Microsoft
Business
Solutions

# Upgrade Toolkit

MICROSOFT BUSINESS SOLUTIONS–NAVISION

# UPGRADE TOOLKIT

## DISCLAIMER

## COPYRIGHT NOTICE

## TRADEMARK NOTICE

## PREFACE

This document contains the information necessary to successfully upgrade from Navision Financials®, Navision Manufacturing, Commerce Portal, Commerce Gateway, Navision Attain® or Microsoft® Business Solutions–Navision® to Microsoft Business Solutions–Navision 4.0 and to migrate to the SQL Server Option for Navision 4.0.

Upgrading to Navision 4.0 is the first step in the process of migrating to the Microsoft SQL Server Option for Navision 4.0.

The tools described in this document are located in the `Upgtk` folder on the product CD. You will find a detailed log of the differences between the specific products and Navision 4.0 in the `changes.doc` file that is located in the same folder.

You should also be familiar with the symbols and typographical conventions used in the Navision manuals. In the list below, you can see how various elements of the program are distinguished by special typefaces and symbols:

| Appearance | Element |
|---|---|
| CTRL | Keys on the keyboard. They are written in small capitals. |
| Design | Menu items and buttons in windows. They always start with a capital letter, and the access key is underlined. |
| **Address** | Field names. They appear in bold and start with a capital letter. |
| ***Department*** | Names of windows, boxes and tabs. They appear in bold italics and start with a capital letter. |
| *Hansen* | Text that you must enter, for example: "...enter *Yes* in this field." It is written in italics. |
| `fin.flf` | File names. They are written with the Courier font and lowercase letters. |

# TABLE OF CONTENTS

# Chapter 1

# The Upgrade Toolkit

The Microsoft® Business Solutions–Navision Upgrade Toolkit 4.0 contains several tools which must be used when upgrading from Navision Financials, Navision Manufacturing, Commerce Portal, Commerce Gateway, Navision Attain 3.01, 3.10, 3,60 or Microsoft Business Solutions–Navision 3.70 to Microsoft Business Solutions–Navision 4.0. These tools and the instructions for using them are described in detail in this document.

The chapter contains:

- 1.1 Description of the Upgrade Toolkit

## 1.1 DESCRIPTION OF THE UPGRADE TOOLKIT

The Upgrade Toolkit consists of a set of tools and procedures that are designed to help you upgrade to Microsoft Business Solutions–Navision 4.0. The description covers the upgrade for the following products:

- Navision Financials 2.00
- Navision Financials 2.01
- Navision Financials 2.50
- Navision Financials 2.60
- Navision Manufacturing 2.60
- Commerce Portal 2.65
- Commerce Gateway 2.65
- Navision Attain 3.01
- Navision Attain 3.10
- Navision Attain 3.60
- Microsoft Business Solutions–Navision 3.70

The tools and procedures that you must use vary depending on the version you are upgrading from.

The Upgrade Toolkit also contains the tools for migrating from Navision 4.0 to the Microsoft SQL Server Option for Navision 4.0 and a description of the procedures involved. Upgrading to Navision 4.0 is the first step in the process of migrating to the Microsoft SQL Server Option for Navision 4. For more information, see Chapter 7.

The Upgrade Toolkit is located in the `Upgtk` folder. The files in the toolkit are grouped in the following folders:

**Data Conversion Tools** - This folder contains the files used to convert data from a prior version to 4.0. The files to upgrade a specific version can be found in the relevant subfolder. For more information, see Chapter 6.

**Documents** - this folder contains the subfolders:

*Localization Instructions* - These documents provide information the data conversion tools and recommendations on how to localize / customize the upgrade tools.

*Upgrade Quick Guides* - This folder contains the Upgrade Quick Guides for each prior version. Partners can add to these documents to include their own customized upgrade steps.

**Feature Enhancements Documents** - These documents provide an overview of the new features in Navision 4.0 when comparted to prior versions of the Navision product. The features are classified by granule.

**NAD Upgrade Documentation** - This folder contains documentation to assist in the upgrade of NAD 2.60 customers to Navision 3.60.

**Multilanguage Tools** - This folder contains the tools to Multilanguage enable an existing database. For more information, see Chapter 5.

**Object Change Tools** - This folder contains tools used to change the existing objects in some of the prior versions so that they can work in the 4.0 version. For more information, see Chapter 2 and Chapter 4.

**Security** - This folder contains the tools for converting database logins to Windows logins. For more information, see Chapter 6.

**SQL Migration** - This folder contains the tools needed for migrating a Navision Server database to Microsoft SQL Server Option for Navision 4.0. For more information, see Chapter 7.

# Chapter 2

# Preparing to Upgrade

There are some preparations that must be made before you can upgrade the customer’s database to Microsoft Business Solutions–Navision 4.0.

These include testing the database and ensuring that the database you are upgrading does not contain any objects that have the same name.

The chapter contains:

- 2.1 Preparing to Upgrade
- 2.2 Testing the Old Database
- 2.3 Checking for Duplicate Names

## 2.1 PREPARING TO UPGRADE

To prepare a customer's old installation for upgrading:

**1** Verify that both your solution developer's license file and the customer's license file have been upgraded to the newest version of Navision.

**2** Identify the user ID and password of a superuser in the system.

**Note**

Before creating a backup of the database, you must ensure that the inventory cost data in the customer's database is up to date by running the ***Adjust Cost-Item Entries*** batch job.

**3** Make a backup of the entire database before you begin the upgrade process. Keep the backup in a safe place, and keep it for a long time.

**Note**

Note how long it takes to complete each of the steps in the upgrade process. You can use this information to estimate the time and cost involved in future upgrades.

**4** Make a copy of the customer's database, and upgrade the copy.

**5** Ensure that no other users are connected to the database before you carry out each part of the upgrade process.

**Note**

Because the customer's license file may be too limited to carry out some of the steps in the following sections, use your solution developer's license from this point on. During the upgrade process, you will use the development environment (C/SIDE®) in both the old and new version. You cannot open a Navision Financials 2.01 database with Navision 4.0.

**Note**

If you are upgrading from any earlier version of Commerce Portal to Navision 4.0, be aware that the ASP pages for your Web portal need to be updated accordingly. Ensure that this has been done before you start the upgrade of your database. Otherwise, your Web portal will not work.

## 2.2 TESTING THE OLD DATABASE

To determine the state of the customer's current database and correct any database errors that may exist, follow the procedures described in this section. This will ensure that no errors exist in the database that will be used as the basis for the upgrade.

**1** Open the customer's database in the version that the customer is currently using.

**2** Run a database test to determine the state of the customer's database.

Test everything except field relationships between tables. If the test fails, you must follow the workflow for repairing damaged databases (contact your local Microsoft Business Solutions Solution Center for details).

**3** Run the remaining part of the database test, that is, test field relationships between tables.

This will allow you to determine the extent of any data inconsistency that can exist in the database. If any error messages appear during the test, note their content and number. You must then decide whether or not these errors will affect the upgrade.

**4** Compile all the objects in the database.

Make a list of the objects that cannot be compiled. At some point, you must decide what to do with the objects that cannot be compiled. They will create problems if you ignore them.

You have tested the customer's old database and are now ready to begin the process of upgrading the database.

## 2.3 CHECKING FOR DUPLICATE NAMES

Navision 4.0 does not allow you to create two objects with duplicate names within the same type of objects. For example, you cannot give two tables the same name. You must therefore ensure that the database you are upgrading does not contain any objects within each objects type that have the same name.

**Note**

You must only carry out this step in the upgrade procedure if you are upgrading from Navision Financials 2.00 or 2.01.

Navision Financials 2.00 and 2.01

To ensure that the database does not contain any duplicate object names:

**1** Open the *Object Designer* and import the `Duplicate Names.fob` file. This file is located in the `Upgtk` folder on the product CD.

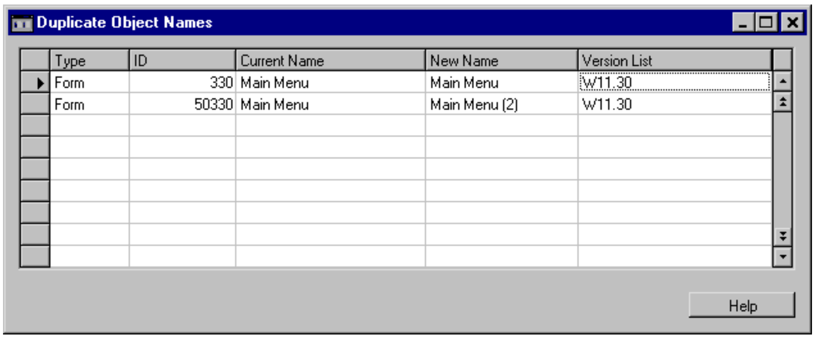
The following objects are imported:

| **Type** | **ID** | **Name** |
|---|---|---|
| Table | 104018 | ***Duplicate Object Name*** |
| Table | 104036 | ***Object Name Suggestion*** |
| Form | 104018 | ***Duplicate Object Names*** |
| Codeunit | 104018 | ***Check Object Names*** |

**2** Select codeunit 104018, ***Check Object Names***, and click Run (ALT+R).

Codeunit 104018 checks whether any objects of the same type share the same name. If any objects share the same name, you must solve these conflicts by renaming some of the objects.

If any names need to be changed, the program will display a list of the current names and the suggested new names in the ***Duplicate Object Names*** window:



| Type | ID | Current Name | New Name | Version List |
|---|---|---|---|---|
| Form | 330 | Main Menu | Main Menu | W11.30 |
| Form | 50330 | Main Menu | Main Menu (2) | W11.30 |

You can modify these suggestions if you wish. When you close the window, a message will appear asking you to confirm whether or not you want to implement the changes displayed in the ***Duplicate Object Names*** window.

Click Yes to implement the changes.

Click No if you want to postpone renaming the objects and continue the upgrade process later.

**3** Make a backup of the customer's Navision Financials database after you have implemented the new object names and call it, for example, `data.fbk`.

You have now ensured that the customer's database does not contain any objects with the same name.

# Chapter 3
# Upgrading to Multilanguage

Microsoft Business Solutions–Navision is multilanguage enabled and allows you to change the application language on the fly.

Since version 3.01, the code base language has been English (United States) and this means that you have to go through some steps before you customize the new database.

If you want to make use of the multilanguage feature for customer-specific objects, you must multilanguage enable them.

The chapter contains:

## 3.1 WHAT IS MULTILANGUAGE?

This means that the user can switch application language on the fly. It also means that the code base for the base application is in English (United States), which requires you to take some extra steps when you upgrade to Navision 4.0 from Navision Financials, Navision Manufacturing, Commerce Portal and Commerce Gateway.

**Note**

Since the databases version 3.01 or later is already multilanguage enabled, you will only have to multilanguage enable customizations where customers want to use multilanguage functionality. For more information about how to enable a Navision compatible product for multilanguage, see the section called *Multilanguage Enabling the Database* on page 28.

In multilanguage terms, all customized databases version 2.00-2.65 will fall into one of two categories:

**The Application Is in English**

Since the application is already in English, you will not have to follow the steps in the multilanguage part of the upgrade that deal with translation.

However, you should still convert customer-specific objects with the *conv-ml* tool and then change the application language from English (United States) into your version of English if needed.

**The Application Is Not in English**

In version 2.00-2.65, objects in customized applications have usually been programmed in the local language(s). This meant that name properties and global variables, for example, were hard-coded into the database so that the user could not easily switch application language.

In Navision, the code base must be in English (United States). This means that, for example, name properties and variables are not translated. When the code base is in English (United States) and multilanguage-enabled captions have been added to the objects in the database, different application languages can run on a common code base. These captions allow you to switch application language on the fly.

The *Name* property of an object, for example, is not translated from English (United States), and it is not used in the user interface of the application. Instead, the user sees the contents of the *Caption* property which gets its information from the *CaptionML* property. The CaptionML property contains captions in all available languages for an object, field, text constant, and so on.

These differences mean that you will have to perform some special steps if your existing database has been translated.

Once you have completed the upgrade and start developing new objects in the customer’s new Navision database, you must follow the guidelines for developing in a multilanguage-enabled database as described in the manual *Application Designer's Guide*.

**Note**

The procedures in this chapter refer to the customer's current database as a Navision Financials 2.60 database, but you can perform the same steps with earlier versions of Navision Financials and with Commerce Portal, Commerce Gateway and Navision Manufacturing.

## How to Use the Databases in this Chapter

The procedures in this chapter will tell you to open Navision Financials databases. It is very important that you open these databases with the new Navision C/SIDE throughout this chapter.

**Note**

You should use copies of the databases for all of these procedures or ensure that you have a valid backup of the database before you open it with a Navision C/SIDE.

To open a Navision Financials database with a Navision C/SIDE, follow this procedure:

**1** Open Navision and open the database that you want to work in.

**2** When prompted, click OK to convert the database.

**3** To open the *Object Designer*, click SHIFT+ F12.

**4** In the *Object Designer*, select all objects and press F11 to compile them.

The database is now fully converted and you can continue with the procedure that you started. The next time you need to open this database with a Navision C/SIDE, you will not be asked to convert the database.

Versions earlier than version 3.01 do not set global language in codeunit 1, ***ApplicationManagement***, as Navision does. This means that when you open a Navision database, the first language that you select will be the global language for all previous version of Navision Financials that you open with that Navision C/SIDE.

**Important**

If you do not compile all objects after converting the database, you will get an incorrect result when you use the Translate, Export functionality.

Translation of Old Base Version

In the section called *Translating the Code Base* on page 19, you are asked to translate your country's old base version into English (United States). Please note that your local Microsoft Business Solutions office may have made that English database available to you in some way. If they have, you should not carry out the translation procedure for your country's old base version.

To ensure that this old base version is comparable, you should perform the database tests described in the section called *Testing the Old Database* on page 7 on the standard database that came with the old version of Navision Financials that the customer is using.

### Overview of the Process of Upgrading to Multilanguage

The purpose of this process is to ensure that customizations can be used with the new Navision database. In order to ensure that, you must perform the following steps in the course of the upgrade process:

1. Get a translation database from your local Microsoft Business Solutions office that will be used to translate the code base of the customer's current database into English. You may also have to translate the code base of the old base version. For more information, ask your local Microsoft Business Solutions office.
2. Generate a translation file with names of objects and fields from the new Navision database and use it to change the language of object and field names in base version objects.

   The processes are described in the sections called *Transferring Object and Field Names* page 27, *Translating for Language Variants* on page 25 and *Restoring ObjectTime.txt* on page 27.
3. Convert old base version and current custom version using the *conv-ml* tool. This process is described in the section called *Conversion* on page 28.
4. Create captions with the *make-ml* tool on customer objects if the customer wants them to be fully multilanguage enabled. This process is described in the section called *Making Captions* on page 31.
5. Merge custom version into Navision either by moving objects in `*.fob` files or by using a merge tool. This process is described in the section called *Merging* on page 39.

### When to Use Which Database

You can open the databases and perform the steps on them in the following order:

Prepare

**Database: New base version, Navision 4.0:**

1. Create translation file as described on page 24.
2. Create object and field names file as described on page 24.
3. Export object file for the merge.

Format Old Base Version

**Database: Old base version, for example Navision Financials 2.60:**

1. Export ObjectTime file as described on page 19.
2. Create translation file, translate it into English in Navision Localization Workbench as described on page 19 and import the translation. Note that this step is only necessary if your local Microsoft Business Solutions office has not made an English database available to you.
3. Import object and field names file from new base version as described on page 25.
4. Compile all objects.
5. Import ObjectTime file as described on page 27.
6. Export object file.
7. Convert all objects with *conv-ml* tool as described on page 28.
8. Remove unique IDs as described on page 36.

Format Current Custom Version

**Database: Customer's current database:**

1. Export ObjectTime file as described on page 19.
2. Create translation file, translate it into English in Navision Localization Workbench as described on page 19 and import the translation.
3. Import object and field names file from new base version as described on page 25.
4. Compile all objects.
5. Import ObjectTime file as described on page 27.
6. Export object file. You can choose to only export the objects that you want to transfer to the new database, but you can also export all objects.
7. Convert all objects with *conv-ml* tool as described on page 28.
8. Remove unique IDs as described on page 36.

Transfer Customizations to the New Base Version.

**Database: A merge tool:**

1. Use a merge tool to merge changes from the old base version and current custom version files into new base version as described in Chapter 4, Merging on page 39.
2. Export new custom version file.

Import Customizations

**Database: New base version, Navision 4.0:**

1. Import object file that you created during the merge.

2. Compile all objects.

This creates the new custom database.

Multilanguage Enable Custom Objects

**Database: New custom version, Navision 4.0:**

1. Export ObjectTime file as described on page 23.
2. Create translation file.
3. Create captions using the *make-ml* tool as described on page 31.
4. Import the translation file and compile all objects.
5. Import ObjectTime file as described on page 27.
6. Post processing as described on page 31.
7. Export ObjectTime file as described on page 23.
8. Create translation file based on the custom objects and translate it in Navision Localization Workbench as described on page 19.
9. Import the translation file and compile all objects.
10. Import ObjectTime file as described on page 27.

The process of merging the old base version and current custom version into the new base version is described in Chapter 4, *Merging* on page 39.

## 3.2 CODE BASE LANGUAGE

The code base consists of all the name properties, variables and so on and in Microsoft Business Solutions–Navision. The code base should be in English (United States).

The actual merging process is easier when the databases are as similar as possible, for example when the old base version and the customer's current version have been transformed into a format like the one used in the new base version. If the code base in your old database has been translated into your local language, you must translate it back into English (United States) to ensure that the upgraded database can run smoothly with multilanguage.

In order to make the merge more efficient, the code base language in the old base version and the current customized version should be the same as in the new base version.

### Language Layers

Navision has language layers that Navision Financials does not have, and the code base language is English (United States) in all versions from 3.01, which it is not in most localized versions of Navision Financials.

As a rule, though, the merge is not made more difficult by the fact that objects in the new base version has more language layers than the old base version. If the difference in the number of language layers is a problem for the merge, the extra language layers in the new base version can be temporarily deleted from the new base version during the merge using the *Export Language Module* function and then imported later into the new customized version. For more information about language modules, see the manual *Application Designer's Guide*.

### Customized Terminology

If the customized database contains customized field names, for example **Item Number** rather than **Item No.**, and you want to keep these customizations, you should use the *make-ml* tool using the local language ID on both the old base version and the current customized version as described in the section called *Remove IDs* on page 36. The captions generated by the *make-ml* tool are not removed when the code base is translated.

You may want to run the old base version and the current custom version through the *make-ml* tool even though there are no terminology customizations since this will ensure that they have the same number of language layers as the new base version does.

If your add-on product is not updated for Navision, you must treat it as part of the customization.

## Prerequisites

To follow the procedures described in this chapter, you need three databases:

*The current translated*, localized, add-on integrated and customized Navision Financials database, referred to in this chapter as the customer's current database.

*A translated*, localized and add-on integrated, but not customized, Navision Financials database, referred to in this chapter as the old base version.

*The new Navision 4.0 database with the English (United States) code base*, referred to in this chapter as the new base version.

**Note**

The procedures and examples use Navision Financials 2.60 as the version number for the customer's current database and for the old base version, but you can follow the same procedures for earlier versions of Navision Financials as well as Navision Manufacturing, Commerce Portal, and Commerce Gateway.

**Note**

You must use the Navision C/SIDE for all three databases. This means that whenever the procedure tells you to open a database, you should open it in Navision.

### Language Settings During Export and Import

When you export and import Navision Financials or Navision objects in text format, you must ensure that the language settings do not change.

If you change the current language settings, Boolean properties such as `Editable=Yes/No` will create conflicts since the language of the Boolean values in object text files depend on the current language.

This means that if you export objects to a `*.txt` file from a database where the current language is German, you must ensure that the current language in the database into which you import the file is also German.

## Translation Database

In order to translate the code base in the customer's current version, you need a translation database. Contact your local Microsoft Business Solutions office to get this file.

The translation database is based on the country's localized base version, and an English version of the country's localized base version. If your local Microsoft Business Solutions office can supply your with an `*.mdb` file for the NLW, this could, for example, be called `Country Base Versions.mdb` or the equivalent.

Your local Microsoft Business Solutions office may also make an English version of the country's localized base version available to you. If not, you must create it yourself by translating your old base version into English (United States) as described in the section called *Translating the Code Base* on page 19.

### Tools

You will also need to install the following tools:

*The Navision Localization Workbench* (NLW) to manage the translation of the code base from the local language to English (United States). See the manual *NLW User's Guide*.

*A tool* for comparing and merging the two object texts.

If you use the NLW for translation, you should have access to a translation database that the local Microsoft Business Solutions office has created. For more information, see the section called *Translation Database* on page 18.

## Translating the Code Base

You are now ready to translate the code base. You must translate the code base for the customer's current database and also for the old base version if your local Microsoft Business Solutions office has not made an already translated database available to you.

The procedures use the customer's current version as an example, but the steps are the same for the old base version. You can start with the old base version or with the customer's current version as you like.

**Note**

For more information about translating files in the Navision Localization Workbench, see the manual *NLW User's Guide*.

Two options in NLW

The Navision Localization Workbench has two options when you use the AutoTranslate function:

AutoTranslation Level: Full Match

AutoTranslation Level: Text Match

This chapter only describes how to use the Full Match option during the upgrade process.

### Creating Translation File and ObjectTime.txt

Before you translate the code base in the customer's database or the old base version, you must create two files for each of the databases, one for the ObjectTime task and one for the actual translation.

To create the files before translation:

1. Open the customer's current database, compile all objects and resolve any errors.

2 Export all text strings by selecting all the objects and clicking Tools, Translate, Export.

In order to speed up the merge process, you should choose to export only those objects that make the current custom version different from the old base version. This is made easier if the customizations are well documented and perhaps marked with an entry in the **Version List** field. If you choose to do this, you must carry out a special step later on as described on page 34.

Save this file as `cust_W1W1_client_260.txt`, where `cust` means that it is the customer's database, the first `W1` means the language of the database, the second `W1` means the country that the database was localized to, `client` indicates that this file was exported with the Translate option, and `260` is the version number. For example, if the database is localized to the German market, the file is saved as `cust_DEDE_client_260.txt`.

3 Import the `ObjectTime.fob` file from the `Upgtk` folder.

4 Select the dataport 104010, ***Import/Export Object Time***, and click Run (ALT+R).

5 In the request form, on the ***Options*** tab, fill in the fields as follows:

| Field Name | Value |
|---|---|
| **File Name** | `ObjectTime.txt` |
| **Direction** | Export |

Note that `ObjectTime.txt` is a generic name that you should not use for both database. For the old base version, the file name could be `ObjectTimeOld.txt`, and for the customer's current version, the file name could be `ObjectTimeCust.txt`.

6 Click OK to create a file with information from the **Modified**, **Date** and **Time** fields from the objects.

You will import these files again when you have completed the translation.

### Full Match Translation of the Code Base

Before you translate, you must set up the correct folder structure in your working folder.

If you working folder is C:\Multilanguage\W1\NF\260, you must create a subfolder called *NLW*, which must contain two subfolders, Source and Target. Move the `cust_W1W1_client_260.txt` file to the Source folder.

To translate the code base:

**1** Open the Navision Localization Workbench and create a new project with the following properties:

| Field | Contents |
|---|---|
| **Project Name** | For example, `Customer Version.mdb`. |
| **Project Folder** | Browse to your working folder, for example C:\Multilanguage\W1\NF\260. |
| **Country** | Select your country. |
| **Source Language** | Select your local language. |
| **Target Language** | Select English (United States). |

You may see a message that you have chosen a target language that is different from the regional settings in the operating system. If you do, you must change the settings in the operating system.

**2** Click File, Add File to add the customer's version, that is the file equivalent to `cust_W1W1_client_260.txt`, as the source file. NLW opens automatically in the Source folder.

**3** In the ***Select File Type*** window, set the following properties:

| Field | Contents |
|---|---|
| **File Type** | Client file. |
| **Version Number** | The version number. |
| **Lock variables during import** | No check mark. |
| **Multilanguage** | No check mark. |

Ensure that the right file name is in the **File** field. This field is not editable.

**4** Click Translate, Auto Translate.

**5** In the ***AutoTranslate*** window, set the following properties:

| Field | Contents |
|---|---|
| **Source project** | Click Browse to select the equivalent to `Country Base Versions.mdb` translation database. |
| **Project folder** | This field is filled in automatically. |
| **Change status to** | AutoTranslated |
| **AutoTranslation level** | Full match. |
| **Translate contents of** | Entire project. |

This translates `cust_W1W1_client_260.txt` into English (United States), using the equivalent to `Country Base Versions.mdb` translation database as the source project. This translation database is described on page 18.

For more information about how to translate a customer database, see the manual *NLW User's Guide*.

**6** When the translation is complete, export the target file as described below.

Before you export the translated file, you can use the Word Count feature to check the number of words that have been translated compared to the number of words that have not been translated. To check the word count, click Tools, Word Count.

