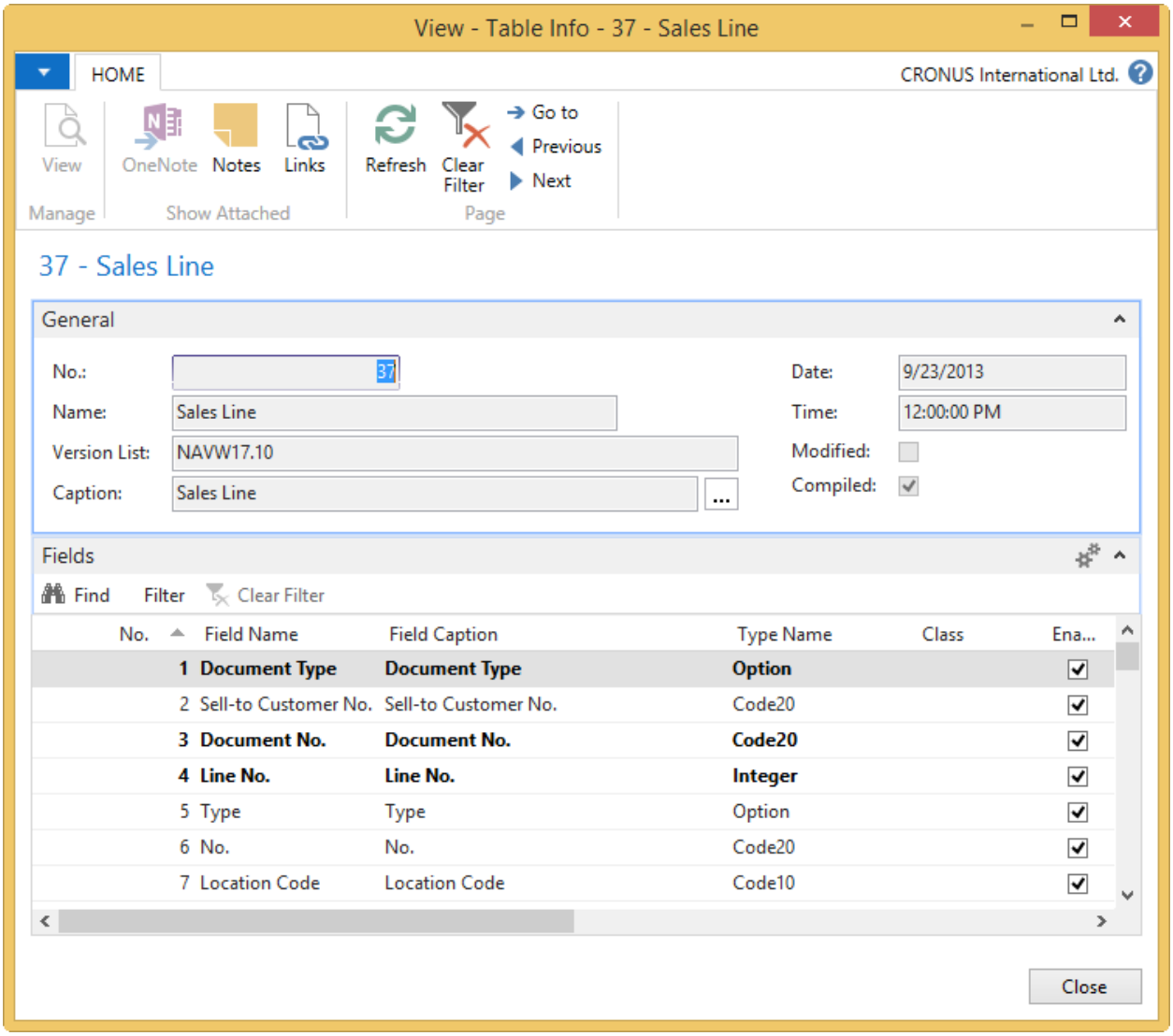
When working on table objects in the object explorer, you can view the summary of the table properties and properties of its fields, and you can modify data in the table, with or without validation.

View Table Information

To view information about the fields of a table (primary keys, field captions and classes, field options, and relations to fields in other tables), select the needed table in the object explorer and then click **Show Table Info** on the **Actions** tab of the ribbon. The window that appears shows the following information:

- General table information (same as on the object line in the object explorer)
  - Version list
  - Date/time
  - Modified and Compiled flags
- List of all table fields and their characteristics
  - Primary key fields in bold
  - Field number, name, and caption
  - Field type and class
  - Enabled flag
  - Option values, if applicable

- Relation to a field in another table, if any



The information presented in the window is for reference purpose only. You cannot modify any values.

3.4.3.1 Edit Table Data

To view or modify data in a table, select the table in the object explorer and then click **Edit Table Data** on the **Actions** tab of the ribbon. In the window that appears, you can browse records stored in the current table and edit field values similarly to running a table object in the object designer. Additionally, the window offers the following possibilities:

- Changing column layout.
  Use the **Process** group in the ribbon to change how data is displayed:
  - Click **Hide All Columns** to only display the values in the currently selected field.
  - Click **Show All Columns** to view all fields.
  - Click **Select Columns** to choose which fields you want displayed.
- Exporting table data.
  Click **Export** on the on the **Actions** tab of the ribbon and then click the

appropriate item to export data stored in the current table to either of the following file formats:
  - .csv
  - Tab delimited .txt
  - .fab
  - .xml

You will be prompted to browse to a location and provide a name for the resulting file.

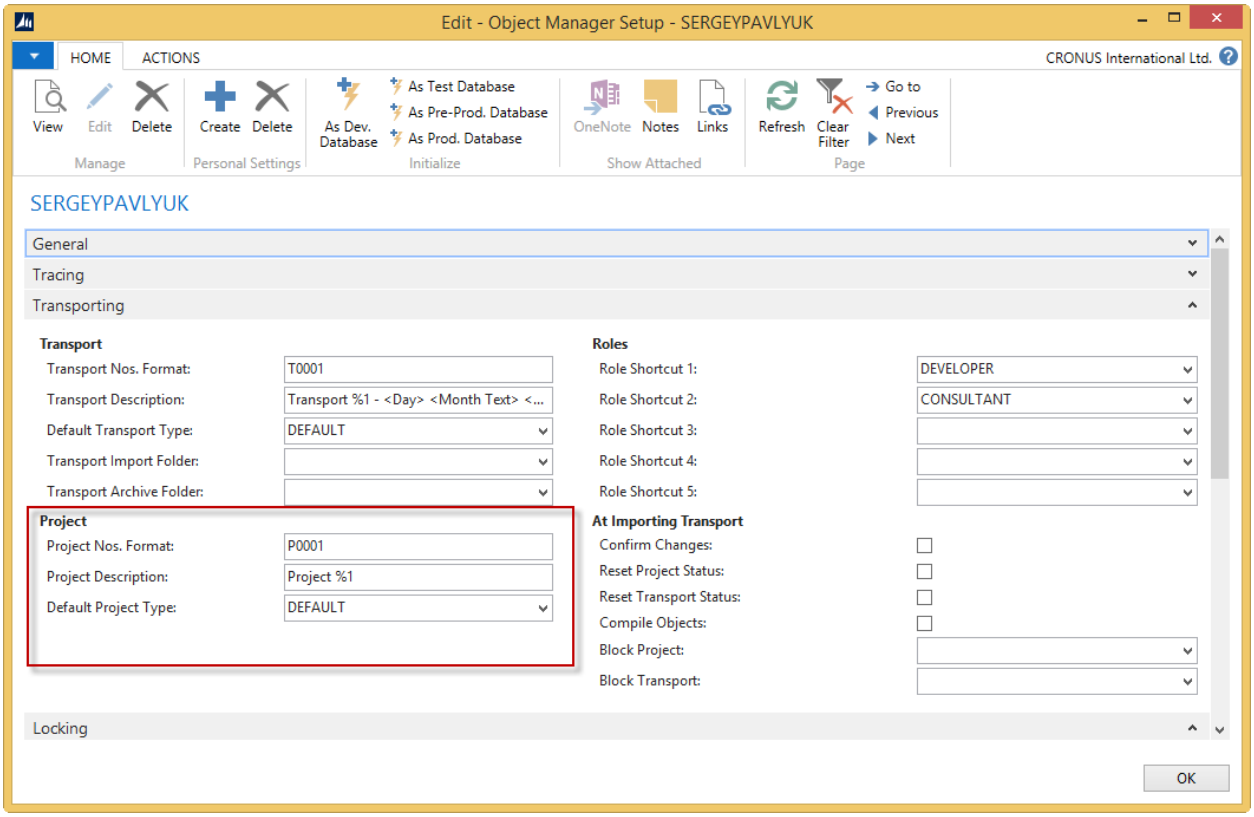
- Viewing data by company.
  Use the **Company Name** field above the lines to view data that is stored in the current table in a different company in your database.
- Editing data with or without validation.
  Use the **Validate** check box above the lines to enable or disable data validation for the currently selected table record.

4 PROJECT

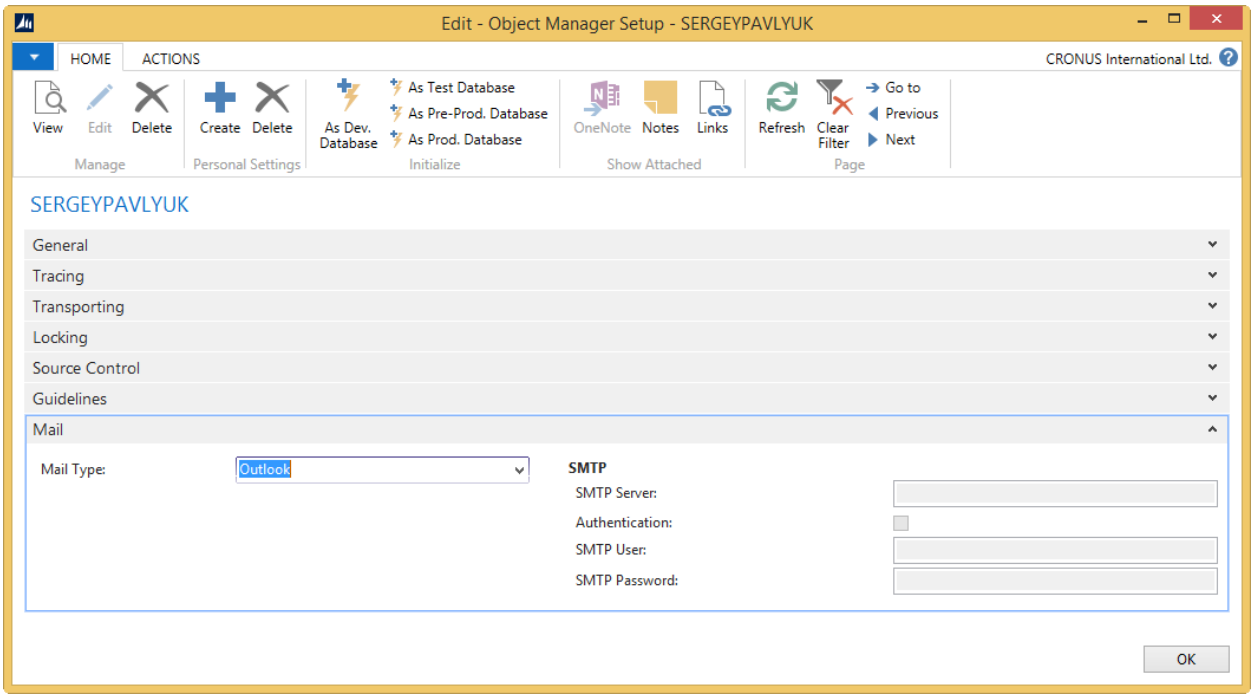
When developing the solutions you can assign a number of objects to a project. You can attach files and other information which is necessary for the consultant or developer involved in the project. You can also monitor the duration between modifications and status changes. Next to that you can assign actions to a project like removing data from tables.

4.1 SETUP

Go to **Departments > Object Manager > Setup > Object Manager Setup**



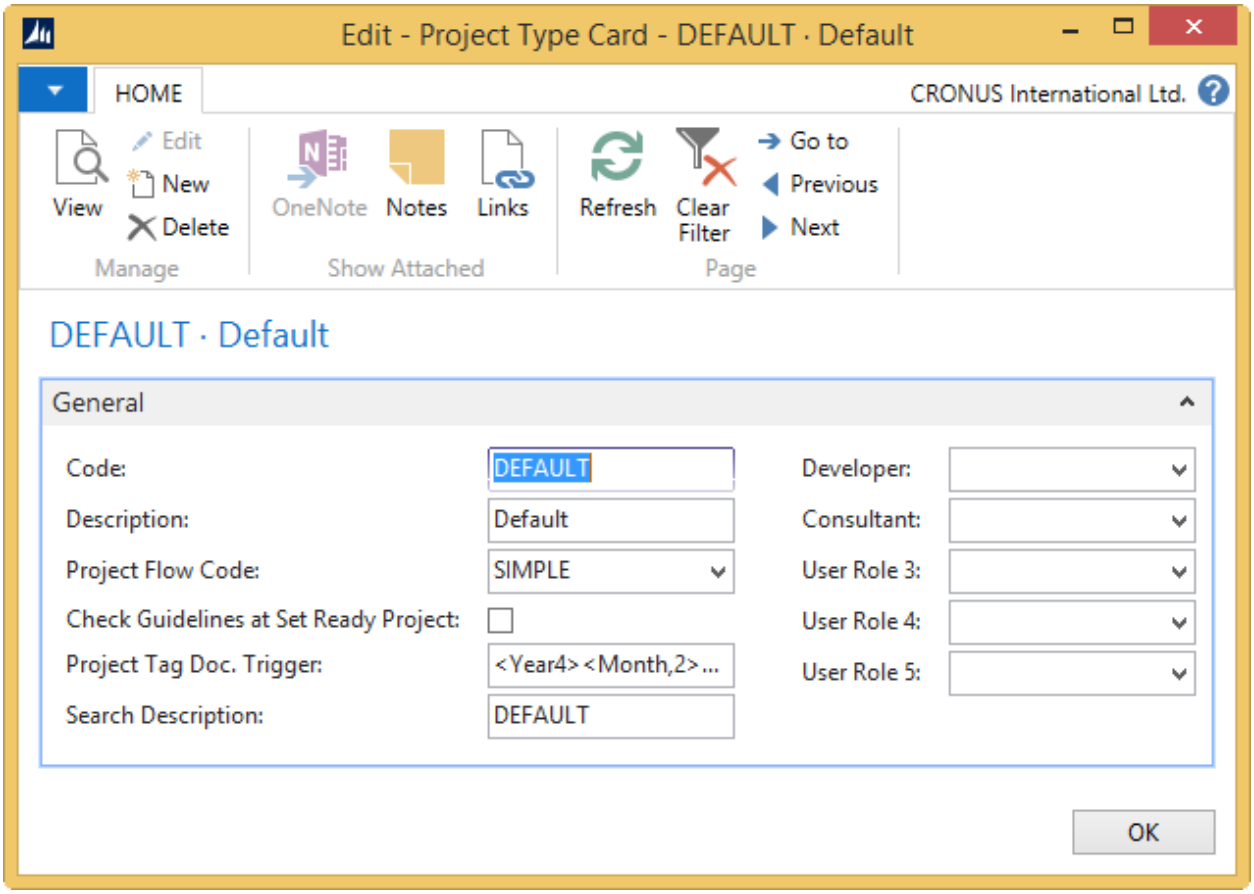
- **Project Nos. Format**
- **Project Description**
  %1 will be replaced by the "Project No.".
  You can use date expressions like: <Day> <Month Text> <Year4>.
- **Default Project Type**
  For more information about Project Types see chapter *4.2 - Project Type*.
- **Role Shortcuts**
  Here you select the roles that are visible in your project card.



- The emails that are sent in the project and transport module can be send by outlook or with a SMTP server.

4.2 PROJECT TYPE

Open Setup > Project Types



Here you can define different types of projects to differentiate for example between hotfixes and longterm projects.

- **Project Flow Code**
  For more information about project flows see section Project Flow.

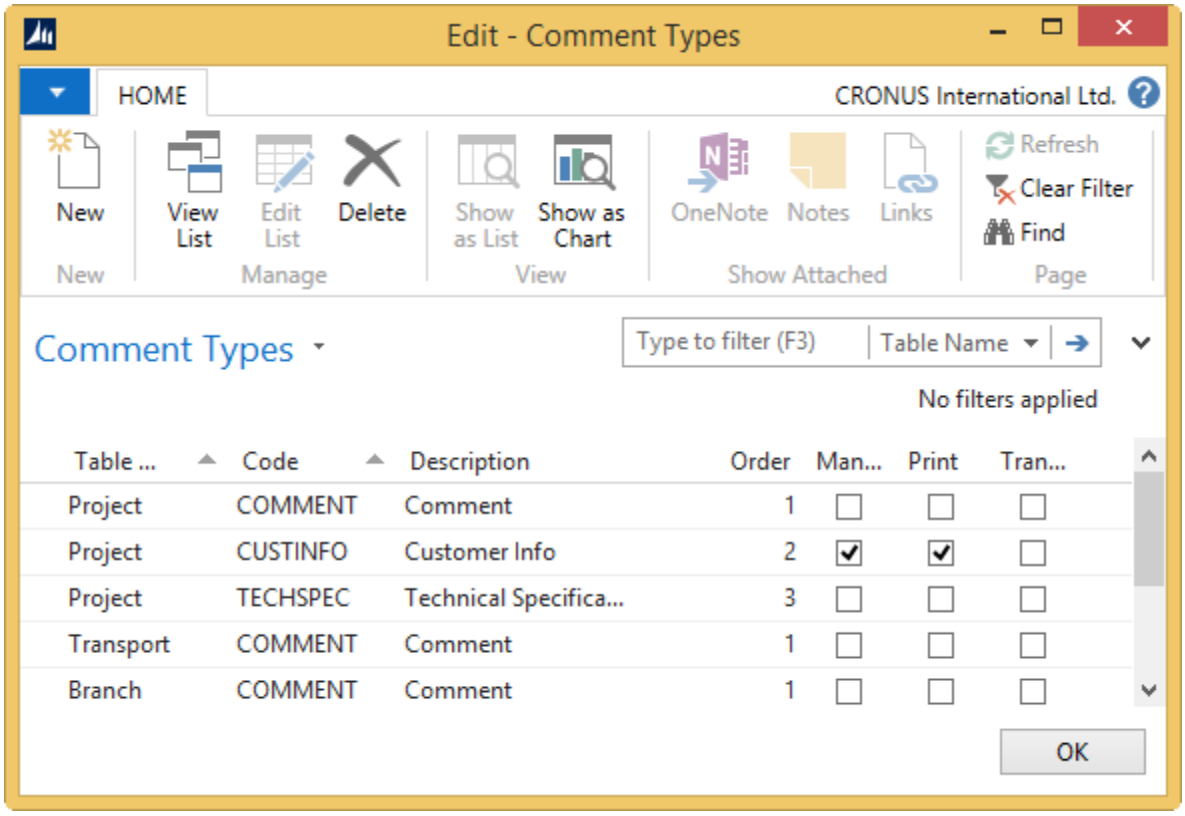
- **Check Guidelines at Set Ready Project**
  This setting will run a guidelines check on the objects in the project when the status of the project is set to ready.
  See more information see chapter Check Guidelines.

- **Project Tag Doc. Trigger**
  Here you can define the text for the automatic documentation trigger insert.
  For more information see section Add Project Tag to Documentation Trigger.

- **Default User Roles**
  Here you can define the default users if this project type.

4.3 COMMENT TYPES

Open Setup > Comment Types.



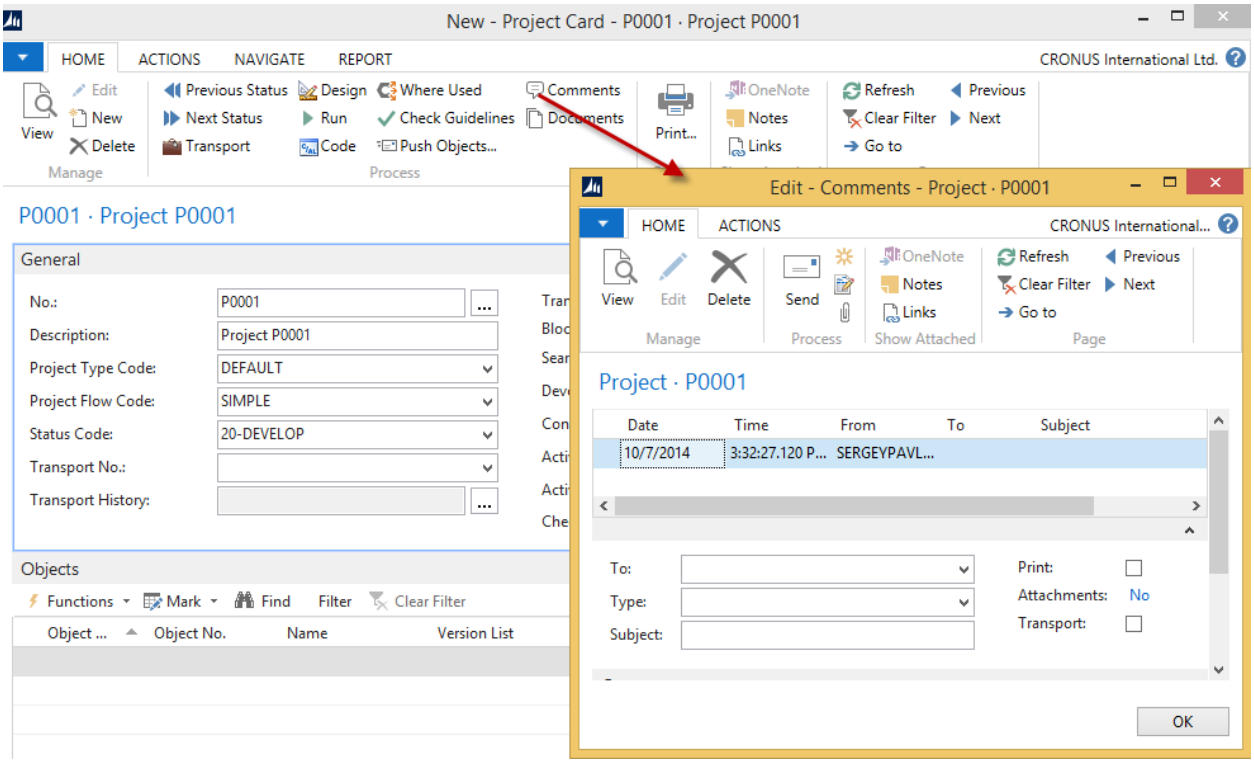
Here you can define Comment Groups for the comments on a project.

- **Order**
  This is the order in which the comments will be printed on the project and transport reports.

- **Mandatory**
  Transport will be only possible if a comment is present.
- **Print**
  Comment(s) will be printed on the project and transport report.
- **Transport**
  Comment(s) will be transported to your customer database.

Use the "Comments" window to write down your comments using either the Text lines or an external editor by pressing"Ext. Editor".

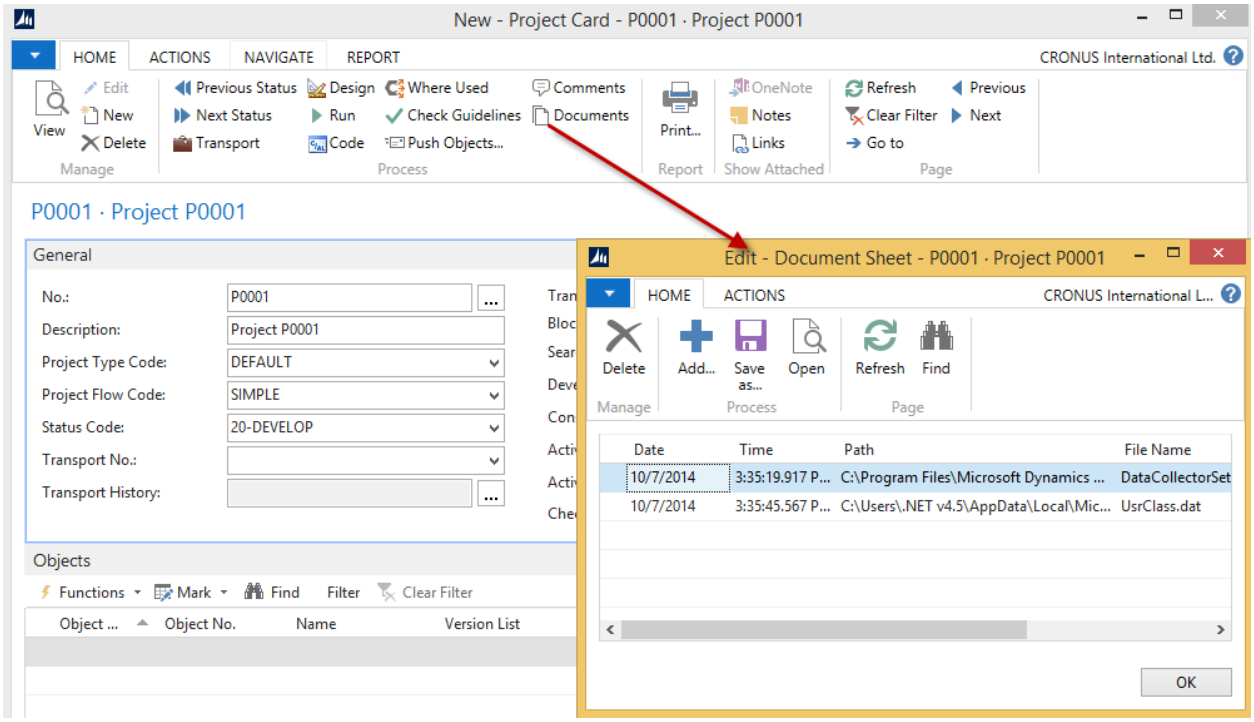
You can also send a comment pressing "Send" provided you entered a user with an associated e-mail address in the "To" field.



4.4 DOCUMENTS

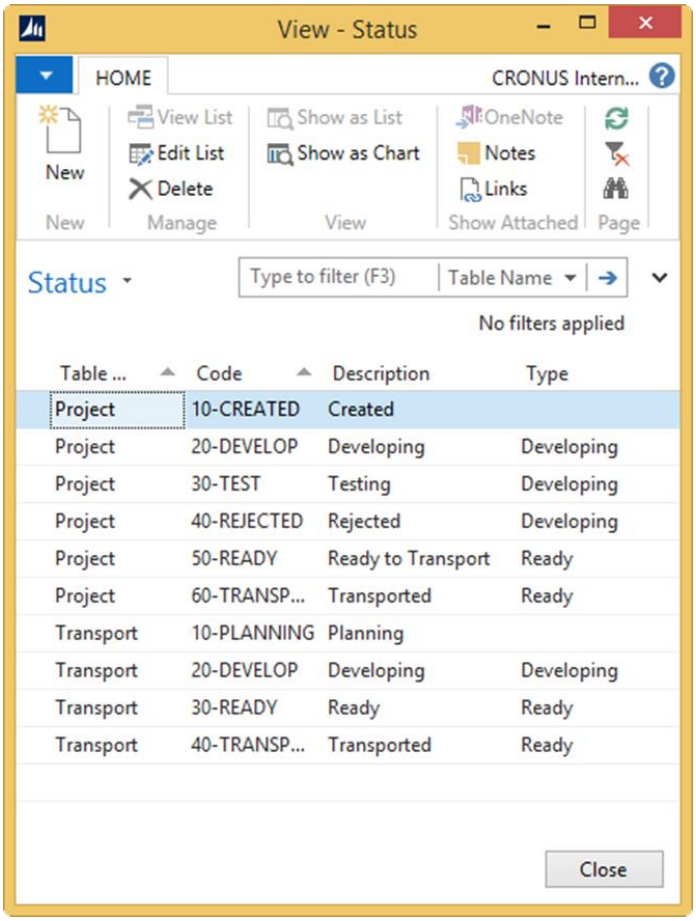
You can add files to a project. For example a customer license or an installation instruction. With the button Document you can add, overwrite and delete files which are of importance to the project.

- **Attach to Transport**
  - As File: File will be saved in the transport folder as a separate file.
  - In FIB: The file is merged into the FIB file.



4.5 PROJECT FLOW

A project flow is a collection of project statuses. You can add a project flow to a project so you can keep track of the progress and which user is responsible for a specific project status.

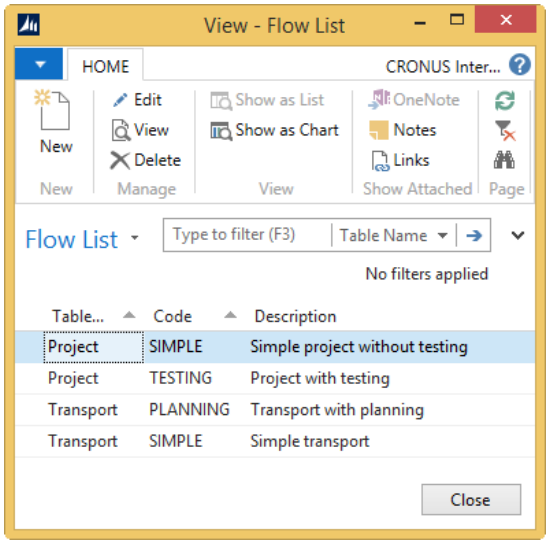


Default there are two flows in the Object Manager but you can add as many flows as you like. You can also create more statuses in the "Status Setup". Go to **Departments > Object Manager > Setup > Status Setup**

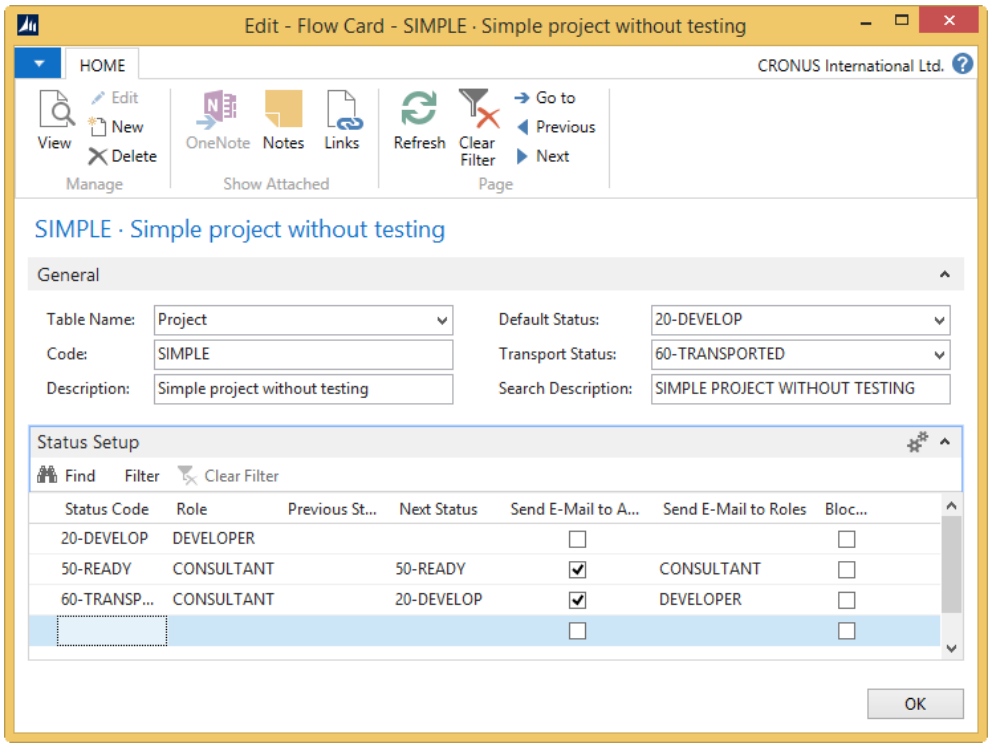
Here you can add an extra status with a brief description and type.

- **Type**
  This is the type of the status. It has the following options:
  - Developing: You can assign modifications to projects that have this status.
  - Ready: You can add projects that have this status to a transport.

Go to **Departments > Object Manager > Setup > Flows:**



Open the **Flow Card**:

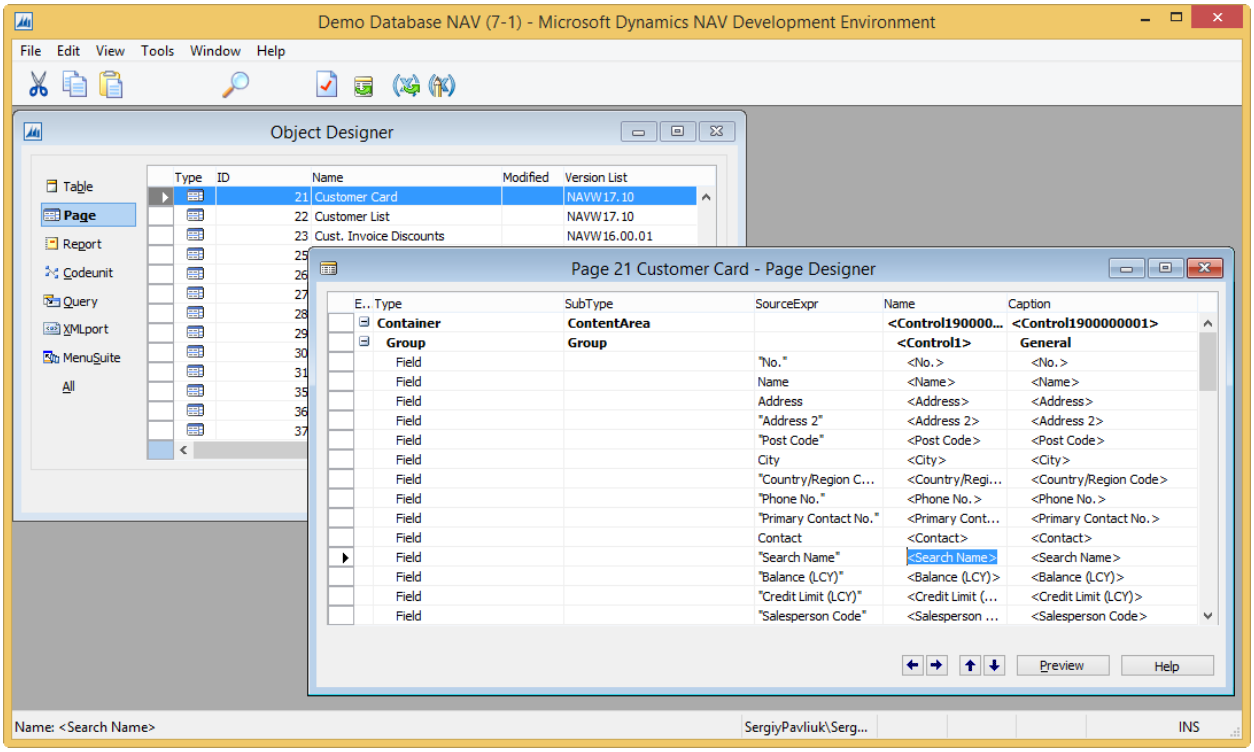


- **Default Status**
  Default status for each new project.

- **Transport Status**
  The project will get this status when it is transported to your customer database.
- **Previous Status**
  User can go back to this status.
- **Next Status**
  User can only choose the next statuses defined here.
- **Send E-Mail**
  When this status is reached an e-mail will be sent to the active user. You will be prompted if you want to send that e-mail.
- **Send E-Mail to Roles**
  When one or more roles are filled in here and this status is reached the user with this role is also sent an e-mail.
- **Block Project**
  When this status is reached, and "Block Project" is check marked, the project will be blocked.

4.6 Assign Modifications to a Project

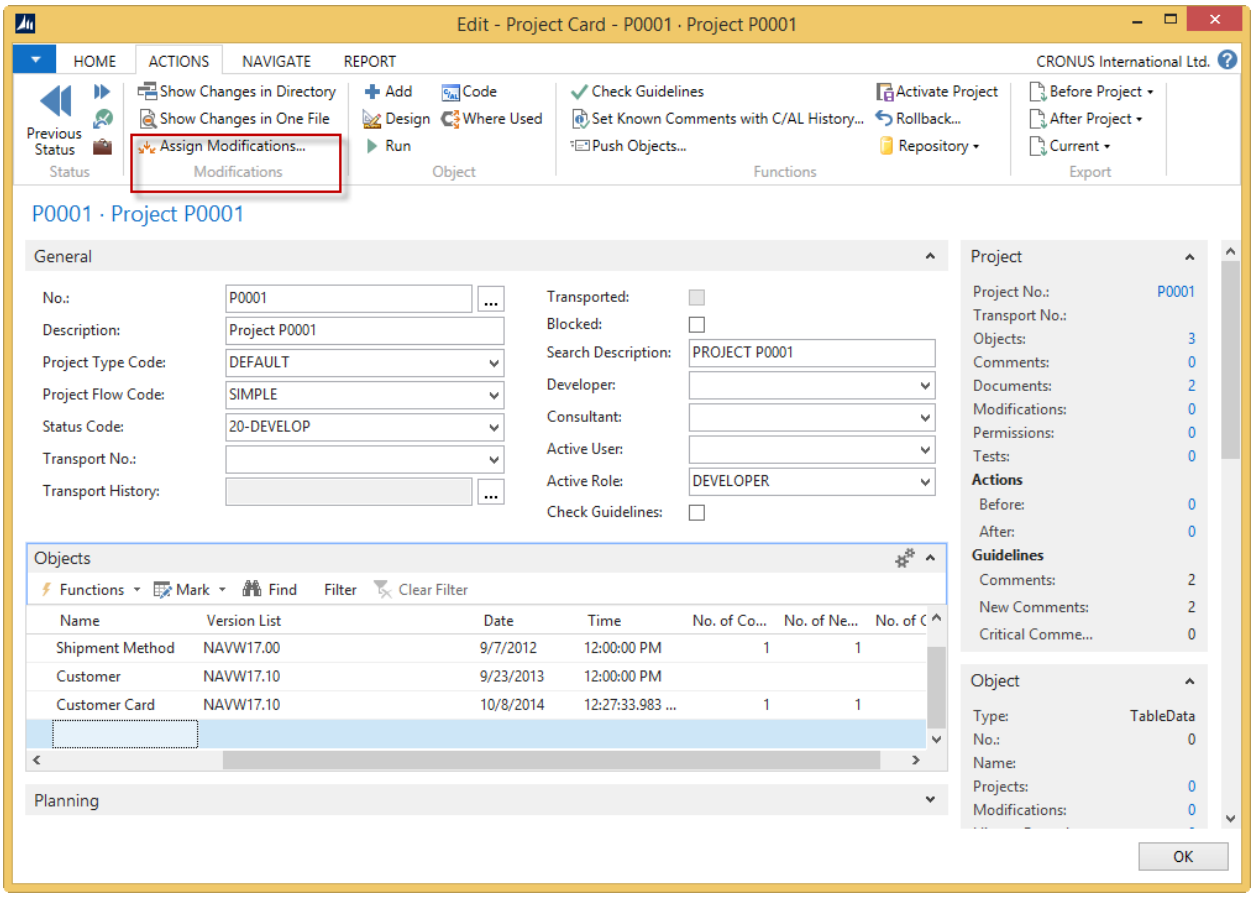
You can add multiple modifications to one project.



Add, for example a new field "Name 3" to the Customer table and add this field as a control to the "Customer Card". Save these objects.

Open **Departments > Object Manager > Objects > Modifications > Assign Modifications**.

Select the objects you want to assign. Press button **Assign**.



Now you can see that the objects are assigned to the project.

To see all modifications on a project, press Modification under button Object or Project in the "Project Card".

In the form Modifications you can set duration from a chosen point. You can use this when you want to know how long you have worked on a particular project.

4.7 Add Project Tag to Documentation Trigger

It is possible to add a predefined string to the documentation trigger of an object. This can be done when you assign an object to a project or with a chosen set of objects in the project card.

On the Project Type card you can setup the format of the string that has to be added to the objects. Default the Object Manager uses the following format:

```
<Year4><Month,2><Day,2> %1 %2: %3
```

You can see that it is possible to add date expressions and 3 aliases that will be replaced with the following.

%1: Project No.

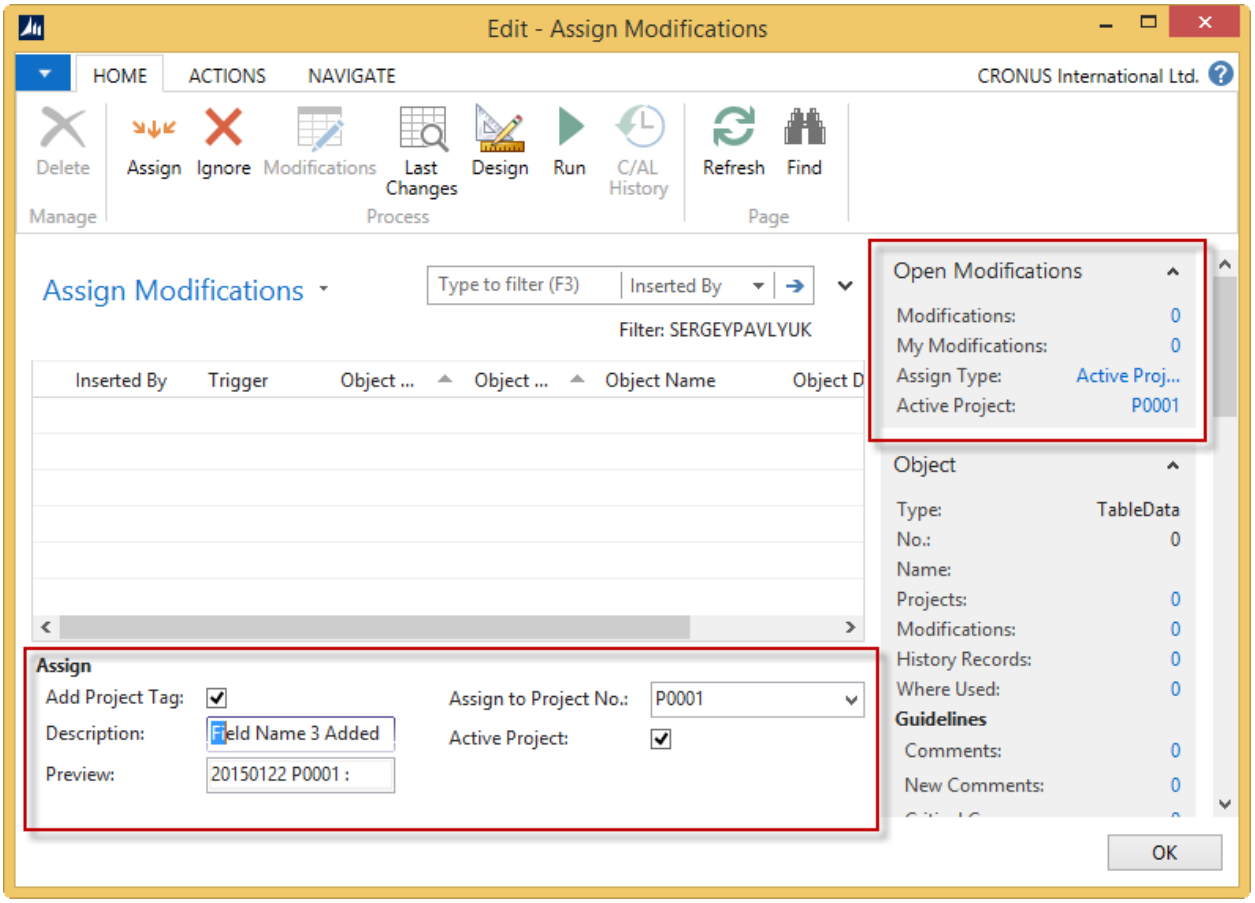
%2: Initials of the active user

%3: Description of the modification

Press Add Project Tag and type a description.

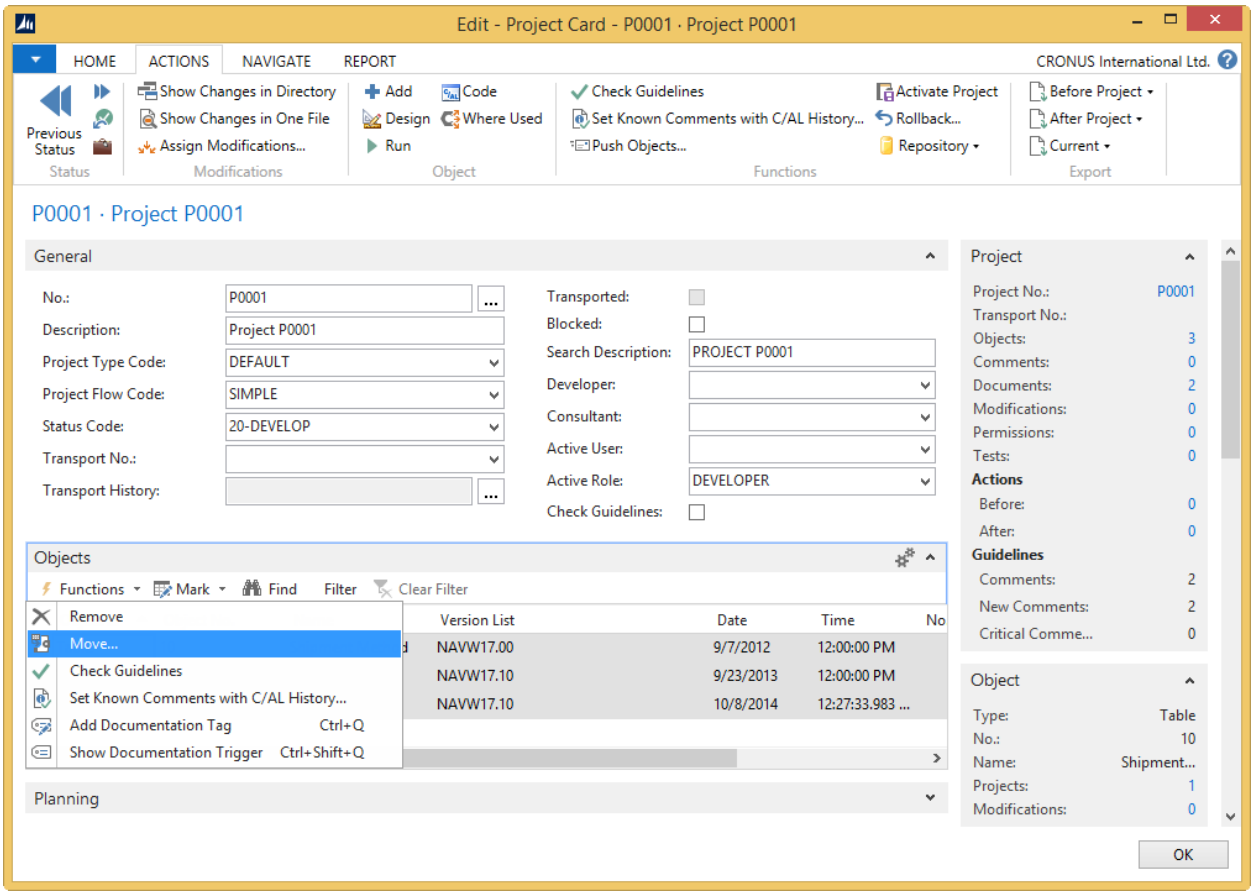
The tag will be added to the objects that are assigned.

It is also possible to add a tag to selected objects in the project card.



4.8 Move Objects and Modifications to another Project

With this function you move objects to another project. And if there are any modifications present the Object Manager will ask to move them too. This is used when you assigned modifications to the wrong project.



When selecting Yes you also move all modifications of the object.

You can also move a particular modification to another project. To do this select the move option in the Modifications form.

In the Modifications form select the modification you want to move to another project. If there are no more modifications left in the original project you will get the following message.

If the object is not yet assigned in the other project you will get the following message.

4.9 Add Permissions to a Project

When you have created some new objects it is required to alter the permissions in the customer database. This can be done by adding permissions to a project. These permissions will be written into the customer database when you transport the project.

The permissions can be recorded with a wizard. When importing the transport file in the customer database there will be an extra step for importing the permissions.

4.10 Check Guidelines

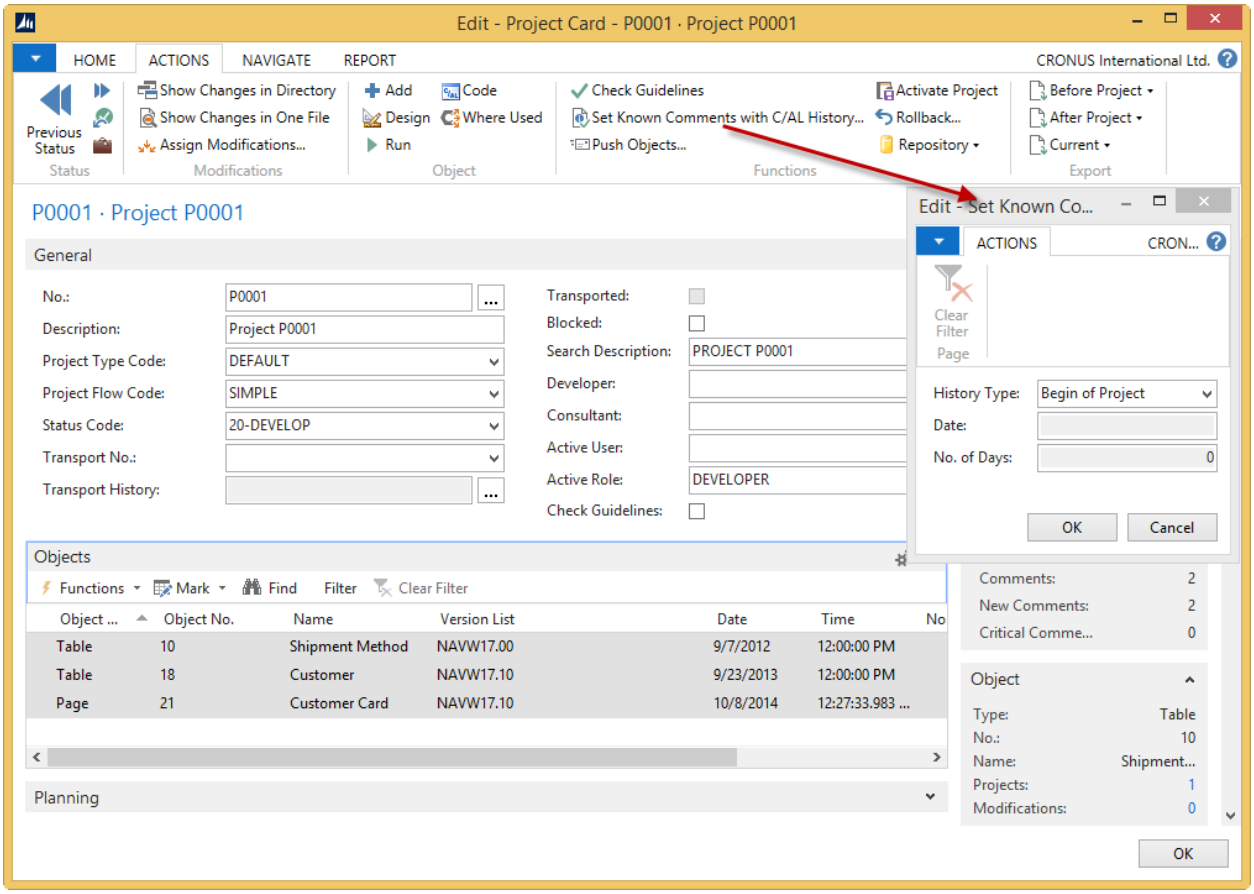
You can check if the Guidelines are met for all the objects in the project.

If in the project type the "Check Guidelines at Set Ready Project" is enabled this will be done automatically when the status of the project is set to ready.

If "Check Guidelines Comments" is set on the project card you will not be able to transport the objects until all code in the objects are according to the guidelines or set to known.

4.11 Set Known Comments with C/AL History

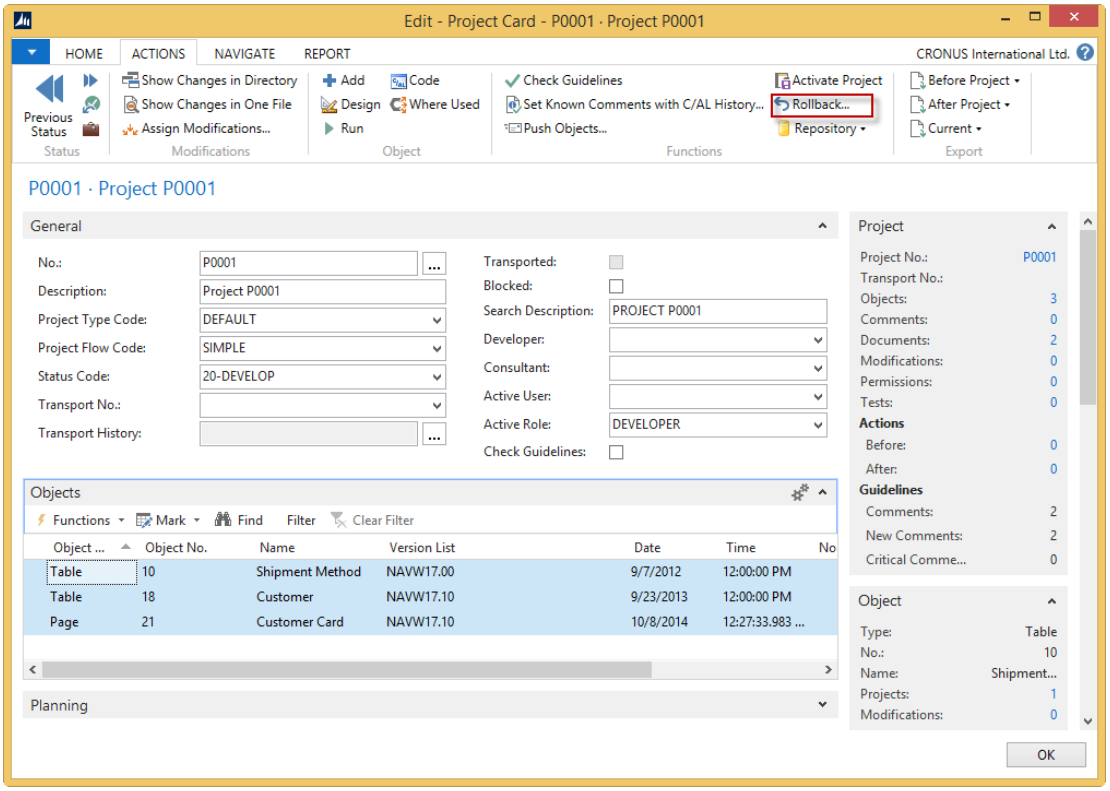
With this function you can set to known all comments that were already present in a point of time, for objects selected.



For more information about Know Comments see section Known Comment.

4.12 Rollback Objects

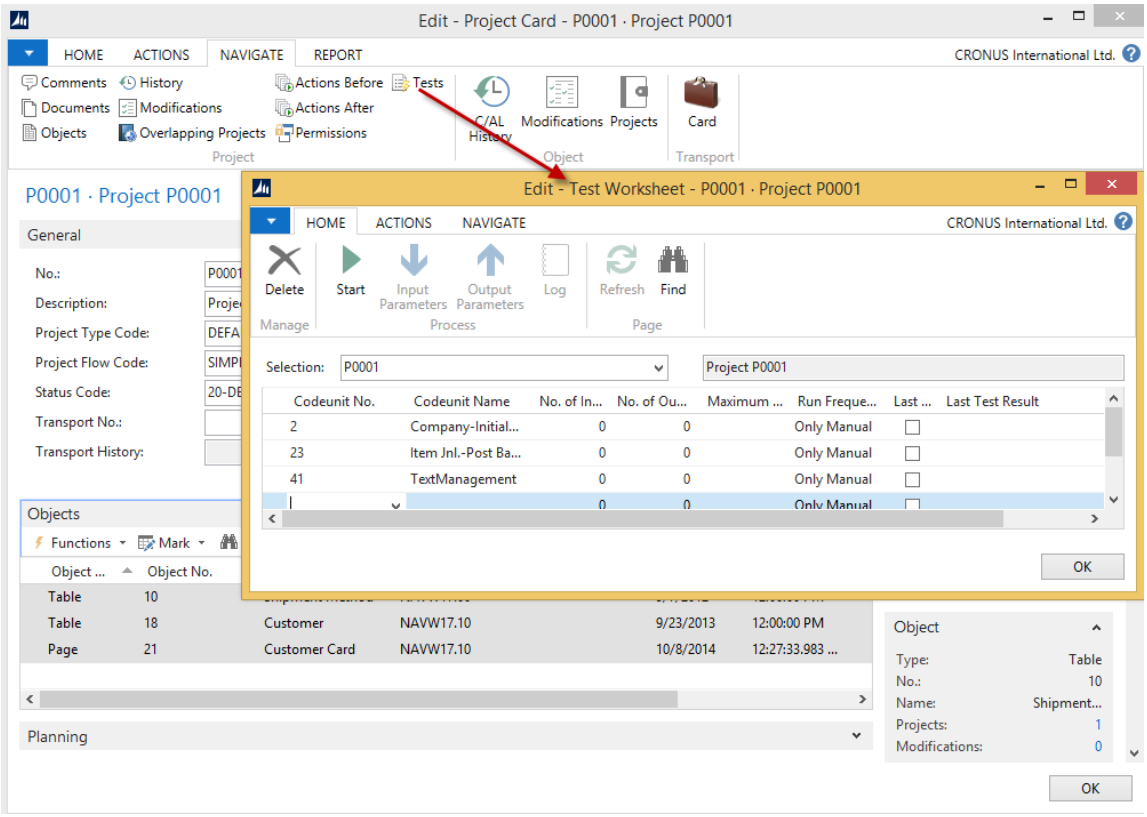
To Rollback object changes due to a project, press Functions – Rollback on the project card to open the "Rollback Objects" window.



For more information, see section Rollback Objects.

4.13 Test Worksheet

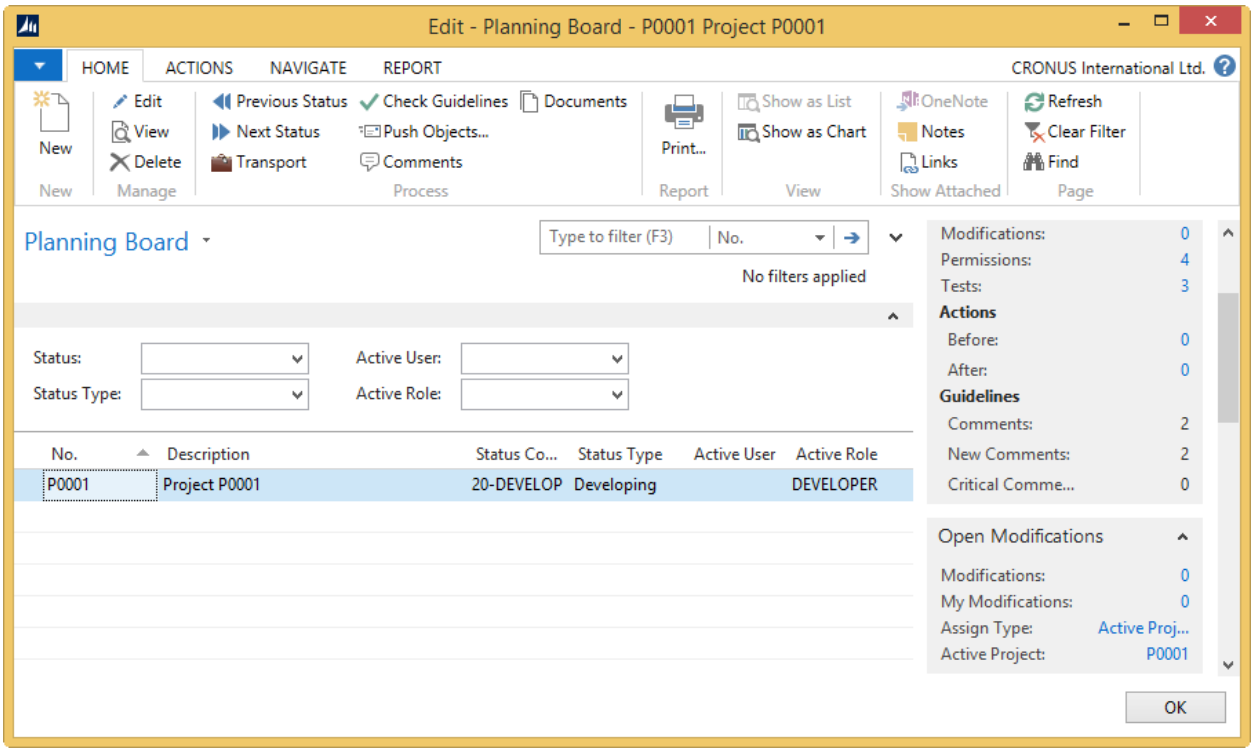
You can add tests to a project for example to check if old functionality still works in the database after transport.



For more information see chapter Test Framework.

4.14 PLANNING BOARD

With the "Planning Board" you have an overview of the projects and their status.

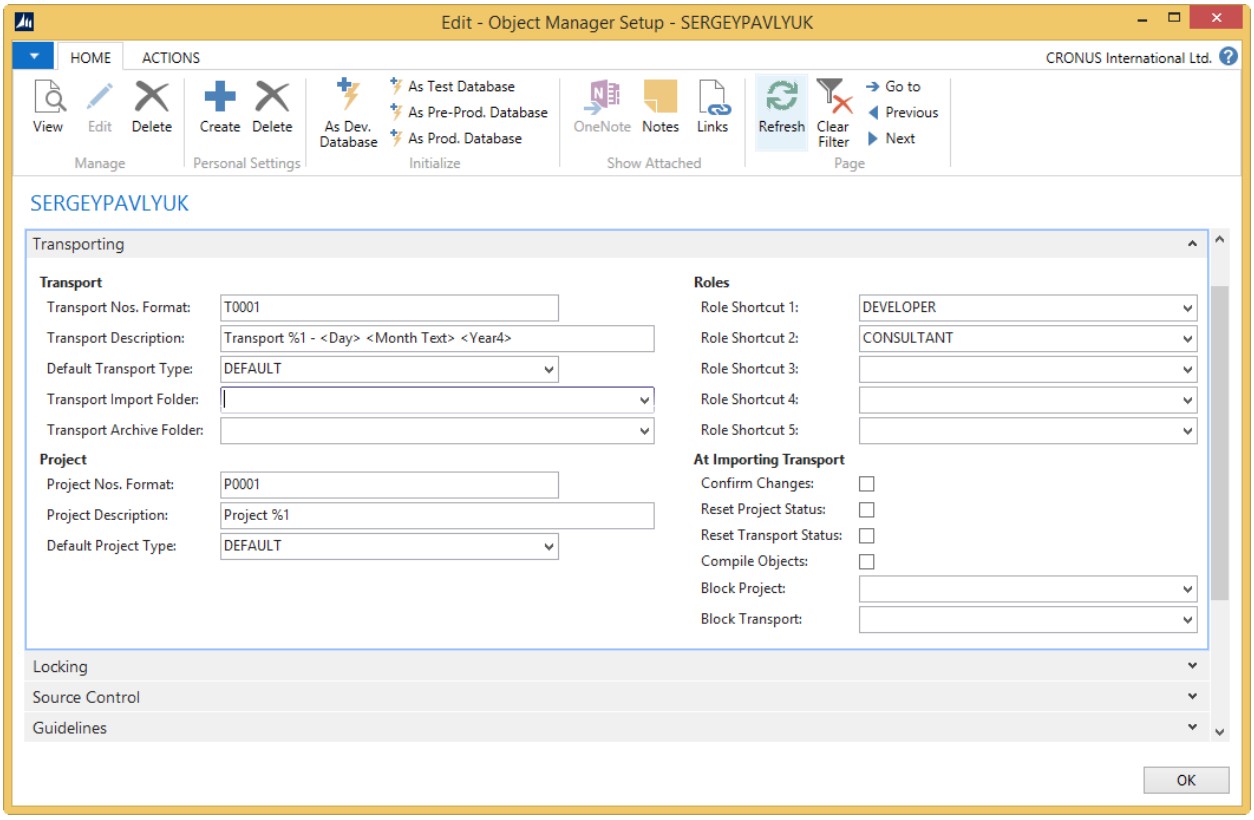


Clicking the checkbox next to the textbox "Active User" in the above right corner shows all the projects where you are the active person.

5 TRANSPORT

With a transport you can transfer objects from your development database to the customer database. A transport can also include master data, actions to run such as reports, permissions and documents.

5.1 SETUP



- **Transport Nos. Format**

- **Transport Description**
  %1 will be replaced by the "Transport No.".
  You can use date expressions like: <Day> <Month Text> <Year4>.

- **Import Transport with SQL trigger**
  This setting makes it possible to import fields and objects outside the active license.

- **Confirm Changes at Import Transport**
  If your customer has changed any objects since the last transport the object compare sheet will open when you import a transport. For more information see section - Confirm Changes at Importing Transport.

- **Reset Project Status at Import Transport**
  All projects will have the default status after you have imported a transport in your test database. For more information see section - Reset Project Status at Importing Transport.

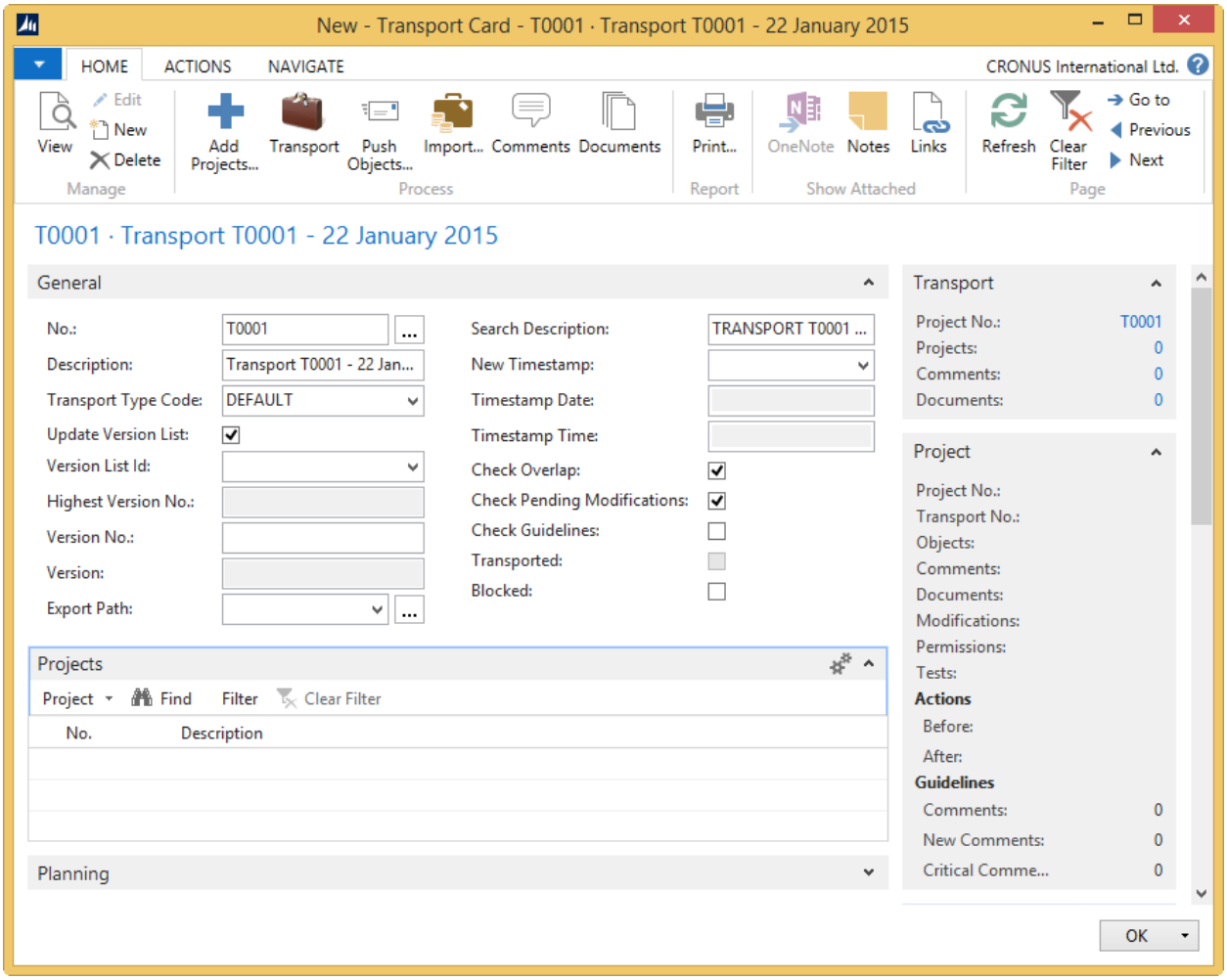
- **Compile Objects after Import Transport**
  After you import a transport all objects will be automatically compiled.

- **Block Project at Import Transport**
  With this setting you can change the blocked status of projects when they are imported with a transport.
  - <Empty>: The status of the project will remain the same as it was when it was transported.
  - No: Projects will be de-blocked when they are imported.
  - Yes: Projects will be blocked when they are imported.

- **Block Transport at Import Transport**
  With this setting you can change the blocked status of a transport when they are imported.
  - <Empty>: The status of the transport will remain the same as it was when it was exported.
  - No: Transport will be de-blocked when it is imported.
  - Yes: Transport will be blocked when it is imported.

5.2 TRANSPORT TYPE

You can differentiate between transports by setting up different types. For example a support issue or hotfix or a release of a functionality.



- **Description**
  Description of the transport type.

- **Transport Flow Code**
  For more information see section - Transport Flows.

- **Update Version List**
  This is the default setting that is copied into new transports. When this setting is enabled the version list of the objects in the transport will be updated when the transport will be transported.

- **Version List Id**
  Select the default "Version List Id". This "Version List Id" will be copied into every new transport. If you use the lookup you will see all used Version List Id's in the current database. It is also possible to fill in another id.
- **Check Guidelines before Transport**
  When this setting is enabled the transport can only be done if there are no unknown comments. For more information see chapter - Check Guidelines.

- **Compile Objects before Transport**
  Objects are compiled when you are transporting. You will get an error if there is an error in an object.

- **Block Project at Transport**
  All projects will be blocked if you transport. Transported projects cannot be de-blocked if this option is enabled.
- **Block Transport at Transport**
  The transport will be blocked at transport. A transported transport cannot be de-blocked if this option is enabled.
- **User Roles**
  You can define the default users here.

- **Export Path**
  This folder will be used to save the transport files.

- **Subfolder for Each Transport**
  The Object Manager creates a new folder for each transport.

  %1 will be replaced by the "Transport No.".
  %2 will be replaced by the "Version List Id" of the transport.
  %3 will be replaced by the "Version List No." of the transport.

  It is also possible to use expressions like: <Day> <Month Text> <Year4>.

- **Transport Files**
  You can choose which files are saved to your disk when you transport a transport.
  - HTML: a report with the transport information.
  - FIB: contains all transport and project data, actions, document and the objects of the transport.
  - OBJ: contains the objects. (See section - Import and Export Files).
  - TXT: contains the objects in text format.
  - FAB: contains all the data of the projects and transport.
  - FAB Before: contains the actions that has to be executed before the objects are imported.
  - FAB After: contains the actions that has to be executed after the objects are imported.

NOTE: You only need the FIB file when importing a transport in your customer database.

5.3 COMMENT GROUPS

Comment Groups are similar to Comment Types in projects. For more information see section - Comment Types.

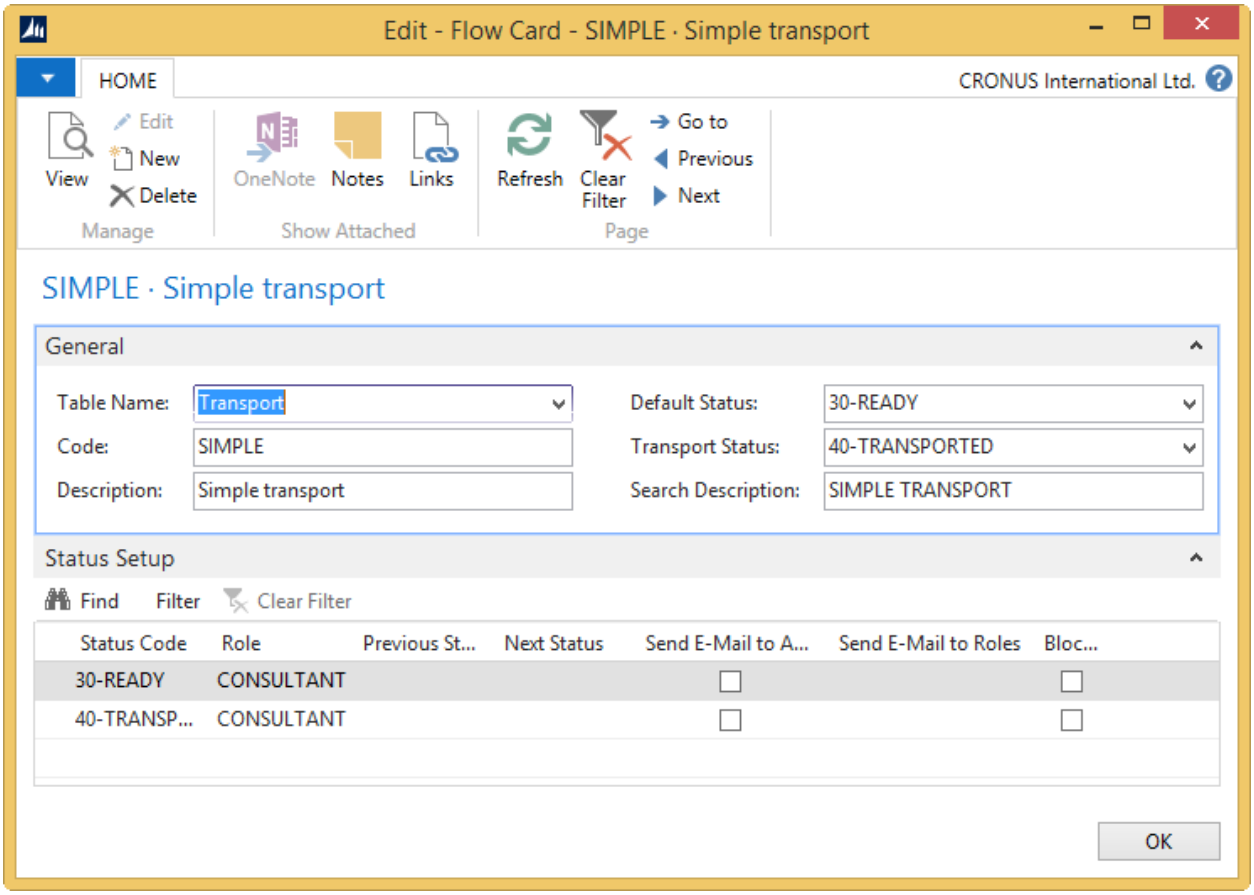
5.4 DOCUMENTS

Documents in transports are similar to Documents in projects. For more information see section - Documents.

5.5 TRANSPORT FLOW

Transport Flows are similar to Flows in projects. For more information see section - Project Flow.

By default a transport gets status type Ready. With this status type you can only add projects that are ready.



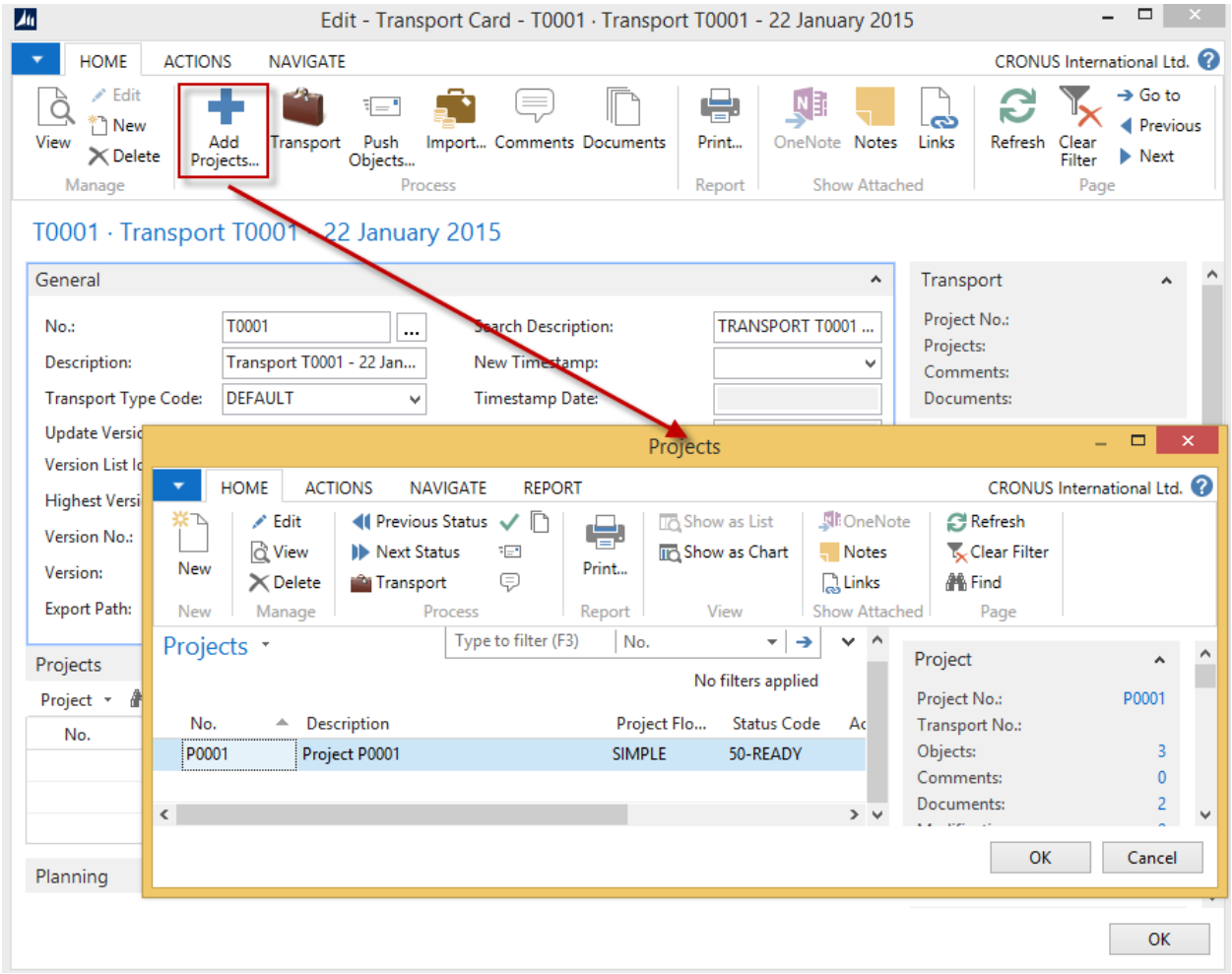
If a transport has status type <empty> or <Developing> you can add all projects. This can be used if you want to plan a future transport and add projects to it that are not yet ready.

5.6 EXPORT TRANSPORT

Make a new Transport Card. Add projects to the transport. By default only projects with status ready are visible in this overview. This prevents you from transporting projects that are not completely finished yet.

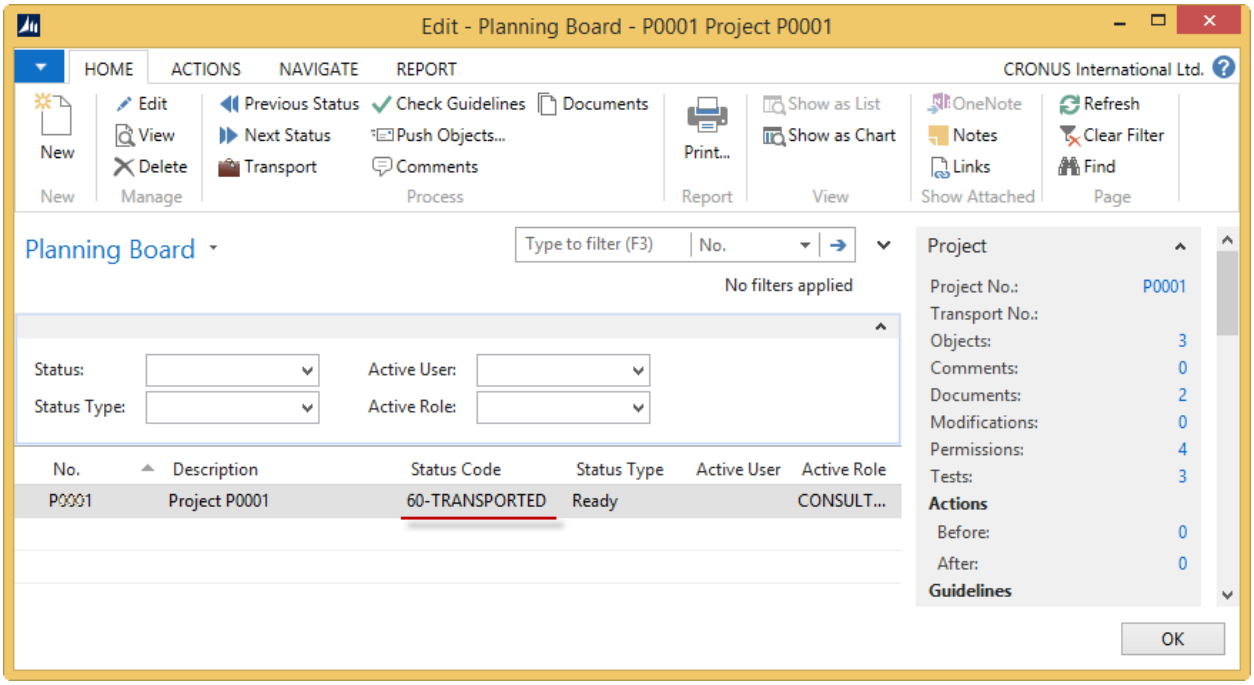
If you work with transport flows it can be possible that your transport has status type <empty> or developing. With this two status types it is possible to add all projects.

By clicking the assist edit button of textbox "Export Path" you can surf to the folder location.



A FIB file is created in the folder as well as the transport report in HTML format and the attached documents of the projects.

In the "Planning Board" you can see that the status of the projects are now set to Transported.



5.7 TIMESTAMP

You can also modify the timestamp to the objects when you transport.

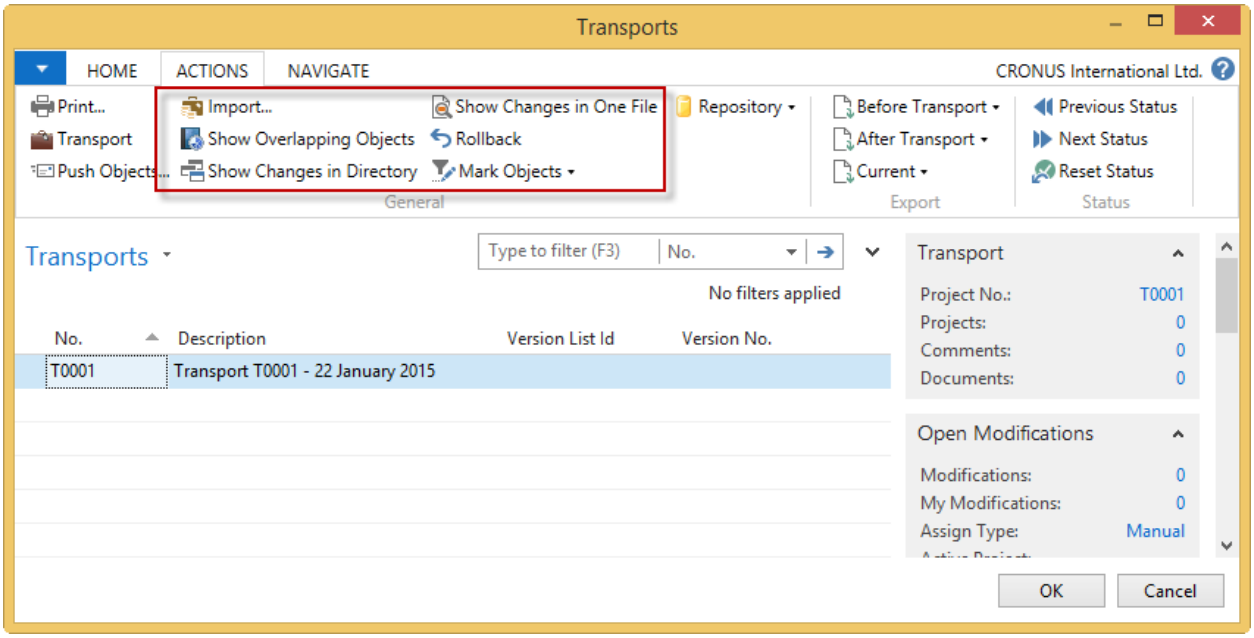
- **Timestamp Options**
  - <Empty>: The timestamp of the objects will be the date and time of the last modification.
  - Moment of Transport: Timestamp will set to the time of transport.
  - Define: You can define your own timestamp in fields Timestamp Date and Time.

NOTE: If you differ too much from the real transport date "Version Control" can act strangely. (See chapter - Version and Source Control)

5.8 OVERLAPPING OBJECTS

When transporting projects you sometimes have the problem of overlapping objects. For example: you used table Item as well in project 1 as in project 2 and you want to transport only project 1.

The Object Manager checks if the transport has overlapping objects.



You can check where these objects are used with the function "Show Overlapping Objects".

The bold lines in this form are in the transport.