You can use the AutoTranslation, Text Match option to get closer to a complete translation. For more information about Text Match translation, see the manual *NLW User's Guide*.

Note that you should not manually translate strings from the old base version.

### Export Target File

You can export the target file from the NLW in one of two ways:

- Using the Generate File option
- Using the Export Comma Separated File option

Generate File

Use this option if your version of NLW has the **Translation Status** field in the ***Generate File*** window.

To generate a file in the NLW, follow this procedure:

**1** Click File, Generate File.

**2** In the ***Generate File*** window, enter the following information:

| Field | Contents |
|---|---|
| **Generate From File** | Enter the name of the project file, for example `cust_DEDE_client_260.txt`. |
| **Generate To File** | Enter the name of the exported text file, for example `cust_ENDE_client_260.txt`. |
| **Generate To Folder** | Click Browse and enter the path to your working folder. |
| **Translation Status** | Auto Translated. |

**3** Click OK to generate the file.

Export Comma Separated File

Use this option if your version of NLW does not have the **Translation Status** field in the ***Generate File*** window.

To export a comma separated file from the NLW, follow this procedure:

**1** Click Export, Comma Separated File.

**2** In the ***Export Comma Separated File*** window, enter the following information:

| **Field** | **Contents** |
|---|---|
| **Export** | Entire project. |
| **Export to File** | Enter the path to your working folder and enter the name of the exported text file, for example `cust_ENDE_client_260.txt`. |
| **Field Separator** | : (Colon) |
| **File Type** | Text for DOS. |
| **Include Field Names** | No. |

**3** In the **Available Fields** field, select *TargetRessourceID* and *Target Text*.

**4** In the ***Export Comma Separated File*** window, click Filter

**5** In the ***Filter*** window, set the following filters:

| **Field** | **Contents** |
|---|---|
| **Translation Status** | Select all options except Not Translated. |
| **Item Name** | Select all options. |

**6** Click OK to return to the ***Export Comma Separated File*** window.

**7** Click OK to export the comma separated file.

### Importing the Translation

You must now import the translation into the customer's database. We recommend that you first take a backup of the database.

To import a translation:

**1** Open the customer's current database.

**2** Open the *Object Designer* and select those objects that you exported to the `cust_ENW1_client_260.txt` file on page 19.

**3** Click Tools, Translate, Import to import `cust_ENW1_client_260.txt`.

**4** Compile the objects and solve any compilation issues that arise.

**5** Close the company and reopen it to display the fully or partly translated application.

You have now translated the name properties of all the objects and all the other resources in the customer's current database from the local language into English (United States). In Navision, all name properties must be in English (United States).

You must now repeat the process of translation for the old base version if your local Microsoft Business Solutions office has not made this database available to you in English.

The result of this process is that the old base version, the customer's current database and the new Navision database will all have the same code base language and you can compare them in order to transfer customizations from the old database to the new.

The translated name properties and variables will make it easier for you to compare and merge the old and the new databases. For more information about merging, see Chapter 4, *Merging* on page 39.

## Transferring Object and Field Names

If you do not have access to the translation database, or if the customer's database has not been heavily modified, you can merely import the Navision object and field names into the old base version and into the customized version. As a result, the merge will be easier as you will not have to manually merge the object and field names.

References to base version field option values, variables and user defined functions are not handled by this process, however, so they must be handled manually during the merge. You can import object and field names from the new base version without risking errors, because once an object or field is assigned a number, guidelines ensure that it is never assigned to other objects or fields.

### Creating New Base Translation File

In order to proceed with the translation and conversion, you need a translation file with all text strings from the new Navision database. You use this file first to create a file with names of objects and fields and then to be used as a memory file in connection with the *conv-ml* tool.

To create the new base translation file:

**1** Open the new base version, Navision.

**2** Click Tools, Language and select what language to work with.

As described on page 18, you must be consistent in the use of current language.

**3** In the *Object Designer*, select all objects and click Tools, Translate, Export.

**4** Save the file as `w1360client.txt`, where `w1` is the country code, `360` is the version of the application, and `client.txt` indicates the file type.

### Creating Object and Field Names File

You must now use the new base translation file to extract names of objects and fields. The target file will then be used to change the language of object and field names in base version objects.

To create a file with object and field names:

1. In Windows Explorer, copy the extraction tool from the `Upgtk` folder to the working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

   ```
   getfld.bat
   ```

   ```
   getfld2.exe
   ```

2. Go to your command prompt (or DOS) and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.

3. Enter the following command line, but use the names of the files that you created in the previous procedure:

   ```
   getfld w1360client.txt w1360names.txt
   ```

   This creates a file with all the object and field names from the new base version.

If you want to also transfer captions from the new base version to the old base version and the customer's current version, you can set this property to *1* in the file `getfld.bat`.

### Importing Object Field Names

You must now import the translation file that you generated in the section called *Creating Object and Field Names File.*

1. Open the customer's current database.

2. In the *Object Designer*, select the customized objects as described on page 19 and click Tools, Translate, Import.

3. In the ***Import Translation*** window, select the file that you created on page 24, `w1360names.txt`.

   This procedure replaces the existing names of objects and fields with the names for the new Navision database.

4. Compile all objects.

Ensure that you carry out this procedure for the customer's current database and for the old base version.

### Translating for Language Variants

Navision can run several versions of a language at the same time. For example, in Switzerland a database can have German (Standard) as one application language and German (Swiss) as another application language.

This means that you may have to perform a kind of translation between almost identical versions of a language, such as versions of English, French, German and

Spanish. For that purpose, we use two tools, *get-ml* and *renlang*, which are both run from the Command Prompt.

The *get-ml* tool extracts those text strings that are shown to the user in the application. The resulting text file is based on a Translate, Export file, but it has been cleaned of code issues such as variables, in other words a file with pure application text for the user interface. Before importing this file back into your database, you can change the words that have a different spelling in your language version compared to the standard version of the language.

The *renlang* tool replaces all occurrences of one language ID, for example A1031, with the language ID of your local language version, for example A2055.

*Get-ml* Tool

To extract the source file for the language version translation:

1. In Windows Explorer, copy the *get-ml* tool from the `Upgtk` folder to your working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

   ```
   get-ml.bat
   
   get-ml2.exe
   ```

2. Go to your Command Prompt and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.
3. Enter the following command line, but exchange the files names for the names of the files that you have created:

   ```
   get-ml w1360client.txt w1360client2.txt 1031
   ```

   where `get-ml` is the name of the tool, `w1360client.txt` is the name of the source file that you created in the section called *Creating New Base Translation File* on page 24, `w1360client2.txt` is the name of the target file and `1031` is the language ID for the standard language version, in this case German (Standard).

You can now translate the target file into your local version of the language, such as changing the wording to your local spelling and run it through the *renlang* tool to change the language ID to your language ID - for example 2055 for German (Swiss) - and import the resulting file using Translate, Import.

*Renlang* Tool

To change language ID in the translated file:

1. In Windows Explorer, copy the *renlang* tool from the `Upgtk` folder to your working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

   ```
   renlang.bat
   
   renlang2.exe
   ```

2. Go to your Command Prompt and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.

3. Enter the following command line, but exchange the files names for the names of the files that you have created:

   ```
   renlang w1360client2.txt w1360client3.txt 1031 2055
   ```

   where `renlang` is the name of the tool, `w1360client1.txt` is the name of the source file that you created with the *get-ml* tool above, `w1360client3.txt` is the name of the target file, `1031` is the language ID for the standard language version, in this case German (Standard) and 2055 is the language ID for the standard language version, in this case German (Swiss).

4. When the tool has finished its work, import the target file, `w1360client3.txt`, to the database by selecting all objects and clicking Tools, Translate, Import.

5. Compile all objects.

### Restoring ObjectTime.txt

As a side effect of translation, the Modified, Date and Time values are also changed for all objects, which is why you must import the same file that you exported in the section called *Creating Translation File and ObjectTime.txt* on page 19, for example `ObjectTimeCust.txt`.

1. Select the dataport 104010, ***Import/Export Object Time***, and click Run (ALT+R).

2. In the request form, on the Options tab, fill in the fields as follows:

| Field Name | Value |
|---|---|
| **File Name** | `ObjectTime.txt` |
| **Direction** | Import |

3. Click OK to restore the original information from the **Modified**, **Date** and **Time** fields for all objects.

Note that you must carry out this procedure for both the customer's current database and the English version of the old base version. The file name depends on the files that you created in the section called *Creating Translation File and ObjectTime.txt* on page 19.

## 3.3 MULTILANGUAGE ENABLING THE DATABASE

Since the Microsoft Business Solutions–Navision database is already multilanguage enabled, you will only have to enable customizations where customers want to use multilanguage functionality.

If the customer wants to fully use the multilanguage functionality, you can multilanguage enable customized objects and add-ons. This process is divided into four main steps:

- Identify the objects that you want to multilanguage enable.
- Use the *conv-ml* tool to create text constants and other changes to the code. This step is mandatory for all objects that will be part of the merge.
- Use the *make-ml* tool to create Caption properties for all customized objects that you want to fully multilanguage enable.
- Post-processing.

All tools related to multilanguage enabling can be found in the `Upgtk` folder on the Navision product CD.

### Identification

Before you start the process of multilanguage enabling a database, you must carefully identify those objects or parts of objects that you want to enable. This is a process very much like the process of identifying which parts of the customer’s current database to transfer to the new database.

If customized objects and customer-specific objects do not have a code base in English (United States) and they are not multilanguage enabled, they will continue to function as usual when they are merged into the multilanguage-enabled Navision database.

But if they contain instances of the C/AL functions TABLENAME or FIELDNAME, or if they read from system tables or virtual tables, these will appear in English rather than your local language.

If you run these objects through the *conv_ml* tool as described below, these references to name properties will be replaced references to captions.

You should then import them into the new Navision database in a **.fob* file. This binary format will eliminate language differences between the two code bases.

### Conversion

The first part of the actual enabling is to convert existing customized objects to cooperate with the new database.

This step is obligatory for all customized objects that you want to transfer to the new database, because the *conv-ml* tool not only creates text constants from the existing text strings, such as `ERROR`, `MESSAGE` and `CONFIRM`. It also replaces references to object names with references to object captions, and it replaces references to field names with references to field captions.

During this process, the tool creates a number of temporary folders and files, so you must ensure that you have 100 MB available space on your computer to hold these files.

**1** Go to the database that you want to enable for multilanguage, for example the customer's current database, and open the *Object Designer*.

**2** Export the objects that you want to convert by selecting them in the *Object Designer* and clicking Files, Export.

Save the file `w1260_object1.txt`, where `w1` is the country code, `260` is the version of the application, `object.txt` indicates the file type, and `1` identifies the file as the base file. Remember to save it to the right working folder, for example C:\Multilanguage\W1\NF\260.

**3** Open the new Navision 4.0 database and export a translation file by selecting all objects and clicking Tools, Translate, Export.

Save the file as `w1360_client.txt`, where `w1` is the country code, `360` is the version of the application, and `client.txt` indicates the file type.

You use this file to ensure that text constants are given the same numbers as in Navision. You should only create such a file if your Navision database uses other text constants than the W1 version of the product.

**4** In Windows Explorer, copy the conversion tools from the `Upgtk` folder to the working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

```
conv-ml.bat

conv-ml2.exe

conv-ml3.exe

conv-ml4.bat
```

**5** Go to your command prompt (or DOS) and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.

**6** Enter the following command line, but exchange the files names for the names of the files that you have created:

```
conv-ml w1260_object1.txt w1260_object2.txt w1360_client.txt 000
ENU 1033
```

where `conv-ml` is the name of the tool, `w1260_object1.txt` is the name of the base file that you created in Step 2 above, `w1260_object2.txt` is the name of the target file (the name of the base file, but version 2 rather than version 1), `w1360_client.txt` is the name of the file that you use to check the process with (a translation file as described in step 3), `000` indicates the start number of the number range for the text constants that are being converted, `ENU` is the language code of the language that the files are in and `1033` is the language ID for that language.

The parameters `w1360_client.txt`, `000`, `ENU` and `1033` are only examples. The parameters have default values so you need not enter anything in them. The default values are:

| **Parameter** | **Default Value** |
|---|---|
| Text Constant Memory File | nul<br>The tool is based on a Navision 4.0 W1 version |
| Starting Number | 0 |
| Language Code | ENU |
| Language ID | 1033 |
| Nationalchars | Empty |

This means that if you are working in English (United States), the command line could be:

```
conv-ml w1260_object1.txt w1260_object2.txt
```

If you are working in, for example, German, the command line could be:

```
conv-ml DE260_object1.txt DE260_object2.txt nul 50000 DEU 1031
```

**Note**

If you are converting an object with general customer modifications, specify a random number between 1,000,000,000 and 1,000,999,999 as the start number of the number range. You can see a list of the number ranges in Chapter 18 in the manual *Application Designer's Guide*.

7 Open the new file, `w1260_object2.txt`, in a text editor. Search for `Textnnn`, where `nnn` is the start number of the number range, to test if the tool worked as planned.

8 When the tool has created the new object text file, return to the database that you are converting, and import the `w1260_object2.txt` file by selecting all objects in the *Object Designer* and clicking Files, Import.

9 Compile all the objects.

Special Characters

The *conv-ml* tool supports a parameter that allows you to enter a list of special characters. Special characters are any characters that are not in the English (US) alphabet. The parameter is called *Nationalchars* and the command line could be:

```
Conv-ml t391.txt t392.txt nul 50000 DAN 1030 "ÆØÅæøå"
```

Where `conv-ml` is the tool, `t391.txt` is the name of the base text file, `t392.txt` is the target file, `nul` is the text constant memory file, `50000` is the start number of the number range for the text constants that are being converted, `DAN` is the language code of the language (Danish) that the files are in, `1030` is the language ID of that language and `"ÆØÅæøå"` are the special characters in that language.

## Making Captions

In a multilanguage-enabled database, the user sees only captions - not names. The *make-ml* tool creates captions and names them after the *Name* property when there are missing captions in an object. This procedure does not replace any existing captions.

To install the *make-ml* tool:

- In Windows Explorer, copy the *make-ml* tool from the `Upgtk` folder to the working folder, for example C:\Multilanguage\W1\NF\260.

  The files that you need are:

  ```
  make-ml.bat
  ```

  ```
  make-ml2.exe
  ```

To make captions by copying the contents of the *Name* property to the *Caption* property:

1. In the new Navision 4.0 application, export a translation file by selecting the objects in the *Object Designer* that you want to give captions and clicking Tools, Translate, Export. Save this file as `ENW1_client_360.txt`.

2. Go to your command prompt (or DOS) and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.

3. Enter the following command line, but exchange the files names for the names of the files that you have created:

   ```
   make-ml ENW1_360_client.txt cap_ENW1_360_client.txt 1033
   ```

   where `make-ml` is the name of the tool, `ENW1_360_client.txt` is the name of the base file that you created in step · above, `cap_ENW1_360_client.txt` is the name of the target file, and `1033` indicates the language ID that is the base application language for this database, in this case English (United States).

4. When the tool has created the new client text file, open the new Navision 4.0 application, and import the `cap_ENW1_client_360.txt` file by selecting all objects in the *Object Designer* and clicking Tools, Translate, Import.

5. Compile all objects.

## Post-Processing

The post-processing process includes testing for errors. You should go through the objects that you have converted to identify problem areas. The use of option variables is one such problem area.

You can use the *Find* tool in the `Upgtk` folder to find the places where an option variable is used with a FORMAT, STRSUBSTNO, ERROR, MESSAGE, and CONFIRM function or as a source expression in a form, report or dataport.

## Find Tool

The *Find* tool is run from the Command Prompt and must be placed in the same folder as the source file. To create the source file, select the objects in the *Object Designer* and click File, Export. Save the file as a text file, not as a Navision `*.fob` file.

In Windows Explorer, copy the *Find* tool from the `Upgtk` folder to the working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

```
findf.exe

Findfunc.bat
```

To run the *Find* tool, open the Command Prompt and change directory to the folder where the tool and the source file are placed.

Use the following syntax:

```
findfunc  <param1>   <param2>   <param3>
```

The command consists of the following elements:

| Element | Description |
|---|---|
| `findfunc` | Tool name |
| `<param1>` | The function or text string in the code that you want to search for. The tool is case sensitive, so if, for example, you want the tool to find those objects that contain CALCDATE, you must enter this parameter in capitals. |
| `<param2>` | The source file that you want to search in |
| `<param3>` | The target file, that is, the result of the search. The target file will contain a list of code strings where `<param1>` appears. |

Example

```
C:\ML\findfunc DateFormula MyObjects.txt DataFormula.txt
```

This command will produce a list of strings in the code where DateFormula appears. The source file is `MyObjects.txt` and the target file is `DataFormula.txt`.

The target file will contain a list of code strings where one of these text strings appears. You will have to investigate the DateFormula strings in the target file. If the Type is Code and the DateFormula property is set to Yes in one of the strings, the Type must be changed to DateFormula for that string.

## Option Variables

Controls in forms, reports and dataports may contain option variables, that is option fields that are not declared as variables but belong to record variables in forms, request forms and dataports, you have to set the properties on the control as shown in the following examples.

In TextBox controls:

| **Property** | **Value** |
|---|---|
| OptionString | Day,Week,Month,Quarter,Year |
| OptionCaptionML | ENU=Day,Week,Month,Quarter,Year |
| SourceExpr | PeriodType |

Note the use of the new property *OptionCaptionML*.

For an example, see report 98, ***Date Compress General Ledger***, in both Navision Financials and Navision.

In OptionButton controls:

| **Property** | **Value** |
|---|---|
| CaptionML | ENU=Day |
| ShowCaption | Yes |
| OptionValue | Day |
| SourceExpr | PeriodType |

Note the use of the new property *CaptionML*.

For an example, see report 110, ***Customer - Labels***, in both Navision Financials and Navision.

C/AL Functions

If you have used an option variable with a C/AL function that formats the option into a string, you must change the code as shown in the following examples.

Before:

```
MESSAGE(Text000,PeriodType);
```

```
Text000 : TextConst ’ENU="Period is %1"’;
PeriodType : 'Day,Week,Month,Quarter,Year';
```

After:

```
MESSAGE(Text000,SELECTSTR(PeriodType + 1,Text001));
```

```
Text000 : TextConst ’ENU="Period is %1"’;
Text001 : TextConst ’ENU="Day,Week,Month,Quarter,Year"’;
PeriodType : 'Day,Week,Month,Quarter,Year';
```

## 3.4 REPLACING BASE VERSION OBJECTS IN CURRENT CUSTOM DATABASE

In the section called *Creating Translation File and ObjectTime.txt* on page 19, you may have chosen to translate only those objects in the customer's current database that have been customized. If you did, you must now replace all base version objects in the customer's database that have not been customized with those from the English version of the old base version.

**Note**

Before you carry out this procedure, you must have an English version of both the old base version and the customized objects in the customer's current database.

The old base version must have been through all steps described in the previous sections in this chapter. Your local Microsoft Business Solutions office may have made such a fully multilanguage-enabled old base version available to you.

To replace base version objects in the customer's database:

1. In Windows Explorer, create a copy of the English version of the old base version database.
2. Open the copy of the English base version database and delete all companies.
3. Open the customer's current database and export the customized objects to a `*.fob` file, for example `cust_W1260.fob`.
4. In the copy of the English base version, import the file that you created in the previous step, `cust_W1260.fob`. When prompted, choose to open the ***Import Worksheet*** window, and in that window, click Replace All.

   You have now imported the customer-specific objects to the copy of the English version of the old base version.
5. Select all objects and export them to a `*.fob` file, for example `cust_EN260.fob`.
6. Open the customer's current database and import the file that you created in the previous step, `cust_EN260.fob`. When prompted, choose to open the ***Import Worksheet*** window, and in that window, click Replace All.

When you have completed this procedure, you have made sure that the names of functions and so on in the base version objects are the same in both the old base version and the customer's current database. This will make the merge easier.

You are now ready to prepare for the merging process as described in the next chapter.

# Chapter 4

# Customizing the New Standard Objects

Any customizations that have been implemented in the customer’s old database must be implemented in the new database. You must identify and correct any illegal locktable calls that are made in the customer’s database. We also recommend implementing some changes that will facilitate migrating to the SQL Server Option for Microsoft Business Solutions–Navision.

The chapter contains:

- 4.1 Customizing the New Standard Objects
- 4.2 Locating Illegal Locktable Calls

## 4.1 CUSTOMIZING THE NEW STANDARD OBJECTS

The next stage in the upgrade process involves identifying any customizations that have been made in the customer's current database and redesigning a database in Microsoft Business Solutions–Navision 4.0 so that it can accommodate these changes.

The procedures described in this chapter are the fastest and safest way to upgrade a customer's installation. There are other ways of upgrading, but we do not recommend them.

**Note**

The procedures in this chapter refer to the customer's current database as a Navision Financials 2.60 database, but you can perform the same steps with earlier versions of Navision Financials and with User Portal, Commerce Portal, Commerce Gateway, Navision Manufacturing and Navision Attain.

### Identifying Customized Objects

Start by identifying any customer-specific changes that have been made to the customer's old standard application. You then implement those changes in a new standard Navision 4.0 application.

In order to identify the customizations that the customer has made, you must have an old standard database to compare with the user's current database. This old base version must be the one that you had from your local office in English or that you translated yourself according to the procedures in Chapter 3.

To compare the objects in the customer's old database with the standard database that came with the old version that the customer is using, and to implement the necessary changes in a new standard Navision 4.0 database, follow this procedure:

1. Create a backup of the customer's current version of Navision Financials from a client computer and then uninstall Navision Financials on that computer.
2. Install the newest version of Navision on the client computer.
3. Restore the objects from the backup of the old customized Navision Financials database into the new version of Navision. Do not restore the data now.
4. Restore the backup of the old standard Navision Financials database that you have just created into the new version of Navision.

### Remove IDs

Since version 3.01, variables and text constants have unique IDs but Navision Financials 2.60 do not. When you import objects from Navision Financials 2.60 into Navision 4.0, these objects are automatically converted so that variables and text constants have unique IDs.

However, the numbering of unique IDs that C/SIDE assigns to imported objects do not match the numbering of the same variables in the new base version, because C/SIDE has no information about how the variables are numbered in the new base version.

To eliminate this problem, you can remove the IDs from all objects in the new customized database and from the English version of the old base version, import them again and let C/SIDE assign correct IDs.

**Note**

The procedures in this chapter refer to the customer's current database as a Navision Financials 2.60 database, but you can perform the same steps with earlier versions of Navision Financials and with User Portal, Commerce Portal, Commerce Gateway and Navision Manufacturing.

To remove IDs, follow this procedure:

1. In the version of the customer's current database that you created above, open the *Object Designer* and export all objects to a text file by clicking File, Export.

   Save the file as, for example, `ObjectsID_cust.txt`.

2. In Windows Explorer, copy the *remid* tool from the `Upgtk` folder to your working folder, for example C:\Multilanguage\W1\NF\260. The files that you need are:

   ```
   remid.bat
   ```

   ```
   remid2.exe
   ```

3. Go to your Command Prompt and change directory to the working folder, for example C:\Multilanguage\W1\NF\260.

4. Enter the following command line, but exchange the files names for the names of the files that you have created:

   ```
   REMID ObjectsID_cust.txt ObjectsNoIDs_cust.txt
   ```

   where `REMID` is the name of the tool, `ObjectsID_cust.txt` is the name of the source file that you created in Step 1 above and `ObjectsNoID_cust.txt` is the name of the target file.

5. When the tool has finished its work, you can use the target file during the merge process.

You can only use this tool to remove IDs on text files that contain IDs.

You must now repeat this procedure for the English version of the old base version. When you have completed the procedure, you must have two files with the objects from the customer's current database and the English version of the old standard database, respectively. You must generate these two files before the merging process, because removing the IDs makes the merge must easier.

**Attention**

Do not remove IDs from the new base version.

When you have completed the merge process described below and you import the merged objects into the new Navision 4.0 database, C/SIDE will assign correct unique IDs to the customized objects.

## Updating to Dynamic Dimensions

When you are upgrading from any version earlier than 3.01, you must update customizations that use the static dimension ***Department*** table and ***Project*** table to use dynamic dimensions.

**Note**

You must only carry out the procedures in this chapter if you are upgrading from Navision Financials, User Portal, Commerce Portal, Commerce Gateway and Navision Manufacturing.

You update to dynamic dimensions by performing the following tasks in text format:

1. Rename fields relating to the dimension tables, so they reflect the dynamic properties of the new dimension solution.

   This includes both temporary buffer fields, FlowFilter fields and the ordinary code fields. Note that the length of the field is now 20.