Now you have three options:

1. Remove the modifications manually made in project 2 and delete the object from project 2
2. Finish and include project 2 to the transport
3. Disable "Check Overlap" on the "Transport Card" if you know for sure that it would not give any complications in the customer database

5.9 Pending Modifications

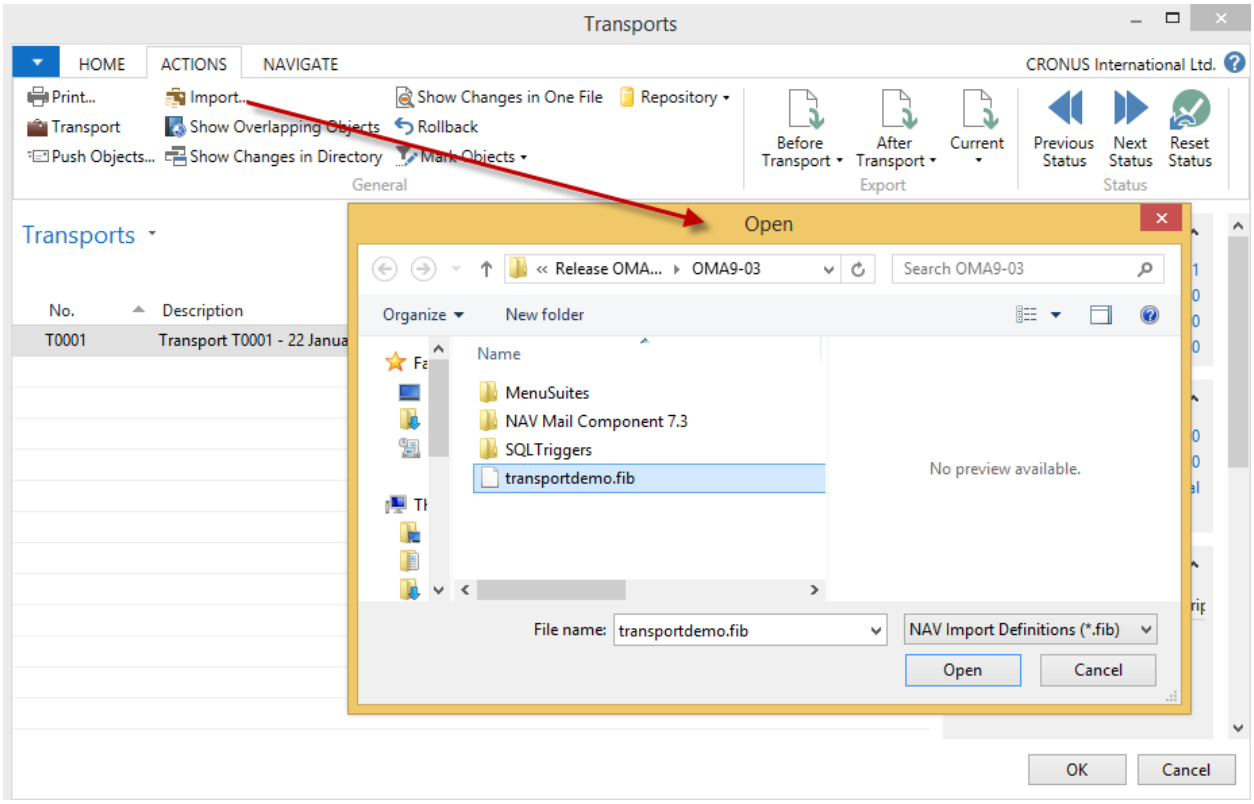
When you transport the objects, the Object Manager checks if there are modifications present that are not yet assigned to a project. This is to be sure that you did not forget any modifications. If you are sure that the pending modifications do not give any complications in your customer database you can disable the "Check Pending Modifications" option on the Transport card.

5.10 Import Transport

The transport (FIB file) can be imported in the customer database.

Select Transports > Import Transport and select the FIB file you want to import.

The "Transport Import Wizard" starts with all the steps that need to be executed.



NOTE: When importing a FIB file into a database, that does not yet contain the projects and transport included in the FIB file, these projects and transport will be created in this database.

5.11 Import Transport with SQL Trigger

If you import a transport with a customer license you sometimes get a license error that a field or object cannot be created. If you enable the option "Import Transport with SQL Trigger" all objects will be imported through a object number available in the customer license. Therefore fields and objects outside the customer license can be imported.

If you enable the option the Object Manager will install a trigger on the table "OM – Update Object". If the SQL database cannot be found or the active user is no DBO the trigger has to be added manual with e.g. SQL Server Management Studio.

Go to the OM – Update Object table and choose the option to create a new trigger.

Then paste the SQL statement (see appendix - Update Object Table SQL Trigger) in the Query window and press execute.

Now it is possible to enable the option "Import Transport with SQL trigger".

5.12 Confirm Changes at Importing Transport

When your customer has changed one or more objects in the database you get a warning when you import a transport.

Opening the "Object Compare Sheet" gives you an overview of the changes the customer has made since the previous transport.

In this example you see that the customer has changed the Item table and the report "Sales - Invoice". It is possible to analyze the changes with your compare tool or confirm the changes with the confirm button.

The option "Confirm Changes at Importing Transport" will always open the "Object Compare Sheet" when importing a transport if objects are changed. All modifications have to be confirmed before you can import a transport.

NOTE: Set options "Save C/AL Before Import" and "Save C/AL After Import" on "If changed" or Yes.

5.13 Object Compare Sheet

To see what your customer has changed since the last transport you can open the "Object Compare Sheet". Use the "Open Transport File" option to compare the changes against a transport file. See section - Confirm Changes at Importing Transport for more information.

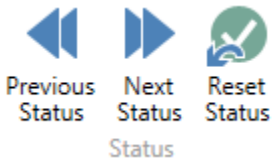
5.14 Reset Project Status at Importing Transport

When you are developing in an environment with a development, test and live database and you want to transport your projects from the test database to the live database you can use the "Reset Project Status at Importing Transport" option in the setup.

When you import a transport you will see an extra option in the transport import wizard.

All the projects that you import with a transport will have the default status of the project flow and the projects will be deleted from the transport. You can recognize this by the grey color in the transport.

Because the field "Transport No." is cleared on the project card it is possible to include them in another transport.



NOTE: Use another "Transport Nos. Format" in your test database then in your development database.

5.15 Rollback Objects

To Rollback object changes due to a transport, press Transport – Rollback to open the "Rollback Objects" window, and then press Functions – Rollback (or F11) to execute the rollback.

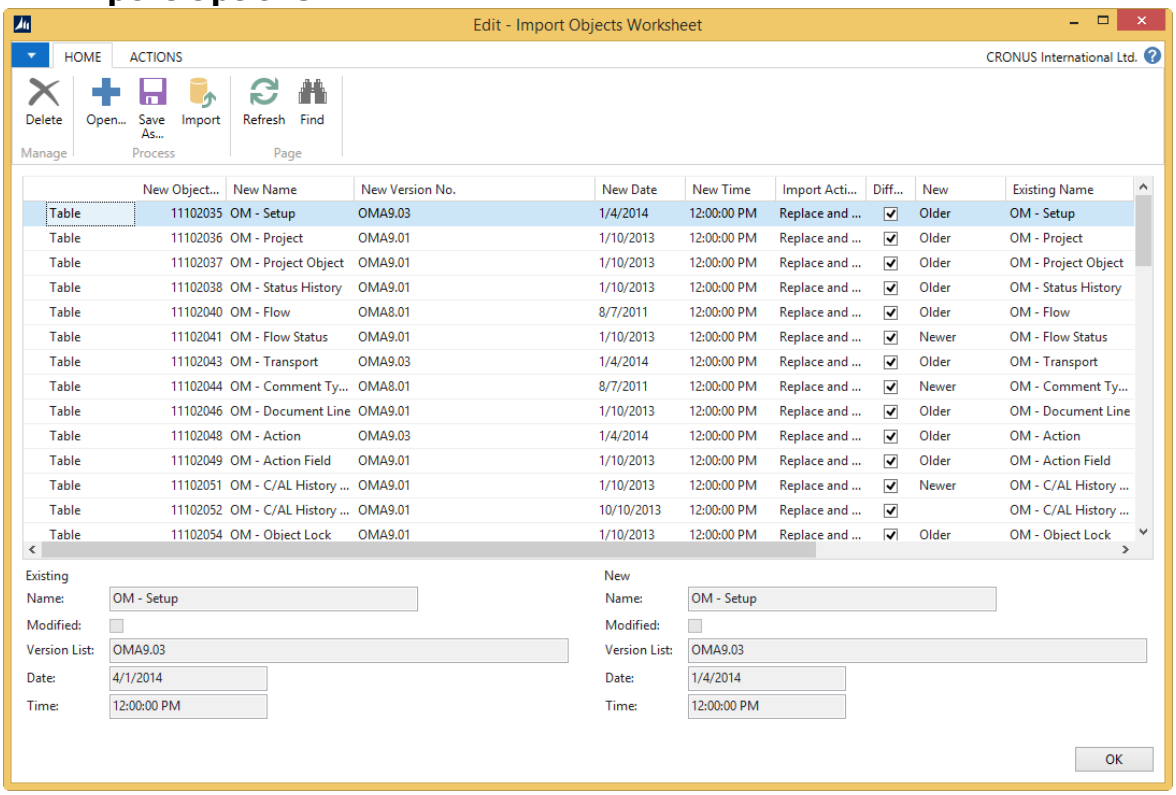
For more information about Check Guidelines see section – Rollback Objects.

6 OBJECTS IMPORT WORKSHEET

The Objects Import Worksheet is used to import objects. You can open files in OBJ, TXT, OBP, and FIB formats. It is also possible to import a directory with split text files.

6.1 IMPORT OBJECTS

6.1.1 Import Options



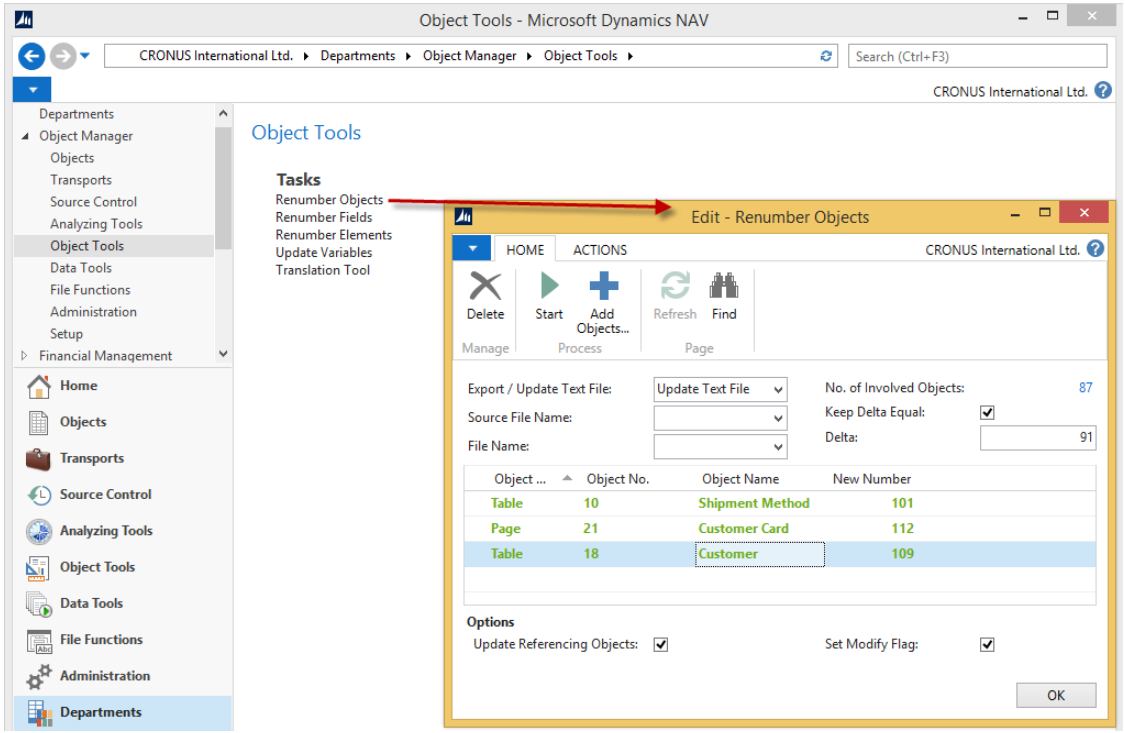
Before importing the objects you can set the action that has to be executed.

- **Replace**
  Replaces the object without compiling.
- **Replace and Compile**
  Replaces the object and compiles it. This is often used in combination with the text format. When there is an error in one of the objects the complete import will be rolled back.
- **Import Properties**
  This action only imports the properties of the objects. Properties are name, date, time, version list and modify flag.
- **Skip**
  Skips the object from import.

6.1.2 Change ID and/or Name

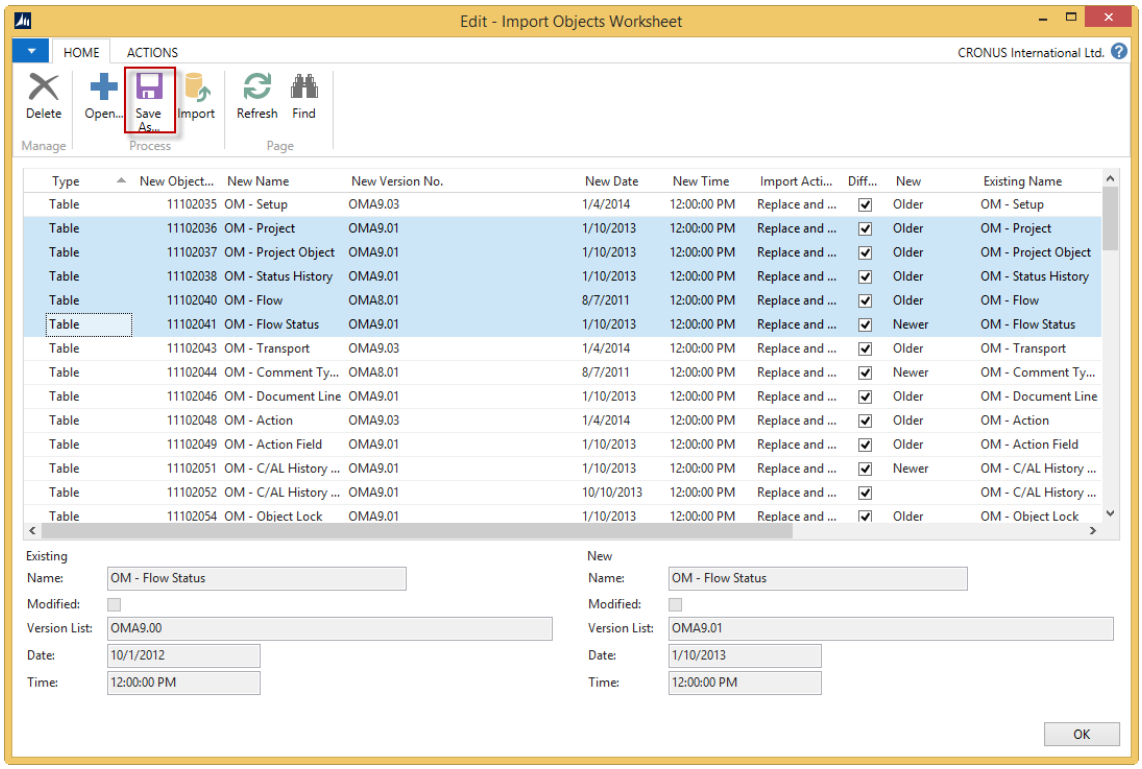
It is possible to give the objects another id and/or name.

NOTE: References to these objects will not be renumbered. Use the "Renumber Objects" tool if you want to renumber with updating all references. (For more information see chapter - Renumber Objects)



6.2 Export Objects

It is also possible to select a number of objects and export them in the corresponding format as they were imported.

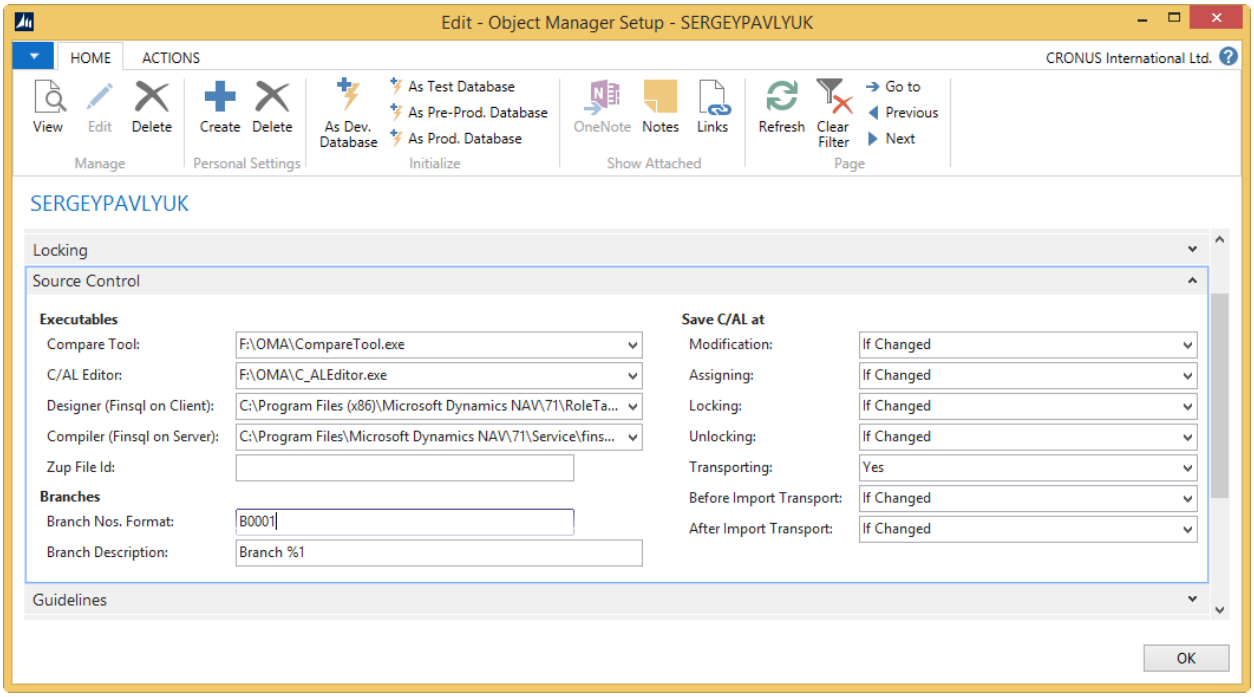


7 VERSION AND SOURCE CONTROL

With this tool you can save versions on different trigger moments.

Also you can start up a compare tool (like Beyond Compare) and compare the code with previous object versions in the system.

7.1 SETUP



Compare Executable

Here you can set the executable that you use for comparing the C/AL code. For example Beyond Compare or UltraCompare.

%1 will be replaced with the left file/directory

%2 will be replaced with the right file/directory

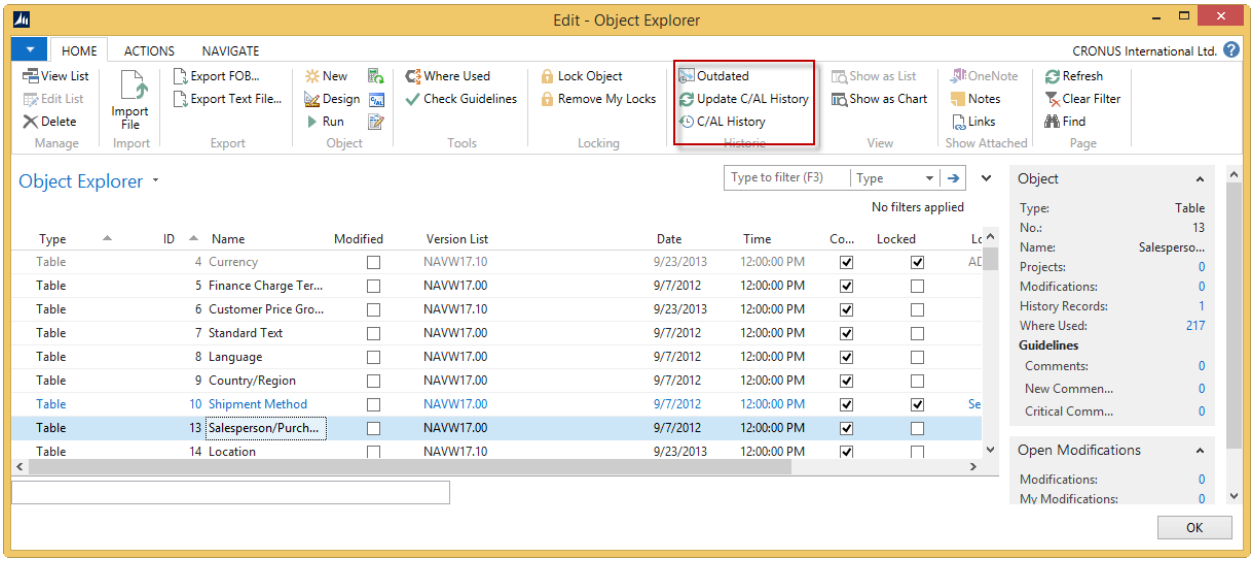
Compare Directory

Here you set the directory that the Object Manager uses to save the C/AL code before opening the compare tool. If you leave it empty your default temp directory will be used.

NOTE: The Object Manager creates folders and files in your "Compare Directory". Periodically you have to clean this folder.

7.2 Update C/AL History

Open the Object Explorer and select outdated objects. An object is outdated when the C/AL History is not up to date.



Press "C/AL History" > Update.

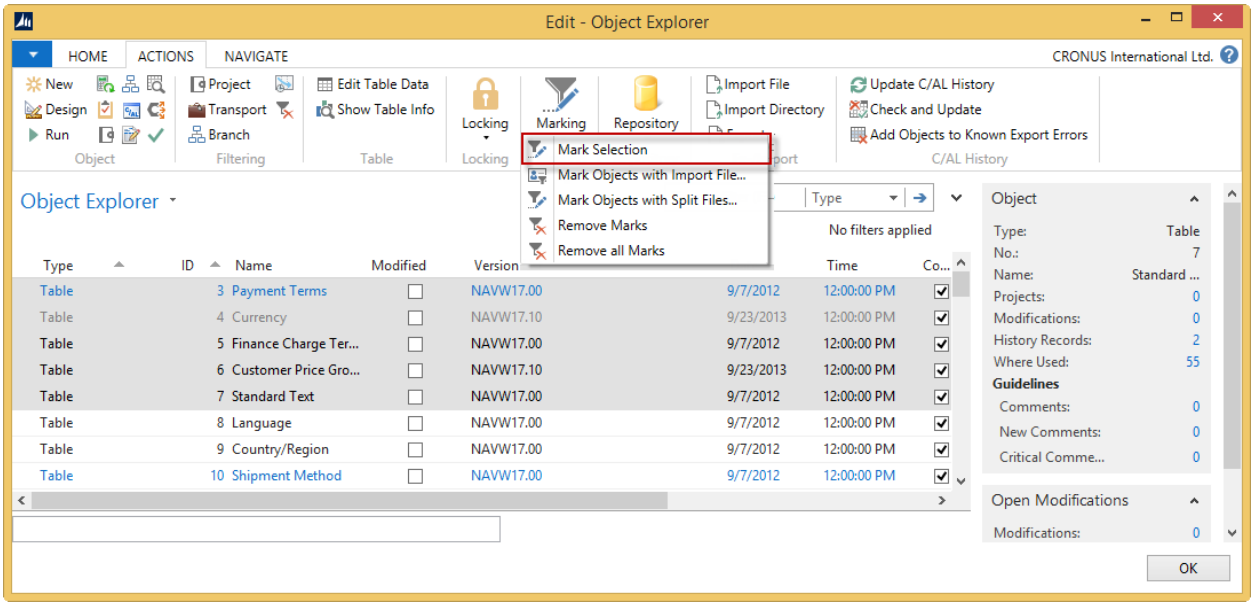
7.3 Check and Update C/AL History

The Object Manager only checks if the properties of an object are changed. If you e.g. imported objects with a text file and the date and time of the objects are still the same but the C/AL of the objects is changed the Object Manager will not update the C/AL History. If you know that there are such changes in your database you will have to use the "Check and Update" option. Now the Object Manager will compare the contents of the objects with the C/AL History. This is a time consuming process.

7.4 Update C/AL History with Text File

When you do not have installed the DLL (for more information see section 2.1-Installing) you can also manually update the "C/AL History" by selecting the outdated objects in the Object Explorer and export the objects in text format and import the objects in the Object Explorer.

1. Select outdated objects.
2. Press "Mark Selection".



3. Press "Select Marked Objects in Designer".
4. Export the objects in text format.
5. Import the text file in the Object Explorer.

NOTE: When you update the C/AL History with a text file the compiled version of the objects is not saved. Therefore it can be difficult to do a rollback.

7.5 Automatic C/AL Saving

When you perform some actions in the Object Manager it is possible to save the current C/AL to the "C/AL History".

Go to Setup > Source Control and set the triggers:

Save C/AL at Modification

Save C/AL at Assigning

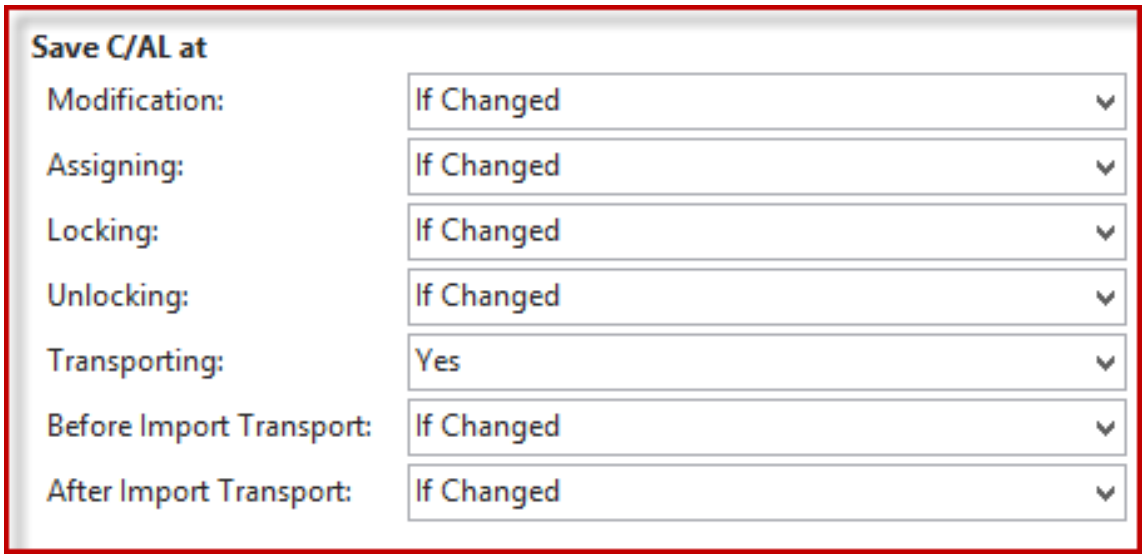
Save C/AL at Locking

Save C/AL at Unlocking

Save C/AL at After Transporting

Save C/AL at Before Importing

Save C/AL at After Importing



You can set the 7 different triggers on 3 values: "No", "If changed" and "Yes".

Yes: C/AL will be saved always when trigger action is executed.

If changed: C/AL will only be saved when the date or version list of the object has changed.

No: C/AL will not be saved.

Example

If you modify an object with the trigger "Save C/AL at Modification" on "Yes" or "If changed", you can view the "C/AL History" with the Object Explorer.

In this example we change the length of the field Name from 30 to 200. Save and compile the Vendor table.

Select the Vendor table and "Show C/AL History" in the Object Explorer. The "C/AL History Objects" form opens.

In this list you can see al changes in the object from the moment you started tracing. In the example you see the first line is the "original" object, the last line is your last change. The field "Action Type" shows which trigger saved the history.

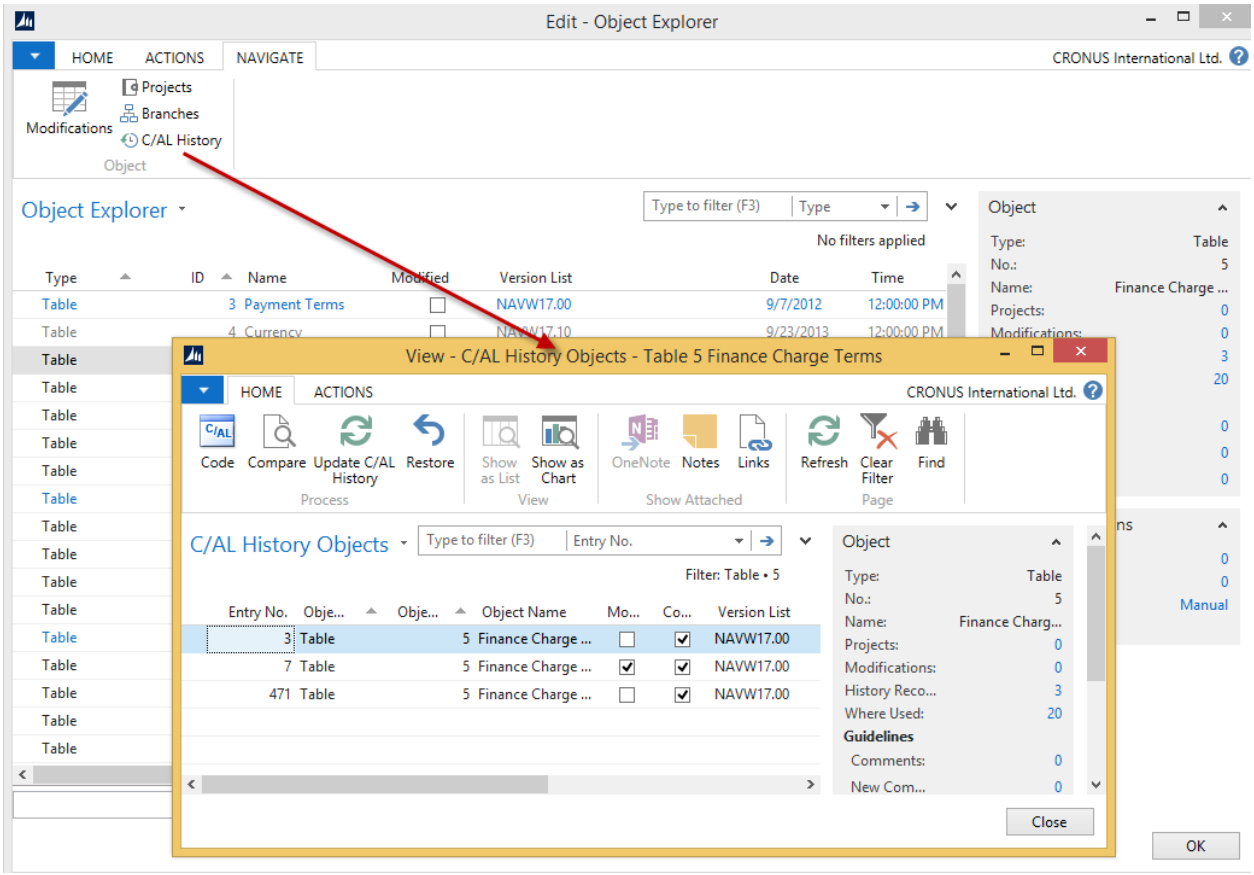
With the button Code you can see the code.

C/AL History Lines.

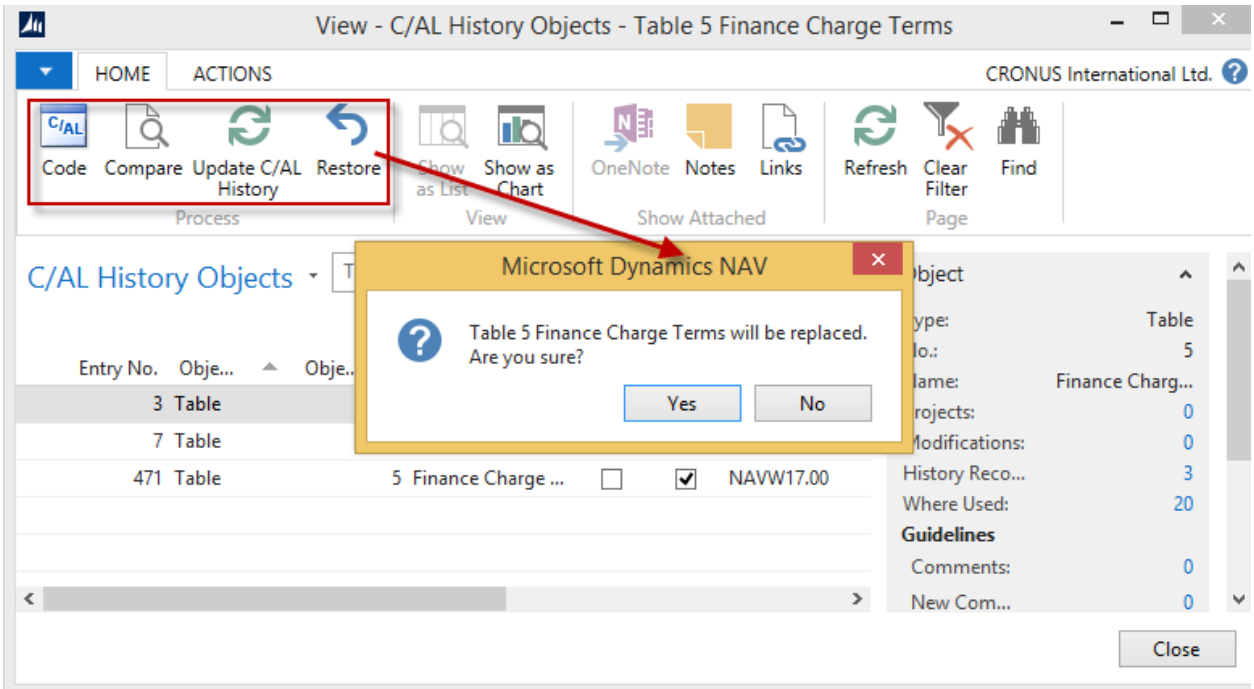
To analyze the changes you can use the function Compare.

7.6 Restore an Object

In the "C/AL History Objects" form you can restore an object to a previous version.



Press Restore.



A new line of "Action Type" Restore is added to the "C/AL History" table.

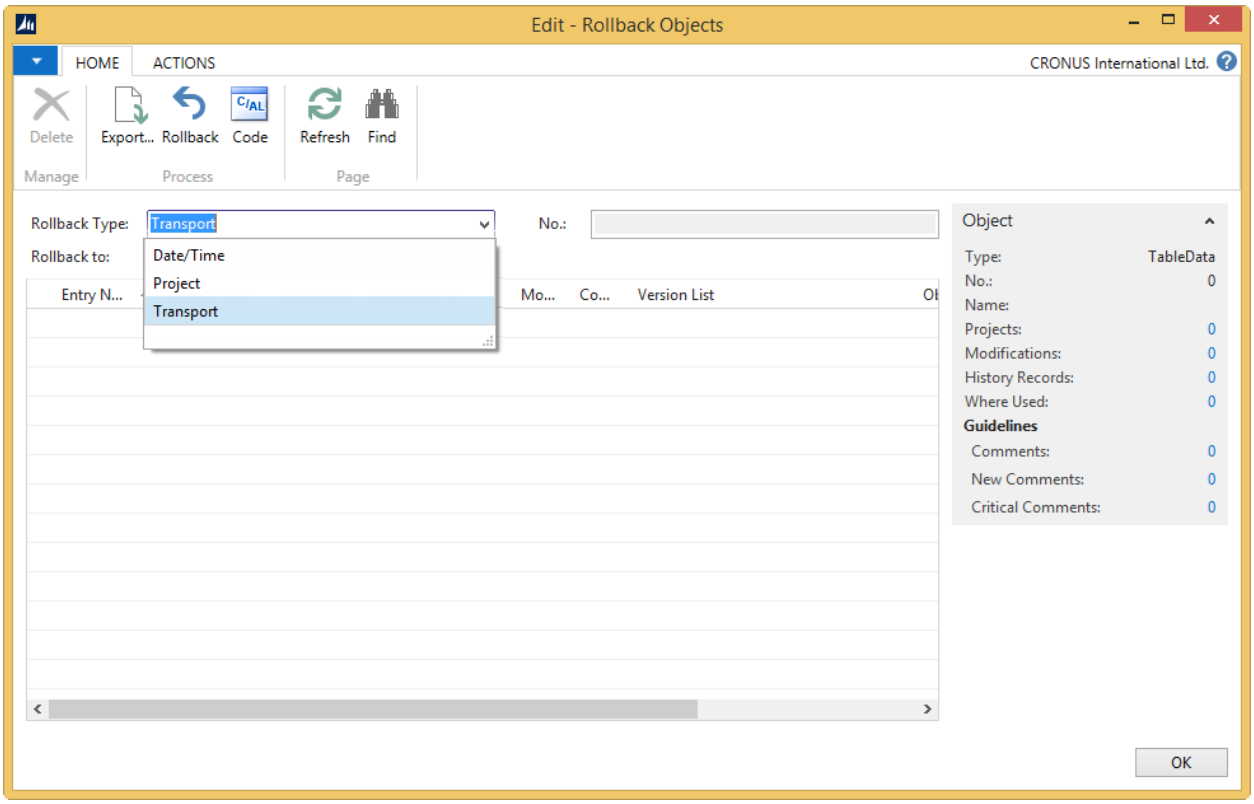
7.7 ROLLBACK OBJECTS

When you want to rollback object changes you open the "Rollback Objects" window. Using the "Rollback Type" you can rollback to a certain date/time combination, for a specific project or specific transport.

Go to **Departments > Object Manager > Source Control > Tasks > Rollback Objects.**

Date/Time
Enter a date/time combination in the "Rollback to" field

Project or Transport
Enter a project/transport number in the "No." field

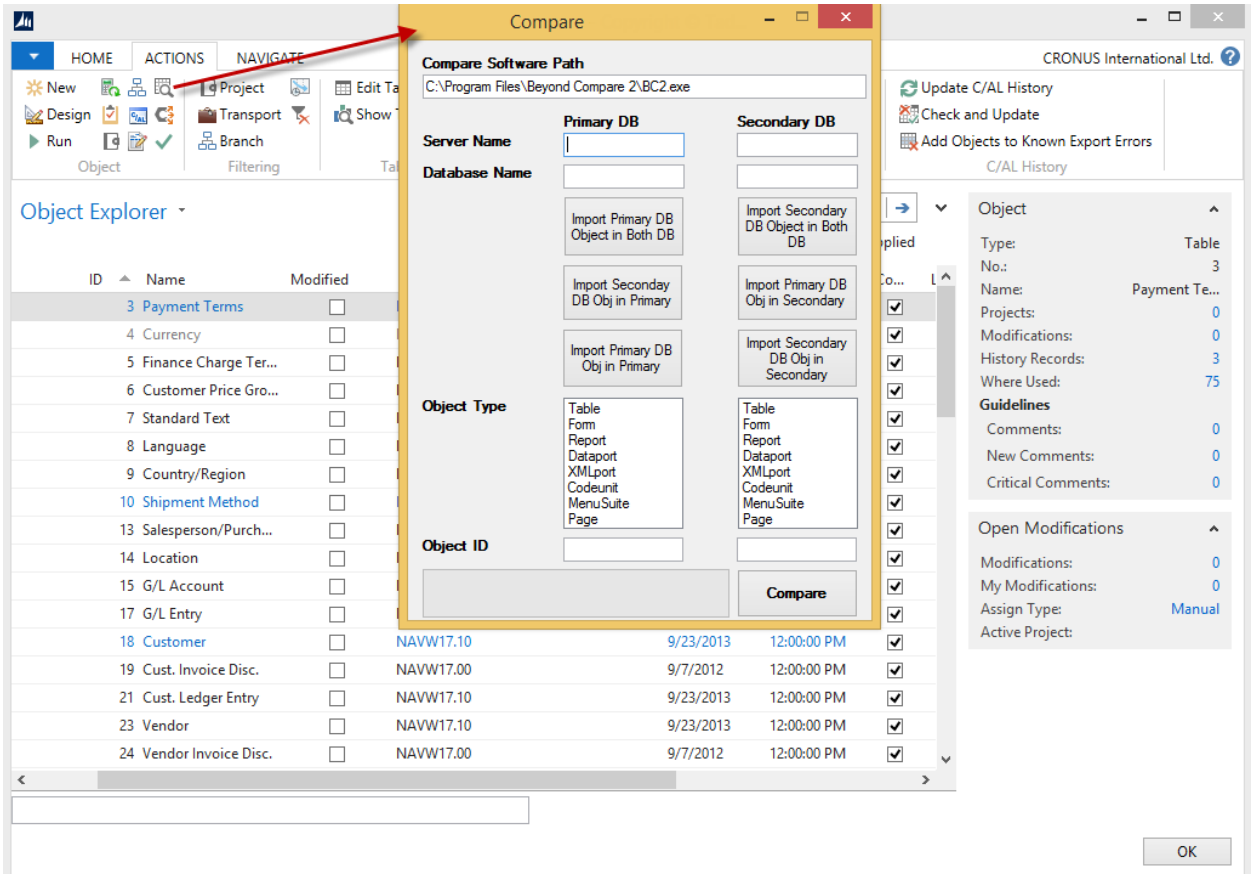


Object Manager will retrieve all the objects that were changed since that date or for that specific project/transport.

Press Functions – Rollback (or F11) to execute the rollback.

7.8 Compare two Versions

"Show Last Changes" will open your compare tool and show the changes you made compared with the previous version of the object.



7.9 Analyze Modifications Made in a Project

You can compare changes of a project by using the function "Show Changes in Directory".

This function opens the compare tool and shows you the differences in objects and code of the specific project.

It is also possible to see the changes in one file. For example if you use UltraCompare which does not have the option of comparing directories.

7.10 Analyze Modifications Made in a Transport

You can compare changes in a transport by using the function "Show Changes in Directory".

This function opens the compare tool and shows you the differences in objects and code of the specific transport.

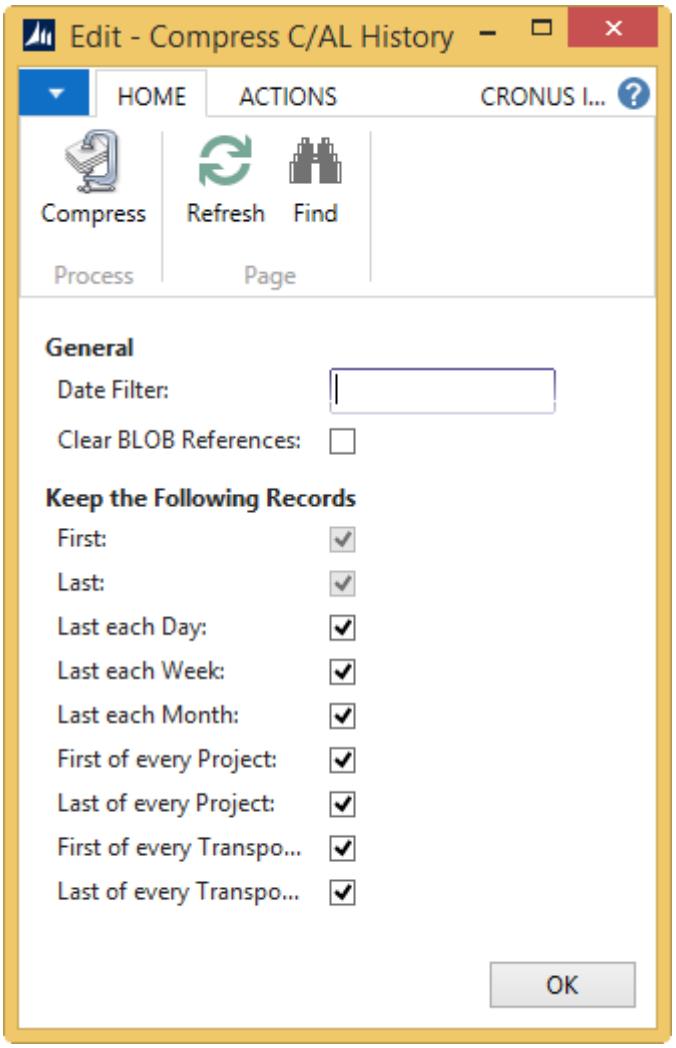
7.11 Analyze Modifications Made in a Period of Time

With the report Compare C/AL Code you can analyze all the modifications that are done in a specific period of time.

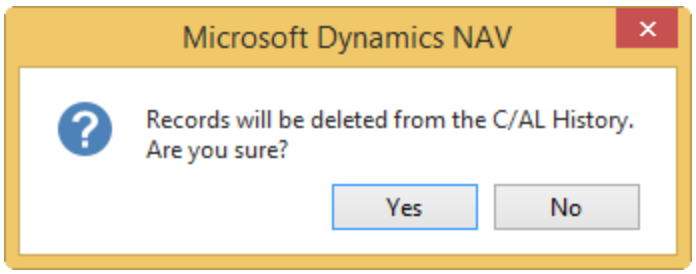
7.12 Compress C/AL History

After a while the C/AL history can be very big. To compress these tables you can use the option "Compress C/AL History" in the Administration menu.

Go to **Departments > Object Manager > Administration > Tasks > Compress C/AL History.**



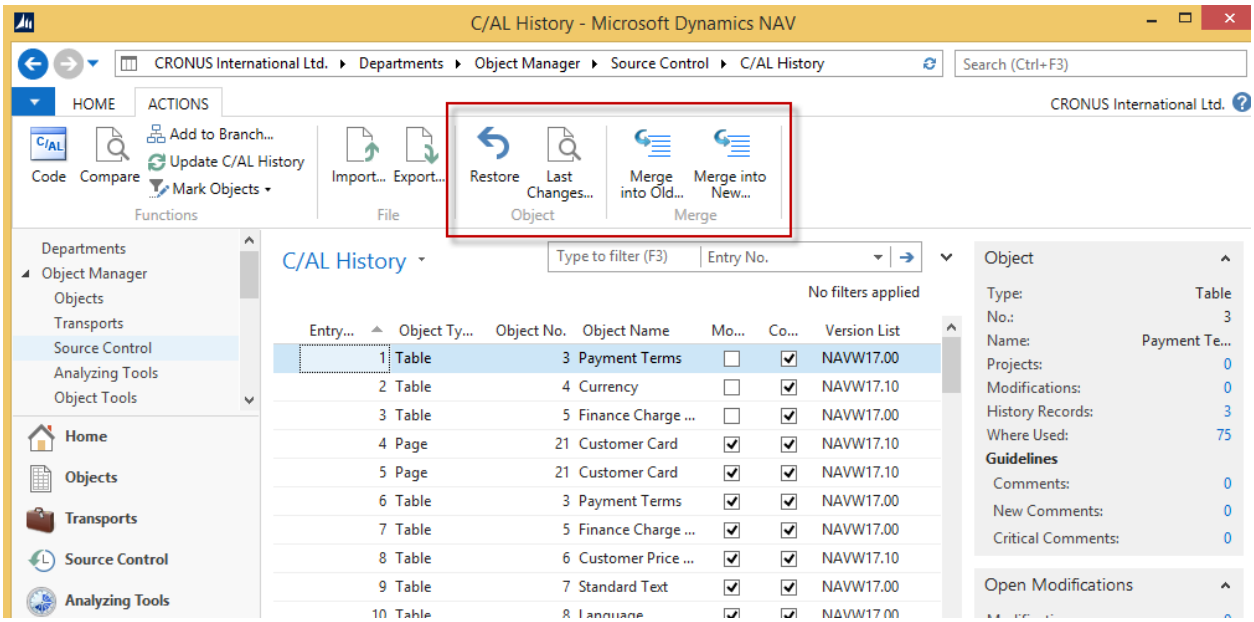
Check the options you want to keep and press Compress to delete all records that does not comply to one of the options.



You can also enter a Date Filter to only compress the C/AL History that is older than e.g. a couple of months. If you choose the option "Clear BLOB References" then all compiled version of the objects in the history will be deleted. After this you cannot restore an object or do a rollback.

7.13 MERGE C/AL HISTORY

With this option it is possible to merge two C/AL History records in an external editor and save your changes automatically back into the database.



There are 2 options:

Merge into Old

The oldest C/AL History record will be imported in the database when you are done.

Merge into New

The newest C/AL History record will be imported in the database when you are done.

Select the two history records you want to merge and press Merge into New.

The merge tool will open where you can merge the differences. In our example the customer table overwritten by a colleague and your change was lost.

We merge the change into the new version and close the merge tool. Now you see the dialog in NAV that is waiting at your confirm.

Compile

Object will be compiled.

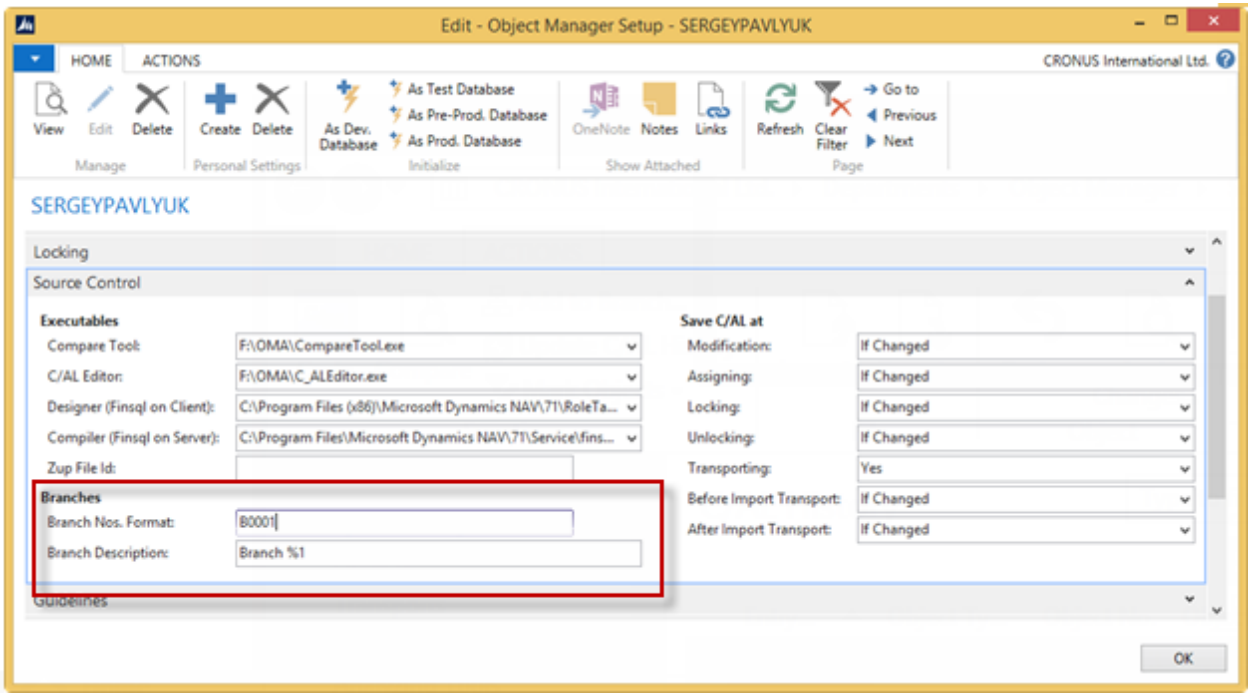
Set Modified Flag

The timestamp of the object will be changed to the current date and time and the modify flag will be set. If this option is disabled the date and time of the merged text file will be used.

8 BRANCHES

A branch is used when your development team needs to work on two distinct copies of a set objects at the same time. With branches it is possible to reserve, group, develop and take a deeper look at selected versions of objects. It makes it possible to work on reserved versions of objects or even develop simultaneously on the same object.

8.1 SETUP



For Branch Nos. Format and Branch Description:

%1 will be replaced by the "Branch No.".

You can use date expressions like: <Day> <Month Text> <Year4>.

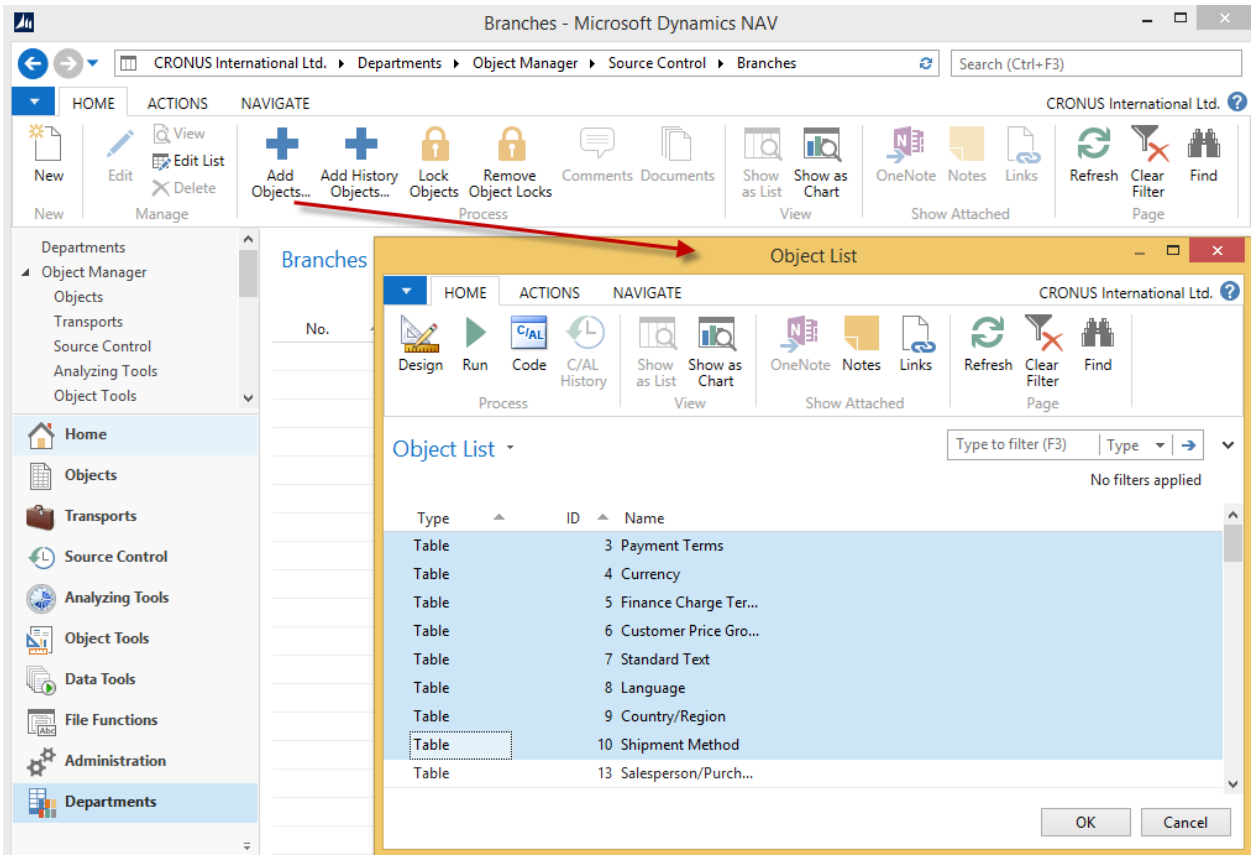
8.2 COMMENT GROUPS

Comment Groups are similar to Comment Groups in projects. For more information see section - Comment Groups.

8.3 DOCUMENTS

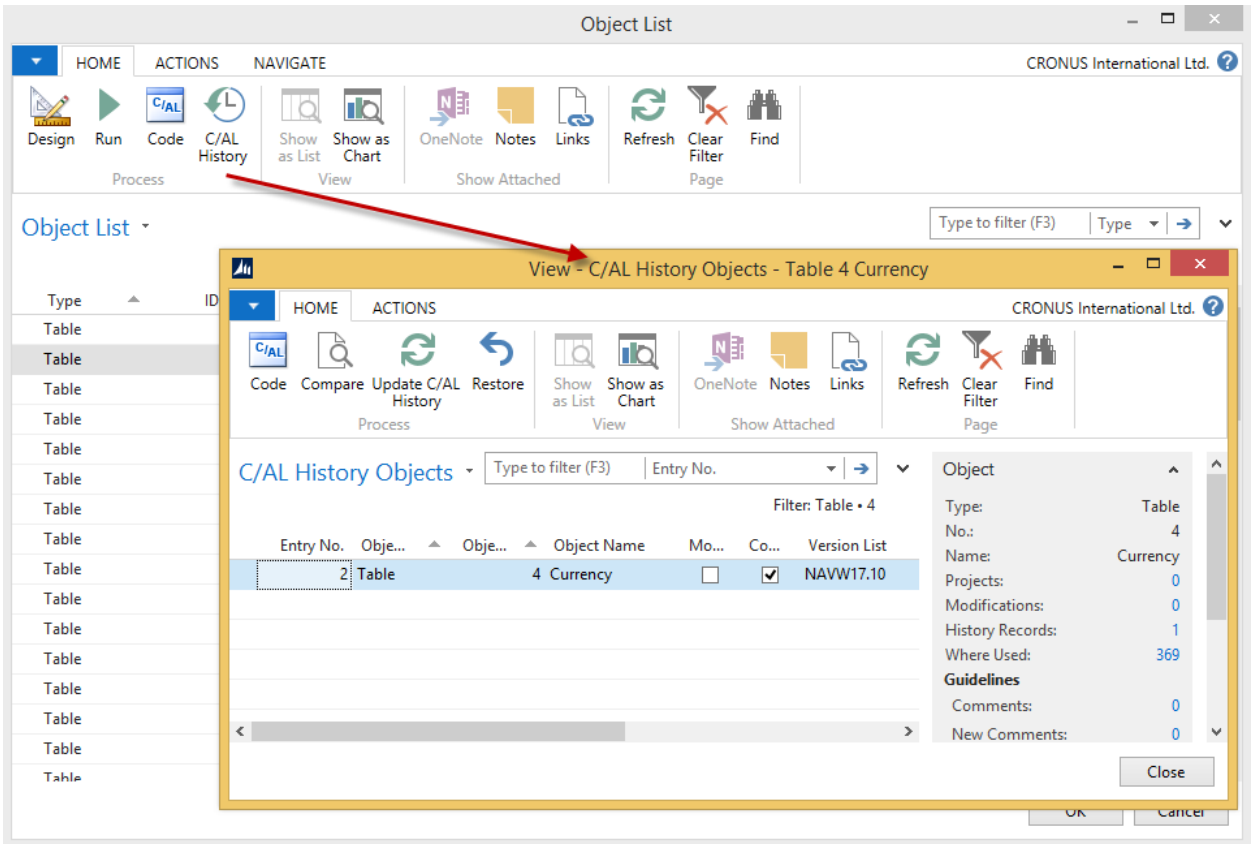
Documents in branches are similar to Documents in projects. For more information see section - Documents.

8.4 Add Objects to a Branch



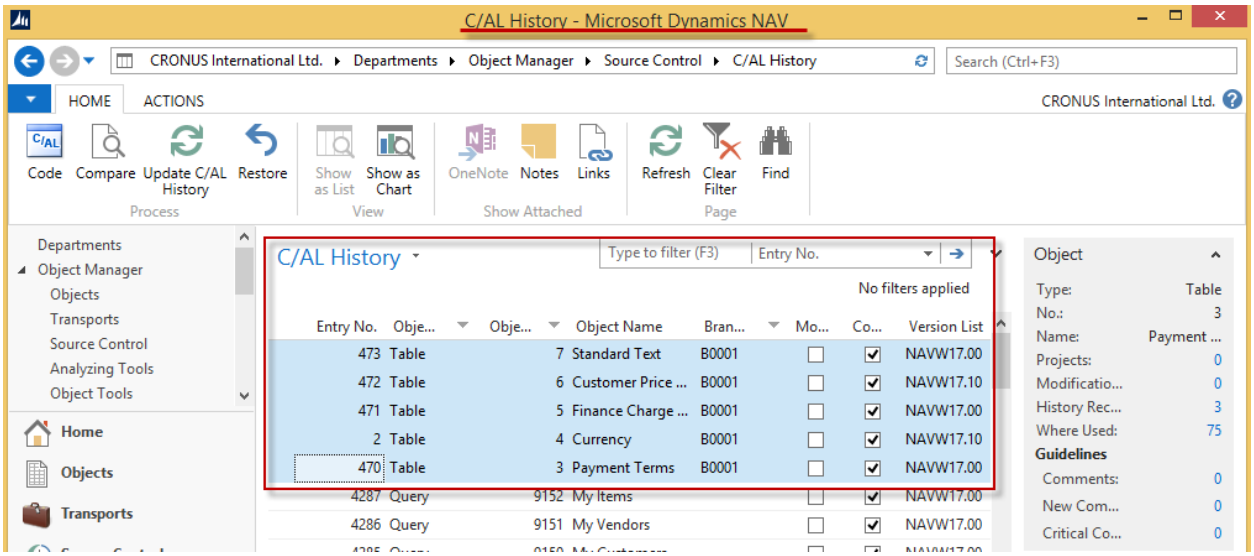
You can add different objects to a branch. The current version is retrieved.

Or you can add a selection from the "C/AL History" table to your branch if needed.



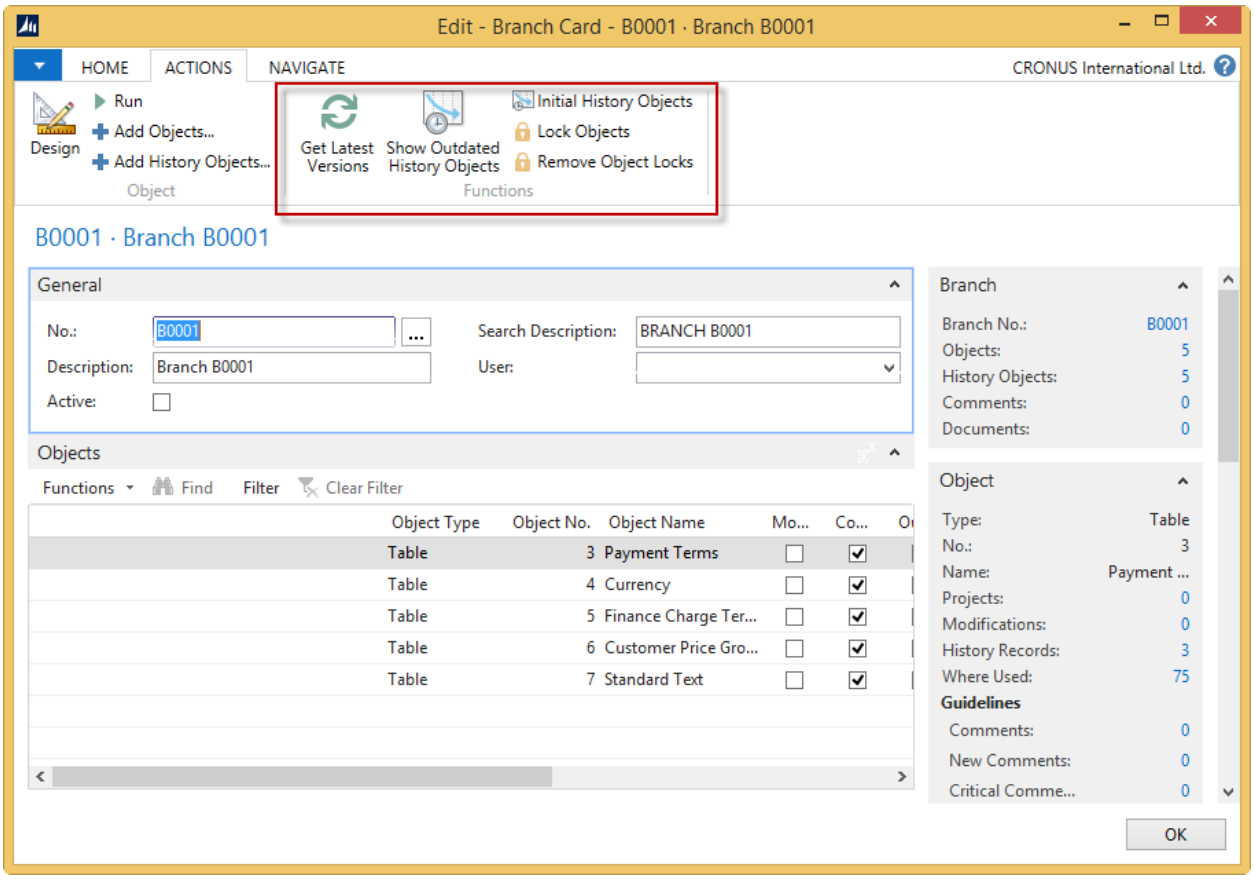
In the example above you see two lines for the table car. The first one is the latest version. The second line is an older version. It is possible to expand and collapse an object to see only the latest version.

In the C/AL History table the branch is added to the selected versions.



You can add an object from the Object Explorer and add C/AL History Lines from the C/AL History to a Branch.

8.5 Outdated Objects in a Branch



When an object in your branch in not the current version they have a Boolean in the outdated field.

There are three possibilities to cope with this conflicts.

Get Latest Versions

If the version of an object in your branch is outdated a new line for the latest version will be added.

Show Outdated History Objects

Shows the latest version of the outdated objects in your branch. This can be used to roll back the objects in the database to the last known version of the branch.

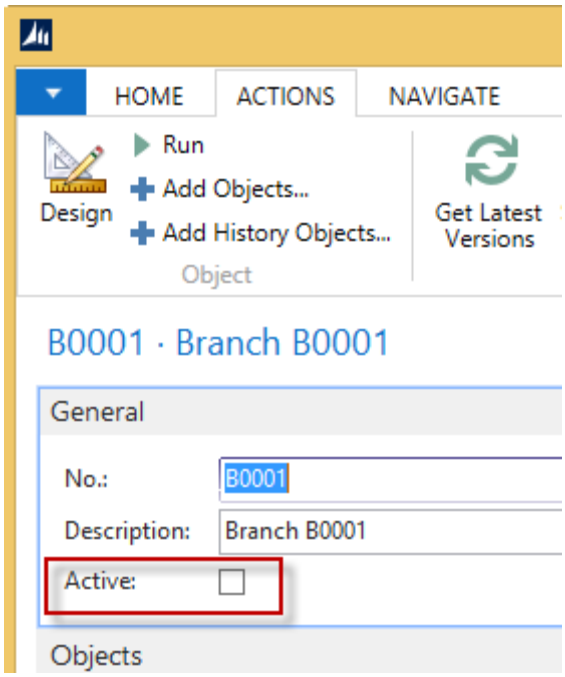
Show Initial History Objects

Show the first version of the objects in the branch. This can be used to roll back the objects in the database to the initial state of the branch.

8.6 Activate a Branch

When you want to work on a branch and reserve the objects in the branch you can activate it. All objects in the branch will be locked by you.

If an object is active in another branch you get an error.

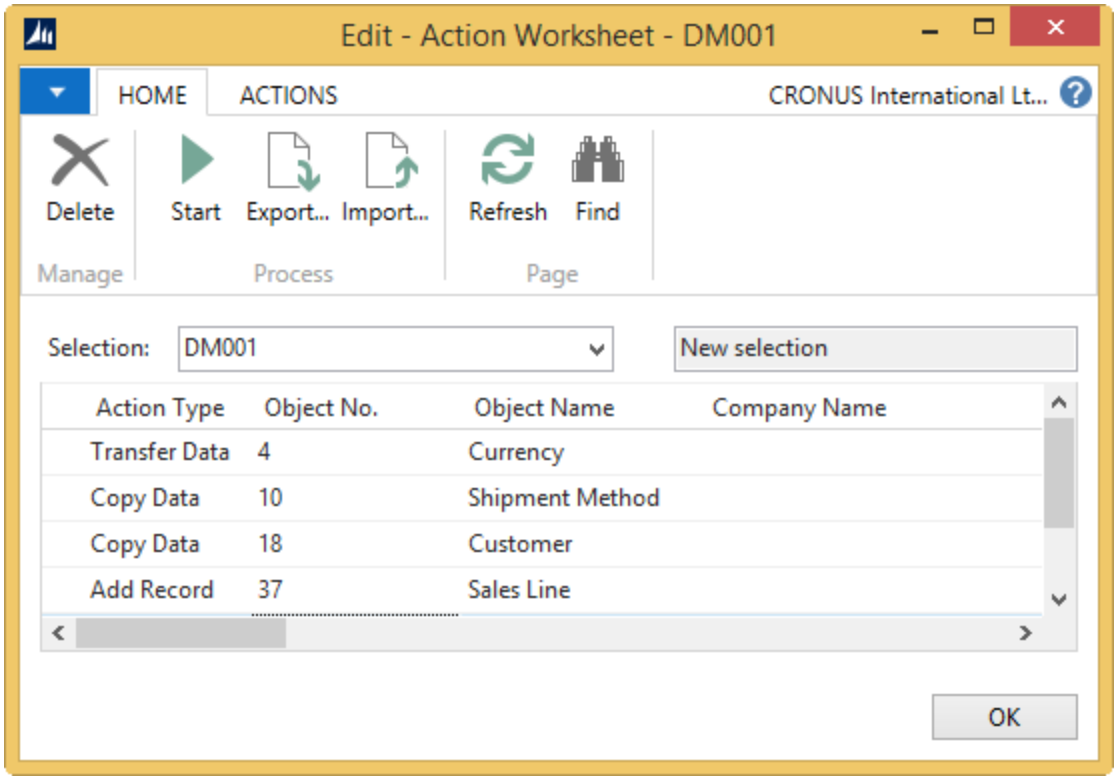


If there is an object in your branch that is outdated you get the following error.

Because of this two checks it is possible to have an object in more than one branch with totally separated functionality.

9 ACTION WORKSHEET

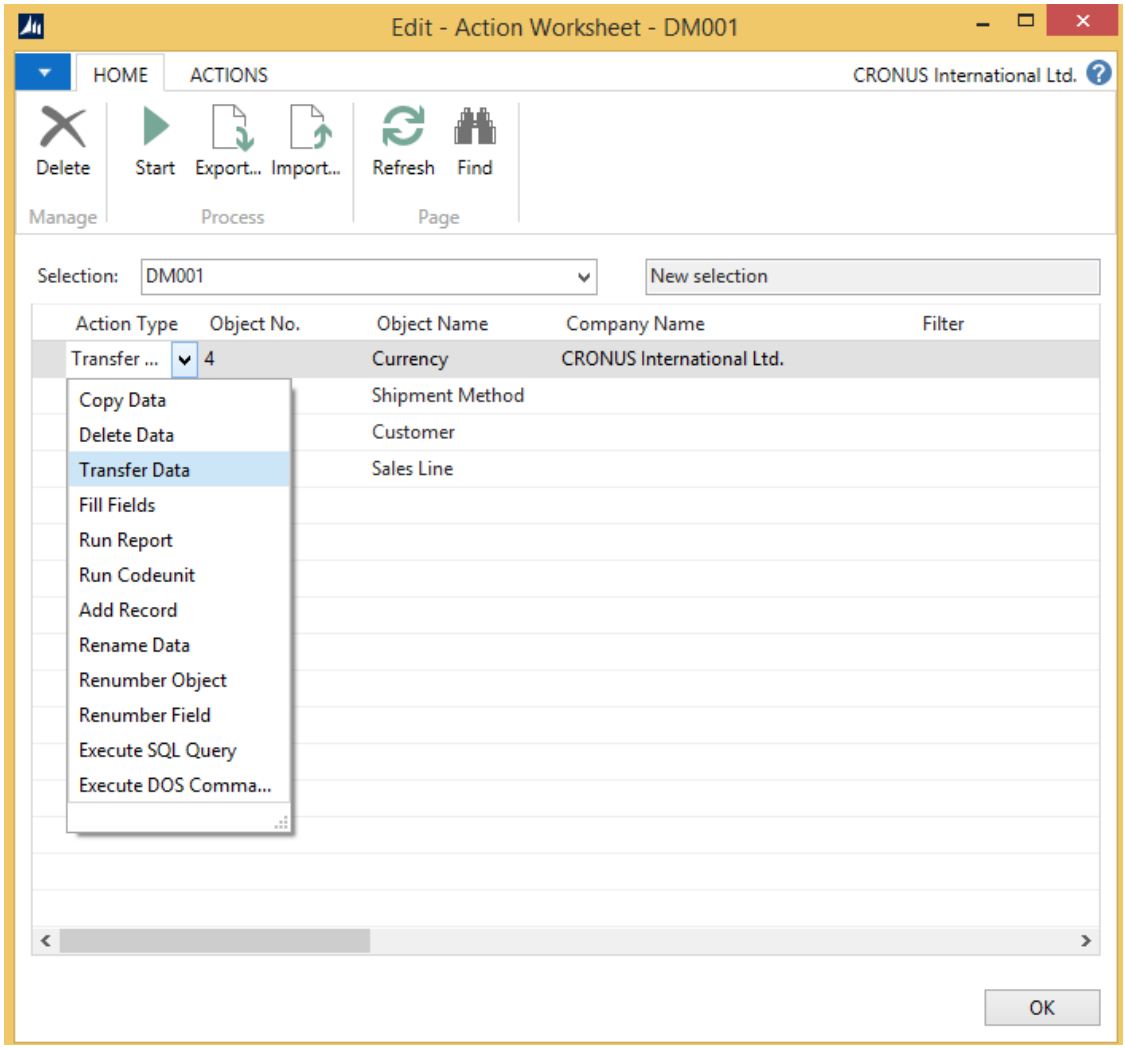
You can perform actions in your own database, like filling fields, but you can also include them in a project so you can transport these actions to the customer database and execute them in there.



You can find the "Action Worksheet" in menu "Data Tools" or access it from the project- or transport card.

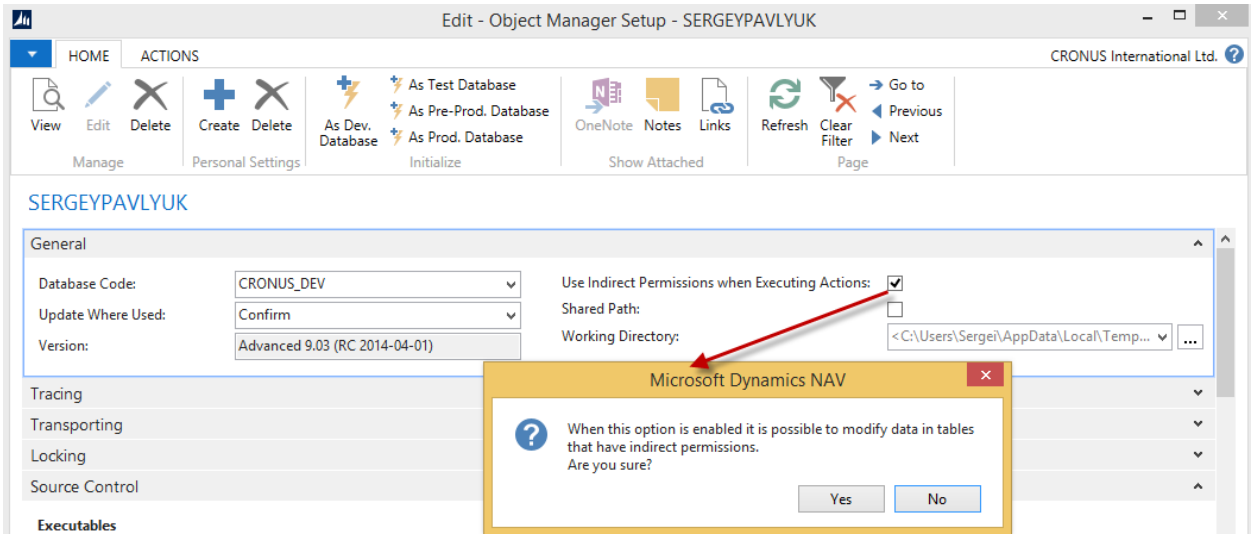
Action Types

- Copy Data: copy data between tables or companies
- Delete Data: delete records in a table or empty fields
- Transfer Data: transfer data between databases
- Fill Fields: fill fields with values
- Run Report: run a report
- Run Codeunit: run a codeunit
- Run Dataport: run a dataport
- Rename Data: Renames data
- Renumber Object: renumbers an object
- Renumber Field: renumbers a field
- Execute SQL Query: executes a SQL query
- Execute DOS Command: executes a DOS command



9.1 SETUP

When you are using a customer license and you want to update data that is only allowed through indirect permissions you can enable the setting "Use Indirect Permissions" when executing actions.



Now it possible to modify data in tables like "G/L Entry".

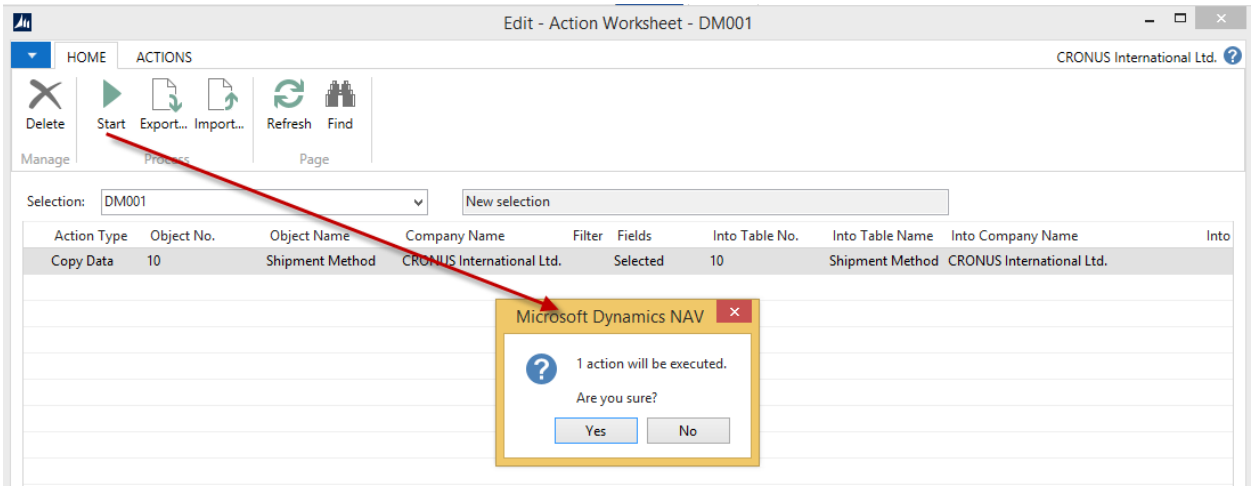
9.2 Copy Data

You can copy data in the same or another table or between two companies.

Example

Copy the Customer Address to Address 2

1. Select the fields that you want to copy
2. Select the table where you want to copy the data to. In our case the same Customer table.
3. Click the assist-edit button of "Into Fields" to map the fields where you want to put the data in.
4. Press Start to get the result.



9.3 Delete Data

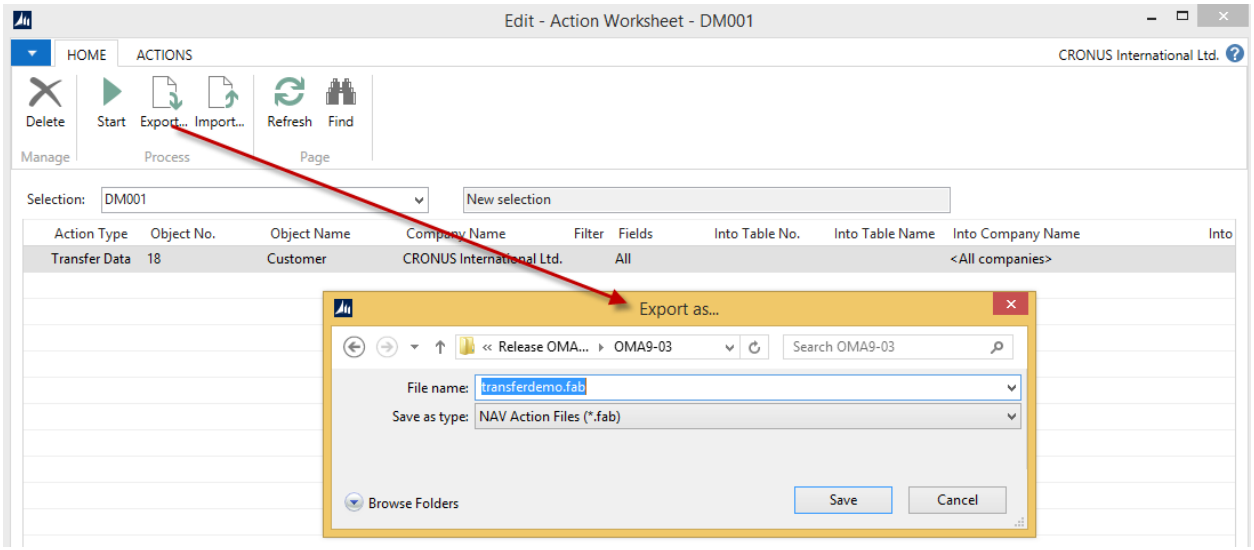
With this action type you can delete data from all records, filtered records and from specific fields. So if you have to delete a field from a table you can first empty it with this action type.

9.4 Transfer Data

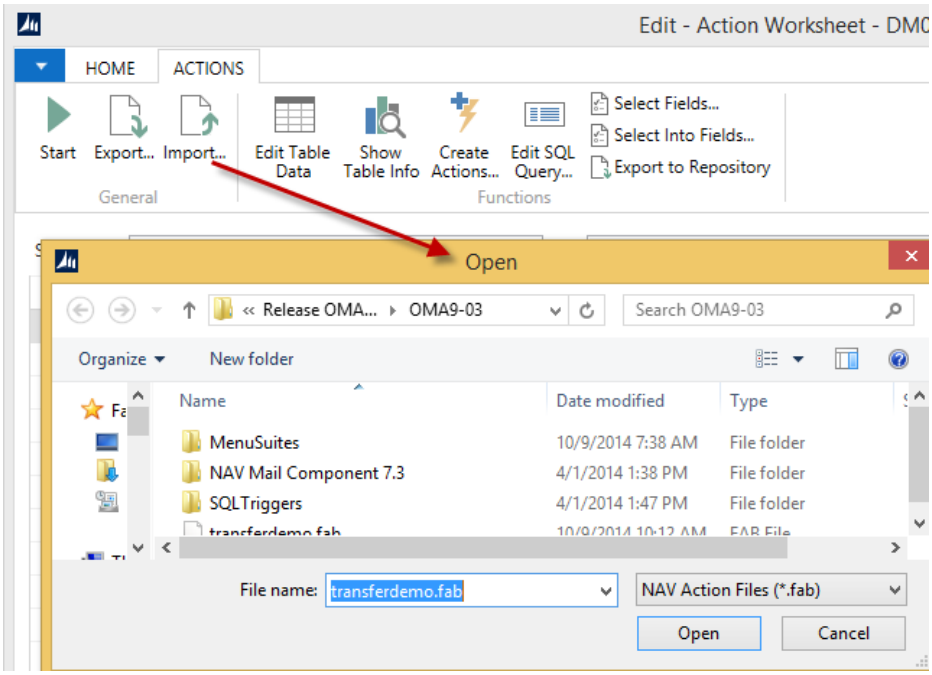
Transfer data from one table to another table in another database.

Example

1. Action Type "Transfer Data"
2. Fill in the "Table No." were you want to transfer data from
3. Press Export



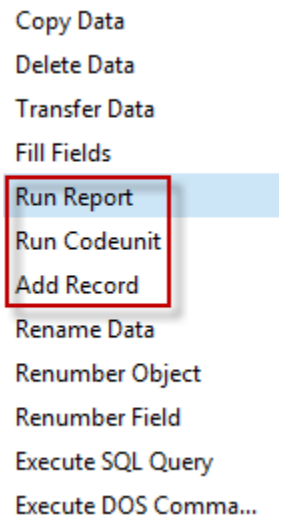
4. The Object Manager Exports the action + data as a FAB file
5. Save the file
6. Open the Customer database
7. Open the "Action Worksheet"
8. Press Import
9. Open the FAB file



10. The Action is imported in the "Action Import Worksheet"
11. Press Start
12. Data from the development database is now in your Customer database

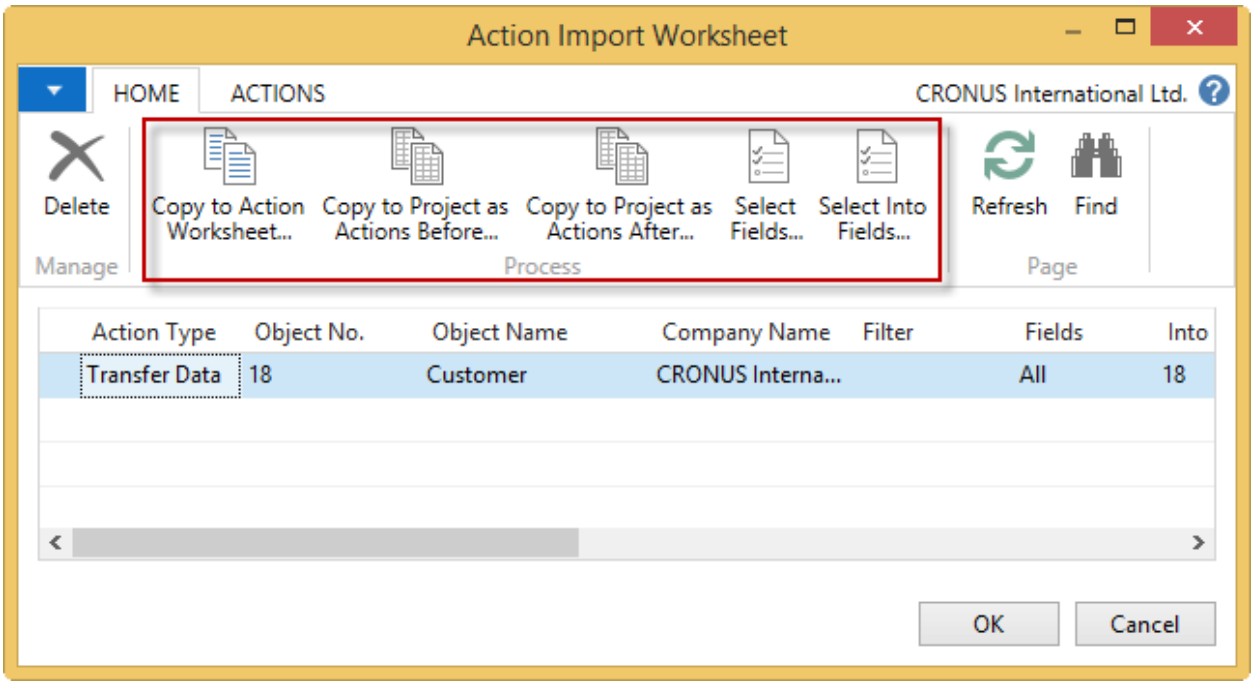
9.5 Run Report, Codeunit or Dataport

You can execute reports, codeunits or dataports.



9.6 Renumber Object

The renumber object function can be used to give an object another number. All references to this object will not be changed. So if you are doing a renumber action in your development database it is preferred to do this with the renumber objects function. For more information see chapter- Renumber Objects. If this renumber action is also needed in your customer database then it is possible to copy this renumber action to a specific project as "action before" with the function "Copy to Project as Action Before".



NOTE: Make the "Actions Before" before you start the renumbering.