2. Replace `TableRelation=Department` with `TableRelation="Dimension Value".Code WHERE (Global Dimension No.=CONST(1))`
3. Replace `TableRelation=Project` with `TableRelation="Dimension Value".Code WHERE (Global Dimension No.=CONST(2))`
4. Change Flowfilters on the old department and project fields according to the following examples from the ***Currency*** table:

```
{ 21  ;   ;Department Filter   ;Code10       ;FieldClass=FlowFilter;
TableRelation=Department }

{ 21  ;   ;Global Dimension 1 Filter;Code20   ;FieldClass=FlowFilter;
TableRelation="Dimension Value".Code WHERE (Global Dimension
No.=CONST(1)); CaptionClass='1,3,1' }


{ 22  ;   ;Project Filter      ;Code10       ;FieldClass=FlowFilter;
TableRelation=Project }

{ 22  ;   ;Global Dimension 2 Filter;Code20   ;FieldClass=FlowFilter;
TableRelation="Dimension Value".Code WHERE (Global Dimension
No.=CONST(2)); CaptionClass='1,3,2' }
```

```
{ 4   ;   ;Department Code     ;Code10         }

{ 5   ;   ;Project Code        ;Code10         }


{ 4   ;   ;Global Dimension 1 Code;Code20
;TableRelation="Dimension Value".Code WHERE (Global Dimension
No.=CONST(1)); CaptionClass='1,1,1' }

{ 5   ;   ;Global Dimension 2 Code;Code20
;TableRelation="Dimension Value".Code WHERE (Global Dimension
No.=CONST(2)); CaptionClass='1,1,2' }
```

## Comparing Versions

You must now use a file comparison tool such as the Navision Developer's Toolkit to compare the files you created in the previous procedure and find and view the object changes in detail. Make a log of these changes.

**Note**

For more information about comparing and merging objects, see the separate documentation for the Navision Developer's Toolkit. This documentation is located on the Navision Tools CD.

When you have compare the old base version and the customer's current database, you must evaluate the changes that have been made.

You must decide which changes you want to duplicate in the standard database that comes with Navision. Some changes may already be part of this new standard Navision database and can be skipped during the upgrade. To make the evaluation easier, you should refer to any documentation that describes the changes that have been made to the customer's database.

Be careful to note any duplicated object names and duplicated field names within an object because these must be renamed. You must also note any duplicated function numbers and control numbers because these must be renumbered.

## Merging

You must now merge the customized objects into the new Navision database. For this process, you should use the Navision Developer's Toolkit.

**Note**

For more information about merging objects, see the separate documentation for the Navision Developer's Toolkit. This documentation is located on the Navision Tools CD.

In the Navision Developer's Toolkit, you must import object files from the old standard version, the customer's current version and the new Navision database. For this process you need the two files that you created in the section called *Remove IDs* on page 36. You also need an object file with all objects from the new Navision database.

**Attention**

Do not remove IDs from the new base version.

Merge the customizations into the new Navision database, which in the Navision Developer's Toolkit is called *the New Custom version*, and correct any errors that the tool finds.

When you have completed the merge, export the New Custom version to a text file, import it into the new Navision database and compile all objects.

You now have a new customized database. This database contains all the objects that have been customized to include the changes that were made in the old database.

These customized objects and all the other Navision 4.0 objects must be exported and saved so that you can import them again after you have restored the backup of the customer's old database into a new database in Navision.

To export the objects:

1. Open the new customized Navision database. Open the *Object Designer* and compile all the objects.

   It should now be possible to compile all the objects without encountering any errors.

2. Export all the objects in the new customized Navision database, and name the object file, for example, `objects.fob`.

3. Make a backup file of the demonstration company and name it, for example, `demo.fbk`.

## 4.2 LOCATING ILLEGAL LOCKTABLE CALLS

You must identify and correct any illegal locktable calls that are made in the customer's database.

**Note**

You must only carry out this procedure if you are upgrading from Navision Financials 2.00 or 2.01.

To locate any illegal locktable calls that may exist in the customer's database:

**1** Open the new customized Navision database. Open the *Object Designer* and export all the objects to a text file called `objects.txt`.

**2** Locate the `Findlock.exe` file in the `Upgtk` folder on the product CD and copy it to a suitable folder on your PC.

**3** Open a command prompt. Set the folder you just copied the `Findlock.exe` file to as the working folder and run:

```
findlock <objects.txt >locks.txt
```

to find all occurrences of possibly illegal `LOCKTABLE` calls.

The `locks.txt` file that is generated will contain a list of all the `LOCKTABLE` calls that might be illegal and the objects that contain these calls.

**4** Open the `locks.txt` file and change the relevant code.

To ensure that the application can run on both Navision Database Server and SQL Server, you must change some of the code. Similar code changes have already been made to the base application in the new Navision that the customer is going to use from now on.

You must check the code to ensure that every `LOCKTABLE` call listed in the file (where the object containing the call has been customized, and where the first argument is `FALSE` or the second argument is `TRUE`) is not made when the application is running on SQL Server. Such calls are not supported on SQL Server.

In addition, if the second argument for `LOCKTABLE` is `TRUE`, you are implementing an optimistic locking scheme. You cannot do this when you are using the SQL Server Option for Navision. You must lock the records before you read them.

The following table illustrates the type of code changes you must make to ensure that your application can run on both servers:

**Works Only with Microsoft Business Solutions Server**

```
IF Rec.FIND(‘-‘) THEN
  REPEAT
  UNTIL Rec.NEXT = 0;
Rec.LOCKTABLE(TRUE,TRUE);
IF Rec.FIND(‘-‘) THEN
  REPEAT
    Rec.MODIFY;
  UNTIL Rec.NEXT = 0;
```

**Works with Both Server Types**

```
IF Rec.RECORDLEVELLOCKING THEN
 Rec.LOCKTABLE;
IF Rec.FIND(‘-‘) THEN
  REPEAT
  UNTIL Rec.NEXT = 0;
IF NOT Rec.RECORDLEVELLOCKING THEN
 Rec.LOCKTABLE(TRUE,TRUE);
IF Rec.FIND(‘-‘) THEN
  REPEAT
    Rec.MODIFY;
  UNTIL Rec.NEXT = 0;
```

**5** Check whether the customer’s application code uses any of the following virtual tables: ***Session***, ***Database File***, and ***Table Information***.

Some of the fields in these tables that are available on Navision Database Server Server are not supported on SQL Server.

You have now ensured that the database does not contain any illegal locktable calls.

For more detailed information about how locking works on the different servers and about these virtual tables and the fields that are available on SQL Server, see the manual *Application Designer’s Guide*.

## Chapter 5

# Upgrading C/SIDE on the SQL Server Option

This chapter describes how to upgrade earlier versions of C/SIDE to the newest version. The procedures described in the chapter only apply to the SQL Server Option for Microsoft Business Solutions–Navision. The chapter contains the following sections:

## 5.1 UPGRADING C/SIDE

If you are using the SQL Server Option for Navision or the SQL Server Option for Navision Financials, there are some things that you must be aware of when you upgrade to the newest version of the SQL Server Option. The internal data formats used by the program depend on the version number of the executables. In order to update these internal data structures you must follow the special procedures described in this chapter.

The following sections describe how to upgrade C/SIDE, that is to say, the database and the executables in the program. This should not be confused with the application. The procedures for upgrading the application are described in Chapter 6 on page 49.

### Upgrading Navision Financials 2.50 to Navision Financials 2.60 C

If you are upgrading from any of these versions of the SQL Server Option for Navision Financials, you must:

1. Make a complete Navision Financials backup of the database.
2. Uninstall all the clients.
3. Install the new SQL Server Option for *Navision 4.0* clients.
4. Create a new database on SQL Server. You must be running on SQL Server 2000. For information about collation types and identifier conversion, see the manual *Installation & System Management: Microsoft* SQL Server Option.
5. If you have to upgrade to SQL Server 2000, you might need to create login accounts on the new server
6. Restore the Navision Financials backup into the new database.
7. Perform the database upgrade procedures described in Chapter 6.

**Note**

Restoring a Navision Financials/Attain backup of a SQL Server database can take a considerable amount of time depending on the size of the database.

We recommend that you make a SQL Server backup of your database before performing the database upgrade procedures described in Chapter 6.

### Upgrading Navision Financials 2.60 D to Navision Financials 2.60 G

If you are upgrading from any of these versions of the SQL Server Option for Navision Financials, there are two ways in which you can upgrade your database.

You can:

1. Make a complete Navision Financials backup of the database.

2. Uninstall all the clients.
3. Install the new SQL Server Option for *Navision 4.0* clients.
4. Create a new database on SQL Server. You must be running on SQL Server 2000. For information about collation types and identifier conversion, see the manual *Installation & System Management: Microsoft* SQL Server Option.
5. Restore the database backup into the new database.
6. Perform the database upgrade procedures described in Chapter 6.

**Alternatively you can:**

1. Install one Navision client. This must be either a *Navision Attain 3.01 B* or a *Navision Attain 3.10* client.
2. If you are running on SQL Server 7.0, you must upgrade to SQL Server 2000. For more information, see the Copy Database Wizard topic in SQL Server Books Online.
3. Connect to your SQL Server database with the new client. When you access the SQL Server database for the first time with the new clients, you are prompted to convert the database. Confirm your choice to convert the database.
4. Uninstall the Attain client, you installed in step 1 above.
5. Install a SQL Server Option for *Navision 4.0* client.
6. Connect to your SQL Server database with this new client. When you access the SQL database for the first time with one of the new clients, you are prompted to convert the database. Convert the database.

We recommend that you make a SQL Server backup of your database before performing the database upgrade procedures described in Chapter 6.

**Note**

Restoring a Navision Financials/Attain backup of a SQL Server database can take a considerable amount of time depending on the size of the database. Converting the database is much quicker.

We recommend that you make a SQL Server backup of your database before performing the database upgrade procedures described in Chapter 6.

### Upgrading Navision 3.01 to Navision 3.70

If you are upgrading from any of these versions of the SQL Server Option for Navision you must:

1. Install a SQL Server Option for *Navision 4.0* client.

2 Connect to your SQL Server database with this new client. When you access the SQL database for the first time with one of the new clients, you are prompted to convert the database. Confirm your choice to convert the database.

### Modifications to Translated System Tables

When you use a new version of Navision to open a SQL database which was created with an older version of Navision, you may get this error:

"The BLOB size field on the Object table does not exist in the SQL Server table or view"

or similar errors about fields that do not exist, and you will not be able to open the database.

This problem occurs because older versions of Navision translated system tables to the local language. When a Navision client connects to the SQL server, it looks up some system tables by name. If these had been translated to, for instance German, then Navision could not read these tables, and could not connect.

The problem is with these tables in Navision:

1. Object (table 2000000001)
2. Session (table 2000000009)
3. Database File (table 2000000010)

The problem will also occur in the field identifiers in table $ndo$dbproperty, which refers to fields in the tables mentioned above.

Notice that the Object table is implemented as a physical table on SQL server, while the Session and Database Files tables are implemented as views on the SQL server.

To be able to connect to the old database with the new version client, you must translate these tables and fields back to English. Use a SQL tool like Enterprise manager to do this. This process cannot be completely automated because the exact process depends on unpredictable factors such as exactly which version of Navision was used to create the database, including which country version, possible customization and other factors. So some manual steps are required. Follow the steps below to resolve the problem:

1. Create a reference database:

   Create a new SQL database using the newest version of the Navision client. You will use this database to copy the correct values into your customer's database. In the next steps this database will be referred to as the reference database.

2. Copy $ndo$dbproperty.identifiers:

   In Enterprise Manager, open the table $ndo$dbproperty in the reference database. Copy the content of the field identifiers, and paste it into the same field in your customer's database.

**3** Change the Object table definition:

From Enterprise Manager, design the Object table in your customer’s database. Copy each field name from the reference database to your customer’s database, overwriting the existing field names. Then close and save the table. After that, make sure that the name of the table is "Object". Rename the table if necessary. Note, that you cannot just copy this table from the reference database because the table contains data which would not be copied.

**4** Copy the Session and Database File views from the reference database:

In the Reference database, highlight the two views Session and Database File. Right-click on the selected views, then click "All Tasks", then Generate SQL Script. Click OK to use the default options, then save the script to a file. Open SQL Query Analyser and select your customer’s database. Open the script and execute it to create the views (existing views will be overwritten).

You have now updated the system tables.

# Chapter 6

# Upgrading the Old Database

This chapter describes the procedures needed to convert the existing data to into the 4.0 database structure.

## 6.1 OVERVIEW

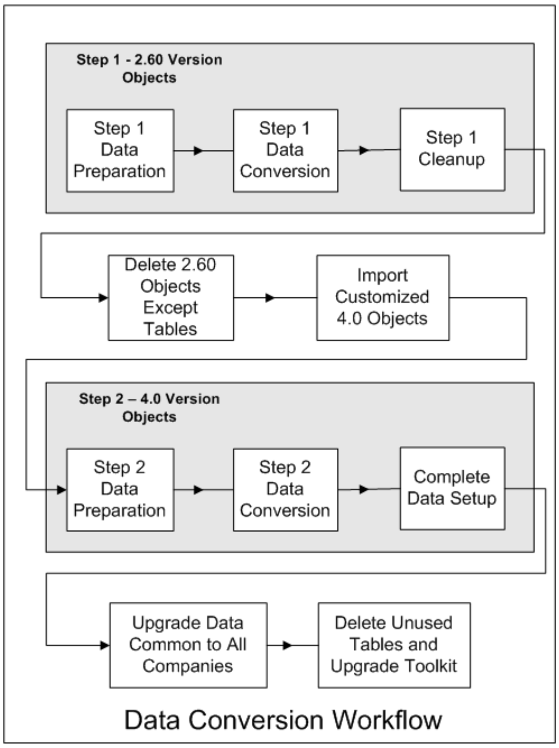
### Converting the Old Database

To upgrade the database, you need to:

- Convert the data to work with the 4.0 version
- Test the data in the 4.0 version
- Install the new 4.0 clients.

Data conversion tools are provided to convert the existing data with the old version table and field structure so it can function with the new table and field structure in the new version. It is only the table objects and table data that are being modified by the data conversion tools. All other objects such as forms, reports, codeunits and dataports are "upgraded" as part of the customization merge processes. The new customized forms, reports, dataports and codeunits are exported after the merge and imported during the data conversion. For more information on this process see Chapter 4, *Customizing the New Standard Objects* on page 35 and the documentation for the Developers Toolkit.

The process involved in performing the data conversion is outlined in the following diagram:


Step 1 - 2.60 Version Objects
Step 1 Data Preparation
Step 1 Data Conversion
Step 1 Cleanup
Delete 2.60 Objects Except Tables
Import Customized 4.0 Objects
Step 2 – 4.0 Version Objects
Step 2 Data Preparation
Step 2 Data Conversion
Complete Data Setup
Upgrade Data Common to All Companies
Delete Unused Tables and Upgrade Toolkit
Data Conversion Workflow

The Step 1 and Step 2 terminology is used as follows:

- *Step 1* - Any tasks that involve changes to the existing data with the old version objects.
- *Step 2* - Any tasks that involve changes to the existing data with the new version objects.

As you can see from the diagram above, Step 1 and Step 2 are made of smaller sub-steps. There are also sub-steps that must be performed between Step 1 and Step 2 and after Step 2. The sub-steps in the diagram above can be further broken down into tasks.

The list of tasks that must be performed during and after the data conversion process and the relevant section in this chapter describing them are shown below:

**Step1 Data Preparation**

- Task 1 - Create the New 4.0 Database
- Task 2 - Import Upgrade Step 1 Objects
- Task 3 - Data/Object Changes Prior to Step 1

**Step 1 Data Conversion**

- Task 4 - Step 1

**Step 1 Cleanup**

- Task 5 - Rename Old Tables

**Change Objects**

- Task 6 - Change Objects
- Task 7 - Import All Customized 4.0 Objects
- Task 8 - Compile Imported Objects

**Step 2 Data Preparation**

- Task 9 - Import Upgrade Step 2 Objects
- Task 10 - Data/Object Changes Prior to Step 2

**Step 2 Data Conversion**

- Task 11 - Step 2

**Setup Completion**

- Task 12 - Complete Data Changes After Step 2
- Task 13 - Initialize the Company

**6.3 Upgrading Data Common to All Companies**

- Roles and Permissions
- Database Key Groups
- Windows Logins

**6.4 Deleting Unused Tables and Upgrade Toolkit**

**6.5 Testing the Database**

**6.6 Installing the Clients**

**Note**

Unless otherwise stated, references to Navision 2.60 also apply to Navision Financials 2.65 Commerce Gateway and Navision Financials 2.65 Commerce Portal.

## Quick Guides

This manual contains the procedures to upgrade the database for all versions prior to 4.0. The Upgrade Toolkit also contains quick guides related to each individual version. Each quick guide lists the actions you must perform when upgrading from a specific version. The following table lists the versions and the related quick guides:

| Version | Quick Guide |
|---|---|
| 2.00-2.60 | `Upgrading Microsoft Business Solutions - Navision 2.00/2.01/2.60` |
| Navision Manufacturing 2.60 | `Upgrading Microsoft Business Solutions - Navision 2.60 MFG` |
| Commerce Portal 2.65 | `Upgrading Microsoft Business Solutions - Navision 2.65 CP` |
| Commerce Gateway 2.65 | `Upgrading Microsoft Business Solutions - Navision 2.65 CG` |
| 3.01 | `Upgrading Microsoft Business Solutions - Navision 3.01` |
| 3.10 | `Upgrading Microsoft Business Solutions - Navision 3.10` |
| 3.60 | `Upgrading Microsoft Business Solutions - Navision 3.60` |
| 3.70 | `Upgrading Microsoft Business Solutions - Navision 3.70` |

## Prior Version Information

Where possible, references to prior versions are made in this chapter and the appendixes providing further information on the data conversion process.

### Commerce Portal and Commerce Gateway

Unless otherwise stated, tasks with a Navision 2.60 reference also need to be performed for Navision Financials 2.65 Commerce Gateway and Navision Financials 2.65 Commerce Portal.

### User Portal

The user portal functionality is not included in the 4.0 version. During the data conversion process the existing data is not removed from the old tables so that the customer can use the data again when an alternative solution becomes available.

### Navision Advanced Distribution

There are no data conversion tools to directly upgrade a customer from Navision Advanced Distribution to Navision 4.0. You must upgrade the customer from NAD 2.60 to Navision 3.60 first and then upgrade from Navsion 3.60 to Navision 4.0. For information on how to upgrade NAD 2.60 to Navision 3.60, see the documents and files included in the NAD Upgrade Documentation folder in the toolkit.

## 6.2 UPGRADING COMPANY-SPECIFIC DATA

This part of the upgrade procedure involves upgrading the customer's database to Microsoft Business Solutions–Navision 4.0. The first task is to restore the backup of the customer's database that was made earlier.

### Step1 Data Preparation

#### Task 1 - Create the New 4.0 Database

To restore the database:

1. Ensure that you have made the preparations and testing described in Chapter 2, page 5.
2. Create a new database in Navision 4.0.

   We recommend that the new database is large enough to contain the objects in the `object.fob` plus twice the size of the existing data in the old customer database. You can see an example of the size of an old and a new database below:

   **Old Database**

   (old objects = 35 MB) + (existing data = 500 MB) = 535 MB

   **New Database**

   (new objects = 60 MB) + (2 x (existing data = 500 MB)) = 1095 MB

   To see the size of data in the old database, run report 104001, ***Table Information***.
3. Restore the backup of the customer's old database that was saved as `data.fbk` into this new database. This is the backup that you made in step 3 on page 6.

**Note**

For Navision Database Server versions 2.60 and later, you do not have to use the backup and restore method. You can simply make a copy of the old database and then open this copy with the new 4.0 client. You should convert the database when prompted to do so and then expand the size of the database as recommended in the backup and restore procedure. For large databases, this alternative method will normally be quicker than the backup and restore method.

### Task 2 - Import Upgrade Step 1 Objects

- Open the *Object Designer* and import the relevant fob file for the version from the table below. This file is located in the `Upgtk` folder on the product CD.

| Version | Fob file |
|---|---|
| 2.00-2.60 | `Upgrade260400.1.fob` |
| Navision Manufacturing 2.60 | `Upgrade260400.1.MFG.fob` |
| Commerce Portal 2.65 | `Upgrade265400.1.CP.fob` |
| Commerce Gateway 2.65 | `Upgrade265400.1.CG.fob` |
| 3.01 | `Upgrade301400.1.fob` |
| 3.10 | `Upgrade310400.1.fob` |
| 3.60 | `Upgrade360400.1.fob` |
| 3.70 | `Upgrade370400.1.fob` |

### Task 3 - Data/Object Changes Prior to Step 1

#### Empty Table Data

Navision 2.00-3.10

- Before upgrading the database you must empty the following tables:

| Table ID | Version | Table Name |
|---|---|---|
| 40 | 2.00-3.10, 2.60 NM | ***Item Price Change*** |
| 83 | 2.00-3.10, 2.60 NM | ***Item Journal Line*** |
| 89 | 2.00-3.10, 2.60 NM | ***BOM Journal Line*** |
| 246 | 2.00-3.10, 2.60 NM | ***Requisition Line*** |
| 5766 | 3.01-3.10 | ***Warehouse Activity Header*** |
| 5768 | 3.01-3.10 | ***Cross Dock Opportunity*** |
| 99000767 | 2.60 NM, 3.01 | ***Capacity Journal Line*** |
| 99000796 | 2.60 NM, 3.01 | ***P.O. Output Journal Line*** |
| 99000813 | 2.60 NM, 3.01 | ***P.O. Consump. Journal Line*** |
| 99000828 | 2.60 NM | ***Planning Line*** |
| 99000853 | 2.60 NM | ***Inventory Profile*** |

### Rounding Precisions for Additional Reporting Currency

All Versions

If you have selected an Additional Reporting Currency, you must ensure that that ***Currency*** table contains a record for that currency. You must also ensure that the additional reporting currency record does not have 0 in either the **Unit-Amount Rounding Precision** or **Amount Rounding Precision** fields.

### Negative Bin Inventory

Navision 2.00-3.60 and Navision Manufacturing 2.60

Before upgrading the database, ensure that none of the locations where bins are used have negative inventory. When you run the first codeunit of the upgrade, it will check the inventory levels per bin in the locations that use bins and show an error if it encounters negative inventory. This is because negative inventory is not supported in versions 3.70 and later.

**Note**

To avoid performance problems while upgrading, add the following key to the ***Item Ledger Entry*** table: **Location Code**, **Bin Code**.

### Invalid Requisition Methods

Navision 2.60-3.60 and Navision Manufacturing 2.60

Table 5408, ***Requisition Method***, is not included in versions 3.70 and later. The fields **Reordering Policy**, **Calc. with Inventory** and **Manufacturing Policy** are now part of table 27, ***Item***, and table 5700, ***Stockkeeping Unit***, which might have caused incorrect requisition methods.

To get a list of the incorrect requisition methods:

- Select report 104026, ***Invalid Requisition Methods***, and click Run (ALT+R).

The settings of a requisition method are considered to be incorrect for one or more of the following reasons:

Reordering Policy = Discrete Lot Size

Calc. below Reorder Point = FALSE

Calc. with Inventory = FALSE in combination with
Reordering Policy = Fixed Reorder Qty. or
Reordering Policy = Maximum Qty.

Calc. with Inventory = TRUE in combination with
Reordering Policy = Order or
Reordering Policy = " ".

Any of the above mentioned settings would prevent the upgrade to Navision 4.0 from running.

### Inconsistent Reservations

Navision 2.00-2.60

A reservation is inconsistent if, for example, a sales order reserves an item across locations. Furthermore, entries where the reservation status is not *Open* are also considered inconsistent.

Inconsistent entries are deleted during the upgrade and you will have to create them again manually in accordance with the new reservation system. Note that in Navision 4.0 you should use transfer orders rather than reserve items across locations.

To run the ***Inconsistent Reservations*** report:

- Select report 104025, ***Inconsistent Reservations***, and click Run (ALT+R).

### Item Tracking

**Note**

You do not need to do this task if the existing standard key, **Item No.**, **Lot No.**, **Prod. Order No.**, **Serial No.** has been enabled in the customer database.

Navision 2.00-2.60 and Navision Manufacturing 2.60

To avoid performance problems in the item tracking upgrade from Navision Financials 2.00, 2.01, 2.60 and Navision Manufacturing 2.60, we recommend creating these 2 new keys in table 32, ***Item Ledger Entry***:

Item No., Serial No.

Item No., Lot No.

### Cleanup of Date Compressed Item Ledger Entries

Navision 2.00-2.60 and Navision Manufacturing 2.60

You may experience problems with the invoiced quantity in upgraded item ledger entries that were originally date compressed. The redistribution of this invoiced quantity may not be correct for every customer due to the different options for the way date compression can be performed. As a result, you may need to perform some cleanup of date compressed item ledger entries before you perform the upgrade and the upgraded entries must be carefully tested after the upgrade.

**Note**

Depending upon the retain fields that were used in the date compression, it may be necessary to adjust codeunit 104049, ***Upgrade 4.00 Step 2 Value***, before you perform the upgrade. You should comment out some of the filters set in codeunit 104049 that follow the comment:

```
// The following filters can be set according to retain fields used
// in the date compression batch job if different retain options are
// set in date compression calls, filters need to be removed.
```

The more filters that are removed, the less exact the distribution will be, but removing filters also increases the chances that the appropriate entries will be handled.

Navision Manufacturing 2.60

### Planning Worksheets

In versions 3.01 and later, the ***Planning Worksheet*** form is no longer based on separate planning worksheet tables but is based on the existing requisition worksheet tables. Table 99000826, ***Planning Wksh. Template***, and table 99000827, ***Planning Wksh. Name***, have been deleted from versions 3.01 and later. The upgrade toolkit will delete all existing entries in these to tables. If the customer has created their own planning worksheet templates and names and wants to continue using them, they will need to be set up again after you have run codeunit 104048, ***Upgrade 4.00 Step 2***. For more information on how to set them up see Task 12 - Complete Data Changes After Step 2, *Planning Worksheets* on page 70.

A report is included in the upgrade toolkit so that you can print out the existing setup before you perform the upgrade. To print a list of the planning worksheet templates and the related planning worksheet names for each template:

- Open the *Object Designer*, select report 104027, ***Mfg. Journals and Plan. Wkshts***, and click Run (Alt+R).

Navision Manufacturing 2.60 and Navision 3.01

### Manufacturing Journals

In versions 3.10 and later, the ***P.O. Output***, ***P.O. Consumption*** and ***Capacity Journal*** forms are no longer based on separate ***P.O Output***, ***P.O Consumption*** and ***Capacity Journal*** tables but are based on the existing item journal tables. As a result, the following tables have been deleted from versions 3.10 and later:

| Table ID | Table Name |
|---|---|
| 99000766 | ***Capacity Journal Template*** |
| 99000768 | ***Capacity Journal Batch*** |
| 99000795 | ***P.O. Output Journal Template*** |
| 99000797 | ***P.O. Output Journal Batch*** |
| 99000812 | ***P.O. Consump. Journal Template*** |

| Table ID | Table Name |
|---|---|
| 99000814 | ***P.O. Consump. Journal Batch*** |

The upgrade toolkit will delete all existing entries in these tables. If the customer has created their own templates and batches for these journals and wants to continue using them, they will need to be set up again after you have run codeunit 104048, ***Upgrade 4.00 Step 2***, for the upgrade (for more information on how to set them up see Task 12 - Complete Data Changes After Step 2, *Manufacturing Journals* on page 70).

A report is included in the upgrade toolkit so that you can print out the existing setup before you perform the upgrade. To print a list of the manufacturing journal templates and the related journal batches for each template:

- Open the *Object Designer*, select report 104027, ***Mfg. Journals and Plan. Wkshts***, and click Run (Alt+R).

### Subcontracted Purchase Lines

Navision 2.00-3.01

You must ensure that the quantity of items received is the same as the quantity invoiced for all Item purchase lines that contain a **Prod. Order No.** The upgrade toolkit checks this during codeunit 104045, ***Upgrade 4.00 Step 1,*** and it will not be able to complete codeunit 104045 until you have invoiced the receipts posted for these subcontracted purchase lines.

Navision Manufacturing 2.60

You must ensure that all item purchase lines containing a **Prod. Order No.** have not been partially invoiced. This means either completely invoiced or not invoiced at all. The upgrade toolkit checks this during codeunit 104045, ***Upgrade 4.00 Step 1,*** and it will not be able to complete codeunit 104045 until you have completely invoiced these subcontracted purchase lines.

In addition, you must ensure that all Item purchase lines containing a **Prod. Order No.** and **Lot No.** or **Serial No.** information are completely received and invoiced. The upgrade toolkit checks this during codeunit 104045, ***Upgrade 4.00 Step 1***, and it will not be able to complete codeunit 104045 until there are no subcontracted purchase lines with **Lot No.** or **Serial No.** information.

### Production Orders with Released Order No. Conflicts

Navision 3.01

The codeunit searches for conflicting numbers in the production orders. A number conflict occurs in the following situations:

- At least one finished production order has a **Released Order No.** that matches the number of a production order with status Released.

- At least two finished production orders have the same **Released Order No.**

**Note**

One way of resolving these conflicts is to renumber the **Released Order No.** on the finished production order that has the conflict.

### Working Calendars

Navision 3.01

The *Working Calendar* in Service Management in version 3.01 is replaced by *Base Calendars* in the 4.0 version. As a result, all data in the working calendar is deleted when you run codeunit 104045, ***Upgrade 4.00 Step 1***. You may wish to make a copy of the holidays you have set up in the working calendar so you can set them up again in the new base calendar.

**Note**

The existing working calendar data is not transferred but must be set up again. The setup time should be since the new base calendars provide features that allow you to quickly set up non-working days as recurring on both a weekly and annual basis.

### Service Pricing

Navision 3.01-3.10

1. Post all services orders that have service pricing in it before you start to upgrade.

   If this is not done, the user will not be able to calculate the service pricing afterwards. The upgrade toolkit checks the service orders during codeunit 104045, ***Upgrade 4.00 Step 1***, and it will not be able to complete codeunit 104045 until you have posted service orders with service pricing.

   Service invoice lines created by the Create Service Price Adjustment function will be upgraded as normal service invoice lines.

2. Before you upgrade, you must make your own documentation of the setup in service pricing by making a report that shows the contents of table 5922, ***Price-Adjusted Serv. Inv. Line***, and table 5926, ***Serv./Cust. Price Gr. Pricing***. This is because data in the service pricing tables (5921-5926) is deleted when you run codeunit 104045, ***Upgrade 4.00 Step 1***. You will need the documentation for completing the setup of service pricing in the 4.0 version. See *Task 12 - Complete Data Changes After Step 2*, Service Pricing on page 70.

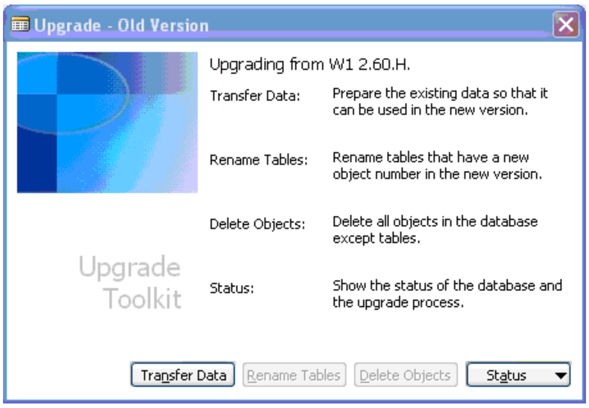
3. If the customer wishes to keep a history of the service item line pricing in posted service orders, you should make a report that shows the contents of table 5923, ***Service Item Line Pricing***. This is because data in the service pricing tables (5921-5926) is deleted when you run codeunit 104045, ***Upgrade 4.00 Step 1***.

## Step 1 Data Conversion

### Task 4 - Step 1

When you are upgrading from any version:

**1** Open the *Object Designer*, select form 104001, ***Upgrade - Old Version***, and click Run (ALT+R).



Upgrading from W1 2.60.H.

| | |
|---|---|
| Transfer Data: | Prepare the existing data so that it can be used in the new version. |
| Rename Tables: | Rename tables that have a new object number in the new version. |
| Delete Objects: | Delete all objects in the database except tables. |
| Status: | Show the status of the database and the upgrade process. |

**2** Click Transfer Data.

This runs codeunit 104045, ***Upgrade 4.00 Step 1***. You must run this codeunit for each company. For more information on what the program does when you run codeunit 104045, see *Appendix A*, page 90.

If you have not completed the data preparation in *Task 3 - Data/Object Changes Prior to Step 1*, the ***Upgrade Error Log*** window will appear. You will not be able to proceed with the upgrade until you correct all errors in this window.

**3** To correct an error, select the error and click Show. This will open the relevant form and where possible show the specific record with the error.

If an error occurs when you are running the codeunit 104045, ***Upgrade 4.00 Step 1***, the program will store the last successfully completed process in the ***State Indicator*** table.

**4** To see a status of what has been upgraded and what remains, click Status, Status Indicator.

**Note**

The ***State Indicator*** table will contain information on which tables were upgraded when the upgrade process failed and which were not.

## Step 1 Cleanup

### Task 5 - Rename Old Tables

Navision 2.00-3.60

When you are upgrading from version 2.00-3.60:

- On form 104001, ***Upgrade - Old Version***, click Rename Tables.

  This runs codeunit 104047, ***Upgrade 4.00 Step Rename Old Tables***.

  You must run this codeunit after you have completed codeunit 104045, ***Upgrade 4.00 Step 1***, for all companies.

  This codeunit renames those tables that exist in both the old and the new version but have a different object number in the new version. If the old version tables were not renamed, conflicts would occur when you try to import the new objects (see *Task 7 - Import All Customized 4.0 Objects* on page 62).

### Task 6 - Change Objects

The next stage in the upgrade procedure involves importing the customer's customized objects into the new Navision 4.0 database. Before importing these objects, you must delete most of the objects that already exist in the database.

1. On form 104001, ***Upgrade - Old Version***, click Delete Objects.
2. This runs codeunit 104002, ***Delete Objects Excl. Tables***. The codeunit deletes all the objects that are not tables. This will ensure that no conflicts occur during the following steps.

### Task 7 - Import All Customized 4.0 Objects

1. Using the *Object Designer*, import the new customized Navision 4.0 objects into the database.

   These are the objects that were exported to the `objects.fob` file in step 2 on page 40.

   When the import starts, a warning appears informing you that some objects with conflicting versions already exist in the database.
2. Click OK and the ***Import Worksheet*** window appears.
3. Click *REPLACE ALL*, and then click OK to import the objects.

   If any errors occur while the program is importing the objects, the process will be canceled. Correct the problem in the new customized database and export the objects.fob file again (see step 2 on page 40). In the database you are upgrading, import the objects again.

## Task 8 - Compile Imported Objects

- Compile all the objects in the customer's new database.

  The database has now been restored into Navision 4.0 and the new customized Navision objects have been imported into the database.

**Compilation Warning:**

Some of the objects you deleted in *Task 6 - Change Objects* on page 62 no longer exist in the 4.0 version. However, the old version tables that were not deleted may have included references to these deleted objects. As a result, you can expect compilation errors concerning some tables. A list of these tables is provided after this note. If there are compilation errors in other objects than those listed, it may be that the customized 4.0 objects you imported contain compilation errors. These errors must be fixed before you proceed with the upgrade.

Navision 2.00-2.60

When upgrading from Navision Financials:

| Table ID | Table Name |
|---|---|
| 28 | ***Item Price*** |
| 40 | ***Item Price Change*** |
| 5000 | ***Prospect*** |
| 5005 | ***Campaign Line*** |
| 5008 | ***Activity*** |

Navision Manufacturing 2.60

When upgrading from Navision Manufacturing 2.60:

| Table ID | Table Name |
|---|---|
| 5000 | ***Prospect*** |
| 5005 | ***Campaign Line*** |
| 5008 | ***Activity*** |
| 99000791 | ***Table 99000791:Production Ord*** |
| 99000792 | ***Table 99000792:Prod. Order Ro*** |
| 99000796 | ***P.O. Output Journal Line*** |
| 99000813 | ***P.O. Consump. Journal Line*** |
| 99000819 | ***Fnshd Prod. Order Comp.*** |
| 99000828 | ***Planning Line*** |

Navision 3.01

When upgrading from Navision Attain 3.01:

| Table ID | Table Name |
|---|---|
| 28 | ***Item Price*** |
| 40 | ***Item Price Change*** |
| 243 | ***Report List*** |
| 5909 | ***Working Calendar*** |
| 99000791 | ***Table 99000791:Production Ord*** |
| 99000792 | ***Table 99000792:Prod. Order Ro*** |
| 99000796 | ***Output Journal Line*** |
| 99000813 | ***Consump. Journal Line*** |
| 99000815 | ***Finished Production Order*** |
| 99000818 | ***Finished Prod. Order Line*** |
| 99000819 | ***Fnshd Prod. Order Comp.*** |

Navision 3.10

When upgrading from Navision Attain 3.10:

| Table ID | Table Name |
|---|---|
| 28 | ***Item Price*** |
| 40 | ***Item Price Change*** |

Navision 3.60

When upgrading from Navision Attain 3.60:

| Table ID | Table Name |
|---|---|
| 5400 | ***Table 5400: Bln*** |

## Step 2 Data Preparation

### Task 9 - Import Upgrade Step 2 Objects

- Open the *Object Designer* and import the relevant fob file for the version from the table below. This file is located in the `Upgtk` folder on the product CD.

| Version | Fob file |
|---|---|
| 2.00-2.65 | `Upgrade260400.2.fob` |
| 3.01 | `Upgrade301400.2.fob` |
| 3.10 | `Upgrade310400.2.fob` |

| Version | Fob file |
|---|---|
| 3.60 | `Upgrade360400.2.fob` |
| 3.70 | `Upgrade370400.2.fob` |
| Commerce Portal 2.65 | `Upgrade265400.2.CP.fob` |
| Commerce Gateway 2.65 | `Upgrade265400.2.CG.fob` |
| Navision Manufacturing 2.60 | `Upgrade260400.2.MFG.fob` |

## Task 10 - Data/Object Changes Prior to Step 2

All versions

### Set Language

- Click Tools, Language and select the language of the old customer database from the list.

For example, if the old customer database was in German, you must select German as the current language.

### Update STX Files

Belgian/Canadian Multilanguage versions prior to 3.01

If you are upgrading from a *Belgian or Canadian* Navision Financials, Navision Manufacturing and Navision Advanced Distribution 2.01 or 2.60 multilanguage solution, note that all date formulas are in English.

This means that if, for example, you use a French fin.stx for the upgrade and French standards for date formulas, running codeunit 104048, ***Upgrade 4.00 Step 2***, will conflict with the old date formulas in English, if you do not protect them.

You must place angle brackets (<>) around the old text field date formulas before evaluating them as binary date formulas.

### Contact Salutations

Navision 2.00-3.01

When you are upgrading from version 2.00-3.01, the upgrade to Navision 4.0 requires some manual changes concerning salutations in the area of Relationship Management.

1. In form 5153, ***Salutations***, enter for example:

   *COMPANY - Company* - To be used later under ***Relationship Management Setup***.

   UNISEX - unisex (unknown gender) - To be used later under RM Setup.

   M - Male

F- Female

F-UMAR - Female Unmarried

Etc.

**2** *Contact - Company*:

You must set up a default company salutation code in form 5094, ***Relationship Management Setup,*** to be selected in the **Company Salutation Code** field - for example, COMPANY.

***3*** *Contact - Person:*

This task is optional. You can set up default person salutation codes in form 5094, ***Relationship Management Setup,*** to be selected in the **Person Salutation Code** field - for example, UNISEX.

**4** Select form 104070, ***Temp Title - Salutation List***, and click Run (Alt+R).

All titles must have a salutation code, and you will have to create these manually as suggested above. You should only consider what kind of person the contact is - for example, Married, Single, Male etc. - and not consider the language.

**Note**

The Salutation feature does not work until salutation formulas have been created. Both a formal and an informal salutation must be created for every language. However, this is not necessary for the upgrade tools to finish.

### Team-Meeting Organizers

Navision 3.01-3.70

When you are upgrading from Navision Attain 3.01-3.70, you must choose meeting organizers for each team which has a Team To-do of the type Meeting assigned to it in the system. To do this:

**1** Select form 104090 ***Team - Meeting Organizers***, and click Run (ALT+R).

**2** Fill in the code of the salesperson who will be the meeting organizer for these team to-dos of the type Meeting.

### Resource Units of Measure

Navision 2.60-3.60 and Navision Manufacturing 2.60

When you are upgrading from version 2.00-3.60, the upgrade to Navision 4.0 requires some manual changes concerning Base Unit of Measure in the area of Resources.

From Navision 3.70, a unit of measure can be related to a base unit of measure. In codeunit 104045, ***Upgrade 4.00 Step 1***, all units of measures previously used by a resource are stored in table 104113, ***Temp Resource Unit of Measure***. The **Qty. Per Unit of Measure** field is per default set to the value 1 and the **Related to Base Unit of Meas.** field is set to YES. The upgrade toolkit uses the code that was on the ***Resource Card*** in the old version as the base unit of measure.

To set up the correct relation between alternatives units of measure and the base unit of measure, do the following:

**1** Open the *Object Designer*, select form 104080, ***Temp Resource Units of Measure***, and click Run (Alt+R).

**2** For each line, enter the correct number of base units in the **Qty. Per Unit of Measure** field. For example, if base unit of measure is set to hour:

| Resource No. | Code | Qty. Per Unit of Measure | Related to Base Unit of Meas. |
|---|---|---|---|
| Lift | Day | 8 | *YES* |
| Lift | Hour | 1 | *YES* |

If the resource code is not related to the base unit of measure, remember to remove the check mark in the **Related to Base Unit of Meas.** field.

### Human Resource Units of Measure

Navision 2.60-3.70 and Navision Manufacturing 2.60

When you are upgrading from version 2.60-3.70, the upgrade to Navision 4.0 requires some manual changes concerning Units of Measure in the Human Resources application area.

From Navision 4.0, a unit of measure can be related to a base unit of measure. In codeunit 104045, ***Upgrade 4.00 Step 1***, all units of measures previously used by a resource are stored in table 5220, ***Temp Human Res. Unit of Measure***. As a default, the **Qty. per Unit of Measure** field is set to the value 1.

To set up the base unit of measure and set up the correct relation between alternatives units of measure and the base unit:

**1** Open the *Object Designer*, select form 5236, ***Human Res. Units of Measure***, and click Run (Alt+R).

**2** Select the base unit of measure and ensure the value of the **Qty. per Unit of Measure** field is 1.

**3** For each other unit of measure, enter the correct number of base units in the **Qty. per Unit of Measure** field. For example, if base unit of measure is set to hour:

| Code | Qty. per Unit of Measure |
|---|---|
| Day | 8 |
| Hour | 1 |

### Detailed Customer and Vendor Ledger Entry Check

Navision 2.00-2.60 and Navision Manufacturing 2.60

The amounts in the detailed customer/vendor ledger entries are in codeunits 104054, ***Upgrade 4.00 Step 2 Customer***, and 104055, ***Upgrade 4.00 Step 2 Vendor***, checked against the amounts that were used to calculate them. This is done as an extra precaution to ensure that no errors are generated in the detailed ledger entries. This check should be carried out in test upgrades and in the final upgrade. However, this step can be omitted during the final upgrade of large installations in order to make the upgrade process a bit faster.

To remove the checks, comment out the following function call lines in the upgrade function in codeunit 104045, ***Upgrade 4.00 Step 1***:

```
AssistCodeunit6.VerifyCustLedgEntries(StateIndicator);

AssistCodeunit7.VerifyVendLedgEntries(StateIndicator);
```

### Cleanup of Date Compressed Item Ledger Entries

Navision 2.00-2.60 and Navision Manufacturing 2.60

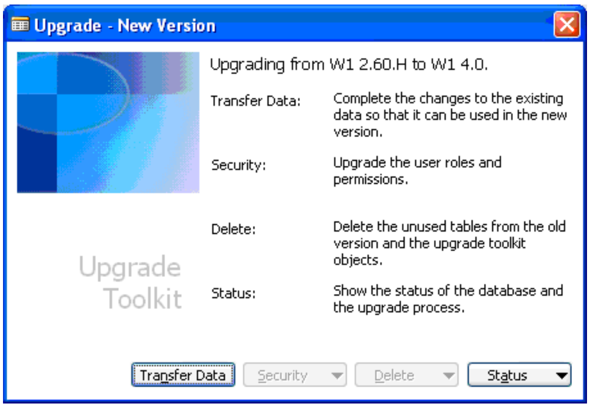
If necessary, modify the date compressed item ledger entry filters in codeunit 104049, ***Upgrade 4.00 Step 2 Value***, if you have not already done so (see *Task 3 - Data/Object Changes Prior to Step 1* on page 55.)

## Step 2 Data Conversion

### Task 11 - Step 2

When you are upgrading from any version:

1 Open the *Object Designer*, select form 104002, ***Upgrade - New Version***, and click Run (Alt+R).



Upgrading from W1 2.60.H to W1 4.0.

| | |
|---|---|
| Transfer Data: | Complete the changes to the existing data so that it can be used in the new version. |
| Security: | Upgrade the user roles and permissions. |
| Delete: | Delete the unused tables from the old version and the upgrade toolkit objects. |
| Status: | Show the status of the database and the upgrade process. |

2 Click click Transfer Data.

This runs codeunit 104048, ***Upgrade 4.00 Step 2***. You must run this codeunit for each company. For more information on what the program does when you run codeunit 104048, see *Appendix B*, page 102.

If you have not completed the data preparation in *Task 10 - Data/Object Changes Prior to Step 2*, the ***Upgrade Error Log*** window will appear. You will not be able to proceed with the upgrade until you correct all errors in this window.

**3** To correct an error, select the error and click Show. This will open the relevant form and where possible show the specific record with the error.

If an error occurs when you are running the codeunit 104048, ***Upgrade 4.00 Step 2***, the program will store the last successfully completed process in the ***State Indicator*** table.

**4** To see a status of what has been upgraded and what remains, click Status, Status Indicator.

**Note**

The ***State Indicator*** table will contain information on which tables were upgraded when the upgrade process failed and which were not. If the upgrade has completed successfully, the ***State Indicator*** table will be empty.

## Setup Completion

### Task 12 - Complete Data Changes After Step 2

#### Business Relations

Navision 2.00-2.60 and Navision Manufacturing 2.60

In the old Contact Management system a prospect can have both a customer number saying which customer the prospect is linked to and have a Prospect Status on the ***Prospect Card*** indicating that the prospect is a customer.

During the upgrade process a new Business Relation will be created for each of these entries. This means that a customer can have two entries in the ***Business Relation*** table. Therefore, when the upgrade is completed you must manually remove the entry that was created on the basis of the Prospect Status information in Contact Management. The entry in the ***Business Relation*** table that was created on the basis of the ***Customer*** table contains more information and this is how you can identify the entry that should be deleted.

Furthermore, if the Prospect Status in the old Contact Management was used to specify different category of customers, for example, good versus bad or domestic versus international, the user should not delete any of the new Business Relations, until this categorization has been created manually in the profile.

This applies both to customers and vendors.

### Logging Old Campaigns

Navision 2.00-2.60 and Navision Manufacturing 2.60

All the old campaigns are converted to un-logged segments. You should manually log or delete the segments one by one based on your knowledge of the old campaigns.

Navision Manufacturing 2.60

### Planning Worksheets

If the customer wishes to use their original planning worksheet templates and names, do the following:

- Using the printout of report 104027, ***Mfg. Journals and Plan. Wksht***, from Task 2 (see page 55), set up the original planning worksheet templates and names using form 293, ***Req. Worksheet Templates.***

Navision Manufacturing 2.60 and Navision 3.01

### Manufacturing Journals

If the customer wishes to use their original manufacturing journal templates and batches, do the following:

- Using the printout of report 104027, ***Mfg. Journals and Plan. Wksht***, from Task 2 (see page 55), set up the original Output and Consumption Journal Templates and Batches in form 102, ***Item Journal Templates***.

### Production Orders

Navision Manufacturing 2.60 and/or Navision 3.01

When using ***Replan Production Order*** batch job (report 99001026) on production orders that have been created before the upgrade, the batch job will not find and delete the related lower level production orders. You must replan those orders manually. Or add to the upgrade toolkit that the lot number on the production orders should be carried on to the **Replan Ref. No.** field in the 4.0 version production order.

### Service Pricing

Navision 3.01-3.10

Use the documentation of the setup in service pricing that you made during *Task 3 - Data/Object Changes Prior to Step 1* on page 60 to verify the content of the new setup and to do the following:

1. Manually transfer data from table 5926, ***Serv./Cust. Price Gr. Pricing***, to table 6081, ***Serv. Price Group Setup***, based on your own documentation.

   The values can only be partially merged by codeunit 104045, ***Upgrade 4.00 Step 1***, due to the different key structure in the two tables.

**2** Check with your own documentation that all records from table 5922, ***Price-Adjusted Serv. Inv. Line***, have been transferred to table 6083, ***Serv. Price Adjustment Detail.***

The values may only be partially merged by codeunit 104045, ***Upgrade 4.00 Step 1,*** due to the different key structure in the two tables.

**Note**

Since not all fields are upgraded, it is important that the setups are checked and approved by the customer before they start to use the database.

### Service Contracts

Navision 3.01-3.10

Correct the service contracts that was marked as *Open* by codeunit 104048, ***Upgrade 4.00 Step 2***, if the sum of the line amounts didn't match the annual amount on the header.

If the customer will not do this manually, it can be automated per service contract by simply removing the check mark in the **Allow Unbalanced Amounts** field on the service contract header. Then the system will ask how the difference should be distributed to the lines.

### Warehouse Employees

Navision 3.01-3.10

Create entries in the ***Whse. Employee*** table manually for users that can work in several locations or blank locations.

An empty warehouse location filter in the ***User Setup*** table in version 3.01 means that the user can work in every location including blank locations.
In version 4.0 you have to create an entry in the ***Whse. Employee*** table for each location the user can work for.