9.7 Renumber Field

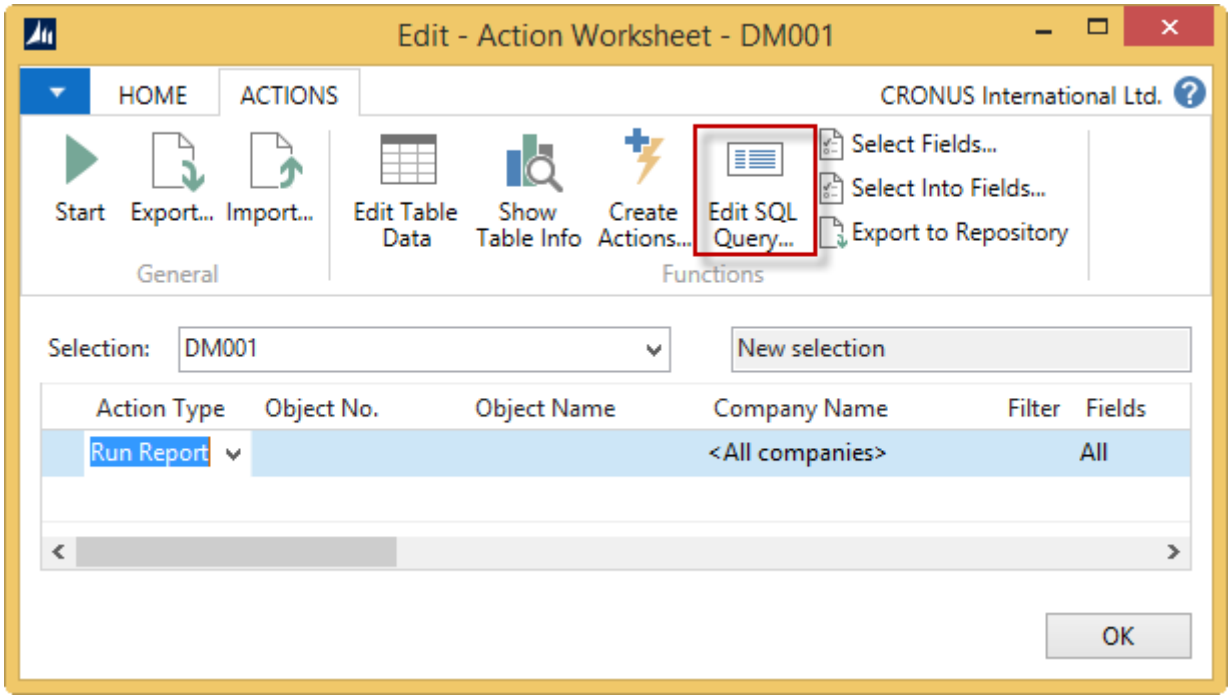
The renumber field function can be used to give a table field another number. All references to this object will not be changed. So if you are doing a renumber action in your development database it is preferred to do this with the renumber fields' function. For more information see chapter - Renumber Fields. If this renumber action is also needed in your customer database then it is possible to copy this renumber action to a specific project as "actions before" with the function "Copy to Project as Action Before".

NOTE: Make the "Actions Before" before you start the renumbering.

9.8 Execute SQL Query

You can execute a SQL query

Press Functions > Edit SQL Query and make your query in your text editor. When finished save the SQLQuery.txt file and press OK in the pending dialog.

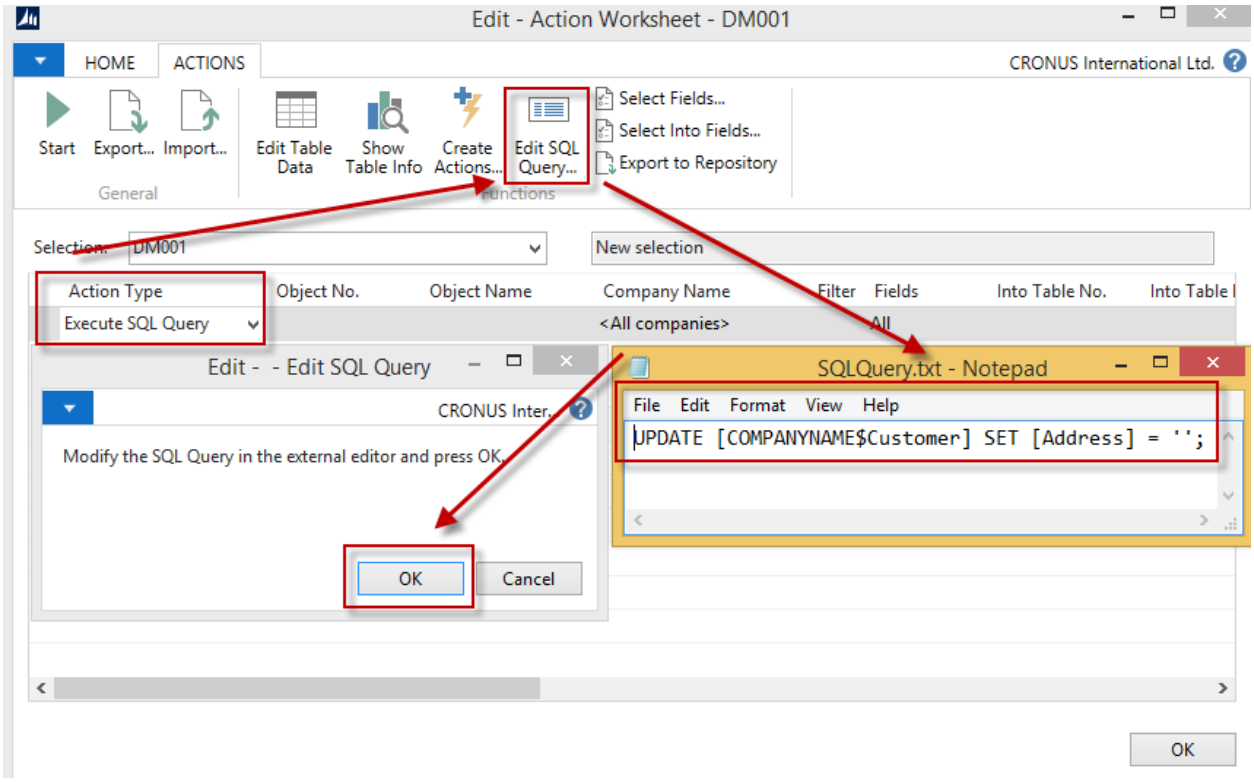


NOTE: The string COMPANYNAME will be replaced by the companies you have selected in the action.

9.8.1 Example SQL Query to Change Data

If you e.g. want to move the content of the address field to the address 2 field in the customer table you can use the following query.

```
UPDATE [COMPANYNAME$Customer] SET [Address 2] = [Address] WHERE [Address] <> '';
UPDATE [COMPANYNAME$Customer] SET [Address] = '';
```



9.8.2 Example SQL Query to Add a View

If you want to send a LinkedObject to your customer database you also want the corresponding view or table to be created in your customer SQL database.

You can do this by creating two actions of type "Execute SQL Query".

The first one is to remove the existing view.

```
IF EXISTS (SELECT * FROM sys.views WHERE object_id = OBJECT_ID(N'[dbo].[COMPANYNAME$No_ of
Customers per Location]'))
DROP VIEW [dbo].[COMPANYNAME$No_ of Customers per Location]
```

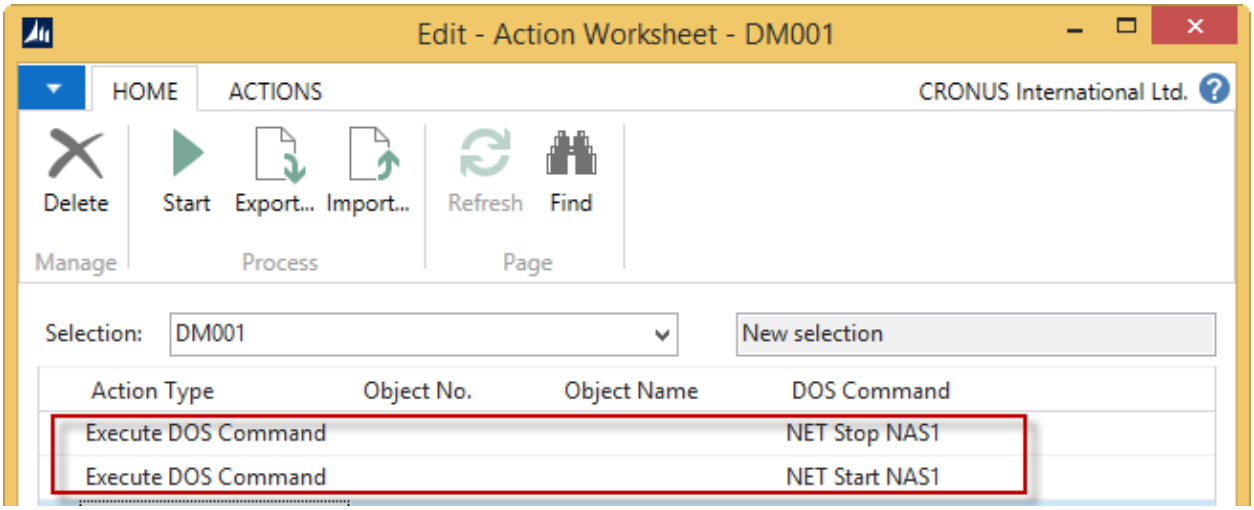
The second is to add the new view.

```
CREATE VIEW [dbo].[COMPANYNAME$No_ of Customers per Location]
AS
SELECT [Location Code], COUNT(No_) AS [No_ of Customers]
FROM dbo.[COMPANYNAME$Customer]
GROUP BY [Location Code]
```

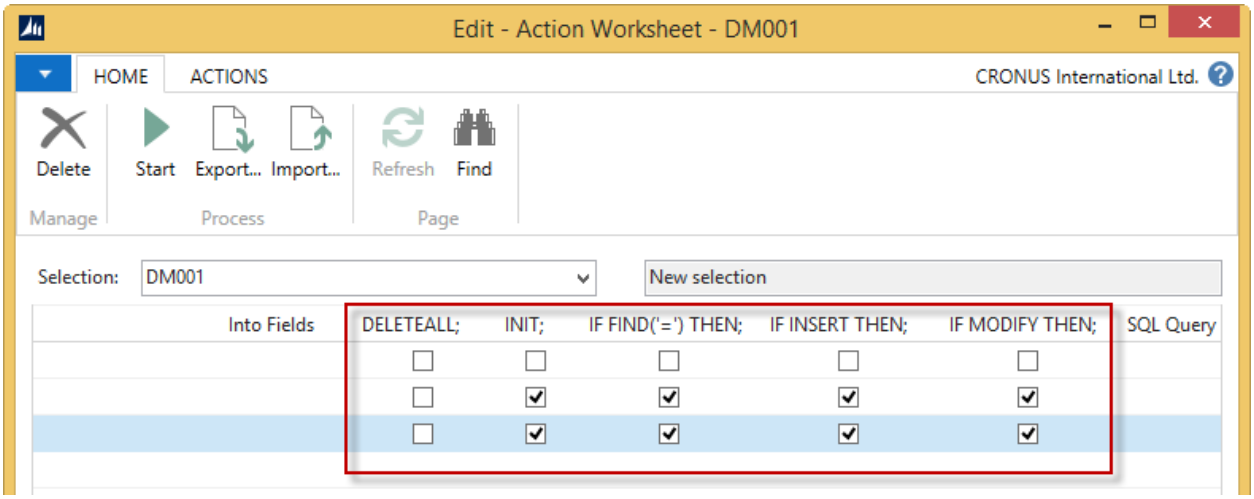
9.9 Execute DOS Command

You can execute a DOS Command as action.

This can be useful if you need to start an application or script during or after an import. This can also be used to restart a NAS.



9.10 Save Options



DELETEALL;

With this option the table will first be emptied before the action is executed.

INIT;

Every new record is first initialized before the action is executed.

IF FIND('=') THEN;

- True: if this option is enabled the action first reads the key fields into the new record and then tries to find the existing record. If found the existing record will be modified, otherwise a new record will be inserted (also depending on the next two options "IF INSERT THEN;" and "IF MODIFY THEN;").
- False: if this option is disabled a new record will be used if used in combination with "INIT;". The previous record will be used if the "INIT;" option is also disabled.

IF INSERT THEN;

If disabled, no new records will be created.

IF MODIFY THEN;

If disabled, no existing records will be modified.

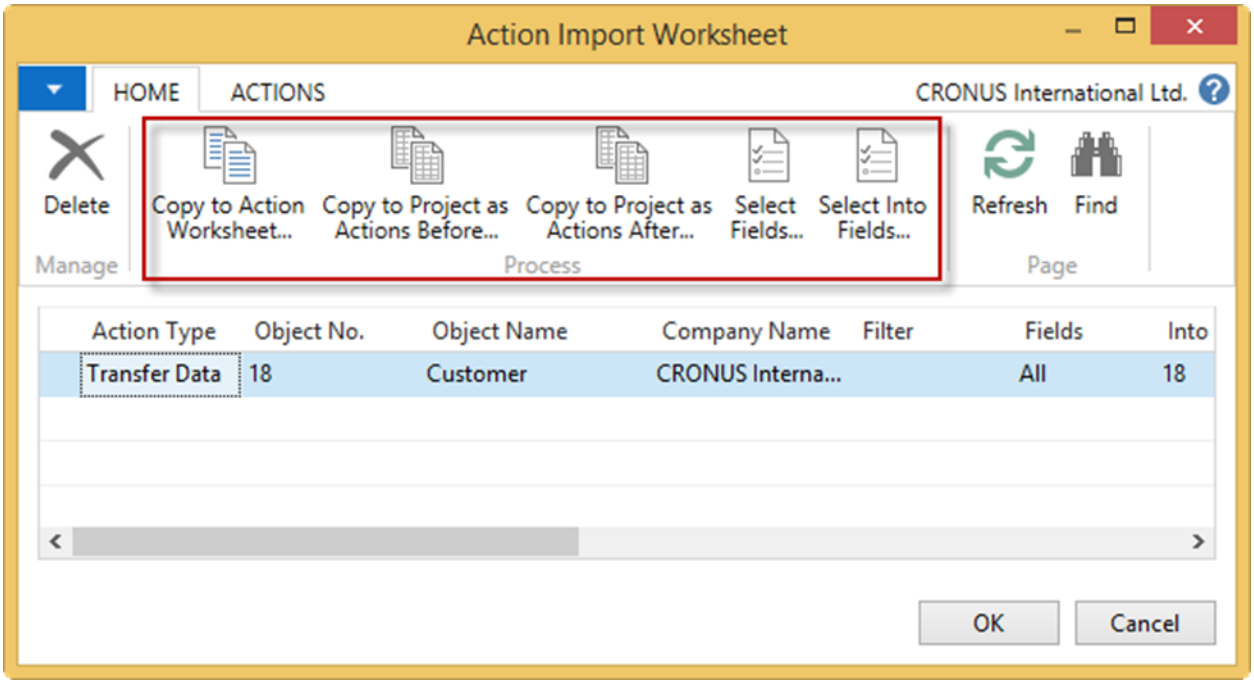
Commit Type

Indicates how many times a commit is executed.

- <EMPTY>: No committing is done. Only when all actions are executed
- At the end: Commit is done when this action is executed.
- After each record
- After 100 records

9.11 Add Actions to a Project

You can add actions to a project. These actions are executed in your customer database when you import the belonging transport. You can perform actions before reading the objects in a database and after.



Example

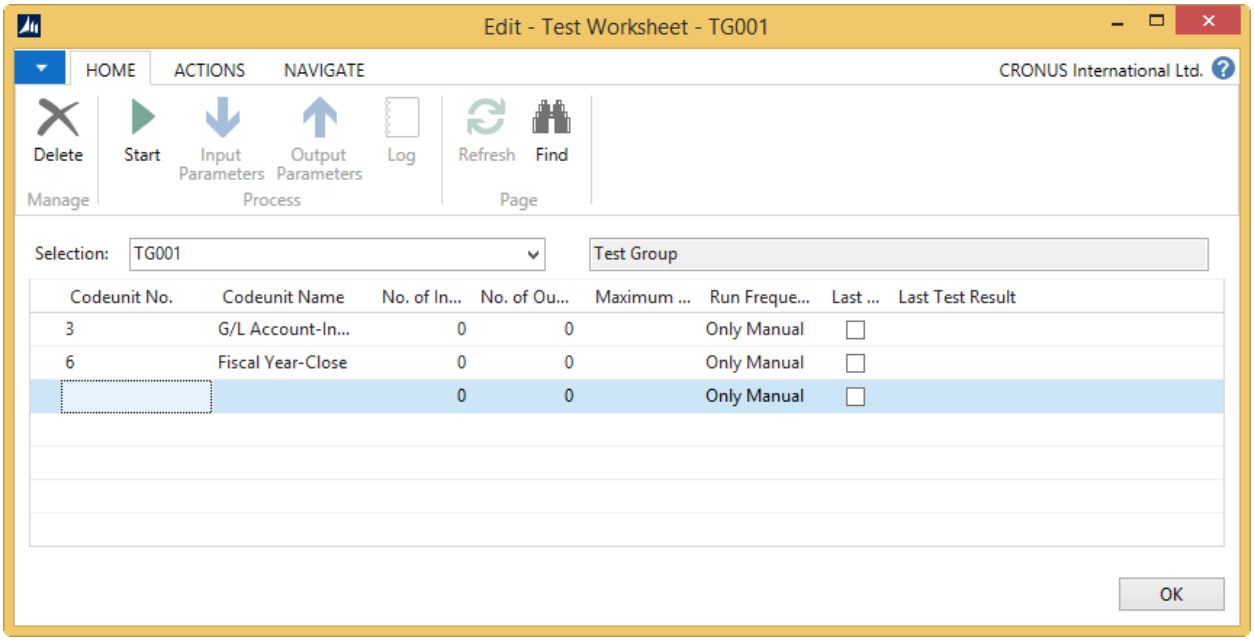
You have an existing table and you changed the property type of a field from Boolean to Option. If you read in a FOB file the conventional way, you will get an error if that field was filled in your customer database. You have to empty that field first for all records.

You can perform an "Action Before" that deletes the contents of that field before reading the new objects in the database. With these functions you can define the actions and export them later with the project into the transport file.

10 TEST FRAMEWORK

The Test Framework is used to automate tests. If you have a certain process that has to be tested before you transport objects to your customer database you can write a codeunit with input and output parameters that will be tested before the transport is executed.

10.1 CREATE TEST



You can find the "Test Worksheet" in menu "Analyzing Tools" or access it from the project- or transport card. If you add a test to a project it will be transported to your customer database and it can be tested before you do the transport.

A test has the following options:

Codeunit No.

The codeunit that will execute the test. See section - An example of a Test Codeunit for an example.

Codeunit Name

The name of the codeunit.

No. of Input Parameters

The number of input parameters that is used in the test codeunit.

No. of Output Parameters

The number of output parameters that is used in the test codeunit. If one of the output parameters has another value the test will fail.

Maximum Duration (ms)

If the duration of the test is longer than this value the test will fail.

Run Frequency

- Only manual: The test will only be executed manually in the test worksheet.
- Before transport: The test will be executed before it is transported. This option is only available if the test is added to a project
- Before every transport: The test is executed every time a transport is done.

Last Test Succeeded

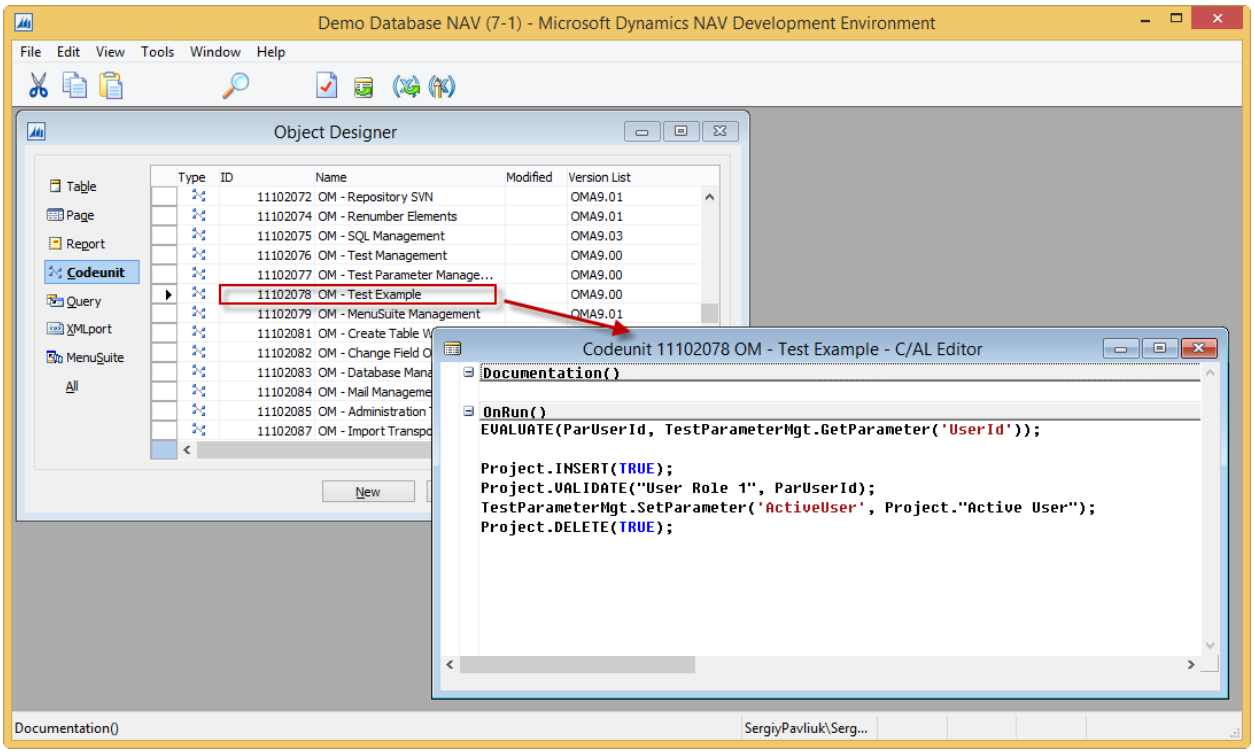
The result of the last executed test.

Last Test Result

The result of the last executed test. If an error has occurred the error message is shown in this field.

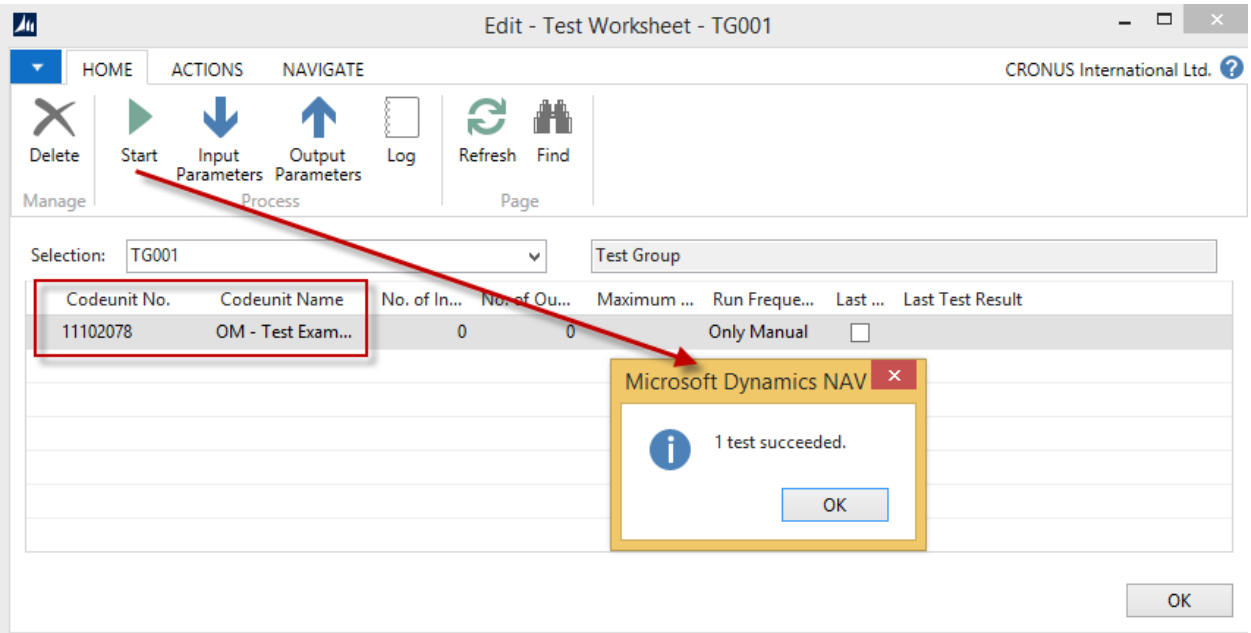
10.2 Example of a Test Codeunit

Codeunit 11102078 - OM - Test Example is a simplified example of how a test codeunit could look like.



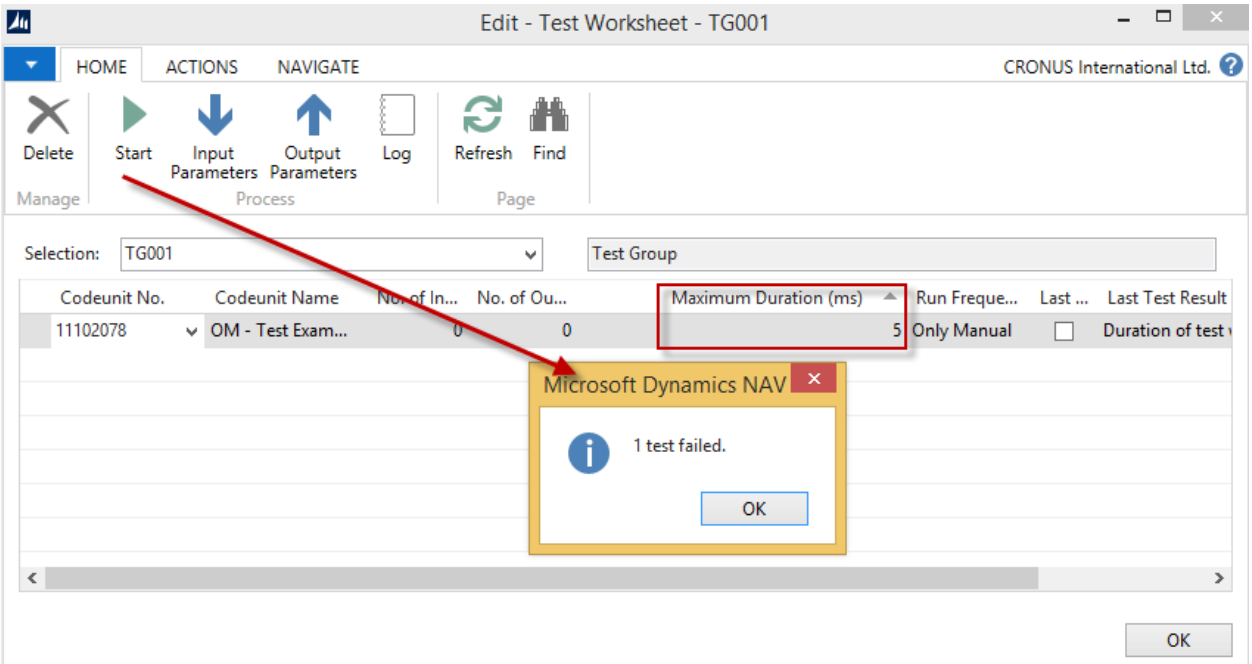
This codeunit will test if in a new project the user that is validated in the first user role will be the active user.

In this example you see that it has 1 input parameter and 1 output parameter.



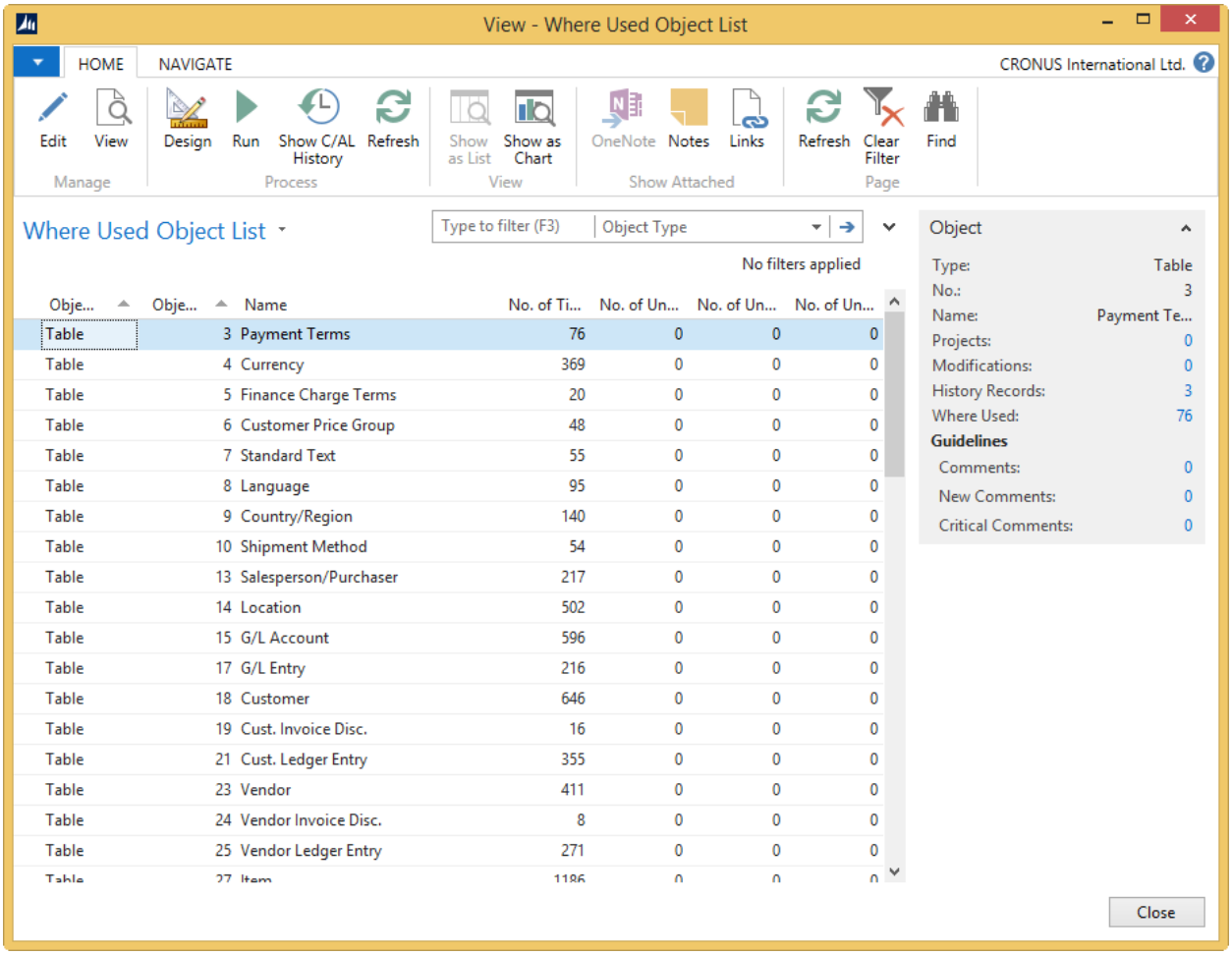
The result of this test will be "Test succeeded".

If you lower the maximum duration to 5 ms. the test will fail and the result will be "Duration of test was 17 ms".



11 WHERE USED FUNCTIONALITY

With this tool you can check where objects, fields, triggers, etc. are used. It is also possible to see if an object is unused and if it is called without validation.



You can search on the following type of objects:

Object

Trigger

Key

Sum Index Field

Field

Global Function

Local Function

Global Variable

Local Variable

Parameter

Return Value

11.1 SETUP

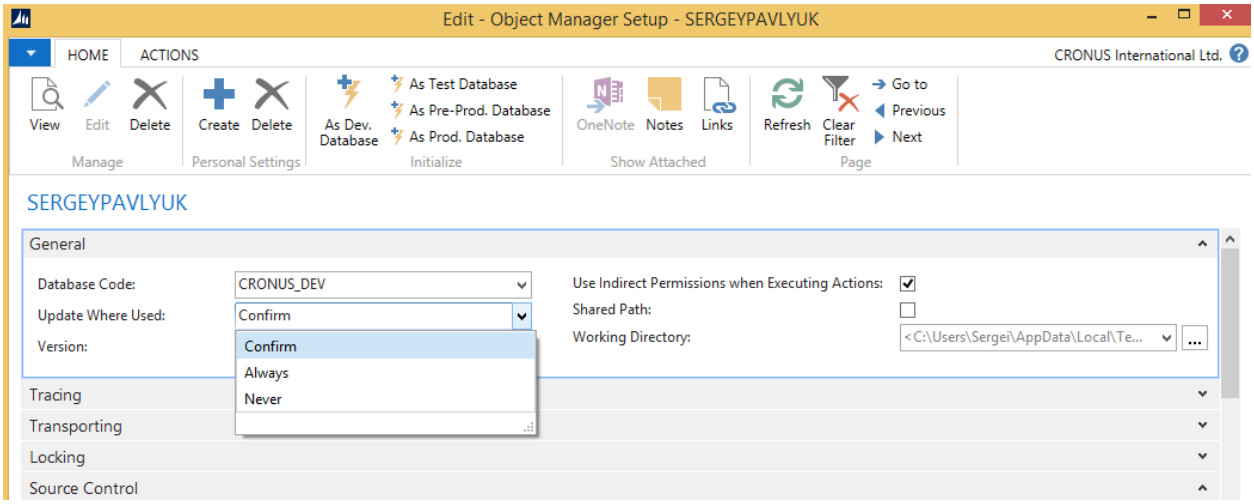
In order to use the Where Used functionality Object Manager needs to update the "Where Used Objects". This is done in three steps:

1. Updating "C/AL History"
2. Updating "Where Used Objects". The C/AL code is analyzed and every item that can be used somewhere is saved
3. Updating "Where Used In". The C/AL code is analyzed and every entry where an item, found in step 2, is used in is saved

Use the "Update Where Used" field on Object Manager Setup window to

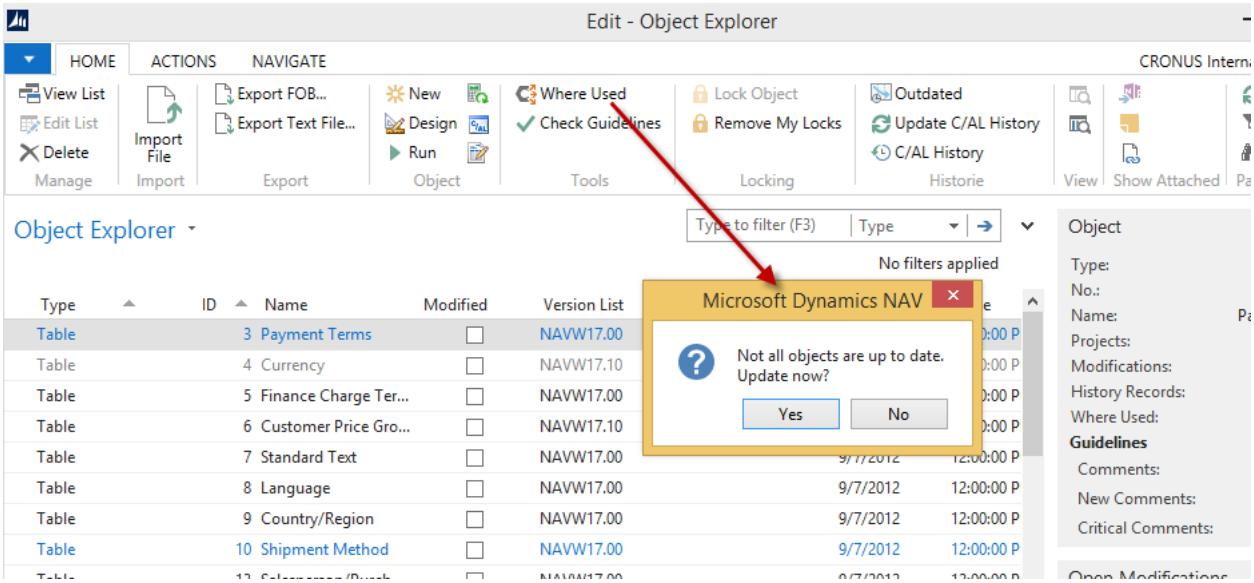
1. Let Object Manager ask you whether or not you want the "Where Used Objects" be updated or
2. Let this process be run automatically each time you open the "Where Used Object Card" or
3. Skip this process

Using the *Confirm*, *Always* or *Never* options respectively.



NOTE: For each object that changes these three steps will be executed again. So your "Where Used Objects" will always be up to date.

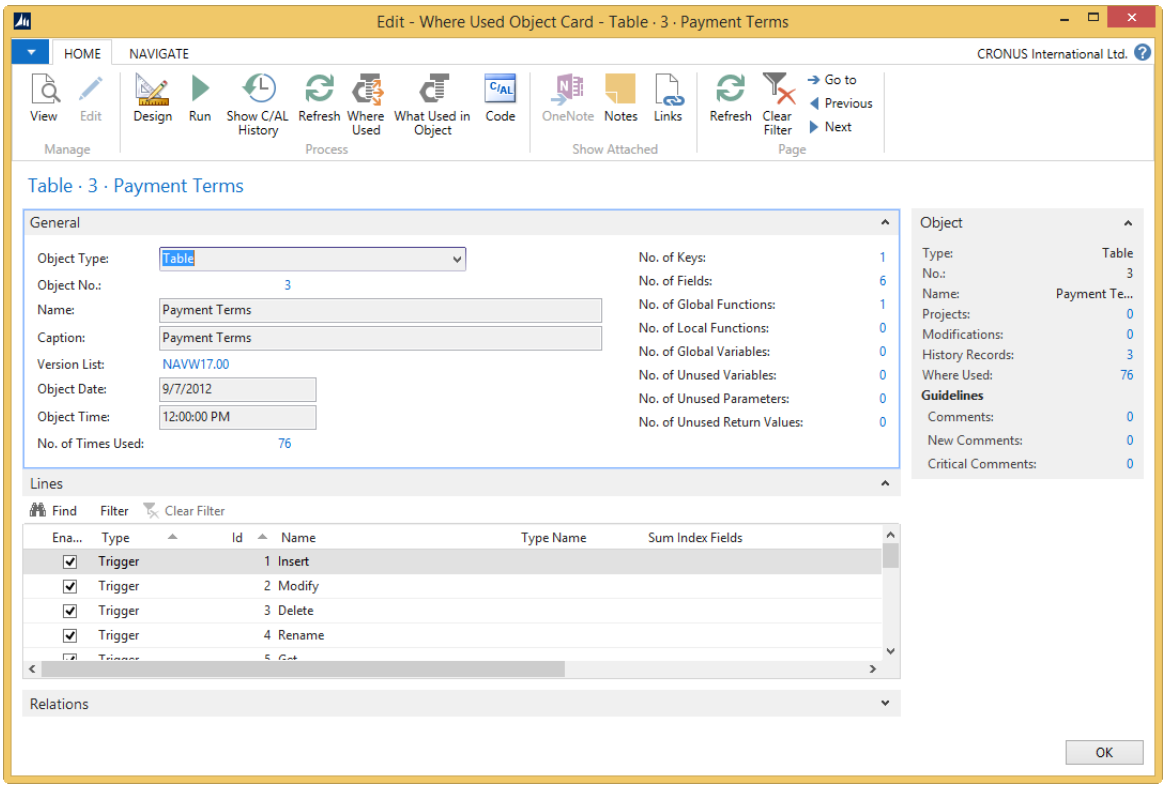
This is a time consuming process. If you cancel the updating process halfway, it will continue, the next time you open the "Where Used Card", from that point on.



If you have not run the setup (see section - Installing) you will have to update the "C/AL History" manually (see section - Update C/AL History) otherwise it will be done automatically.

11.2 Find out Where an Object is used

When everything is updated the "Where Used Object Card" opens.



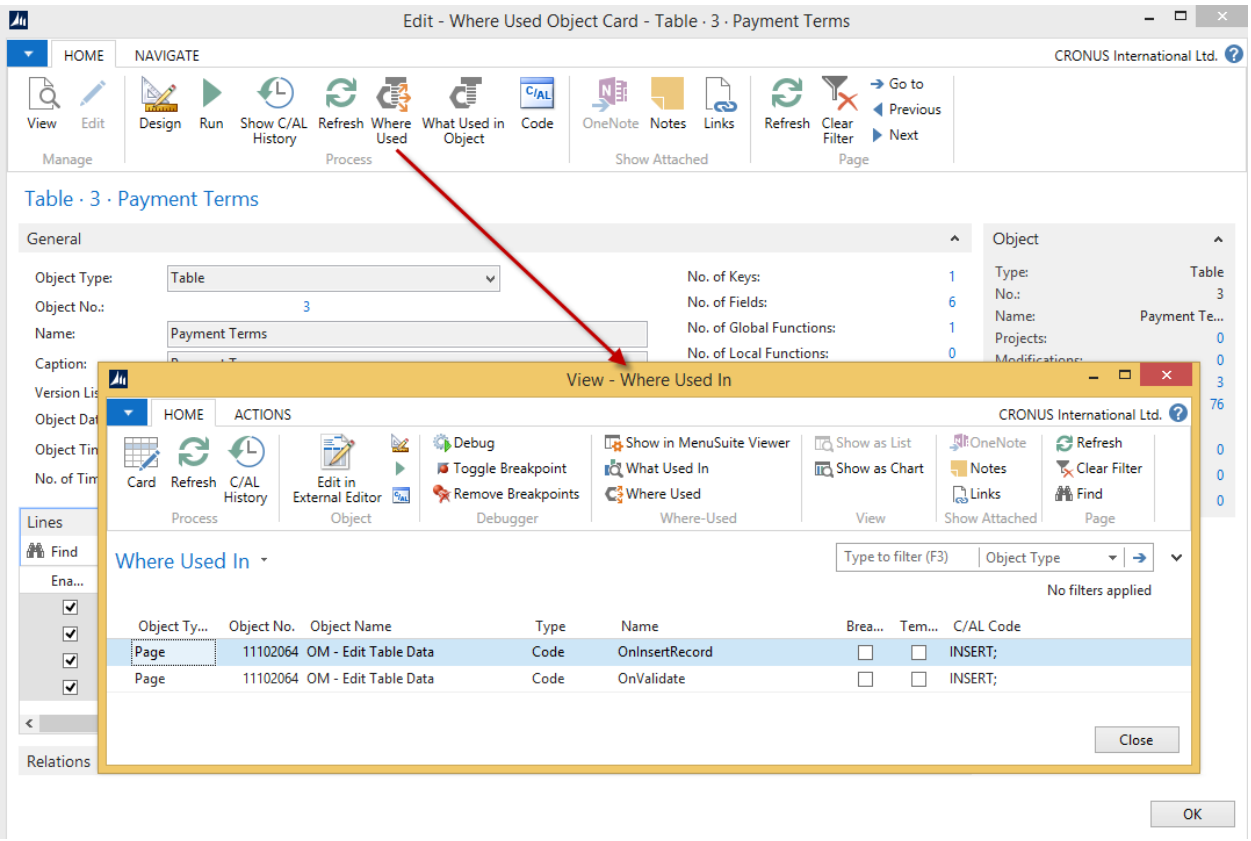
Some lines are colored:

- **Red**: Used but not active
- **Green**: Not used
- **Grey**: Not active
- **Blue**: Global function that is only used local

To find out where an element is used you press the "Where Used" button. For example the Insert trigger of the "Cust. Ledger Entry" table.

The "Where Used In" window opens. Here you see the different lines of code that are calling the Insert trigger.

Once you have selected a line press Code to get a view on the full C/Al code context with the specific line where the trigger is used highlighted.



The "Temporary Record" column in the "Where Used In" window indicates that the C/AL statement uses a temporary record. In our example the call to the Insert trigger is done through the temporary record AppliedCustLedgEntryTemp.

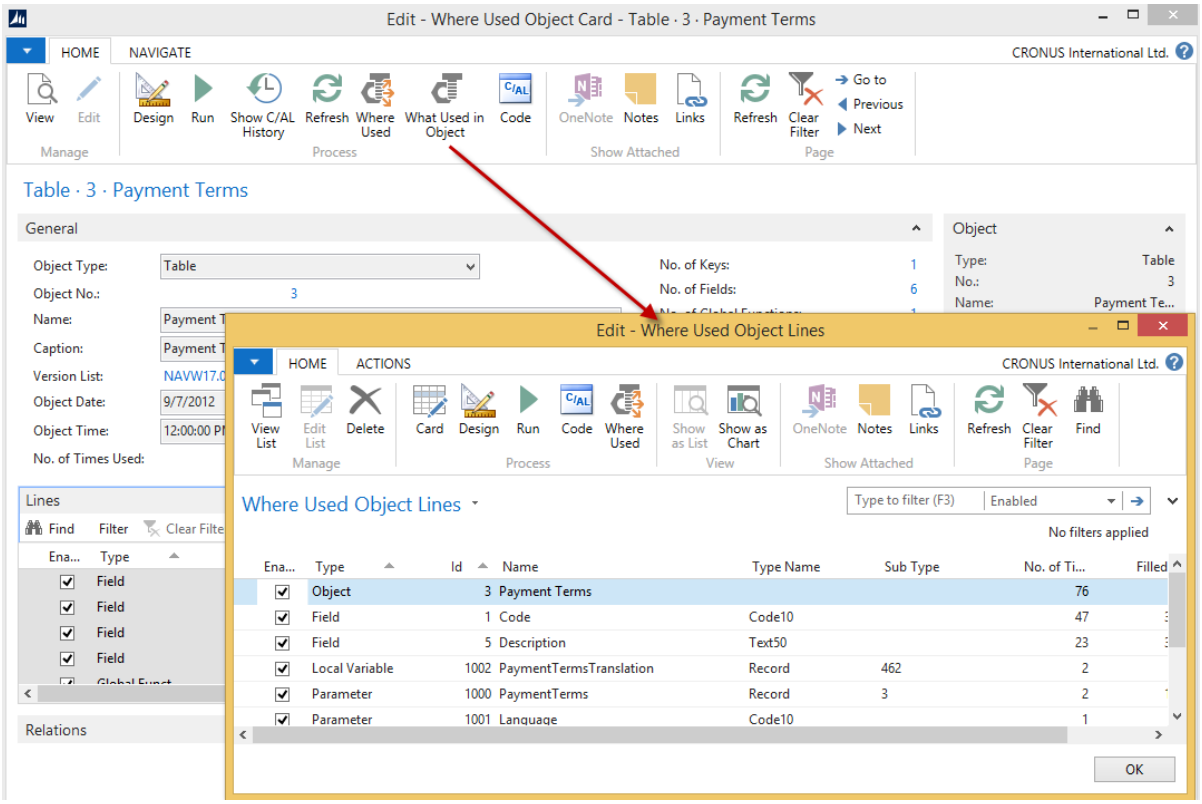
11.3 Relations

On tab relations you can see the relations between tables.

It is also possible to see which objects are used in the object and in which objects the object is used.

11.4 What Used In

If you want to see what is used in a line of code you can use the function "What Used In".



11.5 Delete Where Used Object Lines

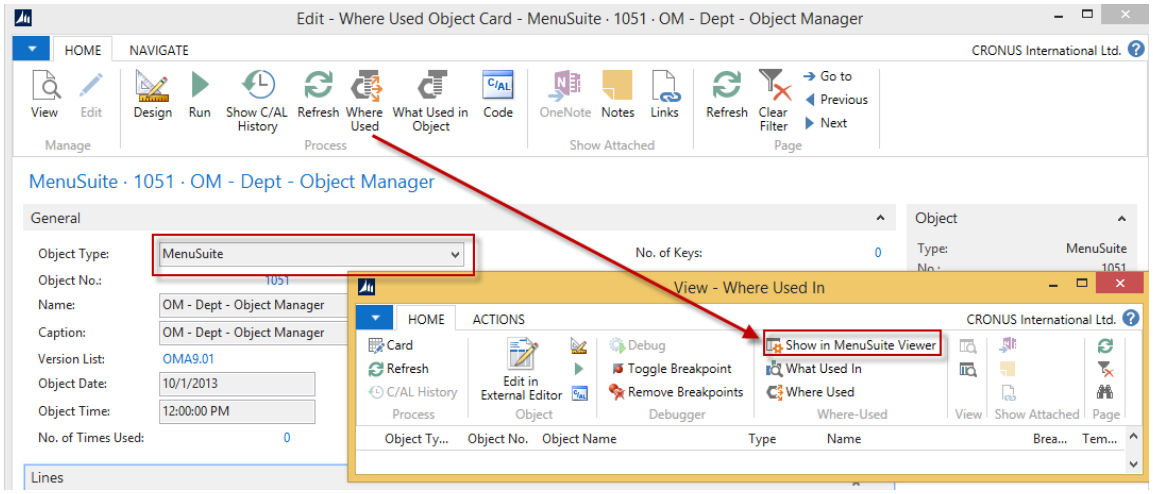
It is possible to delete unused "Where Used Object Lines". It will also delete the entity from the object itself.

For example we have made a field 50.000 in the Customer table but is not used anywhere. We can delete that "Where Used Object Line". The content of the field will be deleted and the field will be deleted from the table.

11.6 MenuSuite Viewer from Where Used

When objects are used in MenuSuites you can see with the function

"Show in MenuSuite Viewer" where it is used.



The Object will be shown as a red line.

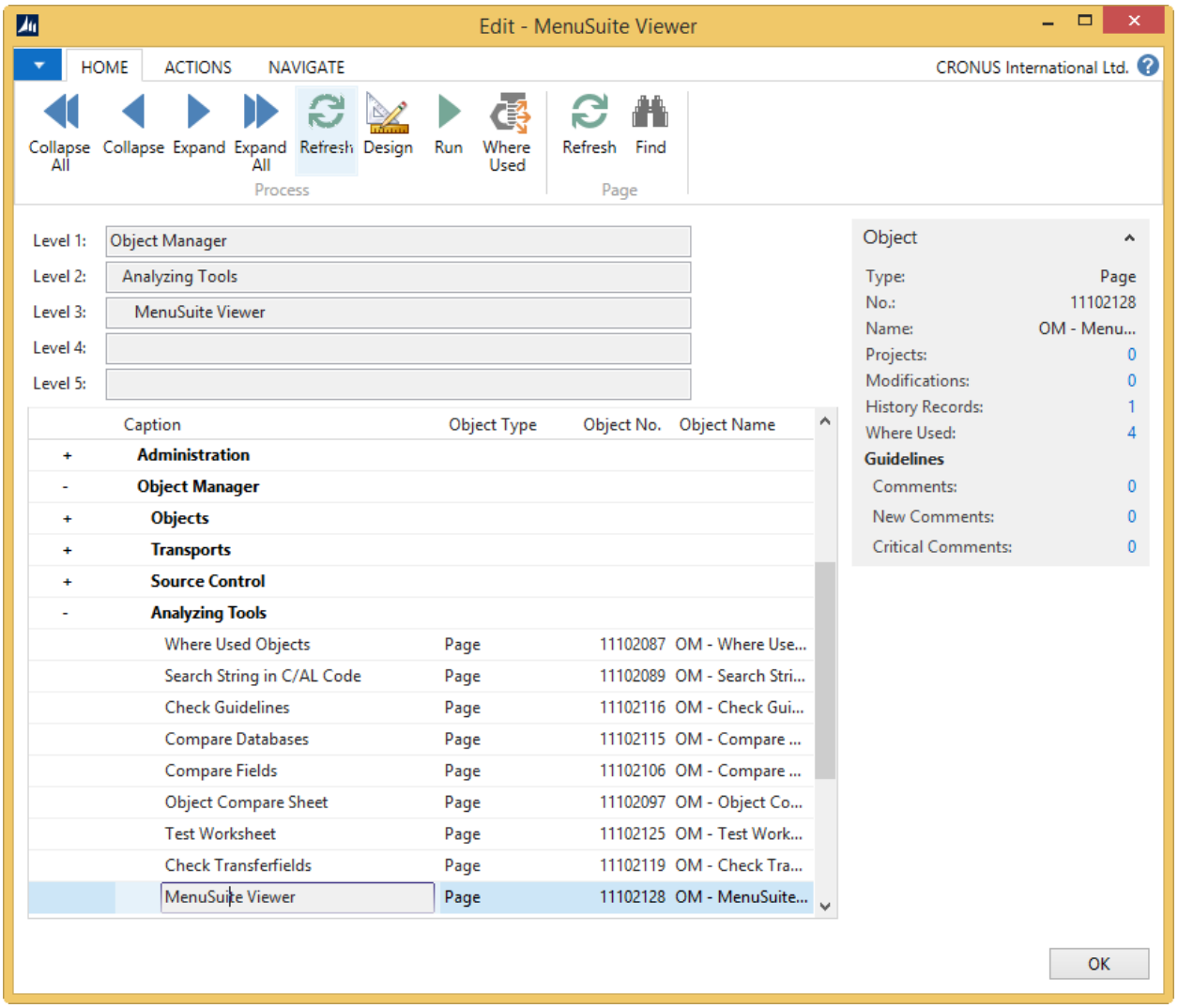
11.7 Enable or Disable Keys and Fields

Using the "Enabled" field on the lines of the "Where Used Object Card", you can enable of disable keys and fields.

Note that this will not work for primary keys, fields containing data, fields being part of an active key, and any other type of lines, like Trigger, Global Function, etc.

12 MENUSUITE VIEWER

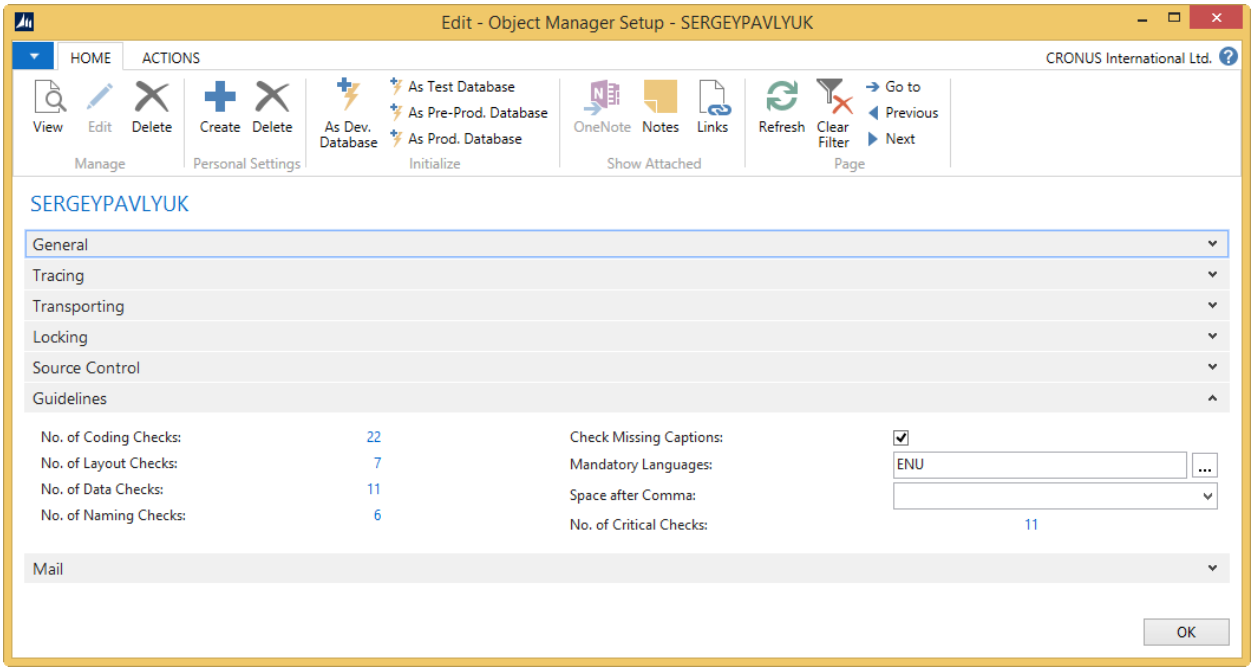
With this tool you can check where an object is located in a MenuSuite of the Classic Client and the Role Tailored Client.



13 CHECK GUIDELINES

This feature will check and/or correct C/AL code of selected objects in NAV. The C/AL code and layout of forms is checked if they meet the Microsoft Guidelines requirements.

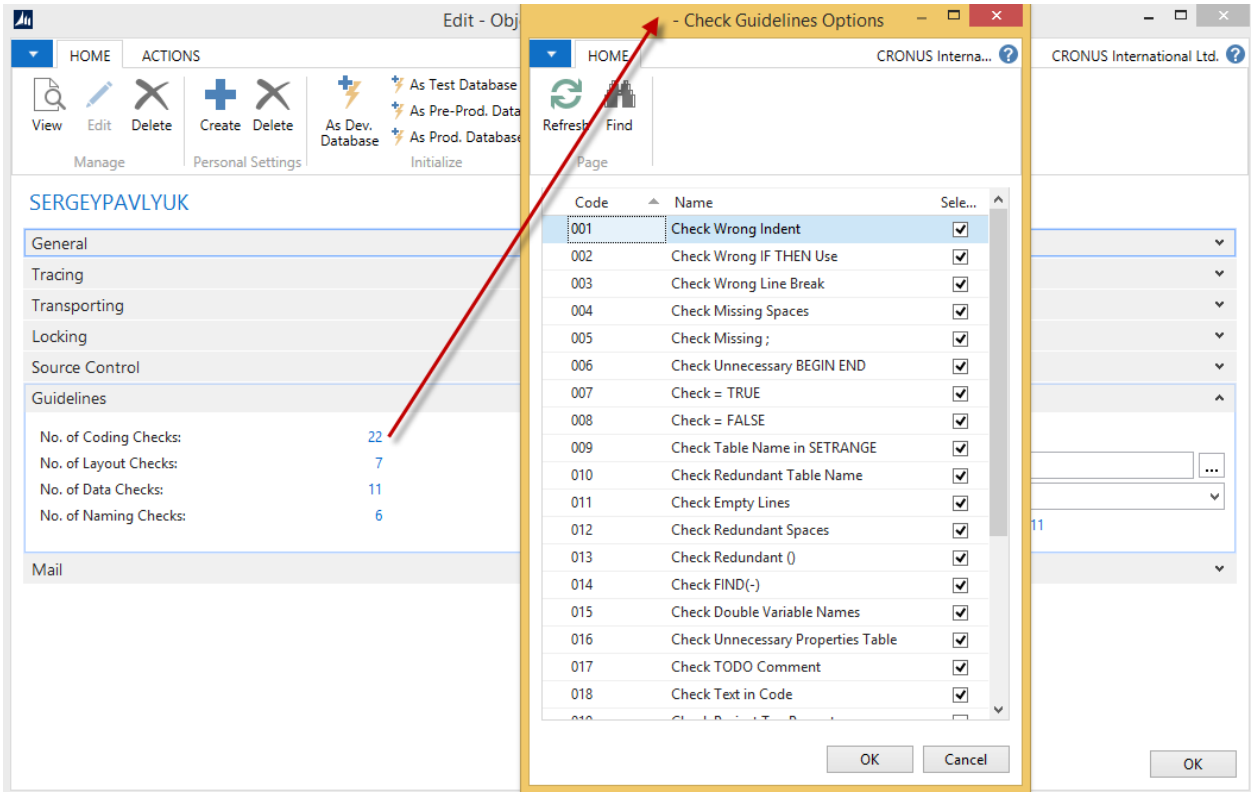
13.1 SETUP



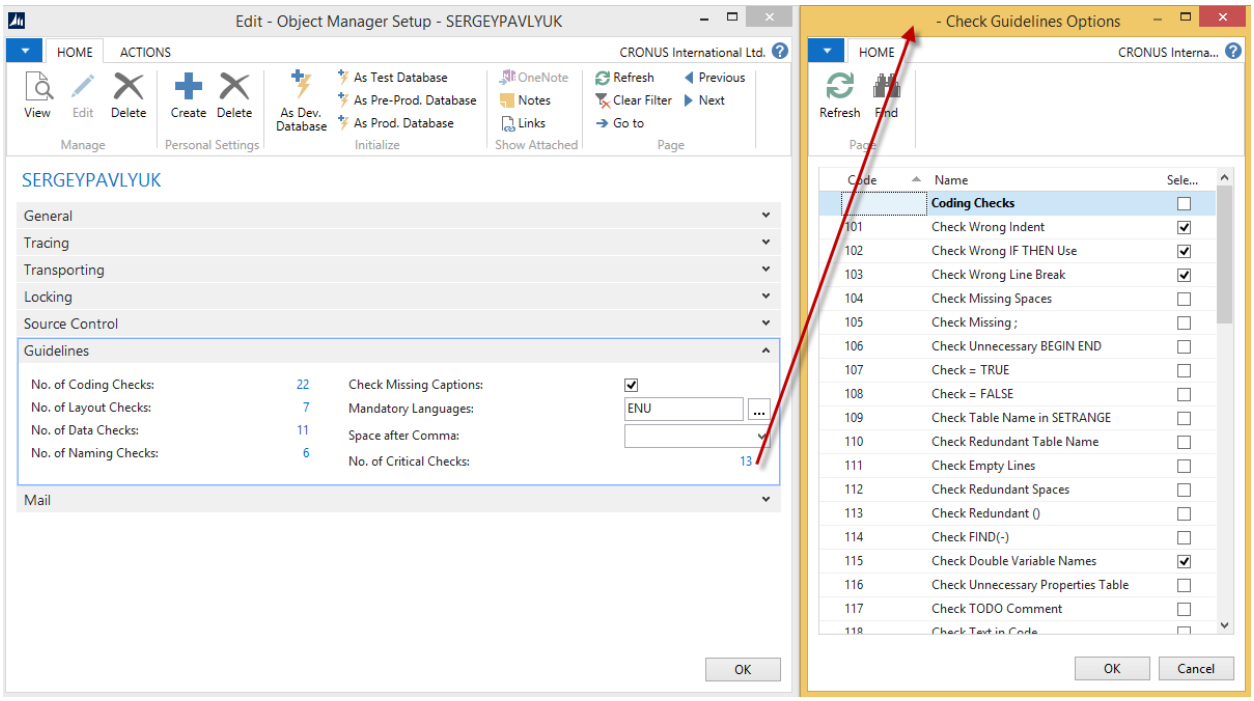
There are 6 types of checks:

- Coding Checks
- Layout Checks
- Data Checks
- Naming Checks
- Missing Captions
- Space after Comma

Zooming in to the Check Guidelines Options window for each check, you will be able to select or deselect any of the guidelines checks.



If you would like to define a check as critical zoom into the "No. of Critical Check" field a select the guideline check. Any object non complying to one of the critical checks will be displayed in red in the "Check Guidelines" window.



13.2 Coding Checks

This list consists of requirements the coding has to meet. Most of these requirements can be autocorrected.

Code	Name	Autocorrect
001	Check Wrong Indent	X
002	Check Wrong 'IF' 'THEN' Use	X
003	Check Wrong Line Break	X
004	Check Missing Spaces	X
005	Check Missing ';'	X
006	Check Unnecessary 'BEGIN' 'END'	X
007	Check '= TRUE'	X
008	Check '= FALSE'	
009	Check Table Name in SETRANGE	X
010	Check Redundant Table Name	X
011	Check Empty Lines	X
012	Check Redundant Spaces	X
013	Check Redundant '()'	X
014	Check 'FIND(`-`)'	
015	Check Double Variable Names	
016	Check Unnecessary Properties Table	
017	Check TODO Comment	
018	Check Text in Code	
019	Check Project Tag Present	
020	Check Wrong SETCURRENTKEY	
021	Check Broken Lines	
022	Check Missing '<>' in CALCDATE	
023	Check MARK	
024	Check FORM.RUN(Integer)	
025	Check Code in OnLookup	
026	Check Comments with Brackets	
027	Check ISSERVICETIER	

13.2.1 Check Wrong Indent

Checks if indentation is correct in code.

Comment: Wrong indent: Remove 1 space(s)

```
IF SalesLine."Document Type" = SalesLine."Document Type"::Order THEN BEGIN
   SalesLine.VALIDATE(Type, SalesLine.Type::Item);
   SalesLine.VALIDATE("No.", Item."No.");
END;
```

Should be:

```
IF SalesLine."Document Type" = SalesLine."Document Type"::Order THEN BEGIN
  SalesLine.VALIDATE(Type, SalesLine.Type::Item);
  SalesLine.VALIDATE("No.", Item."No.");
END;
```

13.2.2 Check Wrong 'IF' 'THEN' Use

Check on use of IF THEN (BEGIN).

Comment: 'IF' must start at new line

```
IF "Document Type" = "Document Type"::Order THEN
  SalesLine.SETFILTER("Quantity Shipped",'<>0')
ELSE IF "Document Type" = "Document Type"::Invoice THEN BEGIN
    SalesLine.SETRANGE("Sell-to Customer No.",xRec."Sell-to Customer No.");
    SalesLine.SETFILTER("Shipment No.",'<>%1','');
  END;
```

Should be:

```
IF "Document Type" = "Document Type"::Order THEN
  SalesLine.SETFILTER("Quantity Shipped",'<>0')
ELSE
  IF "Document Type" = "Document Type"::Invoice THEN BEGIN
    SalesLine.SETRANGE("Sell-to Customer No.",xRec."Sell-to Customer No.");
    SalesLine.SETFILTER("Shipment No.",'<>%1','');
  END;
```

Comment: Keep 'THEN' and 'BEGIN' together

```
IF xRec."Ship-to Code" <> '' THEN
  BEGIN
  GetCust("Sell-to Customer No.");
  "Tax Area Code" := Cust."Tax Area Code";
END;
```

Should be:

```
IF xRec."Ship-to Code" <> '' THEN BEGIN
  GetCust("Sell-to Customer No.");
  "Tax Area Code" := Cust."Tax Area Code";
END;
```

Comment: 'REPEAT' must start at new line

```
IF ServLedgerEntry.FINDSET THEN REPEAT
  DiscountAmount := DiscountAmount + -ServLedgerEntry."Discount Amount";
  ContractDiscAmount := ContractDiscAmount + -ServLedgerEntry."Contract Disc. Amount";
UNTIL ServLedgerEntry.NEXT = 0;
```

Should be:

```
IF ServLedgerEntry.FINDSET THEN
  REPEAT
    DiscountAmount := DiscountAmount + -ServLedgerEntry."Discount Amount";
    ContractDiscAmount := ContractDiscAmount + -ServLedgerEntry."Contract Disc. Amount";
  UNTIL ServLedgerEntry.NEXT = 0;
```

Comment: 'THEN' must start at new line

```
IF ("Account Type" = "Account Type"::"IC Partner") AND
  ("Bal. Account Type" = "Bal. Account Type"::"G/L Account") THEN BEGIN
  GLAcc.SETRANGE("No.","Bal. Account No.");
```

```
  IF GLAcc.FIND('-') THEN
    "IC Partner G/L Acc. No." := GLAcc."Default IC Partner G/L Acc. No";
END;
```

Should be:

```
IF ("Account Type" = "Account Type"::"IC Partner") AND
   ("Bal. Account Type" = "Bal. Account Type"::"G/L Account")
THEN BEGIN
  GLAcc.SETRANGE("No.","Bal. Account No.");
  IF GLAcc.FIND('-') THEN
    "IC Partner G/L Acc. No." := GLAcc."Default IC Partner G/L Acc. No";
END;
```

Comment: Start new line after ELSE

```
IF MapPoint.FIND('-') THEN
  MapMgt.MakeSelection(DATABASE::Contact,GETPOSITION)
ELSE MESSAGE(Text033);
```

Should be:

```
IF MapPoint.FIND('-') THEN
  MapMgt.MakeSelection(DATABASE::Contact,GETPOSITION)
ELSE
  MESSAGE(Text033);
```

13.2.3 Check Wrong Line Break

If a line is broken in two lines you should break it at the first possible break position.

Comment: Break the line at position 14

```
IF AskQuestion THEN BEGIN
  Question := STRSUBSTNO(
      Text031 +
      Text032,ChangedFieldName);
```

Should be:

```
IF AskQuestion THEN BEGIN
  Question :=
    STRSUBSTNO(
      Text031 +
      Text032,ChangedFieldName);
```

If you use a BEGIN in a CASE construction the BEGIN statement must always start at a new line.

Comment: 'BEGIN' must start at new line

```
CASE "Table ID" OF
  DATABASE::"G/L Account": BEGIN
    IF GLAcc.GET("No.") THEN BEGIN
      GLAcc."Global Dimension 1 Code" := NewDimValue;
      GLAcc.MODIFY(TRUE);
    END;
  END;
END;
```

Should be:

```
CASE "Table ID" OF
  DATABASE::"G/L Account":
```

```
    BEGIN
      IF GLAcc.GET("No.") THEN BEGIN
        GLAcc."Global Dimension 1 Code" := NewDimValue;
        GLAcc.MODIFY(TRUE);
      END;
    END;
END;
```

13.2.4 Check Missing Spaces

Checks if statements are glued together where this is not allowed.

Comment: Add space at position 27

```
IF "Applies-to Doc. No." <>'' THEN
  CustLedgEntry.SETRANGE("Document No.","Applies-to Doc. No.");
```

Should be:

```
IF "Applies-to Doc. No." <> '' THEN
  CustLedgEntry.SETRANGE("Document No.","Applies-to Doc. No.");
```

13.2.5 Check Missing ';'

Checks for open ends before an END;

Comment: Add a ';'

```
WITH PaymentTermsTranslation DO BEGIN
  SETRANGE("Payment Term",Code);
  DELETEALL
END;
```

Should be:

```
WITH PaymentTermsTranslation DO BEGIN
  SETRANGE("Payment Term",Code);
  DELETEALL;
END;
```

13.2.6 Check Unnecessary 'BEGIN' 'END'

Checks where a 'BEGIN' and 'END' is used where this is not necessary.

Comments: Remove 'BEGIN' and Remove 'END'

```
IF "Price Includes VAT" THEN BEGIN
  IF NOT VATPostingSetup.GET("VAT Bus. Posting Gr. (Price)","VAT Prod. Posting Group") THE
    FIELDERROR("VAT Bus. Posting Gr. (Price)");
END;
```

Should be:

```
IF "Price Includes VAT" THEN
  IF NOT VATPostingSetup.GET("VAT Bus. Posting Gr. (Price)","VAT Prod. Posting Group") THE
    FIELDERROR("VAT Bus. Posting Gr. (Price)");
```

13.2.7 Check '= TRUE'

Checks where '= TRUE' is added in an equation. This is not necessary and should be removed.

Comment: Remove '= TRUE'

```
IF Complete = TRUE THEN
  MESSAGE(Text003,ReportCaption1,ReportCaption2)
```

Should be:

```
IF Complete THEN
  MESSAGE(Text003,ReportCaption1,ReportCaption2)
```

13.2.8 Check '= FALSE'

Comment: Use NOT instead of '= FALSE'

```
IF "Related to Base Unit of Meas." = FALSE THEN
  "Qty. per Unit of Measure" := 1;
```

Should be:

```
IF NOT "Related to Base Unit of Meas." THEN
  "Qty. per Unit of Measure" := 1;
```

13.2.9 Check Table Name in SETRANGE

Checks if the variablename of the record is stated inside of a setrange function. All other statements (like SETFILTER, SETCURRENTKEY, etc.) are also checked.

Comment: Remove tablename

```
SalesPrice.SETRANGE("Sales Type",SalesPrice."Sales Type"::Campaign);
SalesPrice.SETRANGE(SalesPrice."Sales Code","No.");
SalesPrice.LOCKTABLE;
```

Should be:

```
SalesPrice.SETRANGE("Sales Type",SalesPrice."Sales Type"::Campaign);
SalesPrice.SETRANGE("Sales Code","No.");
SalesPrice.LOCKTABLE;
```

13.2.10 Check Redundant Table Name

Checks for unnecessary recordreferences (Tablenames). This happens the most in reports where the name of the DataItem is not necessary in the code.

Comment: Remove tablename

```
VAT Entry - OnPreDataItem()
"VAT Entry".COPYFILTERS(VATEntry);
```

Should be:

```
VAT Entry - OnPreDataItem()
COPYFILTERS(VATEntry);
```

If you use the WITH statement in your code then the tablename is not necessary in the following code:

Comment: Remove tablename

```
WITH Period DO BEGIN
  Period.SETRANGE("Period Type","Period Type"::Date);
  SETFILTER("Period Start",DateFilter);
  IF FIND('-') THEN
    EXIT("Period Start")
END;
```

Should be:

```
WITH Period DO BEGIN
  SETRANGE("Period Type","Period Type"::Date);
  SETFILTER("Period Start",DateFilter);
```

```
  IF FIND('-') THEN
    EXIT("Period Start")
END;
```

13.2.11 Check Empty Lines

Checks if there are two or more empty lines. And if a function starts with an empty line.

Comment: Remove empty line

```
IF Campaign.GET(GETFILTER("Campaign No.")) THEN
  "Campaign Description" := Campaign.Description;


IF SegHeader.GET(GETFILTER("Segment No.")) THEN
  "Segment Description" := SegHeader.Description;
```

Should be:

```
IF Campaign.GET(GETFILTER("Campaign No.")) THEN
  "Campaign Description" := Campaign.Description;

IF SegHeader.GET(GETFILTER("Segment No.")) THEN
  "Segment Description" := SegHeader.Description;
```

13.2.12 Check Redundant Spaces

Checks unnecessary spaces.

Comment: Remove space at position 44

```
IF CustLedgEntry."Amount to Apply" = 0 THEN  BEGIN
  CustLedgEntry.CALCFIELDS("Remaining Amount");
  CustLedgEntry."Amount to Apply" := CustLedgEntry."Remaining Amount";
END;
```

Should be:

```
IF CustLedgEntry."Amount to Apply" = 0 THEN BEGIN
  CustLedgEntry.CALCFIELDS("Remaining Amount");
  CustLedgEntry."Amount to Apply" := CustLedgEntry."Remaining Amount";
END;
```

13.2.13 Check Redundant '()'

Checks for unnecessary brackets.

Comment: Remove '()'

```
ReservEntry2 := ReservEntry;
ReservEntry2.ClearItemTrackingFields;
ReservEntry2.MODIFY();
```

Should be:

```
ReservEntry2 := ReservEntry;
ReservEntry2.ClearItemTrackingFields;
ReservEntry2.MODIFY;
```

Comment: Remove '(' at position 4, Remove ')' at position 26

```
IF (STRLEN(Parameter) = 2) THEN
  BizTalkNASStartup.RUN;
```

Should be:

```
IF STRLEN(Parameter) = 2 THEN
  BizTalkNASStartup.RUN;
```

13.2.14 Check 'FIND(`-`)'

Checks for FIND(`-`) instructions which should be replaced by FINDSET, FINDFIRST or ISEMPTY. Also spots FIND(`+’) which should be changed to FINDLAST. The FIND(`-`) instructions may cause performance issues on SQL based Navision.

Comment: Replace FIND('-') with 'FINDFIRST', 'FINDSET' or 'ISEMPTY'

```
Job.SETRANGE("Bill-to Customer No.","No.");
IF Job.FIND('-') THEN
  ERROR(Text015,TABLECAPTION,"No.",Job.TABLECAPTION);
```

Should be:

```
Job.SETRANGE("Bill-to Customer No.","No.");
IF Job.FINDFIRST THEN
  ERROR(Text015,TABLECAPTION,"No.",Job.TABLECAPTION);
```

Comment: Replace FIND('+') with 'FINDLAST'

```
CustLedgEntry.SETRANGE(Open,TRUE);
IF CustLedgEntry.FIND('+') THEN
  ERROR(Text012,FIELDCAPTION("IC Partner Code"), CustLedgEntry."IC Partner Code");
```

Should be:

```
CustLedgEntry.SETRANGE(Open,TRUE);
IF CustLedgEntry.FINDLAST THEN
  ERROR(Text012,FIELDCAPTION("IC Partner Code"), CustLedgEntry."IC Partner Code");
```

13.2.15 Check Double Variable Names

Checks for variable names which are used local as well as global variable.

Comment: Variablename 'Bin' already in use as global

13.2.16 Check Unnecessary Properties Table

Checks for properties on fields and tables which are not needed because they are the same as the default.

Comment: Remove property: 'Yes' is default for ‘TestTableRelation’

13.2.17 Check TODO Comment

Checks for “// TODO:” in C/AL code. You can use this as a reminder.

Comment: // TODO found in code

13.2.18 Check Text in Code

Checks if there are hardcoded text messages instead of using textconstants.

Comment: Replace text by text constant

```
CASE globalVariable OF
  'CurrentRow': value := CurrentRow;
  'CurrentCol': value := CurrentCol;
  'RangeStartXlRow': value := RangeStartXlRow;
  'RangeStartXlCol': value := RangeStartXlCol;
  'RangeEndXlRow': value := RangeEndXlRow;
  'RangeEndXlCol': value := RangeEndXlCol;
  'XlWrkSht': value := XlWrkSht;
  ELSE
    ERROR('Global variable %1 is not included for test.',globalVariable);
END;
```

Should be:

```
CASE globalVariable OF
  'CurrentRow': value := CurrentRow;
  'CurrentCol': value := CurrentCol;
  'RangeStartXlRow': value := RangeStartXlRow;
  'RangeStartXlCol': value := RangeStartXlCol;
  'RangeEndXlRow': value := RangeEndXlRow;
  'RangeEndXlCol': value := RangeEndXlCol;
  'XlWrkSht': value := XlWrkSht;
  ELSE
    ERROR(Text001,globalVariable);
```

13.2.19 Check Project Tag Present

Checks if the no. of the project is found in the documentation trigger. This check will only be done if the check is done in a project or a transport.

Comment: Project tag 'P0001' not present

13.2.20 Check Wrong SETCURRENTKEY

Checks if a not existing key is used in the code.

Comment: Not Existing Key

```
WITH ProdOrderLine do begin
  RESET;
  SETCURRENTKEY(Status,"Completely Invoiced");
  SETRANGE(Status,Status::Finished);
  SETRANGE("Completely Invoiced",FALSE);
  EXIT(FIND('-'));
END;
```

13.2.21 Check Broken Lines

Checks if statement is split over multiple lines where this is not necessary.

Comment: Line does not have to be broken

```
IF WhseJnlLine.FINDFIRST THEN
  ERROR(
    Text007,
    FIELDCAPTION("Adjustment Bin Code"));
```

Should be:

```
IF WhseJnlLine.FINDFIRST THEN
  ERROR(Text007,FIELDCAPTION("Adjustment Bin Code"));
```

13.2.22 Check Missing '<>' in CALCDATE

Checks if the DateExpression in a CALCDATE call is entered with < > delimiters surrounding it.

Comment: Add '<>' to CALCDATE

```
AllowedPostingDate := CALCDATE('+1D',AllowedPostingDate);
```

Should be:

```
AllowedPostingDate := CALCDATE('<+1D>',AllowedPostingDate);
```

13.2.23 Check MARK

Identifies any call to MARK and suggests to use a temporary record instead.

Comment: Use temporary record instead of marking

13.2.24 Check FORM.RUN(Integer)

Checks if an integer is used in commands like FORM.RUN(…), REPORT.RUN(...), etc.

Comment: Replace integer with OBJECT::<object name>"

```
FORM.RUN(1);
```

Should be:

```
FORM.RUN(FORM::"Company Information");
```

13.3 Layout Checks

Code	Name
201	Check Missing Image
202	Check Double Access Keys
203	Check Double Shortcut Keys
204	Check Lowercase Shortcut Keys
205	Check Unnecessary Properties
206	Check Usage Not Existing Keys
207	Check Empty Action

13.4 Data Checks

Code	Name
001	Check Key Fields are Integer, Code, Option or Date
002	Check NotBlank on Key Fields
003	Check Testfield on Key Fields
004	Check Flowfields Not Editable
005	Check No. of Options in Field
006	Check Primary Key in Table Relation
007	Check Field Types in Relations
008	Check Missing Relations
009	Check Assignments
010	Check Delete Relating Table
011	Check Constants in Relations
012	Check Transferfields

13.4.1 Check Key Fields are Integer, Code, Option or Date

Checks if keyfields are of type:

- Integer
- Code
- Option
- Date

Comment: Try to avoid 'Decimal' in primary key.

13.4.2 Check NotBlank on Key Fields

Checks if property of keyfield is set to NotBlank. This check is only done on tables with one primary keyfield of type code.

Comment: Add NotBlank property

13.4.3 Check Testfield on Key Fields

Checks if TESTFIELD function is set on key fields in the OnInsert trigger of the table. This check is only done on tables with one primary key field of type code.

Comment: Add TESTFIELD(Code);

13.4.4 Check FlowFields Not Editable

Checks if the Editable property of flowfields is set to No.

Comment: Make not editable

13.4.5 Check No. of Options in Field

Checks if an option in a OptionCaptionML is missing.

Comment: No. of options not the same

13.4.6 Check 'SETRANGE' on Form

Checks if function SETRANGE("Primary Key"); is in the OnAfterGetRecord trigger. This check is only done on forms with a TabControl and a sourcetable.

Comment: Add SETRANGE("Primary Key");

13.4.7 Check Primary Key in Table Relation

Checks if in a table relation the primary key field is used. If it is a relation to a table with a single primary key this is not needed and should be removed.

Comment: Remove Primary Key Field 'Code' from TableRelation

13.4.8 Check Field Types in Relations

Checks fields if related fields in table relation and FlowField CalcFormulas have same data type.

Comment: The related field in table 'G/L Account' is 'Text30'

The “G/L Account Name” field on “G/L Entry” table has data type Text50, while the Name field on the “G/L Account” has data type Text30.

13.5 NAMING CHECKS

This feature checks the correct naming of:

- Temporary records
- Variables (Local and Global)
- Object Names
- Fieldnames
- Functions

Code	Name
001	Check 'tmp' in Name of Temporary Records
002	Check Variable Names
003	Check Object Names
004	Check Field Names
005	Check Reserved Names
006	Check Function Names
007	Check Double Captions

13.5.1 Check 'tmp' in Name of Temporary Records

Checks if 'tmp' is in the name of a temporary record. A warning will be given if a record variable has 'tmp' in its name and is not a temporary record. Other way around when a temporary record has no 'tmp' in its name also a warning appears. The following name parts will be seen as temporary:

- Tmp
- Temp
- Buf
- Buffer

Comment: Add 'tmp', 'temp' or 'buffer' to variablename

Comment: Remove 'Temp' from variablename

(Example: TempCustLedgerEntry is not a temporary record so Temp must be removed from the name)

13.5.2 Check Variable Names

Variable names must start with capital letter. The first and second letter of the name can be lowercase because of the common use of prefixes.

Comment: Start a variablename with a capital

- dimensionValue: Not allowed
- pCustomer: Allowed

Also spaces in the variable name will be noticed.

Comment: Remove space from variablename

13.5.3 Check Object Names

Object names have the same check as variable names except for spaces. Double spaces however are noticed.

13.5.4 Check Field Names

Same checks as Object Names.

13.5.5 Check Reserved Names

Checks if variable-, field- and function names are conflicting with existing functions and commands.

Comment: 'CopyFilters' is a reserved command.

13.5.6 Check Function Names

Checks if a function name contains spaces.

Comment: Remove space from function name

```
PROCEDURE "Export XML"@4();
```

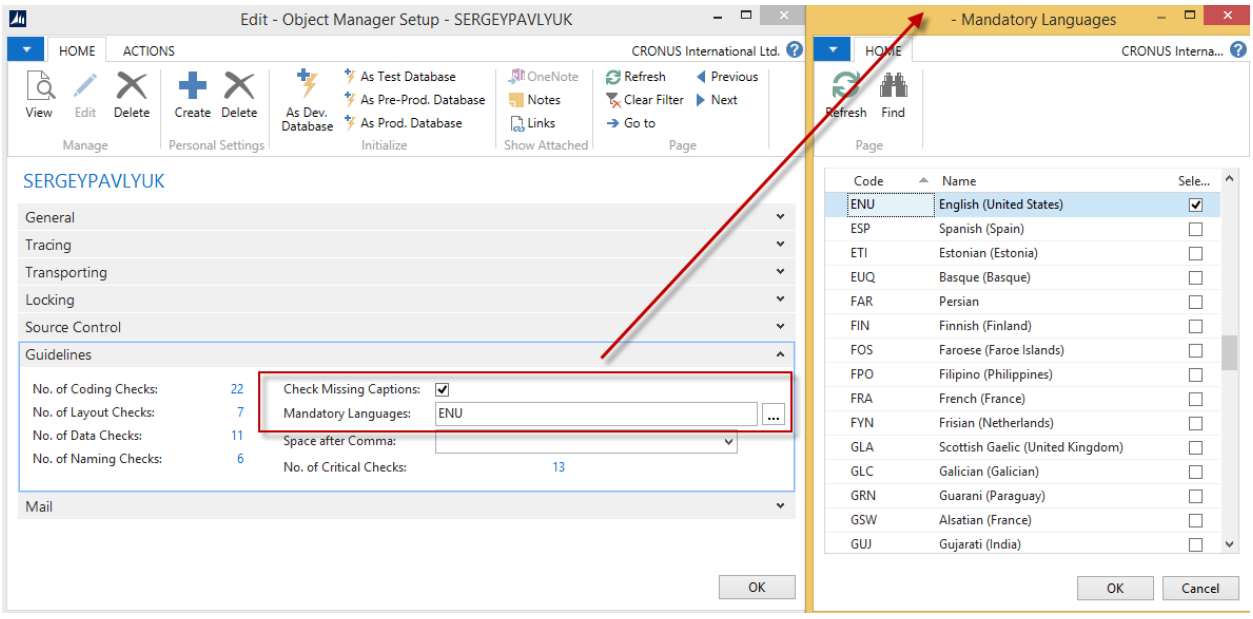
Should be:

```
PROCEDURE ExportXML@4();
```

13.6 Caption Checks

Checks all objects for missing captions and redundant spaces in captions.

To enable this feature check mark "Check Missing Captions" on the "Object Manager Setup" window. Caption checks will only be executed in the range of languages the user has defined in the setup.



Checking the caption typically can show the following comments:

Add caption to object
The object checked does not have any captions defined for.

Add caption
The field or control checked does not have any captions defined for.

Add <language> caption
The field or control checked does not have a caption defined for the designated language.

Remove space from caption
The caption contains a redundant space.

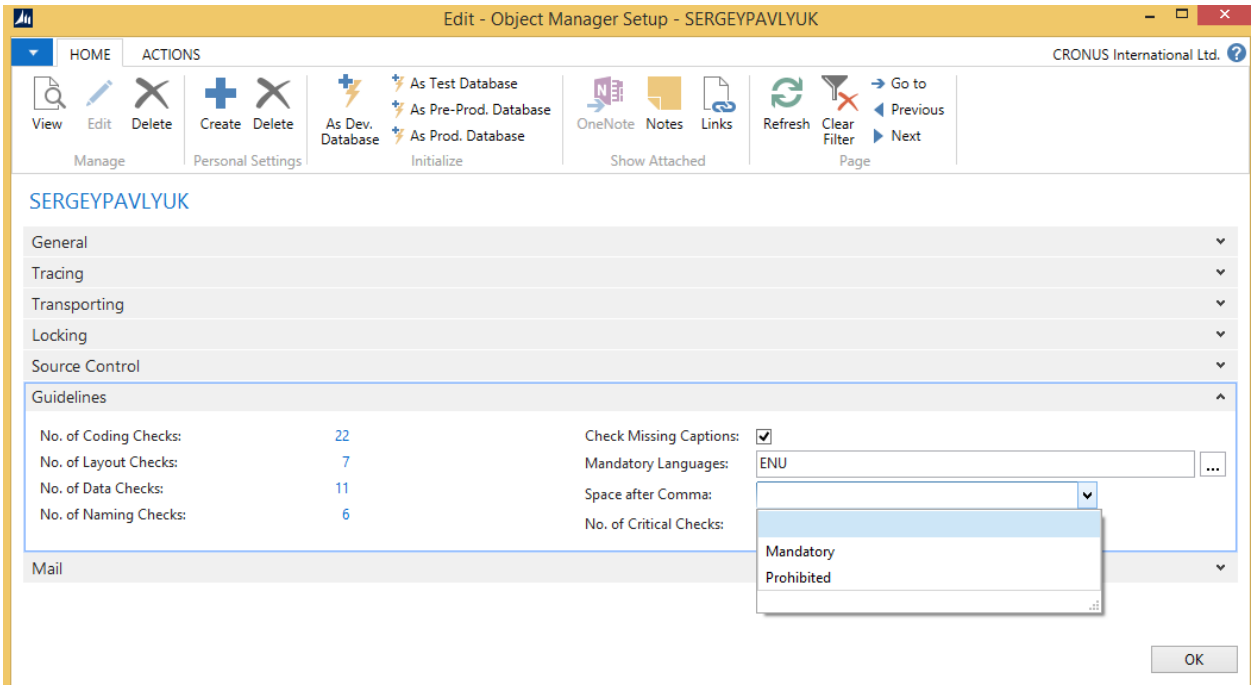
Use the "Update Captions" window to view the individual check results and update them.