### Source Codes and Source Code Setup

All versions

Update the ***Source Code*** table and ***Source Code Setup*** table for any additional source codes in the new version.

## Task 13 - Initialize the Company

All versions

**1** Run codeunit 2, ***Company-Initialize*** .

2 To display the new main menu, close the company and open it again/click ALT+F1.

3 Open form 531, ***Set Up Checklist***. This will automatically update the checklist for the 4.0 data structure.

Navision 2.00-2.60 and Navision Manufacturing 2.60

4 Make an exchange rate adjustment in the upgraded database.

An exchange rate adjustment must be done to ensure that the customer ledger entries and vendor ledger entries are updated correctly. If this is not done, rounding errors can be introduced when the entries are applied. You must always perform an exchange rate adjustment if there are customers with foreign currency.

You have now upgraded the company-specific data.

## 6.3 UPGRADING DATA COMMON TO ALL COMPANIES

The data contained in the tables that are not company specific are not upgraded by the upgrade tool. This includes the following tables:

| Table ID | Table Name |
|---|---|
| 2000000003 | ***Member Of*** |
| 2000000004 | ***User Role*** |
| 2000000005 | ***Permission*** |
| 2000000053 | ***Windows Access Control*** |
| 2000000203 | ***Database Key Groups*** |

You will need to manually update the data in these tables to ensure the data can be used with the 4.0 features.

### Roles and Permissions

**Note**

The following procedure assumes the customer has used the standard roles and permissions provided with the demo company as the basis for their roles and permissions in their database. If this is not the case, the upgrade of roles and permissions may have to be performed manually.

To upgrade user roles and permissions:

**1** On form 104002, ***Upgrade - New Version***, and click Security, Import Roles.

**2** Import the standard user roles for the new version from a text file. See note below.

**3** On form 104002, ***Upgrade - New Version***, and click Security, Import Permissions.

**4** Import the standard permissions for the new version from a text file. See note below.

**Note**

These 2 text files may be provided by your local Microsoft Business Solutions subsidiary with the local upgrade toolkit. If the 2 text files are not included, you can use dataport 104001, ***Import/Export Roles***, and 104002, ***Import/Export Permissions*** to export the new 4.0 standard roles and permissions from the demo company on the 4.0 Product CD.

**5** To update the changes to roles and permissions, click Security, Update Roles and Permissions.

When you run this codeunit, any logins with the prior version roles above will be updated to have the corresponding 4.0 role. In addition, any prior version roles and permissions that do not exist in the 4.0 version will be deleted.

**6** Assign any additional roles for new 4.0 features to the existing database logins and/or Windows logins. The new features in 4.0 that may need to be assigned are shown in the following tables:

| Version Being Upgraded | New Functionality with New Roles in 4.0 |
|---|---|
| 2.00-2.60, 2.65 CP, 2.65 CG. | Manufacturing |
| 2.00-2.60, 2.65 CP, 2.65 CG, 2.60 NM | Inventory Transfers |
| 2.00-2.60, 2.65 CP, 2.65 CG, 2.60 NM | Release documents |
| 2.00-2.60, 2.65 CP, 2.65 CG, 2.60 NM | Relationship Management - Opportunities |
| 2.00-2.60, 2.65 CP, 2.65 CG, 2.60 NM | Service Management |
| 2.00-2.60, 2.65 CP, 2.65 CG, 2.60 NM | Warehouse Management |
| 2.00-3.01, 2.65 CG, 2.60 NM | Commerce Portal |
| 2.00-3.10, 2.65 CP, 2.65 CG, 2.60 NM | ADCS |
| 2.00-3.10, 2.65 CP, 2.65 CG, 2.60 NM | Change Log |
| 2.00-3.10, 2.65 CP, 2.65 CG, 2.60 NM | XBRL |
| All Versions | Business Analytics |
| All Versions | Business Notifications |
| All Versions | Intercompany Transactions |
| All Versions | Sales, Purchase, Inventory Analysis Reports |
| All Versions | Sales and Purchase Budgets |
| All Versions | Navigation Pane Designer |

**7** If the customer has changed the Read/Write/Modify/Delete/Execute rights for some of the standard permissions, you will need to redo these changes manually.

**8** When you have finished the roles and permissions upgrade you should thoroughly test that the existing logins can perform all their tasks without any permissions errors.

## Database Key Groups

Any changes to Database Key Groups must be updated manually.

**Note**

You must perform the update of database key groups AFTER you have performed the upgrade of company data. This is because any changes in the keys assigned to key groups will be imported with the 4.0 tables included in the 4.0 objects fob file.(See *Task 2 - Import Upgrade Step 1 Objects* on page 55).

To upgrade database key groups:

**1** For all versions, add a new database key group, *ConvLoc*.

**2** For all versions except 3.70, update the **Key Group** field for the following database key groups in table 2000000203, ***Database Key Groups***.

| Prior Version Key Group | 4.0 Key Group |
|---|---|
| Acc(D/P) | Acc(Dim) |
| Bank(D/P) | Bank(Dim) |
| Cust(D/P) | Cust(Dim) |
| Item(D/P) | Item(Dim) |
| Vend(D/P) | Vend(Dim) |

### Windows Logins

The security system in Navision Financials was changed considerably in Navision Financials 2.60 to encompass the improved security features provided by Windows 2000.

Navision Financials 2.01 and earlier versions stored all the information about users – their ID, password and name – in the ***User*** table, regardless of the type of authentication the user was employing.

Navision 4.0 stores this user information in two separate tables – the ***User*** table and the ***Windows Login*** table. The ***User*** table stores all the information required for using database server authentication, where the user must supply their user ID and password in order to access a database. The ***Windows Login*** table stores the information required for using the single sign-on system supported by Windows 2000.

When you upgrade from Navision 2.00 or 2.01, all of the information about your users is transferred from the ***User*** table in the old database to the ***User*** table in the new database. This means that you cannot use the Windows 2000 single sign-on system until you have transferred the user information from the ***User*** table to the ***Windows Login*** table. Users who use database server authentication can continue using it without having to change their password.

In relation to the changed security system, the following changes have been made in the terminology:

- Windows authentication means both NT authentication (the unified login), and the single sign-on supported by Windows 2000.

- Database authentication means both Navision Financials authentication and SQL Server authentication.

- Role means a Navision Financials security group.

- Database login means an entry in the ***User*** table. When you use database server authentication, you are granted a database login.

- Windows login means an entry in the ***Windows Login*** window. When you use Windows authentication, you are granted a Windows login.

### Features Provided by Windows Logins

The availability of Windows logins means you must carefully consider your security needs and in particular the type of authentication that you want to use, when you upgrade to Microsoft Business Solutions–Navision 4.0.

The new security system provides you with several new features. These include the ability to assign roles to Windows users and groups within Navision. This means that you do not have to create an entry for each user in Navision, but you can instead make them members of a Windows security group and then assign that group a role within Navision, thereby ensuring that all the members of that Windows security group have the same permissions.

Furthermore, Windows administrators can grant or deny users access to Navision by simply adding them to or deleting them from Windows security groups.

### Converting Database Logins to Windows Logins

Before converting database logins, you must ensure that no other users are currently using the system.

**Note**

If you are currently using database authentication and wish to change to Windows authentication, you can convert the database logins to Windows logins.

To convert the database logins to Windows logins:

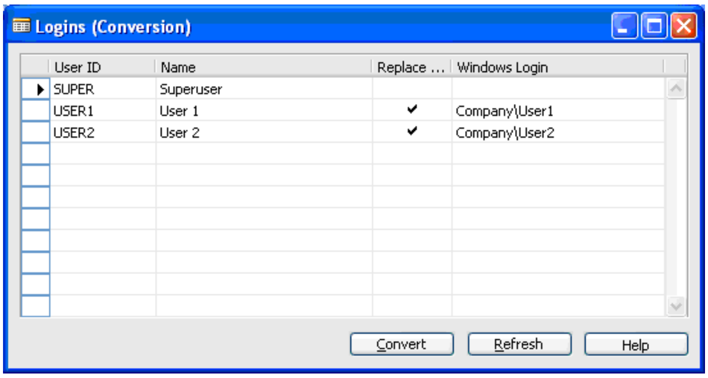
**1** Open the *Object Designer* and import the `Windows Login.fob` file.

This file is located in the `Upgtk` folder on the product CD.

The following objects are imported:

| Type | ID | Name |
|---|---|---|
| Table | 104021 | ***Login (Conversion)*** |
| Form | 104021 | ***Logins (Conversion)*** |
| Codeunit | 104021 | ***Suggest Logins*** |
| Codeunit | 104022 | ***Convert Logins*** |

**2** Select form 102021, ***Logins (Conversion)***, and click Run (ALT+R).



| User ID | Name | Replace ... | Windows Login |
|---|---|---|---|
| SUPER | Superuser | | |
| USER1 | User 1 | ✓ | Company\User1 |
| USER2 | User 2 | ✓ | Company\User2 |

This window lists all of the users that were created in the ***User*** table in the old database. This ***User*** table has already been transferred to the new database and all the users have been given a database login. If both the domain and the client computer you are using are running on Windows 2000 or Windows NT, this window will also list any Windows logins that exist for these users. If this is not the case, you will have to enter the names of their Windows logins manually.

You cannot create any Windows logins from this window.

All of the users who have a Windows login will have a check mark in the **Replace by Windows Login** field. If the customer has some users that must use database authentication, clear the check mark for those users. These users will not be transferred to the ***Windows Login*** table.

**Important**

Leave the database login of at least one superuser in the ***Database Logins*** table, otherwise the Navision security system will not be enabled.

**3** Click Convert and the users that have a check mark in the **Replace by Windows Login** field are transferred to the ***Windows Login*** table.

**4** Close the ***Logins (Conversion)*** window.

The database logins have now been sucessfully converted into Windows logins.

**Note**

The conversion of database logins to Windows logins does not have to be performed as part of an upgrade of Navision. It can be performed at any time if the customer wishes to change from Database logins to Windows Logins without upgrading. This situation can occur if the customer upgrades their version of Windows and the new Windows version offers Windows authentication features.

## 6.4 DELETING UNUSED TABLES AND UPGRADE TOOLKIT

### Delete Old Tables no longer Used in 4.0

You may want to delete the tables from the old version that are only used during the upgrade process.

To delete the unused tables from the old version::

On form 104002, ***Upgrade - New Version***, click Delete, Unused Old Tables.

### Delete Upgrade Toolkit Objects

You may want to delete the upgrade toolkit objects after the upgrade.

To delete the upgrade toolkit objects:

**1** On form 104002, ***Upgrade - New Version***, click Delete, Upgrade Toolkit.

This will delete all upgrade toolkit objects except table 104002. To delete the table:

**2** Open the *Object Designer*, select tabel 104002, ***Status log***, and click F4.

## 6.5 TESTING THE DATABASE

To determine the state of the customer's current database and correct any database errors that may exist, follow the procedures described in this section. This will ensure that no errors exist in the database that will be used as the basis for the upgrade.

**1** Run a database test to determine the state of the database you have just upgraded.

Test everything except field relationships between tables. If the test fails, you must follow the workflow for repairing damaged databases (contact your local Microsoft Business Solutions Solution Center for details).

**2** Run the remaining part of the database test, that is, test field relationships between tables.

This will allow you to determine the extent of any data inconsistency that can exist in the database. If any error messages appear during the test, note their content and number. You must then decide whether or not these errors will affect the upgrade.

**3** To verify that the customer's license file includes all the necessary permissions in the upgraded solution, use the customer's license file to test the functionality in the database.

**4** Test the upgraded item ledger entry invoiced quantities to ensure that any date compressed item ledger entries have been upgraded correctly.

## 6.6 INSTALLING THE CLIENTS

Before you can continue with the upgrade process, you must perform the following tasks:

1. Make a new backup of the new upgraded database, thereby ensuring that you have a new backup of your updated database.
2. Remove all the earlier versions of Navision from the client computers.
3. Install the new version of Navision on all the client computers and on the server.

You have now installed the new version of Navision and can continue with the upgrade process.

## Chapter 7

# Migrating to the Microsoft SQL Server Option for Navision 4.0

This chapter contains instructions for migrating from Microsoft Business Solutions–Navision 4.0 to the Microsoft SQL Server Option for Navision 4.0.

The chapter contains:

- 7.1 Preparing to Migrate
- 7.2 Checking the Old Database
- 7.3 Migrating the Old Database

## 7.1 PREPARING TO MIGRATE

**Note**

If you want to migrate from an earlier version of Navision Financials to the Microsoft SQL Server Option for Microsoft Business Solutions–Navision 4.0, you must first upgrade to Microsoft Business Solutions–Navision 4.0.

If you have successfully upgraded the customer's Navision Financials application to Navision 4.0, you will already have implemented the necessary changes to the objects in the customer's database. You will have also implemented the necessary locking changes in any customer-specific objects that existed in the earlier version of Navision Financials.

To prepare the customer's Navision 4.0 installation for migrating:

**1** Verify that both your solution developer's license file and the customer's license file have been upgraded to SQL Server Option for Navision 4.0.

**2** Identify the user ID and password of a superuser in the system.

**3** Make a backup of the entire database before you begin the migration process. Keep the backup in a safe place, and keep it for a long time.

**Note**

Note how long it takes to complete each of the steps in the migration process. You can use this information to estimate the time and cost involved in future migrations.

**4** Make a copy of the customer's database, and migrate the copy.

**5** Ensure no other users are connected to the system before you carry out each part of the migration process.

**6** Install Microsoft SQL Server 2000 on the server computer.

**Note**

Because the customer's license file may be too limited to carry out some of the steps in the following sections, use your solution developer's license from this point on.

## 7.2 CHECKING THE OLD DATABASE

To determine the state of the customer's current database and correct any database errors that might exist, follow the procedure described in this section. This will ensure that no errors exist in the database that will be used as the basis for the upgrade.

**1** Open the customer's Navision 4.0 database, and check that no other users are currently using the system.

**2** Run a database test to determine the state of the customer's Navision 4.0 database.

Test everything except field relationships between tables. If the test fails, you must follow the workflow for repairing damaged databases (contact your local Microsoft Business Solutions Solution Center for details).

**3** Run the remaining part of the database test, that is, test field relationships between tables.

This will allow you to determine the extent of any data inconsistency that exists in the database. If error messages appear during the test, note their content and number. Decide whether or not these messages will affect the migration process.

**4** Compile all the objects in the database.

Make a list of the objects that cannot be compiled. At some point, you must decide what to do with the objects that cannot be compiled. They will create problems if you ignore them.

**5** Open the *Object Designer* and import the `Migrate.fob` file.

This file is located in the `Upgtk` folder on the product CD. The import begins and a message appears.

Click Yes to import the objects.

The following objects are imported:

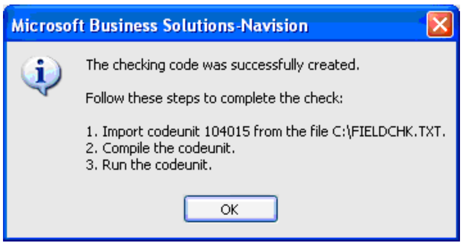
| Type | No. | Name |
|---|---|---|
| Table | 104010 | ***Incorrect Data Value*** |
| Table | 104011 | ***Code Field Information*** |
| Form | 104010 | ***Incorrect Data Values*** |
| Form | 104013 | ***Code Field Information*** |
| Codeunit | 104010 | ***Create Field Checking Code*** |
| Codeunit | 104011 | ***Date Check Management*** |
| Codeunit | 104012 | ***Code Check Management*** |
| Codeunit | 104013 | ***Date Check Indicator Mgt.*** |
| Codeunit | 104014 | ***Date Check Indicator Mgt. 2*** |
| Codeunit | 104015 | ***Field Check*** |

**Note**

If you are using the `migrate.fob` file included in the 4.0 Upgrade Toolkit to migrate to the Microsoft SQL Server Option for Navision 3.70 or earlier, you will get import/compilation errors if you are not using 4.0 client and executables. This is because the codeunit 104011, ***Data Check Management***, contains text variables that are 1024 characters in length. This increased text variable length was only introduced with the 4.0 C/SIDE and executables.

**6** In the *Object Designer*, select codeunit 104010, ***Create Field Checking Code***, and click Run (ALT+R).

Codeunit 104010 generates some new code in a text file with this path and name: `c:\fieldchk.txt`. You can change the file name and path by modifying the codeunit.



The checking code was successfully created.

Follow these steps to complete the check:

1. Import codeunit 104015 from the file C:\FIELDCHK.TXT.
2. Compile the codeunit.
3. Run the codeunit.

Import the `c:\fieldchk.txt` file.

The following objects are imported:

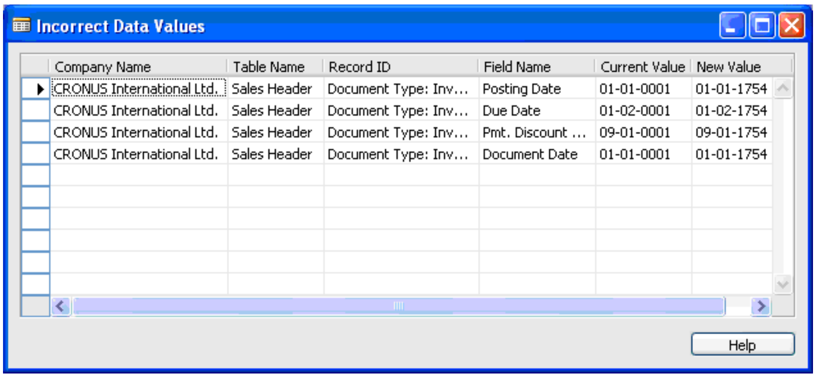
| Type | No. | Name |
|---|---|---|
| Codeunit | 104015 | ***Field Check*** |

**7** Select codeunit 104015, ***Field Check***, and compile it.

**8** Run codeunit 104015, ***Field Check***.

Codeunit 104015 checks that every date, text, code and decimal value in the database can be stored in a SQL Server database. It also checks that nonzero decimal values will not be rounded to zero when they are stored in the SQL Server database. A progress indicator will help you monitor this process. This involves the program reading most of the records in the database and may therefore take some time.

If any values need to be changed, the program shows a list of the incorrect values and the suggested new values in the ***Incorrect Data Values*** window:



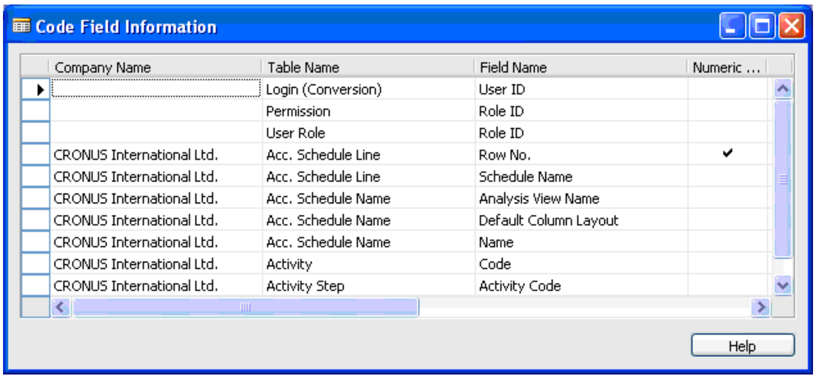
| Company Name | Table Name | Record ID | Field Name | Current Value | New Value |
|---|---|---|---|---|---|
| CRONUS International Ltd. | Sales Header | Document Type: Inv... | Posting Date | 01-01-0001 | 01-01-1754 |
| CRONUS International Ltd. | Sales Header | Document Type: Inv... | Due Date | 01-02-0001 | 01-02-1754 |
| CRONUS International Ltd. | Sales Header | Document Type: Inv... | Pmt. Discount ... | 09-01-0001 | 09-01-1754 |
| CRONUS International Ltd. | Sales Header | Document Type: Inv... | Document Date | 01-01-0001 | 01-01-1754 |

You can modify these suggestions if you want. When you close the window, a message will appear asking you whether or not you want to implement the changes. If you click Yes, the program will implement the changes. The program will read and modify a small number of records in the database during this step.

Codeunit 104015 also checks the code fields in your data. If your code fields contain numeric values of varying lengths, it will list them for you. If you sort by these fields the resulting sorting will be incorrect. Furthermore, any filters using these fields and containing numeric ranges will give unexpected results. One way of overcoming this sorting problem is to represent these code fields as integers. This can be done if the code fields only contain numbers and these numbers do not start with zeros.

A progress indicator will help you monitor the codeunit 104015, ***Field Check***, while it is being run. This step involves the program reading most of the records in the database and may therefore take some time. If there are any inconsistencies, they will be listed in the form 104013, ***Code Field Information***. To open this form:

- In the *Object Designer*, run form 104013, ***Code Field Information***.



| Company Name | Table Name | Field Name | Numeric ... |
|---|---|---|---|
|  | Login (Conversion) | User ID |  |
|  | Permission | Role ID |  |
|  | User Role | Role ID |  |
| CRONUS International Ltd. | Acc. Schedule Line | Row No. | ✓ |
| CRONUS International Ltd. | Acc. Schedule Line | Schedule Name |  |
| CRONUS International Ltd. | Acc. Schedule Name | Analysis View Name |  |
| CRONUS International Ltd. | Acc. Schedule Name | Default Column Layout |  |
| CRONUS International Ltd. | Acc. Schedule Name | Name |  |
| CRONUS International Ltd. | Activity | Code |  |
| CRONUS International Ltd. | Activity Step | Activity Code |  |

This window lists all of the code fields used in the database. It also contains information about whether the code field is numeric only, a compatible integer and if it is zero padded. The window displays the minimum and maximum number of digits that the field contains. The window also displays the SQL Data Type that is used in the field and whether it contains any numbering conflicts and the name of any linked tables.

This multitude of columns means that there are numerous ways of sorting this information.

The **SQL Data Type** field shows how a code field is represented on SQL Server. You can change the **SQL Data Type** property for each code field. If you set the **SQL Data Type** field to *Integer* for a code field, you will be allowed to store only positive numbers in the code field. This will ensure that numeric sorting is done correctly.

For more information about numbering and sorting in the SQL Server Option for Navision, see the manual *Application Designer's Guide*.

You can represent a code field that is already in use as an integer, only if the **Numeric Only** field and the **Compatible Integer** field are checked and the **Zero Padded** field is cleared.

You must check whether any of the following fields are listed in the ***Numbering Conflicts*** window.

| Table Name | Table ID | Field Name | Field No. |
|---|---|---|---|
| ***G/L Account*** | 15 | **No.** | 1 |
| ***Acc. Schedule Line*** | 85 | **Row No.** | 3 |
| ***VAT Statement Line*** | 256 | **Row No.** | 4 |

If any of these fields appear in the ***Numbering Conflicts*** window, you should be aware that any totals based on them may be inconsistent. When possible, this situation can be corrected either by changing the data, or by changing the SQL Data Type property for these fields to *Integer*.

## 7.3 MIGRATING THE OLD DATABASE

Now that the old database has been checked and modified to ensure that it is compatible with SQL Server, you can migrate to the SQL Server Option for Navision 4.0.

1. Open the customer's 4.0 database in Navision 4.0, and check that no other users are currently using the system.
2. Make a backup of the customer's Navision 4.0 database and name it, for example, `data.fbk`.
3. Create a new database with the SQL Server Option for Navision 4.0.
4. Restore everything from the backup saved in `data.fbk`.

   Restoring the backup into the SQL Server Option for Navision 4.0 database will take some time and will generate a very large transaction log. We recommend that you make a new SQL Server backup of the database before you start to work with it. This will truncate the transaction log and give you a new SQL Server backup as your starting point.

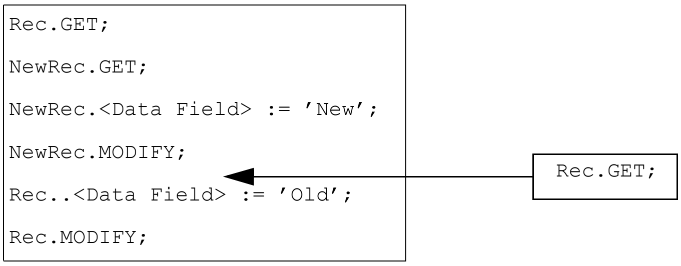
**Note**

To verify that the customer's license file includes all the necessary permissions in the migrated solution, you must use the customer's license file in the following step.

5. Test the newly migrated database to ensure that you do not encounter version control conflicts.

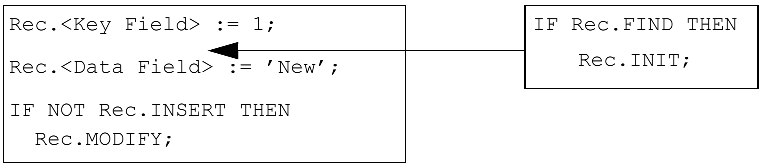
If you receive error messages informing you that a record has been modified by another user, even though you are the only user currently using the system, you have version control conflicts. These problems occur because the SQL Server Option for Navision 4.0 has stricter version control than previous versions of Navision Financials. Any attempt to modify or delete a record will fail if the timestamp on the version of the record that you have read is different from the timestamp on the actual record stored in the database.

The following tables contain examples of the old incorrect code and the new correct code that must be inserted to avoid version control conflicts:



```
Rec.GET;
NewRec.GET;
NewRec.<Data Field> := 'New';
NewRec.MODIFY;
Rec..<Data Field> := 'Old';
Rec.MODIFY;
```

```
Rec.GET;
```



```
Rec.<Key Field> := 1;

Rec.<Data Field> := 'New';

IF NOT Rec.INSERT THEN
  Rec.MODIFY;
```

```
IF Rec.FIND THEN
    Rec.INIT;
```

# Appendix A

## Codeunit 104045, *Upgrade 4.00 Step 1*

This appendix describes the tasks that the Upgrade Toolkit performs when you run codeunit 104045, ***Upgrade 4.00 Step 1***.

## A.1 CODEUNIT 104045, *Upgrade 4.00 Step 1*

The tasks performed by the codeunit are grouped into functions areas.

### Commerce Portal

Codeunit 104045 performs the following tasks:

- Moves the **Notification Process Code** and **Queue Priority** fields in the ***Customer*** table and the **Queue Priority** field in the ***Vendor*** table to the corresponding Navision tables. The fields have been given new numbers.
- Removes the **Sales Agent Prospect No.** field from the ***Web Site*** table, as Prospects no longer exists in Navision 4.0

### Commerce Gateway

Codeunit 104045 performs the following tasks:

- Changes data type from *Code* and *Text* respectively to *DateFormula* for the **Lead Time Calculation** and **Safety Lead Time** fields in the ***Inbound Product Catalog Line*** and ***Outbound Product Catalog Line*** tables.
- If you are upgrading from Commerce Gateway 2.65, codeunit 104045 removes the information from the **Lot No.**, **Serial No.** and **Reserve** fields in the ***Outbound Purch. Document Line*** table and the ***Outbound Sales Document Line*** table.

### Date Formulas

Codeunit 104045 performs the following task:

- Transfers date formulas from all the base tables, for example, the ***Payment Terms*** table, to temporary tables, and deletes them from the base tables. This is done because date formulas are stored in DateFormula fields in Navision. In Navision Financials they were stored in text and code fields.

### Item Ledger Entries and Item Tracking

Codeunit 104045 performs the following tasks:

- Removes the information from the **Lot No.** and **Serial No.** fields in the base tables, for example, the ***Sales Line*** table, transfers it to a temporary table and deletes this information from the base tables. This information will be transferred to the new item tracking tables when you run codeunit 104048, ***Upgrade 4.00 Step 2***, in *Task 11 - Step 2* on page 68.
- Transfers all the value related information from the ***Item Ledger Entry*** table and the ***Item Application Entry*** table to a temporary table and deletes this value related information from the base tables.
- Stores the item tracking information of the old database in temporary upgrade tables that are carried to the new database.

See *Appendix D* on page 134 for a specification of the upgrade concerning Item Tracking in versions 2.60 to 3.10.

- Transfers **Serial No.** and **Lot No.** fields from table 32, ***Item Ledger Entry***, to table 104048, ***Temp Item Ledger Entry***.
- Creates item entry relations based on tables 111, 121 and saves them in table 104092, ***Temp Item Entry Relation***. Clears **Serial No.** and **Lot No.** fields in tables 111, 121.
- Clears **Serial No.** and **Lot No.** fields in tables 113, 115, 123, 125.
- Saves remaining item tracking lines for tables 37, 39, 83, 246, 5406, 5407, 99000929 in tables 104093, ***Temp Item Tracking Line***, and 104094, ***Temp Tracking Specification***, and clears **Serial No.** and **Lot No.** fields on tables 37, 39, 83, 246, 5406, 5407, 99000929.
- Saves item tracking codes and their relations to items in table 104046, ***Temp Item Tracking Code.***

See Appendix D on page 134 for a specification of the 2.60-4.0 upgrade concerning Item Tracking.

Navision 3.01 or later versions

Codeunit 104045 performs the following tasks:

- In the ***Item Application Entry*** table, sets the **Consumption** and **Output is Adjusted** fields to false and sets the **Output Value Entry No.** field to *0*.
- In the ***Item Application Entry*** table, clears the **Output Completely Invoiced Date** field for outbound entries if the corresponding item ledger entry has not yet been invoiced.

Navision 2.00-3.10

Codeunit 104045 performs the following tasks:

- Checks that there is no negative inventory per bin in the locations where bins are used. If there is negative inventory, the codeunit shows an error.
- Transfers the entries in the ***Item Ledger Entry*** table to a temporary table. For locations where bin codes are used, if the codeunit finds entries where the bin code is blank, the program inserts a bin code named ’BLANK’ into the **Bin Code** field.

### Customer Ledger and Vendor Ledger Entries

Codeunit 104045 performs the following task:

- Transfers all the amount information from the ***Customer Ledger Entry*** table and the ***Vendor Ledger Entry*** table into temporary tables, and deletes this information from the base tables. This is done to create detailed customer ledger entries and detailed vendor ledger entries based on the information in the ***Customer Ledger Entry*** table and the ***Vendor Ledger Entry*** table.

### Dimensions

Codeunit 104045 performs the following task:

- Transfers information from the following fields to a temporary table and then deletes the information from the base table:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 15 | ***G/L Account*** | 50 | **Department Posting** |
| 15 | ***G/L Account*** | 51 | **Project Posting** |
| 168 | ***Job Budget Line*** | 17 | **Department Code** |
| 168 | ***Job Budget Line*** | 18 | **Project Code** |
| 5207 | ***Employee Absence*** | 9 | **Department Code** |
| 5207 | ***Employee Absence*** | 10 | **Project Code** |

This is done because department and project codes are no longer stored only in the base tables. Department and project codes are now treated as dynamic/replaceable global dimensions.

## Reservations

Codeunit 104045 performs the following tasks:

- In Navision Financials companies, codeunit 104045 deletes reservation entries where the reservation status is not open. It also deletes illegal reservation entries.

  All other reservation entries are transferred to the ***Temp Reservation Entry*** table.

Navision Attain 3.01 and Navision Manufacturing 2.60

Codeunit 104045 performs the following tasks:

- Changes field 10, **Source Type** in table 337, ***Reservation Entry***, from field type *Option* to *Integer*.

This change required other fields in other tables to be changed accordingly:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 99000799 | ***Order Tracking Entry*** | 20 | **For Type** |
| 99000799 | ***Order Tracking Entry*** | 26 | **From Type** |

The necessary data is transferred into the imported temporary tables and is later transferred back into the corresponding Navision 4.0 tables with new definitions.

## Other

Codeunit 104045 clears the following fields:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 27 | ***Item*** | 5406 | **Indirect Cost per Unit** |
| 339 | ***Item Application Entry*** | 16 | **Adjusted Cost** |

| **Table ID** | **Table Name** | **Field No.** | **Field Name** |
|---|---|---|---|
| 339 | ***Item Application Entry*** | 5801 | **Adjusted Cost (ACY)** |

- Clears the following field because it is a FlowField in Navision:

| **Table ID** | **Table Name** | **Field No.** | **Field Name** |
|---|---|---|---|
| 90 | ***BOM Component*** | 5 | **Bill of Materials** |

- Truncates the following field because it has been shortened in Navision:

| **Table ID** | **Table Name** | **Field No.** | **Field Name** |
|---|---|---|---|
| 14 | ***Location*** | 2 | **Name** |

- Clears the following option field where the option was set to Bold. This is done because the Bold option has been removed:

| **Table ID** | **Table Name** | **Field No.** | **Field Name** |
|---|---|---|---|
| 85 | ***Acc. Schedule Line*** | 16 | **Show** |

### Prices and Discounts

Navision 2.00-3.10 and Navision Manufacturing 2.60

Codeunit 104045 clears the following fields:

| **Table ID** | **Table Name** | **Field No.** | **Field Name** |
|---|---|---|---|
| 6 | ***Price Group*** | 3 | **Allow Quantity Disc.** |
| 6 | ***Price Group*** | 4 | **Allow Cust./Item Disc.** |
| 27 | ***Item*** | 13 | **Sales Qty. Disc. Code** |
| 36 | ***Sales Header*** | 36 | **Allow Quantity Disc.** |
| 37 | ***Sales Line*** | 9 | **Quantity Disc. Code** |
| 37 | ***Sales Line*** | 26 | **Quantity Disc. %** |
| 37 | ***Sales Line*** | 43 | **Allow Quantity Disc.** |
| 37 | ***Sales Line*** | 55 | **Cust./Item Disc. %** |
| 39 | ***Purchase Line*** | 26 | **Quantity Disc. %** |
| 99 | ***Item Vendor*** | 5400 | **Unit of Measure Code** |
| 99 | ***Item Vendor*** | 3 | **Currency Code** |
| 99 | ***Item Vendor*** | 4 | **Starting Date** |
| 110 | ***Sales Shipment Header*** | 36 | **Allow Quantity Disc.** |
| 111 | ***Sales Shipment Line*** | 9 | **Quantity Disc. Code** |

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 111 | ***Sales Shipment Line*** | 26 | **Quantity Disc. %** |
| 111 | ***Sales Shipment Line*** | 43 | **Allow Quantity Disc.** |
| 111 | ***Sales Shipment Line*** | 55 | **Cust./Item disc. %** |
| 112 | ***Sales Invoice Header*** | 36 | **Allow Quantity Disc.** |
| 113 | ***Sales Invoice Line*** | 9 | **Quantity Disc. Code** |
| 113 | ***Sales Invoice Line*** | 26 | **Quantity Disc. %** |
| 113 | ***Sales Invoice Line*** | 43 | **Allow Quantity Disc.** |
| 113 | ***Sales Invoice Line*** | 55 | **Cust./Item disc. %** |
| 114 | ***Sales Cr. Memo Header*** | 36 | **Allow Quantity Disc.** |
| 115 | ***Sales Cr. Memo Line*** | 9 | **Quantity Disc. Code** |
| 115 | ***Sales Cr. Memo Line*** | 26 | **Quantity Disc. %** |
| 115 | ***Sales Cr. Memo Line*** | 43 | **Allow Quantity Disc.** |
| 115 | ***Sales Cr. Memo Line*** | 55 | **Cust./Item disc. %** |
| 121 | ***Purch. Rcpt. Line*** | 26 | **Quantity Disc. %** |
| 123 | ***Purch. Inv. Line*** | 26 | **Quantity Disc. %** |
| 125 | ***Purch. Cr. Memo Line*** | 26 | **Quantity Disc. %** |
| 6651 | ***Return Shipment Line*** | 26 | **Quantity Disc. %** |
| 6660 | ***Return Receipt Header*** | 36 | **Allow Quantity Disc.** |
| 6661 | ***Return Receipt Line*** | 9 | **Quantity Disc. Code** |
| 6661 | ***Return Receipt Line*** | 26 | **Quantity Disc. %** |
| 6661 | ***Return Receipt Line*** | 43 | **Allow Quantity Disc.** |
| 6661 | ***Return Receipt Line*** | 55 | **Cust./Item disc. %** |

- Copies all records from the ***Item Vendor*** table to the ***Temp Item Vendor*** table.
- Copies all records from the ***Price Group*** table to the ***Temp Price Group*** table.
- Copies the **Cust./Item Disc. Gr.** and **Allow Quantity Disc.** fields from the ***Customer*** table to the ***Temp Customer*** table for customers that have the **Allow Quantity Disc.** field set to Yes.
- Copies the **Cust./Item Disc. Gr.** and **Allow Quantity Disc.** fields from the ***Customer Template*** table to the ***Temp Customer Template*** table for customer templates that have the **Allow Quantity Disc.** field set to Yes.
- Transfers all records from the ***Item Price*** table to the ***Temp Item Price*** table.
- Transfers records (that are associated to items) from the ***Item Sales Qty. Disc.*** table to the ***Temp Item Sales Qty. Disc.*** table. The ***Item Sales Qty. Disc. Temp*** table will also record the item number for these quantity discounts.
- Transfers the **Line Discount Calculation** field to the ***Temp Sales & Receivables Setup*** table and sets it to the *Qty. Disc. + Cust./Item Disc.* option in the ***Sales &***

***Receivables Setup*** table. (This is only used in the calculation of the upgraded line discount.)

## Bins

Codeunit 104045 performs the following tasks:

- Transfers all bin related information from the ***Item Ledger Entry*** table to a temporary table and deletes the bin code from the base table.
- Blanks the following fields from the following tables:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 337 | ***Reservation Entry*** | 5402 | **Bin Code** |
| 281 | ***Phys. Inventory Ledger Entry*** | 5403 | **Bin Code** |
| 5802 | ***Value Entry*** | 5403 | **Bin Code** |

- Transfers all entries in the ***Bin Code*** table to a temporary table.

Navision Attain 3.60

Codeunit 104045 performs the following tasks:

- Transfers all entries in the ***Location*** table to a temporary table, and clears the following field:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 14 | ***Location*** | 7300 | **Use Zones and Bins** |

## Manufacturing

Codeunit 104045 performs the following tasks:

- Checks that the ***Consumption Journal*** table is empty.
- Checks that the ***Output Journal*** table is empty.
- Checks that the ***Capacity Journal*** table is empty.
- Checks that, for outstanding subcontracted purchase order lines, if the item has an associated item tracking code in the ***Temp Item Tracking Code*** table, that the subcontracted purchase order lines are completely received and invoiced.
- Checks that there are no partially invoiced outstanding subcontracted purchase order lines.
- Blanks the fields in the following tables:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 99000765 | ***Manufacturing Setup*** | | |
| | | 24 | **Finished Order Nos.** |
| 242 | ***Source Code Setup*** | | |

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| | | 5407 | **Compress Work Center Ledg.** |
| | | 5408 | **Compress Machine Center Ledg.** |
| | | 5409 | **Compress Prod. Order Ledger** |
| 99000793 | ***Prod. Order Capacity Need*** | | |
| | | 15 | **Cost Amount** |
| | | 25 | **Overhead Amount** |
| | | 24 | **Direct Cost Amount** |
| | ***Prod. Order Component*** | | |
| | | | **Completely Consumed** |

- Transfers records with the following non-zero fields and its key information from table 99000792, ***Prod. Order Routing Line***, to the ***Temp Prod. Order Routing Line*** table:

| Field No. | Field Name |
|---|---|
| 90 | **Expected Operation Cost Amt.** |
| 91 | **Expected Capacity Need** |
| 96 | **Expected Capacity Ovhd Cost** |

- Transfers uninvoiced subcontracting purchase receipt lines from table 121, ***Purch. Rcpt. Line***, to the ***Temp SubCntr. Purch. Rcpt. Ln.*** table.

## Relationship Management

Codeunit 104045 performs the following tasks:

- Deletes the **Title** field. With the new Salutation feature, you can make salutations that consist of both *Dear* and *Mr.* - for example, *Dear Mr. John Roberts*.

  Documents where the **Contact Salutation** field is used must be changed to fit the new Salutation feature. Because the new salutation will include *Dear* and a comma after the name, you no longer need to add the *Dear* or comma in your documents.

- Replaces the **Virtual Customer** record with a new customer template, and thereby improves the *Quotation to Contacts* feature.

  If the **Virtual Customer** record is used on any sales quotes, then it must be specified under ***Relationship Management Setup*** in order to successfully upgrade the **Virtual Customer** record to a customer template in Navision 4.0.

- Creates the specified **Virtual Customer** under ***Relationship Management Setup*** as a customer template, and updates sales quotes containing **Virtual Customer** records with the new customer template.

- Move the interaction template setup from the Relationship Management Setup to a temporary table.

- Transfers the **Team Code** from team To-Dos with a **Type of Meeting** to a temporary table so a sales person from the team can be set up as the meeting organizer for the to-do.

### Service Management

Codeunit 104045 performs the following tasks:

- Introduces a new feature called *Customer Templates*, that is used in connection with quotations for contacts. In Navision Attain 3.01, Service Management used a customer template using blocked customers.
- Copies the numbers of all the customers that have been set up as service customer templates to the ***Temp Service Customer Template*** table and removes the **Service Customer Template** field from every template.

**Note**

Do not delete the customers as they are used by the Step 2 upgrade tool codeunits.

- Removes data on Service Order and Posted Service Order regarding Service Price Management.
- Transfers the existing service pricing data to the new service pricing tables (6080-8083). After the transfer of data, the codeunit deletes the data in the old service pricing tables (5921-5926). Codeunit 104048 transfers data between tables as shown below. After the transfer of data the codeunit deletes the data in the old tables.

| Table ID | Field No. | Name | | Table ID | Field No. | Name |
|---|---|---|---|---|---|---|
| 5921 | | ***Price Adjustment Group*** | > | 6082 | | ***Service Price Adjustment Group*** |
| | 1 | **Code** | | | 1 | **Code** |
| | 3 | **Description** | | | 2 | **Description** |
| 5922 | | ***Price-Adjusted Serv. Inv. Line*** | > | 6083 | | ***Serv. Price Adjustment Detail*** |
| | 1 | **Type** | | | 2 | **Type** |
| | 2 | **No.** | | | 3 | **No.** |
| | 3 | **Work Type Code** | | | 4 | **Work Type** |
| | 6 | **Gen. Prod. Posting Group** | | | 5 | **Gen. Prod. Posting Group** |
| | 7 | **Price Adjustment Group Code** | | | 1 | **Price Adjustment Group Code** |

| Table ID | Field No. | Name | | Table ID | Field No. | Name |
|---|---|---|---|---|---|---|
| 5924 | | ***Service Price Group Code*** | > | 6080 | | ***Serv. Price Group Code*** |
| | 1 | **Code** | | | 1 | **Code** |
| | 2 | **Description** | | | 2 | **Description** |
| 5925 | | ***Service/Cust. Price Group*** | > | 6081 | | ***Serv. Price Group Setup*** |
| | 1 | **Service Price Group Code** | | | 1 | **Service Price Group Code** |
| | 2 | **Customer Price Group Code** | | | 3 | **Cust. Price Group Code** |
| | 3 | **Starting Date** | | | 5 | **Starting Date** |

- Transfers service items, which related to Contracts, to a new table called ***Service Contract Line*** table.
- Changes the option value for **Contract Type** fields in tables related to Service Contract (*Quote, Order)*.
- Changes option value for **Contract Type** fields in tables related to Filed Service Contract (*Quote, Order)*.
- Replaces **Account No.** and **Prepaid Account No.** fields from ***Service Ledger Entry*** table and tables related to Service Contract with a relation to a new table called ***Service Price Adjustment Group*** table.
- Redesigns and moves ***Filed Service Contract Line*** table to a temporary table. And deletes data in ***Filed Service Contract Line*** table.
- Clear fields related to customer, item, service contract and job if the related record has been deleted after posting.
- Clears the **Detailed Ledger** field in the ***Service Contract Header*** table. From version 4.0, service ledger entries are always created as detailed, so the field is deleted.

## Warehouse Management

Codeunit 104045 performs the following tasks:

Navision 2.00-3.10

- If the Warehouse Location Filter in the ***User Setup*** table is filled, codeunit 104045 transfers this Warehouse Location Filter information from the ***User Setup*** table to a temporary table and deletes this information from the ***User Setup*** table. This is done because the warehouse user assignment is handled in a different table in version 4.0.
- Transfers all entries in the ***Whse. Request*** table to a temporary table and deletes them in the ***Whse. Request*** table. This is done because warehouse requests for released production orders are stored in a different table in version 4.0. Also the source document definition of warehouse requests is changed.

- Blanks the following information in the ***Warehouse Setup*** table, because these fields are deleted in version 4.0:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 5769 | ***Warehouse Setup*** | | |
| | | 4 | **Whse. Assign Nos.** |
| | | 8 | **Item Journal Template** |
| | | 9 | **Item Journal Batch** |
| | | 11 | **Cross-Dock Due Date Calc.** |
| | | 12 | **Use Cross-Dock** |
| | | 16 | **Autofill Qty. to Handle** |

Transfers all entries in the ***Warehouse Source Filter*** table to a temporary table and deletes them in the ***Warehouse Source Filter*** table. This is done because its date filter handling is changed in version 4.0. Also you cannot use warehouse filter for released production orders anymore, because warehouse requests for released production orders are handled in a different table.

- Transfers all entries in the ***Posted Whse. Activity Header*** table to a temporary table and deletes them in the ***Posted Whse. Activity Header*** table. This is done because the type of posted warehouse activity headers is changed.
- Transfers all entries in the ***Posted Whse. Activity Line*** table to a temporary table and deletes them in the ***Posted Whse. Activity Line*** table. This is done because the type of posted warehouse activity lines and the way item tracking information is stored for posted warehouse activities are changed.
- Transfers all entries in the ***Posted Whse. Activ. Trkg. Line*** table to a temporary table and deletes them in the ***Posted Whse. Activ. Trkg. Line*** table. This is done because the way item tracking information is stored for posted warehouse activities is changed.
- Transfers all entries in the ***Warehouse Comment Line*** to a temporary table.

# Appendix B

## Codeunit 104048, *Upgrade 4.00 Step 2*

This appendix describes the tasks that the Upgrade Toolkit performs when you run codeunit 104048, ***Upgrade 4.00 Step 2***.

## B.1 CODEUNIT 104048, *Upgrade 4.00 Step 2*

The tasks performed by the codeunit are grouped into functions areas.

### Debit and Credit Amounts in Ledger Entries

Navision 2.00

- When you run codeunit 104048, it will run codeunit 104040, ***Upgrade 2.00 Step 2***.

This codeunit updates errors in the debit amount and credit amount fields in the existing G/L Entries and Bank Ledger entries so that they can be used in the 4.0 version.

### Business Relations and Relationship Management No. Series

Navision 2.00-2.60 and Navision Manufacturing 2.60

- When you run codeunit 104048, it will run codeunit 104060, ***Init. RM Setup***.

This codeunit inserts a default value for the **Contact No. Series** and **To-Do No. Series** and transfers the following values from the old Contact Management Setup to the Relationship Management Setup.

**Default Customer Status** field to the **Bus. Rel. Code for Customers** field.

**Default Vendor Status** field to the **Bus. Rel. Code for Vendors** field.

For the default customer and vendor status codes, the **Code** and **Description** field values are also transferred from table 5001, ***Prospect Status*** to table 5053, ***Business Relation***.

### Item Ledger Entries, Value Entries and Item Tracking

Codeunit 104048 perform the following tasks:

- Transfer the **Lot No.** and **Serial No.** information from the temporary table to the appropriate item tracking tables, for example, the ***Item Tracking Line*** table and delete the records in the temporary table.
- Fills the following new fields with appropriate values:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 37 | ***Sales Line*** | 5752 | **Completely Shipped** |
| 39 | ***Purchase Line*** | 5752 | **Completely Received** |
| 111 | ***Sales Shipment Line*** | 58<br>68 | **Qty. Shipped Not Invoiced**<br>**Bill-to Customer No.** |
| 121 | ***Purch. Rcpt. Line*** | 58 | **Qty. Rcd. Not Invoiced** |

- Transfers the value related information from the temporary tables to the ***Value Entry*** table. The codeunits also update the ***Item Application Entries*** table and delete the records in the temporary table. Furthermore, the **Average Cost** field on the ***Item Card*** is updated, and the value related entries in the ***Item Ledger Entry*** table that are now redundant are deleted.

**Note**

When upgrading from version 2.00-2.60, the codeunit 104048, ***Upgrade 4.00 Step 2,*** does not fill the **Valuation Date** field in the ***Value Entry*** table. This is because filling this field automatically through the upgrade would be a rough estimate at best and could therefore lead to incorrect cost calculation when calculating average costs in the future.