If a translation for a specific language has been defined as a translation rule you can insert a missing caption for that language using the "Apply Guidelines to Selection" feature on the "Check Guidelines Code" window.

If a lot of captions are missing you can add those with the translation tool.

13.7 Space after Comma

The guidelines of NAV says that you cannot put a space after a comma. Many developers like to place there a comma anyway. In the setup you can choose which method the Object Manager has to check. If you choose the option Mandatory your code have to look like this:



```
IF NOT PrinterSelection.GET(USERID, ReportID) THEN
  IF NOT PrinterSelection.GET('', ReportID) THEN
    IF NOT PrinterSelection.GET(USERID, 0) THEN
      IF PrinterSelection.GET('', 0) THEN;
```

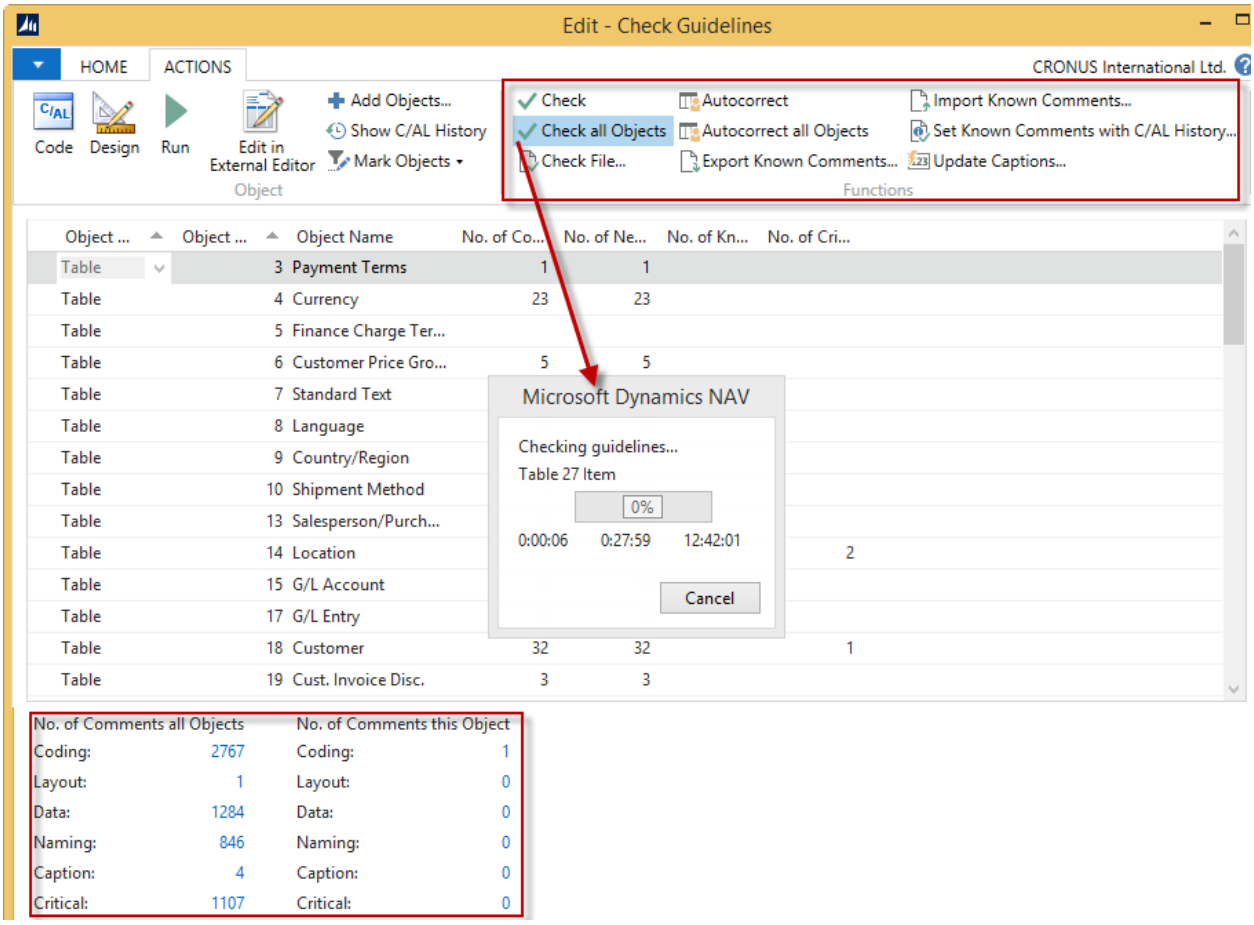
If you choose the Prohibited setting your code has to look like this:

```
IF NOT PrinterSelection.GET(USERID,ReportID) THEN
  IF NOT PrinterSelection.GET('',ReportID) THEN
    IF NOT PrinterSelection.GET(USERID,0) THEN
      IF PrinterSelection.GET('',0) THEN;
```

It is also possible to leave this option blank. The Object Manager will not check for spaces after comma.

13.8 CHECK GUIDELINES

The Check Guidelines tool can be found in menu Analyzing Tools - Check Guidelines.



Add the objects you want to check. If you have a range of objects you want to add you also can use the function “Add Objects”.

Enter your filters.

The objects will be added in the worksheet. To check the object(s) start function Check Selection or Check all Objects. When finished all criteria you selected in the setup is checked.

The No. of Comments per Object can also be seen in the lines.

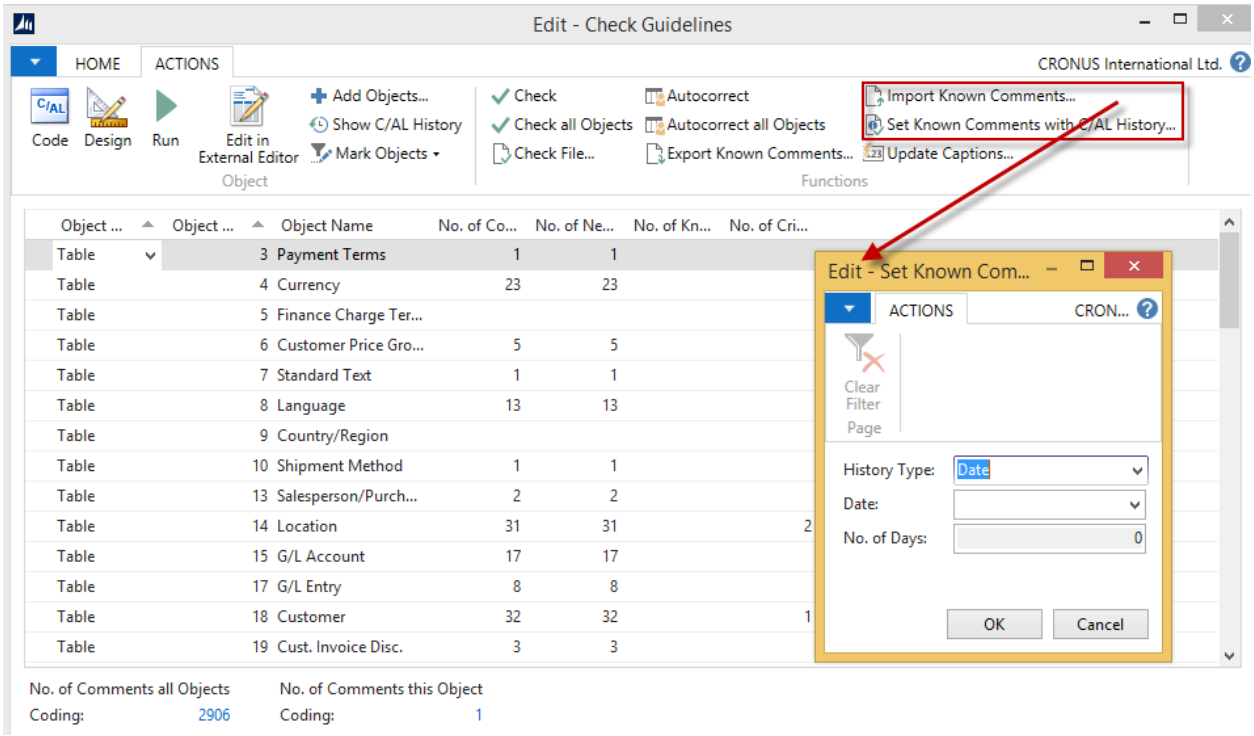
By using the Code button in this form on the selected object you can go to the C/AL where the Check Guidelines tool found the comments.

Here you can see the comments on a piece of code of codeunit 1 – Application Management.

If you want to loop through all comments you can use the function button to go to previous and next comment.

13.9 KNOWN COMMENTS

You can set a comment as Known Comment if you don't want to solve the comment. This is often done when the comments apply to standard NAV objects. The benefit of using the know comment option is that you can easily see a new comment because this comment is not set to 'known'.



13.9.1 Set Known Comments with C/AL History

With this function you can set all comments to known that were already present on a certain moment. This can be useful when you work on a particular project for a week and you want to know which comments were newly created in this week.

Set the date and press OK.

All comments that were already in the objects at January 1st will be set the known. The comments that will remain as new are the comments that are newly created since January 1st.

13.9.2 Import and Export Known Comments

With this function you can im- and export all your known comments from or to other databases. So you don't have to set them again in a new database.

13.10 Auto Apply Guidelines

You have the ability to let the Object Manager correct some of the check coding guidelines comments. Beware that automatic changing by the tool isn't possible for all guideline checks.

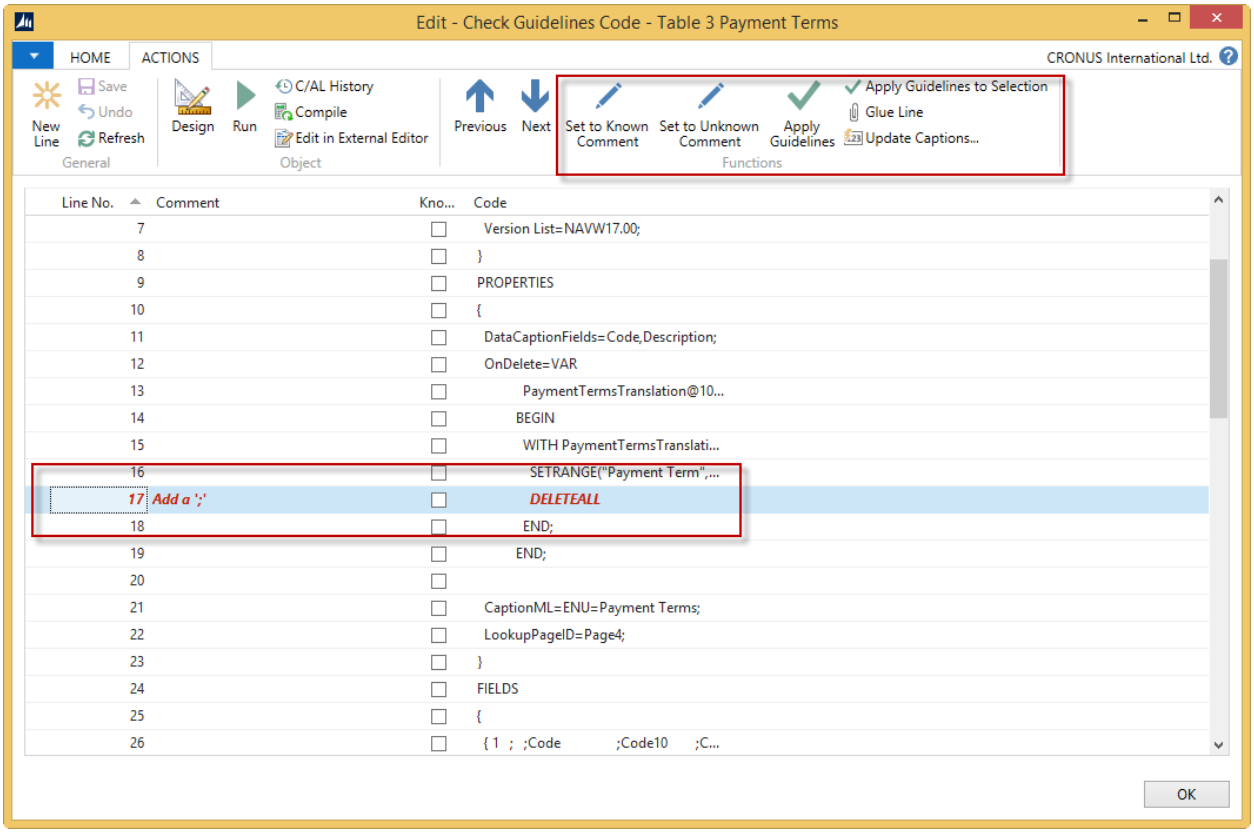
NOTE: Only auto apply guidelines in a development environment and test the changes before transporting your objects to the production environment.

It is possible to apply the guidelines to a selection of code. All comments of the objects. Or a selection of objects.

13.11 Manual Apply Guidelines

There are also coding guidelines checks that cannot be auto applied. E.g. the "Check 'FIND(`-`)'" check can be replaced with more possibilities so this can only be done manually.

The fastest way to do this is directly in the code form.



Here it is possible to edit a line of code and press the save button. The objects will be saved and the guidelines will be rechecked and the comment will be deleted.

Layout checks can be best solved in the Object Designer. To open the object in design mode you can press Ctrl+D.

The Object Designer will open the selected object in design modus.

Also Data and Naming checks have to be done directly in the object itself or by modifying the results in the Check Guidelines Code form.

13.12 PERFORM GUIDELINES ON TEXT FILE

The function Check File gives the possibility to check objects in text format.

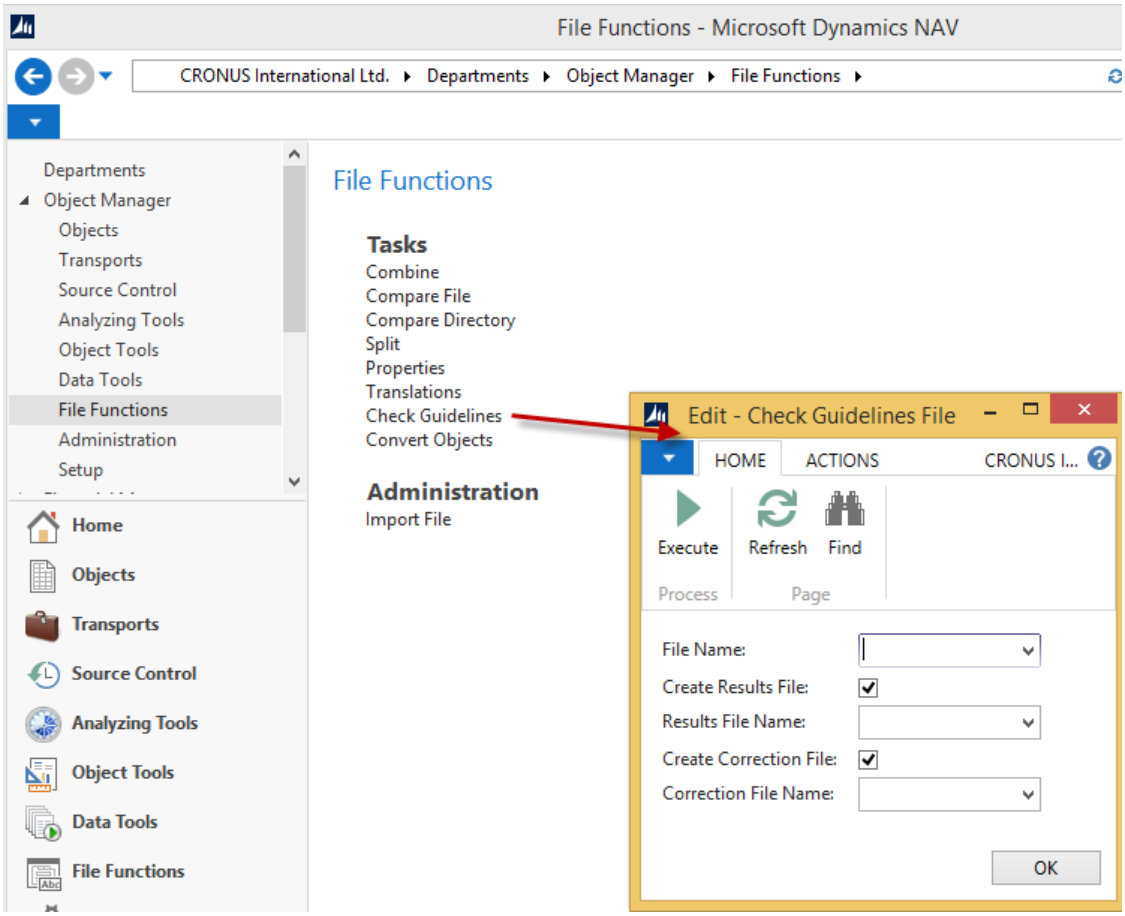
Select the file you want to check.

Create Results File

If you set Create Results File the comments will be published to a text file.

Create Correction File

If you set Create Correction File the Guidelines Tool will automatically correct (there where autocorrect is possible) the text file.



Press OK

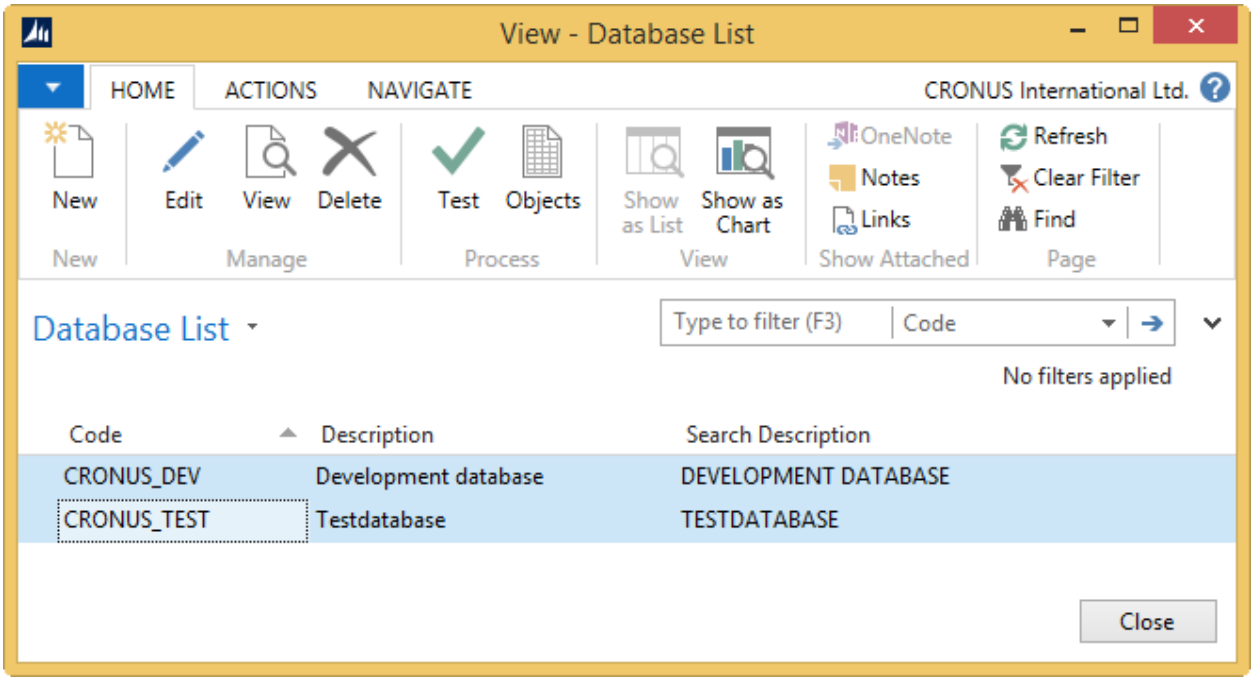
The results and corrected file are created.

14 COMPARE DATABASES

With the Compare Database functionality the user is able to compare objects of two separate databases. You are able to compare current database with another or compare 2 databases other than the current you are in.

14.1 SETUP

Open **Setup > Databases**:



In the Database Card you set the properties for the databases to you want to connect to compare.

14.2 COMPARING DATABASES

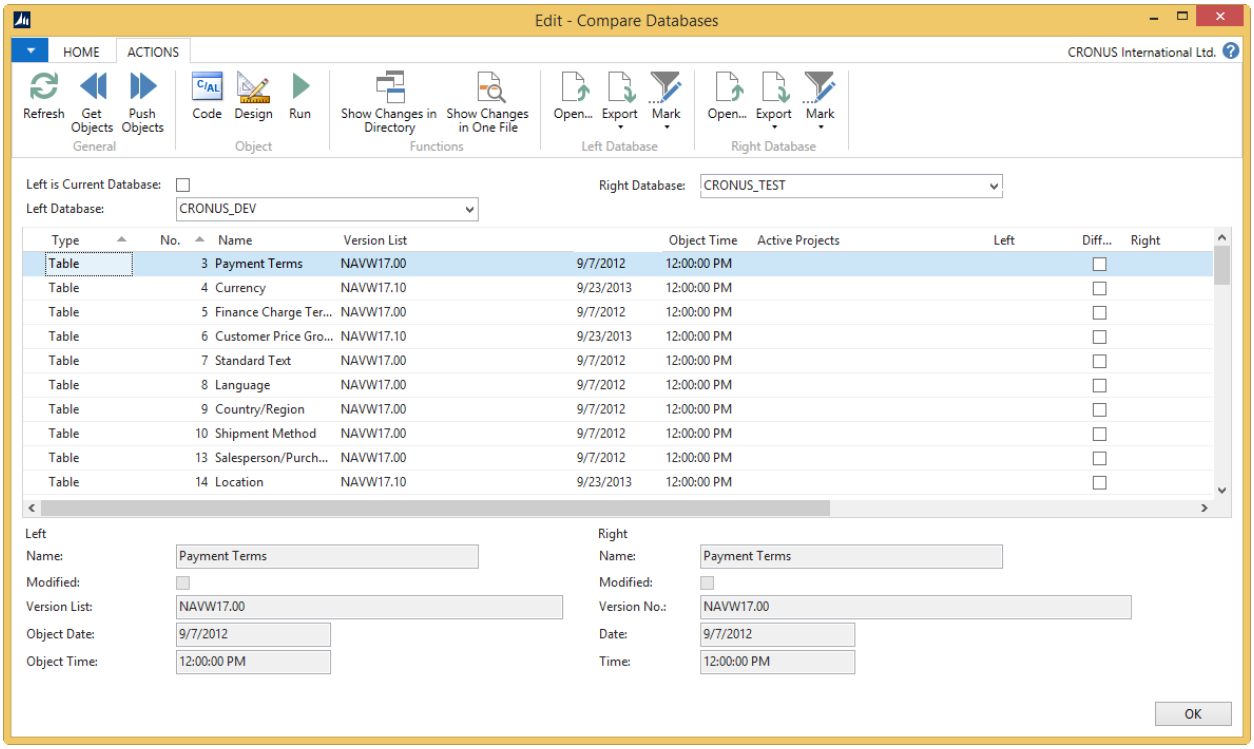
Open **Analyzing Tools > Compare Databases**.

If you want to compare your current database set "Left is Current Database". If you want to compare another database you select a database in the field "Left Database".

Select a database in the **"Right Database**" you want to compare, and click **Refresh**.

You can set a filter on "Different" to show only the objects which are not equal or do not exist in one of the databases.

You can analyze the differences with your compare tool with the menu options "Show Changes in Directory" or "Show Changes in One File"

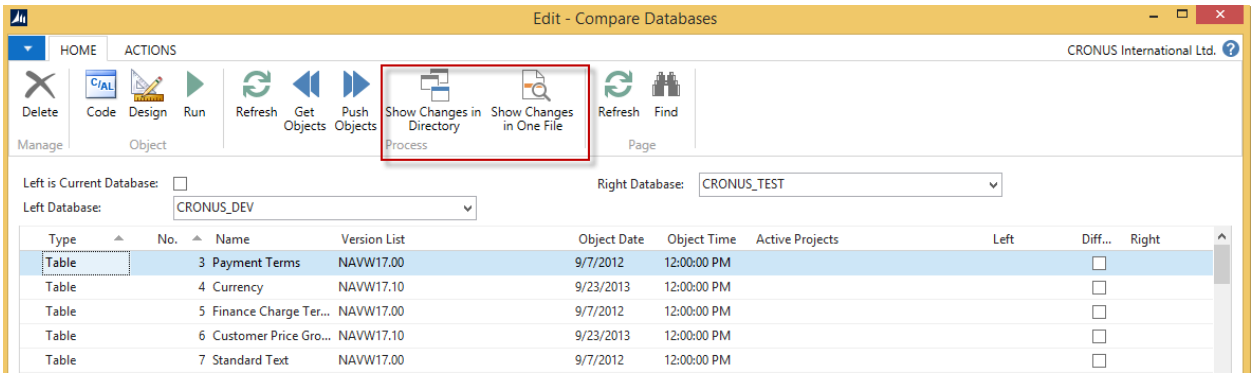


For any of the objects in the left database the "Active Projects" column lists all active projects the object is part of.

Note: If lines are red the C/AL History has to be updated in the database.

14.3 Comparing Files

By opening fob, fib, object .txt and obp files as left and right you can compare the content with each other.



Show Changes in Directory or On File

15 CHECK TRANSFERFIELDS

NAV uses the TRANSFERFIELDS command to copy data from one table to another. When there are conflicting field types or field length in this tables you will get an ERROR. For example:

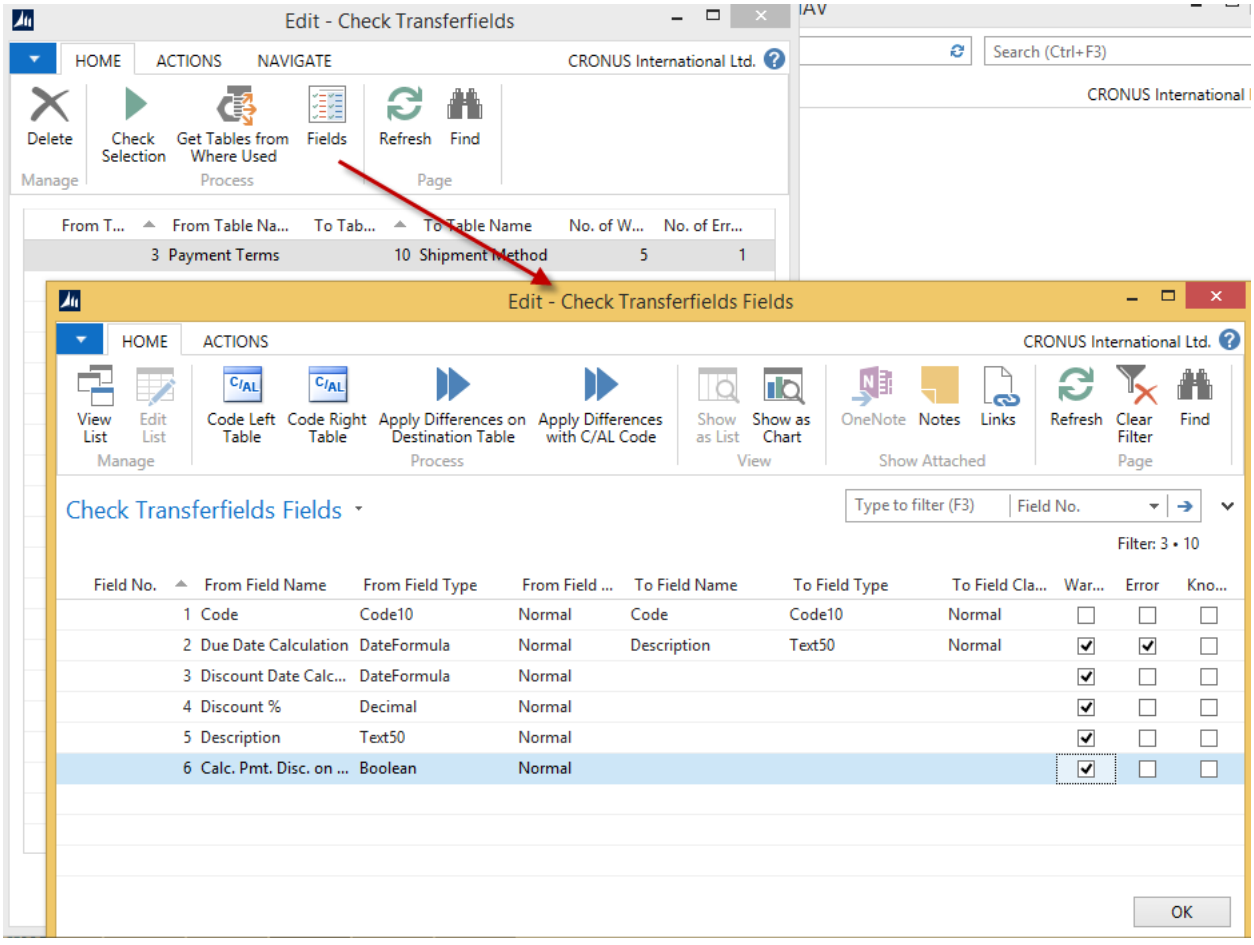
"*The content of the Customer Name 3 field cannot be copied because the size of the field is too small*"

The conflicting field name leads to related WARNING.

To check if you have any of these conflicting fieldtypes in your database you can use the Check Transferfields form.

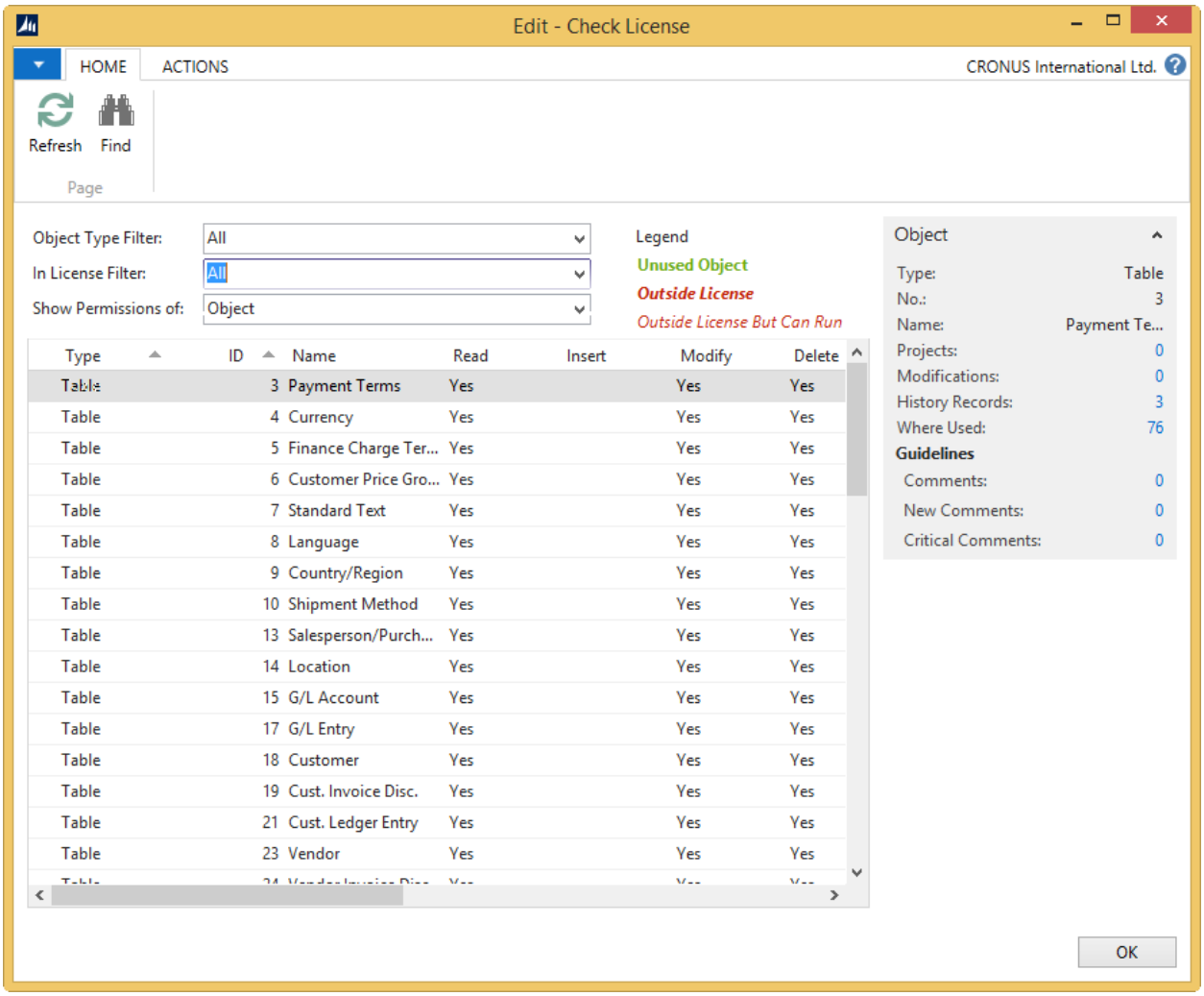
15.1 INITIALIZE TRANSFERFIELDS TABLES

There are two methods to populate the tables to check. The first one is to use the option "Initialize Tables". This option fills in the tables that uses the transferfields command in a default Cronus database. If you have customizations in your database it is better to use the option "Get Tables from Where Used". This option search your where used base for the transferfields command.



16 CHECK LICENSE

With the Check License form you can see which objects are unused in your license (in other words: available object numbers) and which objects are outside your license.

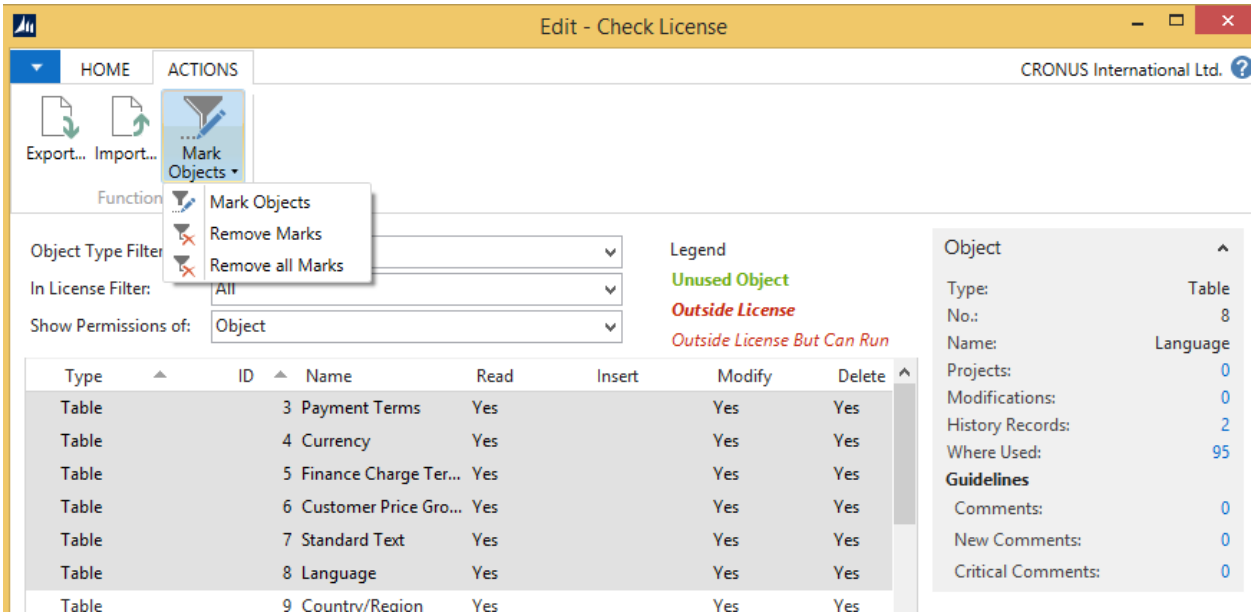


16.1 IMPORT AND EXPORT LICENSE FILES

Pressing Functions – Export you can export license information of the active license to a .lic file. At any time you can import a .lic file into the "Check License" window to verify the license against the object set in OMA.

16.2 MARK OBJECTS

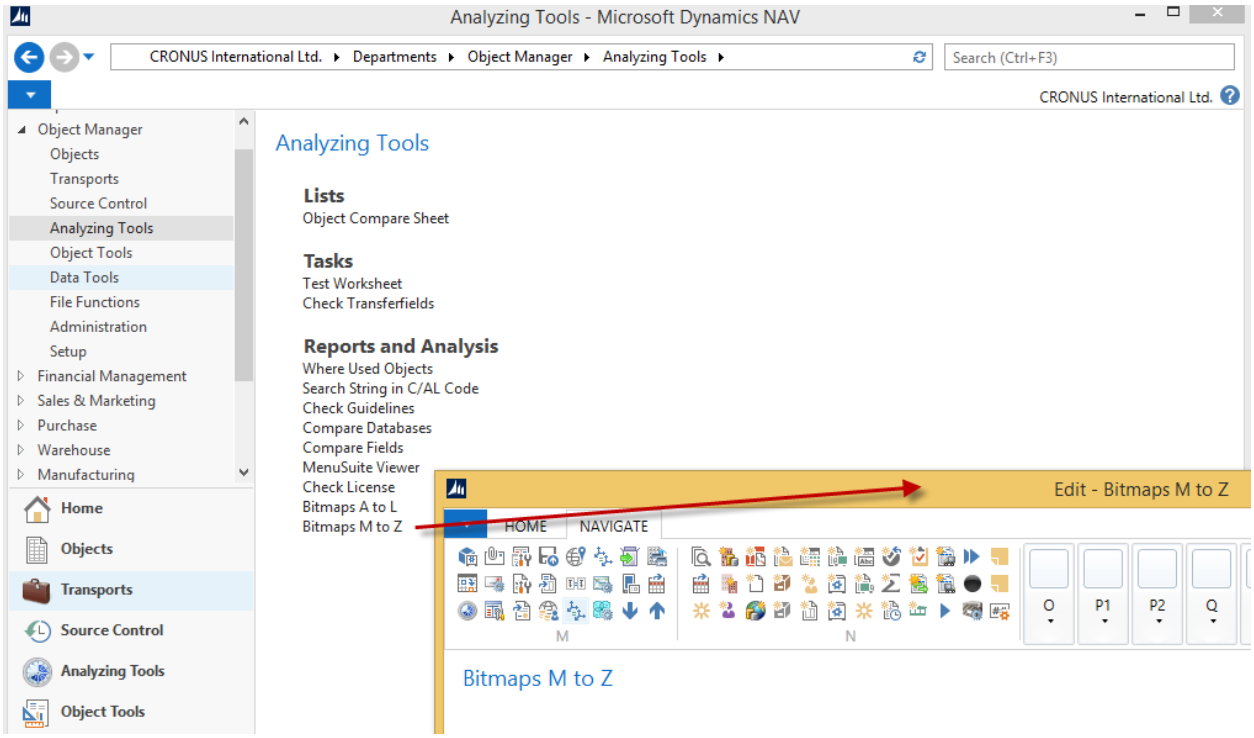
To mark objects in the "**Check License**" window press **Functions > Mark > Mark Objects.**

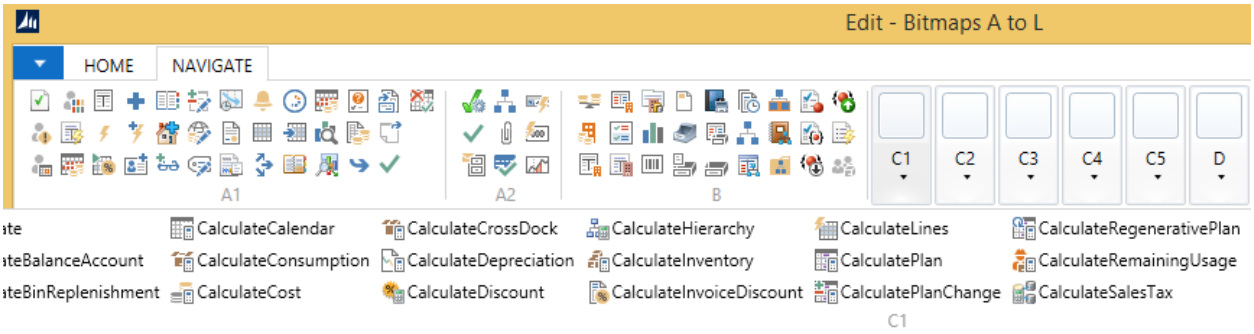


Objects will be marked by means of a # sign added as a prefix to the Version List value.

17 BITMAPS

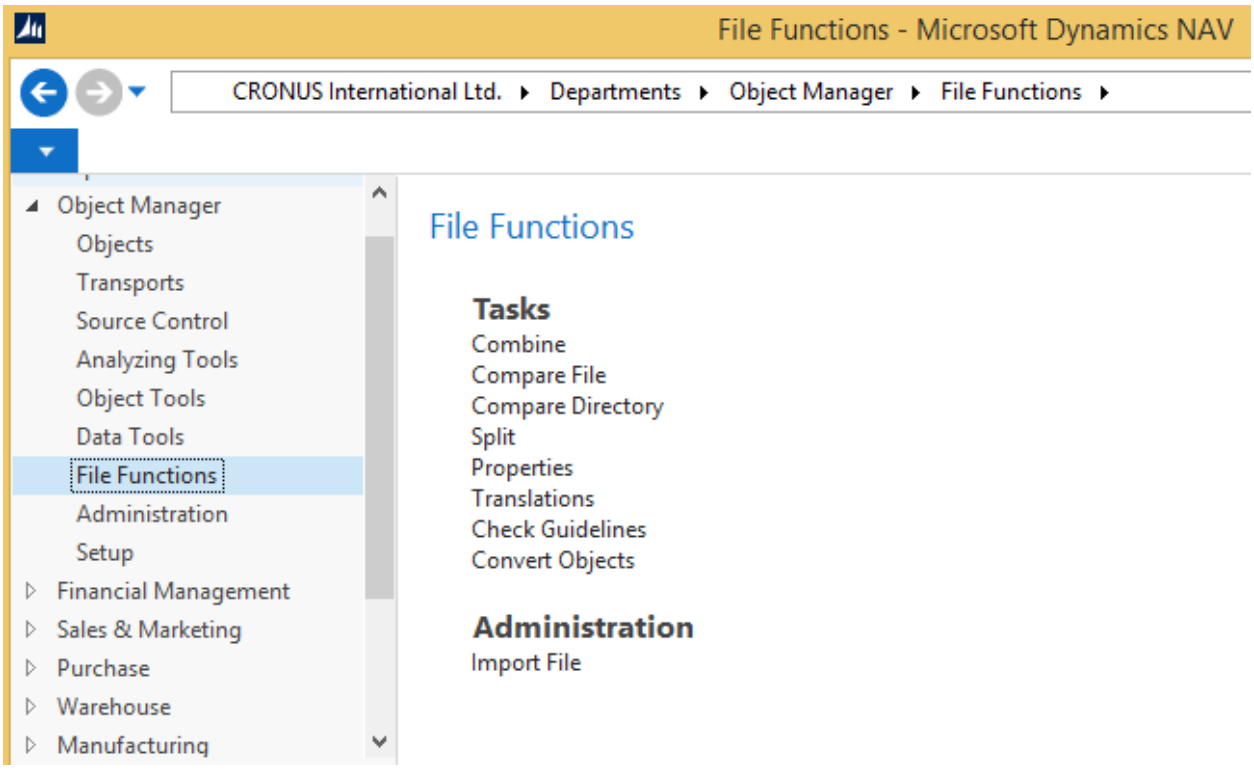
With the Bitmaps form you can see which bitmaps are available in NAV.