- The following fields are cleared:

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 94 | ***Inventory Posting Group*** | 6 | **Inventory Account** |
| 32 | ***Item Ledger Entry*** | 9 | **Inventory Posting Group** |
| 32 | ***Item Ledger Entry*** | 10 | **Source Posting Group** |
| 32 | ***Item Ledger Entry*** | 15 | **Unit Amount** |
| 32 | ***Item Ledger Entry*** | 16 | **Unit Cost** |
| 32 | ***Item Ledger Entry*** | 17 | **Amount** |
| 32 | ***Item Ledger Entry*** | 21 | **Discount Amount** |
| 32 | ***Item Ledger Entry*** | 22 | **Salespers./Purch. Code** |
| 32 | ***Item Ledger Entry*** | 24 | **User ID** |
| 32 | ***Item Ledger Entry*** | 25 | **Source Code** |
| 32 | ***Item Ledger Entry*** | 30 | **Cost is Adjusted** |
| 32 | ***Item Ledger Entry*** | 37 | **Closed by Entry No.** |
| 32 | ***Item Ledger Entry*** | 38 | **Closed by Date** |
| 32 | ***Item Ledger Entry*** | 39 | **Closed by Quantity** |
| 32 | ***Item Ledger Entry*** | 42 | **Adjusted Cost (Qty.)** |
| 32 | ***Item Ledger Entry*** | 43 | **Adjusted Cost (Invoiced Qty.)** |
| 32 | ***Item Ledger Entry*** | 44 | **Cost Posted to G/L** |
| 32 | ***Item Ledger Entry*** | 45 | **Journal Batch Name** |
| 32 | ***Item Ledger Entry*** | 46 | **Reason Code** |
| 32 | ***Item Ledger Entry*** | 48 | **Invt. Qty. Not Adjusted** |
| 32 | ***Item Ledger Entry*** | 49 | **Adjust Invoice Cost** |
| 32 | ***Item Ledger Entry*** | 53 | **Transferred from Entry No.** |
| 32 | ***Item Ledger Entry*** | 57 | **Gen. Bus. Posting Group** |
| 32 | ***Item Ledger Entry*** | 58 | **Gen. Prod. Posting Group** |
| 32 | ***Item Ledger Entry*** | 67 | **Add.-Currency Adj. Cost (Qty.)** |

| Table ID | Table Name | Field No. | Field Name |
|---|---|---|---|
| 32 | ***Item Ledger Entry*** | 68 | **Add.-Curr. Adj. Cost (Inv Qty)** |
| 32 | ***Item Ledger Entry*** | 69 | **Add.-Curr. Cost Posted to G/L** |
| 32 | ***Item Ledger Entry*** | 72 | **Additional-Currency Unit Cost** |
| 339 | ***Item Application Entry*** | 5 | **New Entry No.** |
| 339 | ***Item Application Entry*** | 6 | **Fixed Applied-to Entry No.** |
| 339 | ***Item Application Entry*** | 7 | **Fixed Application** |
| 339 | ***Item Application Entry*** | 8 | **Creation Date** |
| 339 | ***Item Application Entry*** | 9 | **Valid** |
| 339 | ***Item Application Entry*** | 10 | **Valuated** |
| 339 | ***Item Application Entry*** | 12 | **Reserved Quantity** |
| 339 | ***Item Application Entry*** | 13 | **Remaining Quantity** |
| 339 | ***Item Application Entry*** | 14 | **Open** |
| 339 | ***Item Application Entry*** | 15 | **Positive** |
| 339 | ***Item Application Entry*** | 17 | **Cost is Adjusted** |
| 339 | ***Item Application Entry*** | 18 | **Document No.** |
| 339 | ***Item Application Entry*** | 19 | **Entry Type** |
| 339 | ***Item Application Entry*** | 20 | **Item No.** |
| 339 | ***Item Application Entry*** | 22 | **Location Code** |

Codeunit 104048 performs the following tasks on the ***Value Entry*** table:

- Clears the **Sales Amount (Actual)** field except when the value in the **Item Ledger Entry Type** field is *Sale.*
- Changes the sign in the **Sales Amount (Actual)** field when the value in the **Item Ledger Entry Type** field is *Sale.*
- In versions 2.00 - 3.10, depending on the value in the **Expected Cost** field, transfers data between fields as follows:

| Transferred from field | Transferred to field |
|---|---|
| **Amount** | **Sales Amount (Expected)**<br>or **Sales Amount (Actual)** |
| **Adjusted Cost** | **Cost Amount (Expected)**<br>or **Cost Amount (Actual)** |
| **Adjusted Cost (ACY)** | **Cost Amount (Expected) (ACY)**<br>or **Cost Amount (Actual) (ACY)** |

- Totals the expected costs for a value entry and fills the **Cost Amount (Expected)**, **Cost Amount (Expected) (ACY)**, **Sales Amount (Expected)** and **Purchase Amount (Expected)** fields accordingly.

- Fills the **Item Ledger Entry Quantity** field with the quantity from the corresponding item ledger entry.

Navision Financials 2.60

Codeunit 104048 performs the following task:

- Updates date compressed item ledger entries so that they are valid for the current version of Navision. The date compression in Navision Financials sometimes created item ledger entries where the quantity and invoiced quantity had different signs. Codeunit 104048 therefore redistributes the invoiced quantity of the date compressed entries to other existing item ledger entries.

Navision Financials and Navision Manufacturing 2.60

Codeunit 104048 performs the following tasks:

- Reads item tracking codes from table 104046 ***Temp Item Tracking Code*** and creates them in table 6502. Updates **Item Tracking Code** field in table 27. Deletes records from table 104046.
- Transfers **Serial No.** and **Lot No.** fields from table 104048 to table 32. Deletes records from table 104048.
- Transfers item entry relation records from table 104093 to table 6507. Deletes records from table 104092.
- Creates value entry relation records in table 6508 for posted sales and purchase invoices that have item tracking.
- Inserts reservation entries from table 104093 into table 337 if they are not already there. Deletes records from table 104093.
- Inserts tracking specification records from table 104094 into table 336. Deletes records from table 104094.

  See Appendix D on page 134 for a specification of the 2.60-4.0 upgrade concerning Item Tracking.

Navision Manufacturing 2.60

Codeunit 104048 performs the following task:

- Fills the **Item Rcpt. Entry No.** field of a subcontracting purchase receipt line with the **Entry No.** of the linked capacity ledger entry.

- Uses the data in the temporary upgrade tables to split item ledger entries and value entries, and place data in table 336, 337, 6507 and 6508.

See *Appendix D* on page 134 for a specification of the 2.60-4.0 upgrade concerning Item Tracking.

Navision 3.01 and later versions

Codeunit 104048 performs the following tasks:

- If there are any value entries where the valuation date is 12.31.9999, the codeunit resets the valuation date to the posting date of the value entry.
- If the **Expected Cost Posting to G/L** field is set to yes in the inventory setup, the codeunit fills the **Expected Cost Posted to G/L** and **Exp. Cost Posted to G/L (ACY)** fields in the ***Value Entry*** table with the expected cost amount.

## Date Formulas and Date-Time Fields

Codeunit 104048 perform the following tasks:

- Copies the date formulas from the temporary tables into the new date formula fields in the base tables, and deletes the records in the temporary tables.
- Calculates and updates the new **Date-Time** fields in several tables based on the information in the existing **Date** and **Time** fields. These new **Date-Time** fields combine the values in the separate date and time fields into one field.

## Detailed Customer and Vendor Ledger Entries

Codeunit 104048 perform the following tasks:

- Copies the amount information from the temporary tables into new ***Detailed Cust. Ledg. Entry*** table and the ***Detailed Vendor Ledg. Entry*** table and deletes the records in the temporary tables. The fields that used to contain amounts in the ***Cust. Ledger Entry*** table and the ***Vendor Ledger Entry*** table are cleared and are now defined as FlowFields, which are used to evaluate the same numbers as before.

  In earlier versions of Navision Financials or Navision Manufacturing, when a payment discount was given as a result of two customer ledger entries, for example, an invoice entry and a payment entry being applied where the currency is not the same in each entry, the customer ledger entries do not reflect the payment discount in the foreign currency in which the payment is made, but only in the local currency used in the invoice (Payment Disc. Given (LCY)). Therefore the detailed customer ledger entries that have been created in Navision will not reflect this in the same way as they would if they had been posted in Navision 4.0. The amount (LCY) in a Detailed Customer Ledger Entry will reflect the Payment Discount Amount (LCY), but the Amount in foreign currency will be 0. The amounts (FlowFields) on the customer ledger entry will show the correct figures despite this deviation in the detailed customer ledger entries.

  This also applies to vendor ledger entries.

## Reservations

Codeunit 104048 perform the following task:

- Copies reservation entries from the temporary table to the ***Reservation Entry*** table. The entries in the temporary table are deleted.
- The new imported tables contain the Navision 4.0 table definitions. The codeunit converts the data in the previously imported temporary tables and transfers the converted into these new tables.

**Note**

In version 4.0, there is a new field in the ***Reservation*** table called **Reserved Pick & Ship Qty.** The program fills this field with the pick quantity when the user creates a pick line, and it maintains this field when the user registers the pick and places the quantity in the shipping bin. This field helps to protect against double allocation when a given quantity of an item is allocated both by reservations and in the warehouse.

When the user starts to use the program after upgrading to 4.0, this field is empty. Consequently, the user may find that the availability is under-calculated when first using it, but only regarding the reserved orders that have been picked and have not left the warehouse/inventory. After approximately 1 to 3 days, when the items that were picked before the update are shipped, the calculation of available quantity will work correctly.

The less time it takes from the time the pick is created until it is finally shipped (or consumed), the sooner the program will adjust itself.

### Bins

Codeunit 104048 perform the following task:

- Copies the bin codes from the temporary table to the ***Bin*** table. The entries in the temporary table are deleted.
- Totals the bin quantity information in the temporary table for item ledger entries. If bin quantities are found, it creates a bin in the ***Bin*** table (if one does not exist already), it creates bin content in the ***Bin Content*** table for the item and bin, it creates appropriate warehouse entries in the ***Warehouse Entry*** table to correspond to the item ledger entries, and it sets the **Bin Mandatory** field to true in the ***Location*** table.

### Dimensions

Codeunit 104048 perform the following tasks:

- Copies the department and project codes from tables such as the ***Customer*** table and the ***Sales Header*** table into the dimension sub-tables. In the ***General Ledger Setup*** table, Department Code is set up to be Global Dimension 1 Code and Project Code is set up to be Global Dimension 2 Code. This is done because department and project codes are no longer stored only in the base tables. Department and project codes are now treated as dynamic/replaceable global dimensions.

  The information in the temporary table is also copied to the ***Default Dimension*** table, and the records in the temporary table are deleted.

### Sales and Purchase Comments

Codeunit 104048 perform the following task:

- Ensures that properties stored in the ***Sales Comment Line*** table and the ***Purch. Comment Line*** table are not changed. This means that records stored as Shipment, Receipt, Posted Invoice or Posted Credit Memo are still stored with that property rather than as a return order.

## Manufacturing

Codeunit 104048 perform the following task:

Navision Manufacturing 2.60 and Navision 3.01

- Deletes all records in the ***Prod. Order Register*** table
- Deletes all records in the ***Capacity Register*** table
- Transfers all table data from the objects that were renamed earlier to the following new tables:

| Old Table ID | New Table ID | New Table Name |
|---|---|---|
| 99000791 | 5405 | ***Production Order*** |
| 99000792 | 5409 | ***Prod. Order Routing Line*** |
| 99000793 | 5410 | ***Prod. Order Capacity Need*** |
| 99000806 | 5411 | ***Prod. Order Routing Tool*** |
| 99000807 | 5412 | ***Prod. Order Routing Personnel*** |
| 99000808 | 5413 | ***Prod. Order Rtng Qlty Meas.*** |
| 99000809 | 5414 | ***Prod. Order Comment Line*** |
| 99000810 | 5415 | ***Prod. Order Rtng Comment Line*** |
| 99000811 | 5416 | ***Prod. Order BOM Comment Line*** |

- In the ***Production Document Dimension*** table, the codeunit changes the **Table ID** field from 90000791 to 5405.
- Transfers data between tables as follows:

| Transferred from table | Transferred to table | Description |
|---|---|---|
| ***Finished Production Order*** | ***Production Order*** | Status set to Finished |
| ***Finished Prod. Order Line*** | ***Prod. Order Line*** | Status set to Finished |
| ***Fnshd Prod. Order Rtng Line*** | ***Prod. Order Routing Line*** | Status set to Finished |
| ***Fnshd Prod. Order Comp.*** | ***Prod. Order Component*** | Status set to Finished |
| ***Fnshd Prod. Order Rtng Tool*** | ***Prod. Order Routing Tool*** | Status set to Finished |
| ***Fnshd Prod. Order Rtng Persnl*** | ***Prod. Order Routing Personnel*** | Status set to Finished |
| ***Fnshd Prod. Order Rtng Qlty Meas.*** | ***Prod. Order Rtng Qlty Meas.*** | Status set to Finished |
| ***Fnshd Prod. Order Cmt. Ln.*** | ***Prod. Order Comment Line*** | Status set to Finished |

| Transferred from table | Transferred to table | Description |
|---|---|---|
| ***Fnshd Prod. Order Rtng Cmt. Ln.*** | ***Prod. Order Rtng Comment Line*** | Status set to Finished |
| ***Fnshd Prod. Order BOM Cmt. Line*** | ***Prod. Order BOM Comment Line*** | Status set to Finished |
| ***Posted Prod. Doc. Dimension*** | ***Production Document Dimension*** | Status set to Finished |
| ***Fnshd Prod. Order Ledg. Entry*** | ***Prod. Order Ledger Entry*** | It will also move the corresponding associated dimensions. |

- Transfers all table data from the **Prod. Order Ledger** table using the following criteria:

  All production order ledger entries of type *Consumption* will update the following related item ledger entry fields:

| Field Name |
|---|
| **Prod. Order No.** |
| **Prod. Order Line No.** |
| **Prod. Order Comp Line No.** |

All production order ledger entries of type *Input/Output* to do with output will have the following related item ledger entry fields updated:

| Field Name |
|---|
| **Prod. Order No.** |
| **Prod. Order Line No.** |

All production order ledger entries of type *Input/Output* representing capacity will:

- insert capacity ledger entries with corresponding value entries (for direct cost and indirect cost) and their associated dimensions.

- insert value entries (one direct cost entry and the other a variance entry) that were posted earlier as *Purchase Variance* to *Subcontracted Variance* related to the output item.

- Transfers table data from the ***Temp. Prod. Order Rtng Line*** table to the following fields in the ***Prod. Order Routing Line*** table:

| Field ID | Field Name |
|---|---|
| 90 | **Expected Operation Cost Amt.** |
| 91 | **Expected Capacity Need** |
| 96 | **Expected Capacity Ovhd Cost** |

- Deletes all records in the ***Work Center Ledger Entry*** table.
- Deletes all records in the ***Machine Center Ledger Entry*** table.
- Deletes all records in the ***Average Cost Adjustment*** table, that refer to an item that does not use costing method Average.

## Prices and Discounts

**Important**

The pricing feature uses the ’best price’ method. This is a method to resolve a conflict between more than one valid sales or purchase price and/or line discount. The system resolves this conflict by choosing the lowest applicable price possible. In addition, there is a built-in assumption in the new pricing functionality that prices are lower when the Sales Type is *Customer Group* (*Price* or *Discount*) than when the Sales Type is *All Customers*. What this means is that a conflict can occur in the upgrade if an item with a *Blank* sales type had a unit price (for example, 100 LCY) that was lower than the unit price for the item with a *Customer Group* sales type (for example, 110 LCY). This is because of the best price method.

Codeunit 104048 performs the following tasks:

- Transfers data between tables as follows:

| Transferred from table | Transferred to table | Description |
|---|---|---|
| ***Temp Item Price*** | ***Sales Price*** | Prices with blank Price Group will be converted to Sales Type = All Customers. It will also set an ending date if two prices exist that have similar conditions, but have different starting dates. |
| ***Cust./Item Discount*** | ***Sales Line Discount*** | |
| ***Temp Item Sales Qty. Disc.*** | ***Sales Line Discount*** | All Discounts will be converted to specific Items and the Sales Type will be All Customers. |
| ***Temp Item Vendor*** | ***Purchase Price*** | It will set an ending date if two prices exist that have similar conditions, but have different starting dates. |
| ***Item Purch. Qty. Disc.*** | ***Purchase Line Discount*** | |

- Creates sales line discounts, which will depend both on the line discount calculation in the ***Sales & Receivables Setup*** window and whether there were items that had both a Cust./Item Disc Gr. and a Sales Qty. Disc. Gr. Setup.
- Creates new price groups, when relevant.

- Updates the **Allow Line Disc.** field in the ***Customer Price Group*** table by using the information from the **Allow Quantity Disc.** and **Cust./Item Disc. Gr.** fields in the ***Temp Customer Price Group*** table.
- Updates the **Allow Line Disc.** and **Customer Disc. Group** fields in the ***Customer*** table by using the information from the **Allow Quantity Disc.** and **Cust./Item Disc. Gr.** fields in the ***Temp Customer*** table.
- Updates the **Allow Line Disc.** and **Customer Disc. Group** fields in the ***Customer Template*** table by using the information from the **Allow Quantity Disc.** and **Cust./Item Disc. Gr.** fields in the ***Temp Customer Template*** table.

### Relationship Management

Codeunit 104048 performs the following tasks:

- Assigns all contact Companies with the Relationship Management Setup Def. Company Salutation Code.
- Assigns all contract persons with the salutation code that corresponds to the title code they had before.
- Reorganizes the data in the to-do table to fit the new structure. The existing to-do is changed into a to-do of the Organizer system type. If there is a contact linked to the to-dos, the codeunit creates a new to-do of the Contact Attendee system type. Finally, it creates Attendee records for meetings.

Navision Attain 3.01

Codeunit 104048 performs the following task:

- Creates a customer template based on the old **Virtual Customer** record.
- Updates all sales quotes that were using the old virtual customer template with the new customer template.
- Changes existing team to-dos into a to-do of the Organizer system type. For team meeting to-dos, new to-dos are created with the Organizer and Salesperson Attendee system type and creates Attendee records for the salespeople in the team

### Service Management

Codeunit 104048 performs the following tasks:

- Transfers all customers from service customer templates into the ***Customer Template*** table.

**Important**

You must manually delete these customers if they are not going to be used in the customer database (look out for ledger entries or other data that the customer wished to keep).

- Redesigns Service Comment Line to be consistent with the rest of the application and therefore upgrades data in Service Comment Line.

- Updates new **Document Type** field in ***Service Order Log*** window with correct option value (*Quote,Order*).
- Updates new **Document Type** field in ***Loaner*** window with correct option value (*Quote,Order*).
- Updates new **Document Type** field in ***Loaner Entry*** window with correct option value (*Quote,Order*).
- Updates new **Document Type** field in ***Service Item Log*** window with correct option value (*Quote,Order*).
- Updates new **Document Type** field in ***Service Order Allocation*** window with correct option value (*Quote,Order*).
- Moves ***Service Contracts*** table with type *Template* to new ***Service Contract Template*** table.
- Changes the option value for **Contract Type** field in ***Service Order Log*** table, ***Service Item Log*** table and ***Contract Change Log*** table (*Quote*,*Order*).
- Fills ***Filed Service Contract Line*** table with data from the temporary table created by codeunit 104045.
- Changes the option value for **Type** field in ***Service Contract Dimension*** table (*Quote*,*Order*).
- Moves dimension for Service Contract Template from ***Service Contract Dimension*** table to ***Default Dimension*** table.
- Changes the option value for **Contract Type** field in ***Contract/Service Discount*** table (*Quote*,*Order,Template*).
- Updates new and old fields for Discount handling in relevant tables.
- Corrects the calculated **Actual Response Time (Hours)** field and the Service Time (Hours) field on the un-posted Service Orders.
- Updates the Service Item Log for entries indicating the Service Item have been added to a contract.
- Updates new **Line Cost** field in Service Contract Line with correct value.
- Updates new **Line Discount Amount** field in Service Contract Line with correct value.
- Updates new **Profit** field in Service Contract Line with correct value.

### Warehouse Management

Codeunit 104048 performs the following tasks:

- If the **Require Pick** field on the ***Location*** table was set to true in 3.01, the codeunit sets both the **Require Pick** field as well as the new **Require Shipment** field to true on the ***Location*** table.
- If the **Require Pick** field on the ***Warehouse Setup*** table was set to true in 3.01, the codeunit sets both the **Require Pick** field as well as the new **Require Shipment** field on the table to true.

- Creates a warehouse employee for each user, where Location Filter information is found in the temporary ***User Setup*** table and deletes all entries in the temporary table.
- For released production orders, the codeunit creates entries in the ***Pick Request*** table according to the warehouse request entries in the temporary table.
  It creates entries in the ***Whse. Request*** table according to the temporary table for all other source documents and converts their source document information.
  All entries in the temporary table are deleted.
- Creates entries in the ***Warehouse Source Filter*** table using the temporary table. Because Released Production Orders are no longer handled in the ***Whse. Request table*** the according information is not transferred.
  The filling of the **Qty. to Handle** fields in the warehouse is inverted. **Qty. to Handle** field is now filled by default. Therefore the meaning of field 15, **Assign Available Inventory** is inverted. This field is renamed to **Do Not Fill Qty. to Handle** and the information is inverted.
  The date filter handling is changed in the ***Warehouse Source Filter*** table. New fields are added, where you can enter the date filters. The new fields are filled with the information of the old fields as follows:

| New field No. | New Field Name | Old Field No. | Old Field Name |
|---|---|---|---|
| 7300 | **Planned Delivery Date** | 10 | **Planned Delivery Date Filter** |
| 7301 | **Planned Shipment Date** | 21 | **Planned Shipment Date Filter** |
| 7302 | **Planned Receipt Date** | 22 | **Planned Receipt Date Filter** |
| 7303 | **Expected Receipt Date** | 23 | **Expected Receipt Date Filter** |
| 7304 | **Shipment Date** | 24 | **Shipment Date Filter** |
| 7305 | **Receipt Date** | 25 | **Receipt Date Filter** |
| 7306 | **Sales Shipment Date** | 28 | **Sales Shipment Date Filter** |

The old fields are now defined as flow filters. All entries in the temporary table are deleted.

- Creates entries in the ***Registered Whse. Activity Hdr.*** table using the temporary table, converts their types and deletes all entries in the temporary table.
- Creates entries in the ***Registered Whse. Activity Line*** table using the temporary table. The item tracking information of registered warehouse activity lines are now stored in the lines themselves. If item tracking information exists for an entry in the temporary table this entry is split according to the quantity in the ***Temp Posted Whse. Item Tracking Line*** table and the item tracking information is transferred to the newly created ***Registered Whse. Activity Line*** table. All entries in the ***Temp Registered Whse. Activity Line*** table and ***Posted Whse. Item Tracking Line*** table are deleted.
- Fills the new fields **Qty. Picked**, **Qty. Picked (Base)** and **Completely Picked** in the ***Prod. Order Component*** table for all lines that have to be handled in the

warehouse. This includes all component lines, where the flushing method is *Manual*, the planning level code is zero and that picking is required according to the location setup.
In version 4.0 the information, what is already picked, is stored in the component line.
Qty. Picked of a component line is calculated as the difference of Expected Quantity and Remaining Quantity. If Qty. Picked is equal to the Expected Quantity the line is Completely Picked.

**Note**

When you register a pick for a released production order the consumption is no longer posted automatically. This is changed to decouple the production and the warehouse process.

Navision Attain 3.10 Codeunit 104048 performs the following tasks:

- If the **Require Pick** field on the ***Location*** table was set to true in 3.10, the codeunit sets both the **Require Pick** field as well as the new **Require Shipment** field to true on the ***Location*** table.
- If the **Require Pick** field on the ***Warehouse Setup*** table was set to true in 3.10, the codeunit sets both the **Require Pick** field as well as the new **Require Shipment** field on the table to true.
- If the **Require Pick** field on the ***Location*** table was set to true in 3.60, the codeunit sets both the **Require Pick** field as well as the new **Require Shipment** field to true on the ***Location*** table.
- If the **Require Pick** field on the ***Warehouse Setup*** table was set to true, the codeuit sets both the **Require Pick** field as well as the new **Require Shipment** field on the table to true.

Navision Attain 3.60 Codeunit 104048 performs the following task:

- If the **Use Zones and Bins** field on the ***Location*** table was set to true in 3.60, the codeunit sets the fields **Bin Mandatory** and **Directed Put-away and Pick** to true.

# Appendix C

## Costing Foundation in 3.10 and later versions

In order to reduce the complexity of the costing functionality, we have eliminated some redundant information in the Navision Manufacturing area. We achieved this by combining objects and fields with similar behaviour patterns and deleting those that were irrelevant or could be avoided.

This appendix contains a number of tables. These show how the information is transferred from entries in the deleted tables to new or existing tables.

## C.1 FIELD MAPPING FROM DELETED TABLES

The following tables show how information is transferred from deleted entries to new or existing entries.

The following abbreviations are used in the text:

ILE = Item Ledger Entry

CLE = Capacity Ledger Entry.

Where there is no mention of a **Mapped to Table ID**, the field has been deleted

### Table 99000794, *Prod. Order Ledger Entry* (Deleted)

At present, the ***Prod. Order Ledger Entry*** table records information related to capacity consumption made by machine center/work centers through prouction orders. In future, this information will be recorded in the ***Capacity Ledger Entry*** table (except for the value part, which will be recorded in the ***Value Entry*** table). The list below shows how the fields have been mapped from the ***Prod. Order Ledger Entry*** to the ***Capacity Ledger Entry***, the ***Item Ledger Entry*** and the ***Value Entry*** tables to make this transition possible.