18 FILE FUNCTIONS

Different file functions that can be executed:

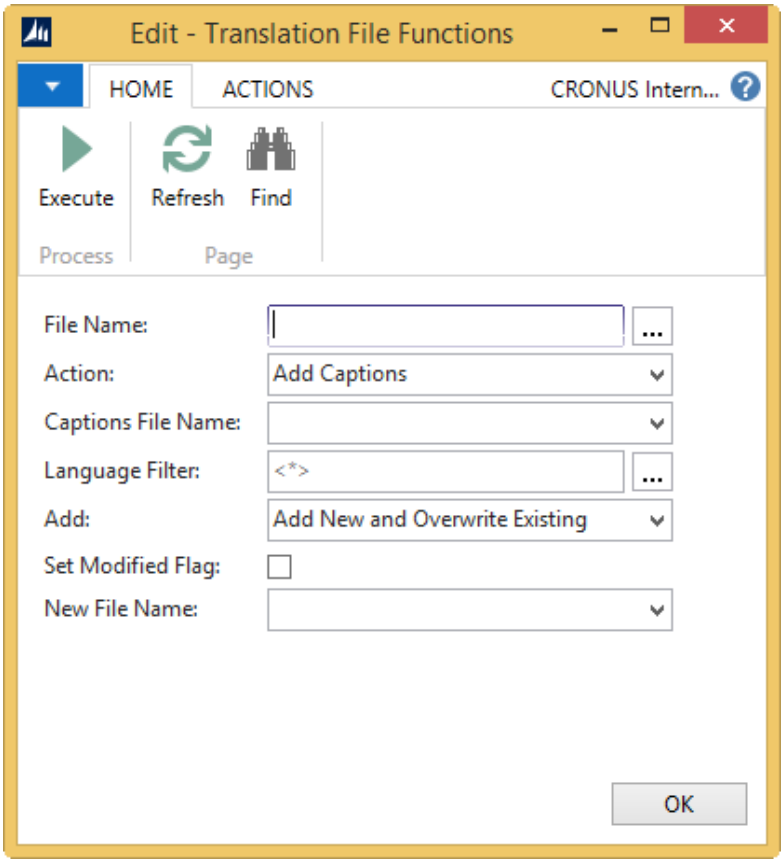


- Importing files
- Comparing files
- Comparing directories
- Splitting files
- Combining files
- Updating object properties in files
- Updating translation, i.e. captions, in files
- Checking guidelines in object file
- Converting Objects

NOTE: All these functions can also be accessed from the "Object Explorer" window through the "Functions" menu button.

18.1 TRANSLATIONS

Select **File Functions > Translations** to open the "Translation File Functions" window.



File Functions >Translation

In "Filename" you enter the file for which you want to update the translations. The result will be written to the file defined in "New Filename".

You can perform the following actions:

- Create Translation File
  given an object text file all strings will be extracted and placed into a translation (.txt) file
- Delete Captions
  given an object text file all captions will be deleted from the object txt file
- Add Captions
  given an object text file captions will be added from the file as defined in "Captions Filename", which can either be an object text file or an translation file

18.2 Compare File

Select **File Functions > Compare File** to open the windows file dialog to choose an object text file. This file will be compared with the same object residing in your database, using the compare executable as defined in the setup (see section **Setup**).

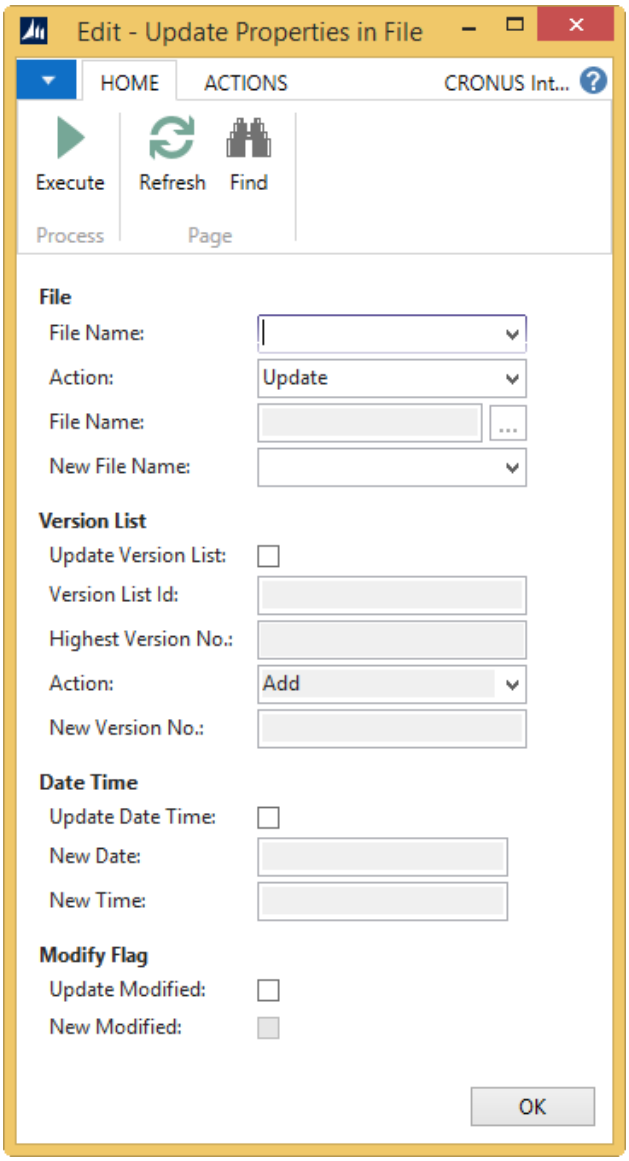
18.3 Compare Directory

Select **File Functions > Compare Directory** to open the **"Select a Folder"** window to choose a windows directory containing object text files. These files will

be compared with the same objects residing in your database, using the compare executable as defined in the setup (see section **Setup**).

18.4 PROPERTIES

Select **File Functions > Properties** to open the "Update Properties in File" window.



File Functions>Properties

In "Filename" you enter the file for which you want to update the object properties. This can either be a objects text file or a OBP file. The result will be written to the file defined in "New Filename".

You can update Version List, Date/ Time and Modified object properties by marking the corresponding check boxes and defining how you what to update these properties.

You can perform the following actions:

- Update
  given an object text file all object properties will be updated as described above
- Add
  given an object text file all object properties will be added from the file as defined in “Filename”, which can either be an object text file or an object properties (.obp) file
- Remove
  given an object text file all object properties will be removed
- Create OBP File
  given an object text file all object properties will be extracted and placed into an object properties (.obp) file

19 APPENDIX

A. Change List

OMA9.01

Objects can be converted from Microsoft Dynamics NAV 2013 to previous versions.

OBJ files can be split and combined.

Objects belonging to a project can be exported in their current state, before the project or after the project.

Status can be changed for a number of projects from the project list and plan board.

Bug fix: rollback objects were picked incorrectly from history when rolling back with a time value.

OMA9.02

Support for Microsoft Dynamics NAV 2013 R2

OMA9.03

- New codeunit "OM - Check New Transport" added to check for a new transport.
- New option in transport new time stamp: Date of Transport.
- New time stamp added to transport type.
- Assign Modifications page/form is default filtered on active user
- You get a proper message if you are importing a transport with a renumber field action and you don't have an active license that allows importing a text file.
- Marking added to all object tools.
- 2013: You get a proper message if marking of objects is not possible if the version list is full.
- 2013: Extra test added to the "Check Settings" to see if dotnet is ok.

- Bug fix: Removed quotes from field names in setup table.
- Bug fix: Check guidelines. If a semicolon was missing on the last line it was not seen as missing.
- Bug fix: Check guidelines. If REPEAT BEGIN where on the same line the BEGIN and END where sometimes removed wrongfully.
- Bug fix: Where used. In a statement like FieldRef.SETRANGE(VarName) the VarName was seen as a fieldname instead of a variable.
- Bug fix: If you removed a project from a transport in NAV2013 it still was connected to the transport.
- Bug fix: Paths in setup extended to length 250.
- Bug fix NAV2013 - Get a user name with a domain.
- Bug fix NAV2013 - Marking an object is possible If the version list is shorter than 248 instead of 80.
- Bug fix NAV2013: Where used - Properties on Actions on a CueGroup where not parsed.

- Bug fix NAV2013: Where used - Table data was not analyzed if the object was not present in the object table as table data. Now info from table information is used.
- Bug fix NAV2013: Where used - On overflow was given is a long key was shown on the card page.
- Bug fix NAV2013: Command Line - Timeout of 5 seconds removed from the import object command.
- Bug fix NAV2013: Command Line - A longer pause is added to wait until the log file is free to open.
- Bug fix NAV2013: Command Line - Parameter validatetablechanges added to the command line.
- Bug fix NAV2013: Wrong page was used if you did a lookup to a language.
- Bug fix NAV2013: Check Guidelines - If an action was missing a caption is was not seen as missing.
- Bug fix NAV2013: Trace modifications - In some installations it was not possible to modify an object with the trace trigger. This is fixed.

Please contact IDYN Support (support@idyn.nl) if you have any questions about this release.

B. Developer vs. Customer License

In the following table you can see which functionality is available in combination with a developer license and which functionality is available with a customer license. Some parts will only be available with a customer license when the C/AL history is updated with a developer license.

	Dev. License	Cust. License	C/AL Update Needed
Objects			
Object Explorer	X		
Import Objects	X	X	
Object Locks	X		
Modifications			
Start Tracing	X		
Modifications	X		
Assign Modifications	X		
Projects			
Projects	X	X	
Planning Board	X	X	
Project History	X	X	
Transports			
Transports	X	X	
Object Compare Sheet	X	X	
Import Transport	X	X	
Source Control			
C/AL History	X	X	X
Branches	X		
Rollback Objects	X		
Repository	X		
Compare C/AL Code	X	X	X
Analyzing Tools			
Where Used Objects	X	X	X

Compare Databases	X	X	X
Search String in C/AL Code	X	X	X
Check Guidelines	X	X	X
Test Worksheet	X	X	
Check Transferfields	X	X	
Client Monitor Analyzer	X	X	
MenuSuite Viewer	X	X	X
Check License	X	X	
Bitmaps	X	X	
Colors	X	X	

Object Tools

Create Table Wizard	X		
Renumber Objects	X		
Renumber Fields	X		
Renumber Elements	X		
Update Variables	X		
Change Field Options	X		
Translation Tool	X		
Downgrade Objects	X		

Data Tools

Action Worksheet	X	X	
Record Permissions Wizard	X	X	

Administration

Compress C/AL History	X		
Backup and Restore	X		

C. Differences between Fobs

	3.7	4.0	4.0 SP2	5.0	2009	2009 R2	2013	2013 R2
Tables	x	x	x	x	x	x	x	x
Forms	x	x	x	x	x	x		
Reports	x	x	x	x	x	x	x	x
Dataports	x	x	x	x	x	x		
XMLPorts		x	x	x	x	x	x	x
Codeunits	x	x	x	x	x	x	x	x
Menusuites		x	x	x	x	x	x	x
Pages					x	x	x	x
FINDSET			x	x	x	x	x	x
Integration Mgt.						x		x
NAV Locking						x		x

D. Setup Initialization Methods

	Development	Test	Pre-Production	Production
Lock Object at Design	TRUE	FALSE	FALSE	FALSE
Remove Locks at Closing Menu	Confirm	No	No	No
Save C/AL at Modification	If Changed	No	No	No
Save C/AL at Assigning	If Changed	No	No	No
Save C/AL at Locking	If Changed	No	No	No
Save C/AL at Unlocking	If Changed	No	No	No
Save C/AL after Transporting	Yes	No	No	No
Save C/AL before Import Tr.	If Changed	Yes	Yes	Yes
Save C/AL after Import Tr.	If Changed	Yes	Yes	Yes
Reset Project Status at Import	FALSE	TRUE	FALSE	FALSE
Block Project at Imp. Trans.	<empty>	No	Yes	Yes
Block Transport at Imp. Trans.	<empty>	Yes	No	Yes
Transport Nos. Format	T0001	TT0001	T0001	T0001

E. Object Table SQL Trigger

```
IF EXISTS
(
 SELECT name FROM sysobjects
 WHERE name = 'Object_TraceModifications' AND type = 'TR'
)
DROP TRIGGER Object_TraceModifications


GO


CREATE TRIGGER [dbo].[Object_TraceModifications]
  ON  [dbo].[Object]
  AFTER INSERT,DELETE,UPDATE
AS
BEGIN


 SET NOCOUNT ON


 DECLARE @SetupPresent INTEGER;
 DECLARE @SetupTriggerTestStatus INTEGER;
 DECLARE @SetupSQLCheckObjectLockType INTEGER;
 DECLARE @SetupTraceModifications INTEGER;
 DECLARE @SkipSQLTrigger INTEGER;
 DECLARE @Username NVARCHAR(100);
 DECLARE @LockedBy NVARCHAR(100);
 DECLARE @IsLocked INTEGER;
 DECLARE @IsModification INTEGER;
 DECLARE @CheckIsLocked INTEGER;
 DECLARE @CalledFromRepository INTEGER;
 DECLARE @ObjectType INTEGER;
 DECLARE @ObjectId INTEGER;
 DECLARE @ObjectName NVARCHAR(100);
 DECLARE @ObjectDate DATETIME
```

```
DECLARE @ObjectTime DATETIME
DECLARE @ObjectTypeText NVARCHAR(100);
DECLARE @ObjectIdText NVARCHAR(100);
DECLARE @Message NVARCHAR(100);
DECLARE @NoOfInserts INTEGER;
DECLARE @NoOfDeletes INTEGER;
DECLARE @NoOfModifies INTEGER;
DECLARE @Action INTEGER;
DECLARE @TokenNo INTEGER;
DECLARE @LockObjectAtSaving INTEGER;
DECLARE @OldVersionList NVARCHAR(80);
DECLARE @NewVersionList NVARCHAR(100);

SELECT TOP 1
  @SetupPresent = 1,
  @SetupTriggerTestStatus = [Trigger Test Status],
  @SetupSQLCheckObjectLockType = [SQL Check Object Lock Type],
  @SetupTraceModifications = [Trace Modifications],
  @SkipSQLTrigger = [Skip SQL Trigger],
  @LockObjectAtSaving = [Lock Object at Saving]
FROM [OM - Setup]
WHERE [Primary Key] = '';

IF @SetupPresent = 1
BEGIN

  IF @SetupTriggerTestStatus = 1
  BEGIN
    UPDATE [OM - Setup]
    SET [Trigger Test Status] = 3;
  END ELSE BEGIN

    IF @SkipSQLTrigger = 0
```

```
    BEGIN
      SELECT
        @ObjectType = d.[Type],
        @ObjectId = d.[ID],
        @ObjectName = d.[Name],
        @ObjectDate = d.[Date],
        @ObjectTime = d.[Time]
      FROM Deleted d;

      SELECT
        @ObjectType = i.[Type],
        @ObjectId = i.[ID],
        @ObjectName = i.[Name],
        @ObjectDate = i.[Date],
        @ObjectTime = i.[Time]
      FROM Inserted i;

      SET @CalledFromRepository = 0;
      SELECT @CalledFromRepository = 1
      FROM [OM - Repository Log] rl
      WHERE rl.[Object Type] = @ObjectType
      AND rl.[Object No_] = @ObjectId
      AND rl.Status = 1;

      IF @ObjectType > 0 AND (@ObjectId < 1000000000 OR @ObjectId >= 2000000000) AND
@CalledFromRepository = 0
      BEGIN
        SET @LockedBy = '';
        SET @TokenNo = 0;
        SET @IsLocked = 0;

        SELECT
          @LockedBy = ol.[Locked By],
          @LockedBy = ol.[Locked By],
```

```
  @TokenNo = ol.[Token No_],
  @IsLocked = 1
FROM [OM - Object Lock] ol
WHERE ol.[Object Type] = @ObjectType
AND ol.[Object No_] = @ObjectId;

SET @Username = UPPER(SYSTEM_USER);
IF (CHARINDEX('\', @Username) > 0)
  SET @Username = SUBSTRING(@Username, CHARINDEX('\', @Username) + 1, 100);

SELECT @NoOfInserts = COUNT([Type]) FROM Inserted;
SELECT @NoOfDeletes = COUNT([Type]) FROM Deleted;
SELECT @NoOfModifies = COUNT(i.[Type])
FROM Inserted i
  INNER JOIN Deleted d
    ON i.[Type] = d.[Type] AND i.[ID] = d.[ID];

SET @IsModification = 0;
SET @CheckIsLocked = 0;

-- INSERT
IF @NoOfModifies = 0 AND @NoOfInserts > 0
BEGIN
  SET @Action = 1;
  SET @IsModification = 1;
END;

-- MODIFY
IF @NoOfModifies > 0
  SET @Action = 2;

-- DELETE
IF @NoOfDeletes > 0 AND @NoOfInserts = 0
```

```
BEGIN
  SET @Action = 3;
  SET @IsModification = 1;
  SET @CheckIsLocked = 1;
END;

-- RENAME
IF @NoOfModifies = 0 AND @NoOfDeletes > 0 AND @NoOfInserts > 0
BEGIN
  SET @Action = 4
  SET @IsModification = 1;
  SELECT @Action = 0, @IsModification = 0
  FROM Inserted i, [OM - Update Object] uo
  WHERE i.[Type] = uo.[Object Type]
  AND i.[ID] = uo.[New Object No_];
END;

-- MODIFY
IF @Action = 2
  SELECT
    @IsModification = 1,
    @CheckIsLocked = 1
  FROM Inserted i
    INNER JOIN Deleted d
      ON i.[Type] = d.[Type]
      AND i.[ID] = d.[ID]
  WHERE
  (
    i.[Date] <> d.[Date]
    OR CONVERT(VARCHAR(20), i.[Time], 108) <> CONVERT(VARCHAR(20), d.[Time], 108)
    OR i.[Name] <> d.[Name]
    OR REPLACE(REPLACE(REPLACE(i.[Version List], ',', ''), '#', ''), 'LOCKED', '') <>
      REPLACE(REPLACE(REPLACE(d.[Version List], ',', ''), '#', ''), 'LOCKED', '')
```

```
  )
  AND i.[Version List] <> '! CHECK OBJECT VALID !'
  AND d.[Version List] <> '! CHECK OBJECT VALID !';

IF @SetupSQLCheckObjectLockType <> 0 AND @CheckIsLocked = 1 AND
   @LockedBy <> @Username AND @LockedBy <> ''
BEGIN
  SELECT
    @ObjectTypeText =
      CASE @ObjectType
        WHEN 1 THEN 'Table'
        WHEN 2 THEN 'Form'
        WHEN 3 THEN 'Report'
        WHEN 4 THEN 'Dataport'
        WHEN 5 THEN 'Codeunit'
        WHEN 6 THEN 'XMLport'
        WHEN 7 THEN 'MenuSuite'
        WHEN 8 THEN 'Page'
        ELSE ''
      END,
    @ObjectIdText = @ObjectId;

  SET @Message =
    CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
    'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
    'Object %s %s - %s is locked by %s' + CHAR(13) + CHAR(10);
  RAISERROR(@Message, 16, 1, @ObjectTypeText, @ObjectIdText, @ObjectName, @LockedBy);
  ROLLBACK TRANSACTION;
END;

IF @LockObjectAtSaving = 1 AND @IsModification = 1 AND @LockedBy = ''
BEGIN
```

```
IF EXISTS(
 SELECT 1
 FROM [OM - Repository Setup]
 WHERE [Use Repository] = 1)
BEGIN
 SET @Message =
  CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
  'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
  'The option ''Lock Object at Saving'' cannot be ' +
  'used in combination with repository' + CHAR(13) + CHAR(10);
 RAISERROR(@Message, 16, 1);
 ROLLBACK TRANSACTION;
END;

INSERT INTO [OM - Object Lock]
 (
  [Object Type], [Object No_], [Locked By],
  [Lock Date],
  [Lock Time],
  [Deleted], [Token No_], [Branch No_]
 )
SELECT
 @ObjectType, @ObjectId, @Username,
 CAST(CONVERT(VARCHAR(20), GETDATE(), 112) + ' 00:00:00' AS DATETIME),
 CAST('17540101 ' + CONVERT(VARCHAR(20), GETDATE(), 108) AS DATETIME),
 0, 0, '';
SELECT @OldVersionList = [Version List]
FROM Inserted;

IF CHARINDEX('#', @OldVersionList) = 1
 SET @NewVersionList = '#LOCKED,' + SUBSTRING(@OldVersionList, 2, 100)
ELSE
 SET @NewVersionList = 'LOCKED,' + @OldVersionList;
```

```
  IF LEN(@NewVersionList) <= 80
    UPDATE [Object]
    SET [Version List] = @NewVersionList
    WHERE [Object].[Type] = @ObjectType
    AND [Object].[ID] = @ObjectId
    AND CHARINDEX('LOCKED', [Version List]) = 0;

  SET @IsLocked = 1;
  SET @LockedBy = @Username;

END;

IF @SetupSQLCheckObjectLockType = 2 AND @CheckIsLocked = 1 AND @IsLocked = 0
BEGIN
  SELECT
    @ObjectTypeText =
      CASE @ObjectType
        WHEN 1 THEN 'Table'
        WHEN 2 THEN 'Form'
        WHEN 3 THEN 'Report'
        WHEN 4 THEN 'Dataport'
        WHEN 5 THEN 'Codeunit'
        WHEN 6 THEN 'XMLport'
        WHEN 7 THEN 'MenuSuite'
        WHEN 8 THEN 'Page'
        ELSE ''
      END,
    @ObjectIdText = @ObjectId;

  SET @Message =
    CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
    'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
```

```
    'Object %s %s - %s is not locked' + CHAR(13) + CHAR(10);
   RAISERROR(@Message, 16, 1, @ObjectTypeText, @ObjectIdText, @ObjectName);
   ROLLBACK TRANSACTION;
  END;

  IF @SetupTraceModifications = 1 AND @IsModification = 1
   INSERT INTO [OM - Modification]
    (
     [Object Type], [Object No_], [Object Name],
     [Object Date], [Object Time],
     [Status], [Inserted By],
     [Insert Date],
     [Insert Time],
     [Assigned to Project No_], [Assigned By],
     [Assign Date], [Assign Time], [Auto Assigned],
     [Transport No_], [Object Date Time],
     [Locked By], [Token No_], [Traced By SQL], [SQL Trigger], [SQL Status],
     [System User], [Host Name]
    )
   SELECT
    @ObjectType, @ObjectId, @ObjectName,
    @ObjectDate, CAST('17540101 ' + CONVERT(VARCHAR(20), @ObjectTime, 108) AS DATETIME),
    0, @Username,
    CAST(CONVERT(VARCHAR(20), GETDATE(), 112) + ' 00:00:00' AS DATETIME),
    CAST('17540101 ' + CONVERT(VARCHAR(20), GETDATE(), 108) AS DATETIME),
    '', '',
    CAST('17530101 00:00:00' AS DATETIME), CAST('17530101 00:00:00' AS DATETIME), 0,
    '', CAST(CONVERT(VARCHAR(20), @ObjectDate, 112) + ' ' +
    CONVERT(VARCHAR(20), @ObjectTime, 108) AS DATETIME),
    @LockedBy, @TokenNo, 1, @Action, 1,
    SUBSTRING(SYSTEM_USER, 1, 50), SUBSTRING(HOST_NAME(), 1, 50);

 END;
```

```
    END;
   END;

 END;

END;

Update Object Table SQL Trigger

IF EXISTS
(
 SELECT name FROM sysobjects
 WHERE name = 'OM_Update_Object' AND type = 'TR'
)
DROP TRIGGER OM_Update_Object

GO

CREATE TRIGGER [dbo].[OM_Update_Object]
  ON  [dbo].[OM - Update Object]
  AFTER INSERT,DELETE,UPDATE
AS
BEGIN
 SET NOCOUNT ON;

 UPDATE [Object]
 SET [Object].[ID] = Inserted.[New Object No_]
 FROM [Object], Inserted
 WHERE [Object].[Type] = Inserted.[Object Type]
 AND [Object].[ID] = Inserted.[Object No_]
 AND Inserted.[Update Properties] = 0;
```

```
  UPDATE [Object]
  SET
    [Object].[ID] = Inserted.[New Object No_],
    [Object].[Modified] = Inserted.[Modified],
    [Object].[Version List] = Inserted.[Version List],
    [Object].[Date] = Inserted.[Object Date],
    [Object].[Time] = Inserted.[Object Time]
  FROM [Object], Inserted
  WHERE [Object].[Type] = Inserted.[Object Type]
  AND [Object].[ID] = Inserted.[Object No_]
  AND Inserted.[Update Properties] = 1;

  DELETE FROM [OM - Update Object] WHERE [Object Type] = -1;

END;
```

F. Ask SQL Trigger

```
CREATE TRIGGER [dbo].[OM_Ask_SQL_900]
  ON  [dbo].[OM - Ask SQL]
  AFTER INSERT
AS
BEGIN
 SET NOCOUNT ON

 DECLARE @Question INTEGER;
 DECLARE @Answer NVARCHAR(100);

 SELECT @Question = Question
 FROM inserted;

 SET @Answer = '';

 IF @Question = 1
 BEGIN
  SET @Answer = '001';
 END;

 IF @Question = 2
 BEGIN

  SET @Answer = '1';

  DELETE FROM [OM - Where Used Status];

  INSERT INTO [OM - Where Used Status] ([Object Type], [Object No_])
  SELECT ob.[Type], ob.ID
  FROM [Object] ob
   LEFT OUTER JOIN [OM - Known Object Export Error] ee
```

```
  ON ob.[Type] = ee.[Object Type]
  AND ob.ID = ee.[Object No_]
 LEFT OUTER JOIN [OM - Where Used Object] wu
  ON ob.[Type] = wu.[Object Type]
  AND ob.ID = wu.[Object No_]
 LEFT OUTER JOIN
 (
  SELECT ho2.[Object Type], ho2.[Object No_], MAX(ho2.[Entry No_]) [Entry No_]
  FROM [OM - C_AL History Object] ho2
  GROUP BY ho2.[Object Type], ho2.[Object No_]
 ) sho
  ON ob.[Type] = sho.[Object Type]
  AND ob.ID = sho.[Object No_]
 LEFT OUTER JOIN [OM - C_AL History Object] ho
  ON sho.[Entry No_] = ho.[Entry No_]
  AND ob.[Type] = ho.[Object Type]
  AND ob.ID = ho.[Object No_]
WHERE ob.[Type] <> 0
AND ee.[Object Type] IS NULL
AND
(
 wu.[Object Type] IS NULL OR
 wu.[C_AL Changed] = 1 OR
 wu.[Status] < 2 OR
 wu.[Version No_] < 701 OR
 ho.[Entry No_] IS NULL OR
 REPLACE(REPLACE(REPLACE(ho.[Version List], ',', ''), '#', ''), 'LOCKED', '') <>
 REPLACE(REPLACE(REPLACE(ob.[Version List], ',', ''), '#', ''), 'LOCKED', '') OR
 ho.Modified <> ob.Modified OR
 ho.[Object Name] <> ob.Name OR
 ho.[Object Date] <> ob.Date OR
 ABS(CAST(ho.[Object Time] AS FLOAT) - CAST(ob.[Time] AS FLOAT)) * 60 * 60 * 24 > 1 OR
 wu.[Object Date] <> ob.Date OR
```

```
    wu.[Object Time] <> ob.Time OR
    wu.[Object Modified] <> ob.Modified
  );

  INSERT INTO [OM - Where Used Status] ([Object Type], [Object No_])
  SELECT wu.[Object Type], wu.[Object No_]
  FROM [OM - Where Used Object] wu
    LEFT OUTER JOIN [Object] ob
      ON wu.[Object Type] = ob.[Type]
      AND wu.[Object No_] = ob.ID
  WHERE wu.[Object No_] < 2000000000
  AND ob.[Type] IS NULL;

  SELECT TOP 1 @Answer = '0'
  FROM [OM - Where Used Status];

 END;

 UPDATE [OM - Ask SQL] SET Answer = @Answer
 FROM inserted i, [OM - Ask SQL] ASql
 WHERE i.Question = ASql.Question;

END;
```

G. Block Database Conversion

```
CREATE TRIGGER [dbo].[OM_Block_DB_Conversion_900]
  ON  [dbo].[$ndo$dbproperty]
  AFTER UPDATE
AS
BEGIN
 SET NOCOUNT ON;

 DECLARE @OtherDatabaseVersionNo INTEGER = 0;
 DECLARE @Message NVARCHAR(100);

 SELECT @OtherDatabaseVersionNo = 1
 FROM Inserted i
  INNER JOIN Deleted d
    ON i.programid = d.programid
 WHERE i.databaseversionno <> d.databaseversionno;
 IF @OtherDatabaseVersionNo = 1
 BEGIN
  SET @Message =
    CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
    'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
    'Database cannot be converterd.' + CHAR(13) + CHAR(10) +
    'Please first remove the Block Database Conversion trigger.' + CHAR(13) + CHAR(10);

  RAISERROR(@Message, 16, 1);
  ROLLBACK TRANSACTION;
 END;

END;
```

H. Trace Modification Trigger

```
CREATE TRIGGER [dbo].[Object_TraceModifications_908]
  ON  [dbo].[Object]
  AFTER INSERT,DELETE,UPDATE
AS
BEGIN
 SET NOCOUNT ON


 DECLARE @SetupPresent INTEGER;
 DECLARE @SetupSQLCheckObjectLockType INTEGER;
 DECLARE @SkipSQLTrigger INTEGER;
 DECLARE @CheckObjectValid INTEGER = 0;
 DECLARE @DatabaseLocked INTEGER;
 DECLARE @TraceModifications INTEGER;
 DECLARE @LockedBy NVARCHAR(100);
 DECLARE @IsLocked INTEGER;
 DECLARE @IsModification INTEGER;
 DECLARE @CheckIsLocked INTEGER;
 DECLARE @CalledFromRepository INTEGER;
 DECLARE @ObjectType INTEGER = 0;
 DECLARE @ObjectId INTEGER = 0;
 DECLARE @ObjectName NVARCHAR(100);
 DECLARE @ObjectDate DATETIME
 DECLARE @ObjectTime DATETIME
 DECLARE @ObjectTypeText NVARCHAR(100);
 DECLARE @ObjectIdText NVARCHAR(100);
 DECLARE @ObjectVersionList NVARCHAR(100);
 DECLARE @Message NVARCHAR(100);
 DECLARE @NoOfInserts INTEGER;
 DECLARE @NoOfDeletes INTEGER;
 DECLARE @NoOfModifies INTEGER;
 DECLARE @Action INTEGER;
 DECLARE @TokenNo INTEGER;
```

```
DECLARE @LockObjectAtSaving INTEGER;
DECLARE @Username NVARCHAR(100);
DECLARE @ShortUsername NVARCHAR(100);
DECLARE @LockedAdded INTEGER = 0;

SET @Username = SYSTEM_USER;
SET @ShortUsername = UPPER(SYSTEM_USER);
IF (CHARINDEX('\', @ShortUsername) > 0)
  SET @ShortUsername = SUBSTRING(@ShortUsername, CHARINDEX('\', @ShortUsername) + 1, 100);

SELECT TOP 1
  @SetupPresent = 1,
  @SetupSQLCheckObjectLockType = [SQL Check Object Lock Type],
  @SkipSQLTrigger = [Skip SQL Trigger],
  @LockObjectAtSaving = [Lock Object at Saving],
  @DatabaseLocked = [Database Locked],
  @TraceModifications = [Trace Modifications]
FROM [OM - Setup]
WHERE [Primary Key] = '';

SELECT TOP 1
  @SetupSQLCheckObjectLockType = [SQL Check Object Lock Type],
  @LockObjectAtSaving = [Lock Object at Saving],
  @DatabaseLocked = [Database Locked],
  @TraceModifications = [Trace Modifications]
FROM [OM - Setup]
WHERE [Primary Key] = @ShortUsername;

IF @SetupPresent = 1
BEGIN

  IF @SkipSQLTrigger = 0
  BEGIN
```

```
IF @DatabaseLocked = 1
BEGIN
  RAISERROR('Database is locked.', 16, 1);
  ROLLBACK TRANSACTION;
END;

SELECT
  @ObjectType = d.[Type],
  @ObjectId = d.[ID],
  @ObjectName = d.[Name],
  @ObjectDate = d.[Date],
  @ObjectTime = d.[Time],
  @ObjectVersionList = d.[Version List]
FROM Deleted d
WHERE d.[Type] > 0;

IF @ObjectVersionList = '! CHECK OBJECT VALID !'
  SET @CheckObjectValid = 1;

SELECT
  @ObjectType = i.[Type],
  @ObjectId = i.[ID],
  @ObjectName = i.[Name],
  @ObjectDate = i.[Date],
  @ObjectTime = i.[Time],
  @ObjectVersionList = i.[Version List]
FROM Inserted i
WHERE i.[Type] > 0;

IF @ObjectVersionList = '! CHECK OBJECT VALID !'
  SET @CheckObjectValid = 1;
```

```
    SET @CalledFromRepository = 0;
    SELECT @CalledFromRepository = 1
    FROM [OM - Repository Log] rl
    WHERE rl.[Object Type] = @ObjectType
    AND rl.[Object No_] = @ObjectId
    AND rl.Status = 1;

    SELECT @Username = em.[User Id], @ShortUsername = UPPER(em.[User Id])
    FROM [OM - Expected Modification] em
    WHERE (em.[Object Type] = @ObjectType OR em.[Object Type] = 0)
    AND (em.[Object No_] = @ObjectId OR em.[Object No_] = 0);

    IF (CHARINDEX('\', @ShortUsername) > 0)
      SET @ShortUsername = SUBSTRING(@ShortUsername, CHARINDEX('\', @ShortUsername) + 1, 100);

    IF @ObjectType > 0 AND (@ObjectId < 1000000000 OR @ObjectId >= 2000000000) AND
@CalledFromRepository = 0 AND @CheckObjectValid = 0
    BEGIN
      SET @LockedBy = '';
      SET @TokenNo = 0;
      SET @IsLocked = 0;

      SELECT
        @LockedBy = ol.[Locked By],
        @TokenNo = ol.[Token No_],
        @IsLocked = 1
      FROM [OM - Object Lock] ol
      WHERE ol.[Object Type] = @ObjectType
      AND ol.[Object No_] = @ObjectId;

      SELECT @NoOfInserts = COUNT([Type]) FROM Inserted;
      SELECT @NoOfDeletes = COUNT([Type]) FROM Deleted;
      SELECT @NoOfModifies = COUNT(i.[Type])
      FROM Inserted i
```

```
  INNER JOIN Deleted d
    ON i.[Type] = d.[Type] AND i.[ID] = d.[ID];

SET @IsModification = 0;
SET @CheckIsLocked = 0;

-- INSERT
IF @NoOfModifies = 0 AND @NoOfInserts > 0
BEGIN
  SET @Action = 1;
  SET @IsModification = 1;
END;

-- MODIFY
IF @NoOfModifies > 0
  SET @Action = 2;

-- DELETE
IF @NoOfDeletes > 0 AND @NoOfInserts = 0
BEGIN
  SET @Action = 3;
  SET @IsModification = 1;
  SET @CheckIsLocked = 1;
END;

-- RENAME
IF @NoOfModifies = 0 AND @NoOfDeletes > 0 AND @NoOfInserts > 0
BEGIN
  SET @Action = 4
  SET @IsModification = 1;
END;

-- MODIFY
```

```
IF @Action = 2
 SELECT
  @IsModification = 1,
  @CheckIsLocked = 1
 FROM Inserted i
  INNER JOIN Deleted d
   ON i.[Type] = d.[Type]
   AND i.[ID] = d.[ID]
 WHERE
 (
  i.[Date] <> d.[Date]
  OR CONVERT(VARCHAR(20), i.[Time], 108) <> CONVERT(VARCHAR(20), d.[Time], 108)
  OR i.[Name] <> d.[Name]
  OR REPLACE(REPLACE(i.[Version List], ',', ''), '#', '') <>
    REPLACE(REPLACE(d.[Version List], ',', ''), '#', '')
 );

IF @SetupSQLCheckObjectLockType <> 0 AND @CheckIsLocked = 1 AND
  @LockedBy <> @Username AND @LockedBy <> ''
BEGIN
 SELECT
  @ObjectTypeText =
   CASE @ObjectType
     WHEN 1 THEN 'Table'
     WHEN 2 THEN 'Form'
     WHEN 3 THEN 'Report'
     WHEN 4 THEN 'Dataport'
     WHEN 5 THEN 'Codeunit'
     WHEN 6 THEN 'XMLport'
     WHEN 7 THEN 'MenuSuite'
     WHEN 8 THEN 'Page'
     WHEN 9 THEN 'Query'
     ELSE ''
```

```
    END,
    @ObjectIdText = @ObjectId;

  SET @Message =
    CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
    'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
    'Object %s %s - %s is locked by %s' + CHAR(13) + CHAR(10);
  RAISERROR(@Message, 16, 1, @ObjectTypeText, @ObjectIdText, @ObjectName, @LockedBy);
  ROLLBACK TRANSACTION;
END;

IF @LockObjectAtSaving = 1 AND @IsModification = 1 AND @LockedBy = '' AND @Action <> 4
BEGIN

  IF EXISTS(
    SELECT 1
    FROM [OM - Repository Setup]
    WHERE [Use Repository] = 1)
  BEGIN
    SET @Message =
      CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
      'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
      'The option ''Lock Object at Saving'' cannot be ' +
      'used in combination with repository' + CHAR(13) + CHAR(10);
    RAISERROR(@Message, 16, 1);
    ROLLBACK TRANSACTION;
  END;

  DELETE FROM [OM - Object Lock]
  WHERE [Object Type] = @ObjectType
  AND [Object No_] = @ObjectId;

  INSERT INTO [OM - Object Lock]
```

```
    (
      [Object Type], [Object No_], [Locked By],
      [Lock Date],
      [Lock Time],
      [Deleted], [Token No_], [Branch No_]
    )
   SELECT
    @ObjectType, @ObjectId, @Username,
    CAST(CONVERT(VARCHAR(20), GETDATE(), 112) + ' 00:00:00' AS DATETIME),
    CAST('17540101 ' + CONVERT(VARCHAR(20), GETDATE(), 108) AS DATETIME),
    0, 0, '';

   UPDATE [Object]
   SET [Locked] = 1,
      [Locked By] = @Username
   WHERE [Type] = @ObjectType
   AND [ID] = @ObjectId
   AND [Locked] = 0;

   SET @IsLocked = 1;
   SET @LockedAdded = 1;
   SET @LockedBy = @Username;

  END;

  IF @SetupSQLCheckObjectLockType = 2 AND @CheckIsLocked = 1 AND @IsLocked = 0
  BEGIN
   SELECT
    @ObjectTypeText =
     CASE @ObjectType
       WHEN 1 THEN 'Table'
       WHEN 2 THEN 'Form'
       WHEN 3 THEN 'Report'
```

```
        WHEN 4 THEN 'Dataport'
        WHEN 5 THEN 'Codeunit'
        WHEN 6 THEN 'XMLport'
        WHEN 7 THEN 'MenuSuite'
        WHEN 8 THEN 'Page'
        WHEN 9 THEN 'Query'
        ELSE ''
      END,
    @ObjectIdText = @ObjectId;

  SET @Message =
    CHAR(13) + CHAR(10) + CHAR(13) + CHAR(10) +
    'OBJECT MANAGER ERROR:' + CHAR(13) + CHAR(10) +
    'Object %s %s - %s is not locked' + CHAR(13) + CHAR(10);
  RAISERROR(@Message, 16, 1, @ObjectTypeText, @ObjectIdText, @ObjectName);
  ROLLBACK TRANSACTION;
END;

IF @IsModification = 1 AND @TraceModifications = 1
  INSERT INTO [OM - Modification]
    (
      [Object Type], [Object No_], [Object Name],
      [Object Date], [Object Time],
      [Status], [Inserted By],
      [Insert Date],
      [Insert Time],
      [Assigned to Project No_], [Assigned By],
      [Assign Date], [Assign Time], [Auto Assigned],
      [Transport No_], [Object Date Time],
      [Locked By], [Token No_], [Traced By SQL], [SQL Trigger], [SQL Status],
      [System User], [Host Name]
    )
  SELECT
```

```
  @ObjectType, @ObjectId, @ObjectName,
  @ObjectDate, CAST('17540101 ' + CONVERT(VARCHAR(20), @ObjectTime, 108) AS DATETIME),
  0, @ShortUsername,
  CAST(CONVERT(VARCHAR(20), GETDATE(), 112) + ' 00:00:00' AS DATETIME),
  CAST('17540101 ' + CONVERT(VARCHAR(20), GETDATE(), 108) AS DATETIME),
  '', '',
  CAST('17530101 00:00:00' AS DATETIME), CAST('17530101 00:00:00' AS DATETIME), 0,
  '', CAST(CONVERT(VARCHAR(20), @ObjectDate, 112) + ' ' +
  CONVERT(VARCHAR(20), @ObjectTime, 108) AS DATETIME),
  @LockedBy, @TokenNo, 1, @Action, 1,
  SUBSTRING(@ShortUsername, 1, 50), SUBSTRING(HOST_NAME(), 1, 50);

IF @Action = 1 OR @Action = 2
BEGIN
 INSERT INTO [OM - Object Lock]
  (
   [Object Type], [Object No_], [Locked By],
   [Lock Date],
   [Lock Time],
   [Deleted], [Token No_], [Branch No_]
  )
 SELECT
  ob.[Type], ob.ID, ob.[Locked By],
  CAST(CONVERT(VARCHAR(20), GETDATE(), 112) + ' 00:00:00' AS DATETIME),
  CAST('17540101 ' + CONVERT(VARCHAR(20), GETDATE(), 108) AS DATETIME),
  0, 0, ''
 FROM Inserted ob
  LEFT OUTER JOIN [OM - Object Lock] ol
   ON ob.[Type] = ol.[Object Type]
   AND ob.ID = ol.[Object No_]
 WHERE (ob.Locked = 1 OR @LockedAdded = 1)
 AND ol.[Object Type] IS NULL
 AND ob.[Locked By] <> 'CheckSettings'
```

```
        AND ob.[Type] > 0
        GROUP BY ob.[Type], ob.ID, ob.[Locked By];

        DELETE ol
        FROM [OM - Object Lock] ol
          INNER JOIN Inserted ob
            ON ol.[Object Type] = ob.[Type]
            AND ol.[Object No_] = ob.ID
        WHERE (ob.Locked = 0 AND NOT @LockedAdded = 1)
        AND ol.Deleted = 0;
      END;

      IF @Action = 3
        DELETE ol
        FROM [OM - Object Lock] ol
          INNER JOIN Deleted ob
            ON ol.[Object Type] = ob.[Type]
            AND ol.[Object No_] = ob.ID
            AND ol.Deleted = 0;
    END;

   END;

 END;

END;
```