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 1 | **Entry No.** | | | |
| 2 | **Prod. Order No.** | 32<br>5802<br>5832 | 5401<br>5401<br>50 | |
| 3 | **Operation No.T** | 5832 | 8 | |
| 4 | **Item No.** | 32<br>5802<br>5832 | 5<br>2<br>56 | **Source No.** |
| 5 | **Posting Date** | 32<br>5802<br>5832 | 3<br>3<br>3 | |
| 6 | **Document No.** | 32<br>5802<br>5832 | 6<br>6<br>6 | |
| 7 | **Entry Type** | | | |
| 8 | **Type** | 5832 | 4 | |
| 9 | **No.** | 32<br>5802<br>5832 | 2<br>2<br>2 | **Item No.**<br>**Item No.** |
| 10 | **Description** | 32<br>5802<br>5832 | 7<br>7<br>7 | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 11 | **Quantity** | | | |
| 12 | **Direct Unit Cost** | 5802 | 15 | **Cost per Unit** |
| 14 | **Cost Amount** | 5802 | 17 | **Amount** |
| 15 | **Work Center No.** | 5832 | 9 | |
| 16 | **Work Center Group Code** | 5832 | 68 | |
| 18 | **Starting Time** | 5832 | 43 | |
| 19 | **Ending Time** | 5832 | 44 | |
| 20 | **Concurrent Capacities** | 5832 | 19 | **Concurrent Capacity** |
| 21 | **Location Code** | 32<br>5802 | 8<br>8 | |
| 22 | **Bin Code** | 32<br>5802 | 5403<br>5403 | |
| 25 | **Variant Code** | 32<br>5802 | 5402<br>5402 | |
| 27 | **Self-supply** | | | |
| 28 | **Work Shift Code** | 5832 | 69 | |
| 31 | **Global Dimension 1 Code** | 32<br>5802<br>5832 | 33<br>33<br>33 | |
| 32 | **Global Dimension 2 Code** | 32<br>5802<br>5832 | 34<br>34<br>34 | |
| 33 | **Finished** | | | |
| 34 | **Setup Time** | | | |
| 35 | **Output Quantity** | | | |
| 36 | **Scrapped Quantity** | | | |
| 37 | **Appl.-to Item Entry** | 32 | 28 | **Applies-to Entry** |
| 38 | **Output Quantity (Base)** | 5832 | 16 | **Output Quantity** |
| 39 | **Scrapped Quantity (Base)** | 5832 | 17 | **Scrap Quantity** |
| 40 | **Item Unit of Measure Code** | 5832 | 58 | **Unit of Measure Code** |
| 41 | **Routing No.** | 5832 | 52 | |
| 42 | **Routing Reference No.** | 5832 | 53 | |
| 43 | **Prod. Order Line No.** | 32<br>5802<br>5832 | 5832<br>5881<br>51 | |
| 44 | **Status** | | | |
| 45 | **Prod. Order Comp. Line No.** | 32 | 5833 | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 53 | **Overhead Amount** | | | |
| 62 | **Source Code** | 5802 | 25 | |
| 66 | **Amt. Posted to G/L** | 5802 | 45 | **Cost Posted to G/L** |
| 74 | **Reason Code** | 5802 | 46 | |
| 75 | **User ID** | 5802 | 24 | |
| 76 | **Item Ledger Entry No.** | | | |
| 79 | **Gen. Bus. Posting Group** | 5802 | 57 | |
| 80 | **Gen. Prod. Posting Group** | 5802 | 58 | |
| 87 | **Document Date** | 32<br>5802<br>5832 | 60<br>60<br>60 | |
| 88 | **External Document No.** | 32<br>5802<br>5832 | 61<br>61<br>61 | |
| 90 | **Unit of Measure Code** | 32<br>5832 | 5407<br>28 | <br>**Cap. Unit of Measure Code** |
| 91 | **Run Time** | | | |
| 92 | **Qty. per Unit of Measure** | 5832 | 29 | **Qty. per Cap. Unit of Measure** |
| 93 | **Qty. per Item Unit of Measure** | 32<br>5832 | 5404<br>59 | **Qty. per Unit of Measure** |
| 94 | **Run Time (Base)** | 5832 | 12 | **Run Time** |
| 95 | **Setup Time (Base)** | 5832 | 11 | **Setup Time** |
| 96 | **Quantity (Base)** | 5832 | 10 | **Quantity** |
| 97 | **Invoiced Subcontr. Qty.** | 5832 | 15 | **Invoiced Quantity** |
| 98 | **Purchase Variance Amt.** | 5802 | 43 | **Adjusted Cost (Entry Type: Variance)** |
| 99 | **Variance Posted to G/L** | 5802 | 45 | **Cost Posted to G/L (Entry Type: Variance)** |
| 101 | **Ovhd. Cost Posted to G/L** | 5802 | 45 | **Cost Posted to G/L (Entry Type: Indirect Cost)** |
| 102 | **Last Output Line** | 5832 | 39 | |
| 104 | **Unit Cost (ACY)** | 5802 | 72 | **Cost per Unit (ACY)** |
| 107 | **Amt. Posted to G/L** | 5802 | 45 | **Cost Posted to G/L** |
| 108 | **Purchase Variance Amt. (ACY)** | 5802 | 68 | **Adjusted Cost (ACY) (Entry Type: Variance)** |
| 109 | **Variance Posted to G/L (ACY)** | 5802 | 70 | **Cost Posted to G/L (ACY) (Entry Type:Variance)** |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 111 | **Ovhd. Cost Posted to G/L (ACY)** | 5802 | 70 | **Cost Posted to G/L (ACY) (Entry Type: Indirect Cost)** |
| 112 | **Cost Amount (ACY)** | 5802 | 68 | **Adjusted Cost (ACY)** |
| 113 | **Overhead Amount (ACY)** | 5802 | 68 | **Adjusted Cost (ACY) (Entry Type: Indirect Cost)** |
| 150 | **Completely Invoiced** | 5832 | 42 | |
| 151 | **Applies-to Entry** | | | |
| 152 | **Remaining Quantity (Base)** | 32 | 13 | **Remaining Quantity** |

### Table 99000759, *Machine Center Ledger Entry* (Deleted)

The ***Machine Center Ledger Entry*** table presently records capacity consumption postings related to machine centers. The ***Capacity Ledger Entry*** table will replace both the ***Machine Center Ledger Entry*** and the ***Work Center Ledger Entry*** tables in recording capacity related entries. The list below shows how the fields have been mapped from the ***Machine Center Ledger Entry*** table to the new ***Capacity Ledger Entry*** table.

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 1 | **Entry No.** | 5832 | 1 | |
| 3 | **Machine Center No.** | 5832 | 2 | **No.** |
| 4 | **Posting Date** | 5832 | 3 | |
| 5 | **Document No.** | 5832 | 6 | |
| 6 | **Item No.** | 5832 | 56 | |
| 11 | **Description** | 5832 | 7 | |
| 12 | **Used Time** | | | |
| 13 | **Direct Unit Cost** | | | |
| 14 | **Unit Cost** | | | |
| 15 | **Cost Amount** | | | |
| 16 | **Work Center No.** | 5832 | 9 | |
| 17 | **Work Center Group Code** | 5832 | 68 | |
| 18 | **Starting Time** | 5832 | 43 | |
| 19 | **Ending Time** | 5832 | 44 | |
| 20 | **Concurrent CLE** | 5832 | 19 | **Concurrent Capacity** |
| 23 | **Prod. Order No.** | 5832 | 50 | |
| 25 | **Operation Type** | | | |
| 26 | **Stop Code** | 5832 | 65 | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 27 | **Scrap Code** | 5832 | 66 | |
| 28 | **Work Shift Code** | 5832 | 69 | |
| 31 | **Global Dimension 1 Code** | 5832 | 33 | |
| 32 | **Global Dimension 2 Code** | 5832 | 34 | |
| 35 | **Output Quantity** | | | |
| 36 | **Scrapped Quantity** | | | |
| 37 | **Item Unit of Measure Code** | 5832 | 58 | **Unit of Measure Code** |
| 38 | **Output Quantity (Base)** | 5832 | 16 | **Output Quantity** |
| 39 | **Scrapped Quantity (Base)** | 5832 | 17 | **Scrap Quantity** |
| 62 | **Source Code** | | | |
| 73 | **Journal Batch Name** | | | |
| 74 | **Reason Code** | | | |
| 75 | **User ID** | | | |
| 87 | **Document Date** | 5832 | 60 | |
| 88 | **External Document No.** | 5832 | 61 | |
| 90 | **Unit of Measure Code** | 5832 | 28 | **Cap. Unit of Measure Code** |
| 91 | **Used Time (Base)** | 5832 | 10 | **Quantity** |
| 92 | **Time per Cap. Unit of Meas.** | 5832 | 29 | **Qty. per Cap. Unit of Measure** |
| 93 | **Qty. per Item Unit of Measure** | 5832 | 59 | **Qty. per Unit of Measure** |

### Table 99000755, *Work Center Ledger Entry* (Deleted)

The ***Work Center Ledger Entry*** table presently records capacity postings related to work centers. The ***Capacity Ledger Entry*** table will replace both the ***Machine Center Ledger Entry*** and the ***Work Center Ledger Entry*** tables in recording capacity related entries. The list below shows how the fields have been mapped from the ***Work Center Ledger Entry*** table to the new ***Capacity Ledger Entry*** table.

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 1 | Entry No. | 5832 | 1 | |
| 2 | Work Center No. | 5832 | 9 | |
| 3 | Posting Date | 5832 | 3 | |
| 4 | Document No. | 5832 | 6 | |
| 5 | Item No. | 5832 | 56 | |
| 10 | Description | 5832 | 7 | |
| 11 | Used Time | | | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 13 | Direct Unit Cost | | | |
| 14 | Unit Cost | | | |
| 15 | Cost Amount | | | |
| 17 | Work Center Group Code | 5832 | 68 | |
| 18 | Starting Time | 5832 | 43 | |
| 19 | Ending Time | 5832 | 44 | |
| 20 | Concurrent Capacities | 5832 | 19 | Concurrent Capacity |
| 23 | Prod. Order No. | 5832 | 50 | |
| 28 | Work Shift Code | 5832 | 69 | |
| 31 | Global Dimension 1 Code | 5832 | 33 | |
| 32 | Global Dimension 2 Code | 5832 | 34 | |
| 35 | Output Quantity | | | |
| 36 | Scrapped Quantity | | | |
| 37 | Item Unit of Measure Code | 5832 | 58 | Unit of Measure Code |
| 38 | Output Quantity (Base) | 5832 | 16 | Output Quantity |
| 39 | Scrapped Quantity (Base) | 5832 | 17 | Scrap Quantity |
| 62 | Source Code | | | |
| 72 | Journal Batch Name | | | |
| 73 | User ID | | | |
| 74 | Reason Code | | | |
| 87 | Document Date | 5832 | 60 | |
| 88 | External Document No. | 5832 | 61 | |
| 90 | Unit of Measure Code | 5832 | 28 | Cap. Unit of Measure Code |
| 91 | Used Time (Base) | 5832 | 10 | Quantity |
| 92 | Time per Cap. Unit of Meas. | 5832 | 29 | Qty. per Cap. Unit of Measure |

### Table 99000813, *Consumption Journal Line* (Deleted)

The ***Consumption Journal Line*** table was used to post material consumption for production orders. This possibility will now be directly available through the ***Item Journal Line*** table. The list below shows how the fields have been mapped from the ***Consumption Journal Line*** table to the ***Item Journal Line*** table.

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 1 | Journal Template Name | | | |
| 2 | Line No. | | | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 3 | Prod. Order No. | 83 | 5401 | |
| 5 | Item No. | 83 | 6 | Source No. |
| 6 | Posting Date | 83 | 4 | |
| 7 | Document No. | 83 | 7 | |
| 10 | No. | 83 | 3 | Item No. |
| 11 | Description | 83 | 8 | |
| 12 | Quantity | 83 | 13 | |
| 13 | Direct Unit Cost | 83 | 17 | Unit Cost |
| 14 | Unit Cost | 83 | 17 | |
| 15 | Cost Amount | 83 | 18 | Amount |
| 21 | Location Code | 83 | 9 | |
| 22 | Bin Code | 83 | 5403 | |
| 25 | Variant Code | 83 | 5402 | |
| 26 | Unit of Measure Code | 83 | 5407 | |
| 31 | Shortcut Dimension 1 Code | 83 | 34 | |
| 32 | Shortcut Dimension 2 Code | 83 | 35 | |
| 37 | Appl.-to Item Entry | 83 | 29 | Applies-to Entry |
| 40 | Qty. per Unit of Measure | 83 | 5404 | |
| 43 | Prod. Order Line No. | 83 | 5880 | |
| 45 | Prod. Order Comp. Line No. | 83 | 5884 | |
| 50 | Indirect Cost % | 83 | 37 | |
| 51 | Overhead Rate | 83 | 99000755 | |
| 53 | Overhead Amount | 83 | 5875 | |
| 62 | Source Code | 83 | 26 | |
| 73 | Journal Batch Name | 83 | 41 | |
| 74 | Reason Code | 83 | 42 | |
| 75 | Recurring Method | 83 | 43 | |
| 76 | Expiration Date | 83 | 44 | |
| 77 | Recurring Frequency | 83 | 45 | |
| 79 | Gen. Bus. Posting Group | 83 | 57 | |
| 80 | Gen. Prod. Posting Group | 83 | 58 | |
| 87 | Document Date | 83 | 60 | |
| 88 | External Document No. | 83 | 62 | |
| 89 | No. Series | | | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 90 | Quantity (Base) | 83 | 5413 | |
| 151 | Applies-to Entry | 83 | 29 | |
| 6500 | Item Tracking No. | 83 | 6500 | |

### Table 99000796, *Output Journal Line* (Deleted)

The ***Consumption Journal Line*** table was used to post capacity consumption and output for production orders. This possibilty will now be directly available through the ***Item Journal Line*** table. The list below shows how the fields have been mapped from the ***Output Journal Line*** table to the ***Item Journal Line*** table.

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 1 | Journal Template Name | | | |
| 2 | Line No. | | | |
| 3 | Prod. Order No. | 83 | 5401 | |
| 4 | Item No. | 83 | 3 | |
| 5 | Operation No. | 83 | 5838 | |
| 6 | Posting Date | 83 | 4 | |
| 7 | Document No. | 83 | 7 | |
| 9 | Type | 83 | 5830 | |
| 10 | No. | 83 | 5831 | |
| 11 | Description | 83 | 8 | |
| 12 | Run Time | 83 | 5842 | |
| 13 | Direct Unit Cost | | | |
| 14 | Unit Cost | | | |
| 15 | Cost Amount | | | |
| 16 | Work Center No. | 83 | 5839 | |
| 17 | Work Center Group Code | 83 | 5898 | |
| 18 | Starting Time | 83 | 5873 | |
| 19 | Ending Time | 83 | 5874 | |
| 20 | Concurrent Capacities | 83 | 5849 | Concurrent Capacity |
| 21 | Location Code | 83 | 9 | |
| 22 | Bin Code | 83 | 5403 | |
| 25 | Variant Code | 83 | 5402 | |
| 26 | Stop Code | 83 | 5895 | |
| 27 | Scrap Code | 83 | 5896 | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 28 | Work Shift Code | 83 | 5899 | |
| 31 | Shortcut Dimension 1 Code | 83 | 34 | |
| 32 | Shortcut Dimension 2 Code | 83 | 35 | |
| 33 | Finished | 83 | 5885 | |
| 34 | Setup Time | 83 | 5841 | |
| 35 | Output Quantity | 83 | 5846 | |
| 36 | Scrapped Quantity | 83 | 5847 | Scrap Quantity |
| 37 | Appl.-to Item Entry | 83 | 29 | Applies-to Entry |
| 38 | Output Quantity (Base) | 83 | 5856 | |
| 39 | Scrapped Quantity (Base) | 83 | 5857 | Scrap Quantity (Base) |
| 40 | Item Unit of Measure Code | 83 | 5407 | Unit of Measure Code |
| 41 | Routing No. | 83 | 5882 | |
| 42 | Routing Reference No. | 83 | 5883 | |
| 43 | Prod. Order Line No. | 83 | 5880 | |
| 50 | Indirect Cost % | 83 | 37 | |
| 51 | Overhead Rate | 83 | 99000755 | |
| 53 | Overhead Amount | 83 | 5875 | |
| 62 | Source Code | 83 | 26 | |
| 66 | Post Prod. Order Entry Only | | | |
| 73 | Journal Batch Name | | | |
| 74 | Reason Code | 83 | 42 | |
| 75 | Recurring Method | | | |
| 76 | Expiration Date | | | |
| 77 | Recurring Frequency | | | |
| 79 | Gen. Bus. Posting Group | 83 | 57 | |
| 80 | Gen. Prod. Posting Group | 83 | 58 | |
| 87 | Document Date | 83 | 60 | |
| 88 | External Document No. | 83 | 62 | |
| 89 | No. Series | | | |
| 90 | Unit of Measure Code | 83 | 5858 | Cap. Unit of Measure Code |
| 92 | Qty. per Item Unit of Measure | 83 | 5859 | Qty. per Cap. Unit of Measure |
| 93 | Qty. per Item Unit of Measure | 83 | 5404 | Qty. per Unit of Measure |
| 94 | Run Time (Base) | 83 | 5852 | |
| 95 | Setup Time (Base) | 83 | 5851 | |

| ID | Field Name in Deleted Table | Mapped to Table ID | Mapped to Field ID | Field Name in New Table (if different) |
|---|---|---|---|---|
| 6500 | Item Tracking No. | 83 | 6500 | |

## C.2 FIELD MAPPING TO EXISTING TABLES

The following tables show how information is transferred to new and existing tables from the deleted tables.

The following abbreviations are used in the text:

ILE = Item Ledger Entry

CLE = Capacity Ledger Entry.

**Table 83, *Item Journal Line* (Existing)**

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 1 | Journal Template Name | | |
| 2 | Line No. | | |
| 3 | Item No. | 99000813<br>99000796 | 10<br>3 |
| 4 | Posting Date | 99000813<br>99000796 | 4<br>4 |
| 5 | Entry Type | | |
| 6 | Source No. | | |
| 7 | Document No. | 99000813<br>99000796 | 7<br>7 |
| 8 | Description | 99000813<br>99000796 | 11<br>8 |
| 9 | Location Code | 99000813<br>99000796 | 21<br>9 |
| 13 | Quantity | 99000796 | 35 |
| 15 | Invoiced Quantity | | |
| 16 | Unit Amount | | |
| 17 | Unit Cost | 99000813 | 14 |
| 26 | Source Code | 99000813<br>99000796 | 62<br>62 |
| 29 | Applies-to Entry | 99000813<br>99000796 | 37<br>37 |
| 34 | Shortcut Dimension 1 Code | 99000813<br>99000796 | 31<br>31 |
| 35 | Shortcut Dimension 2 Code | 99000813<br>99000796 | 32<br>32 |
| 37 | Indirect Cost % | 99000796 | 50 |
| 39 | Source Type | | |

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 41 | Journal Batch Name | | |
| 42 | Reason Code | 99000796 | 74 |
| 43 | Recurring Method | | |
| 44 | Expiration Date | | |
| 45 | Recurring Frequency | | |
| 57 | Gen. Bus. Posting Group | 99000813<br>99000796 | 79<br>79 |
| 58 | Gen. Prod. Posting Group | 99000813<br>99000796 | 80<br>80 |
| 60 | Document Date | 99000813<br>99000796 | 87<br>87 |
| 62 | External Document No. | 99000813<br>99000796 | 88<br>88 |
| 5401 | Prod. Order No. | 99000813<br>99000796 | 3<br>3 |
| 5402 | Variant Code | 99000813<br>99000796 | 25<br>25 |
| 5403 | Bin Code | 99000813<br>99000796 | 22<br>22 |
| 5404 | Qty. per Unit of Measure | 99000813<br>99000796 | 40<br>92 |
| 5407 | Unit of Measure Code | 99000813<br>99000796 | 26<br>90 |
| 5413 | Quantity (Base) | | |
| 5830 | Type | 99000796<br>99000767 | 9<br>9 |
| 5831 | No. | 99000796 | 10 |
| 5838 | Operation No. | 99000796 | 5 |
| 5839 | Work Center No. | 99000796 | 16 |
| 5841 | Setup Time | 99000796 | 34 |
| 5842 | Run Time | 99000796 | 12 |
| 5843 | Stop Time | | |
| 5846 | Output Quantity | 99000796 | 35 |
| 5847 | Scrap Quantity | 99000796 | 36 |
| 5849 | Concurrent Capacity | 99000796 | 20 |
| 5851 | Setup Time (Base) | 99000796 | 95 |
| 5852 | Run Time (Base) | 99000796 | 94 |
| 5853 | Stop Time (Base) | | |

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 5856 | Output Quantity (Base) | 99000796 | 38 |
| 5857 | Scrap Quantity (Base) | 99000796 | 39 |
| 5858 | Cap. Unit of Measure Code | 99000796 | 40 |
| 5859 | Qty. per Cap. Unit of Measure | 99000796 | 93 |
| 5873 | Starting Time | 99000796 | 18 |
| 5874 | Ending Time | 99000796 | 19 |
| 5875 | Overhead Amount | 99000796 | 53 |
| 5880 | Prod. Order Line No. | 99000813 | 43 |
| 5882 | Routing No. | 99000796 | 41 |
| 5883 | Routing Reference No. | 99000796 | 42 |
| 5884 | Prod. Order Comp. Line No. | 99000813 | 45 |
| 5885 | Finished | 99000796 | 33 |
| 5887 | Unit Cost Calculation | | |
| 5888 | Subcontracting | | |
| 5895 | Stop Code | 99000796 | 26 |
| 5896 | Scrap Code | 99000796 | 27 |
| 5898 | Work Center Group Code | 99000796 | 17 |
| 5899 | Work Shift Code | 99000796 | 28 |
| 6500 | Item Tracking No. | 99000813 | 6500 |
| 99000755 | Overhead Rate | 99000796 | 51 |

### Table 32, *Item Ledger Entry* (Existing)

| **ID** | **Field Name** | **Mapped from Table ID** | **Mapped from Field ID** |
|---|---|---|---|
| 2 | Item No. | 99000794 | 9 |
| 3 | Posting Date | 99000794 | 5 |
| 5 | Source No. | 99000794 | 4 |
| 6 | Document No. | 99000794 | 6 |
| 7 | Description | 99000794 | 10 |
| 8 | Location Code | 99000794 | 21 |
| 12 | Quantity | 99000794 | 11 |
| 28 | Appl.-to Entry | 99000794 | 37 |
| 33 | Global Dimension 1 Code | 99000794 | 32 |
| 34 | Global Dimension 2 Code | 99000794 | 33 |
| 60 | Document Name | 99000794 | 87 |
| 61 | External Document No. | 99000794 | 88 |
| 5401 | Prod. Order No. | 99000794 | 2 |
| 5402 | Variant Code | 99000794 | 25 |
| 5403 | Bin Code | 99000794 | 22 |
| 5404 | Qty. per Unit of Measure | 99000794 | 92 |
| 5407 | Unit of Measure Code | 99000794 | 90 |
| 5832 | Prod. Order Line No. | 99000794 | 43 |
| 5833 | Prod. Order Comp. Line No. | 99000794 | 45 |

### Table 5802, *Value Entry* (Existing)

In future, the ***Value Entry*** table will also store value related information for capacity consumption entries. Previously, values of capacity consumption was stored on the ***Production Order Ledger Entry*** table . The list below shows how the field mapping of capacity from ***Production Order Ledger Entry*** table should be done:.

| **ID** | **Field Name** | **Mapped from Table ID** | **Mapped from Field ID** |
|---|---|---|---|
| 2 | Item No. | 99000794 | 9 |
| 3 | Posting Date | 99000794 | 5 |
| 5 | Source No. | 99000794 | 4 |
| 6 | Document No. | 99000794 | 6 |
| 7 | Description | 99000794 | 10 |
| 8 | Location Code | 99000794 | 21 |
| 12 | Valued Quantity | 99000794 | 11 |

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 15 | Cost per Unit | 99000794 | 13 |
| 17 | Amount | 99000794 | 14 |
| 24 | User ID | 99000794 | 75 |
| 25 | Source Code | 99000794 | 62 |
| 33 | Global Dimension 1 Code | 99000794 | 31 |
| 34 | Global Dimension 2 Code | 99000794 | 32 |
| 45 | Cost Posted to G/L | 99000794 | 66 |
| 46 | Reason Code | 99000794 | 74 |
| 57 | Gen. Bus. Posting Group | 99000794 | 79 |
| 58 | Gen. Prod. Posting Group | 99000794 | 80 |
| 60 | Document Date | 99000794 | 87 |
| 61 | External Document No. | 99000794 | 88 |
| 5401 | Prod. Order No. | 99000794 | 2 |
| 5402 | Variant Code | 99000794 | 25 |
| 5403 | Bin Code | 99000794 | 22 |
| 5831 | Capacity Ledger Entry No. | 5832 | |
| 5832 | Type | 99000794 | 8 |
| 5834 | No. | 99000794 | 9 |
| 5881 | Prod. Order Line No. | 99000794 | 43 |

### Table 5832, *Capacity Ledger Entry* (New)

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 1 | Entry No. | 99000759<br>99000755 | 1<br>1 |
| 2 | No. | 99000794<br>99000759 | 15<br>3 |
| 3 | Posting Date | 99000759<br>99000755 | 4<br>3 |
| 4 | Type | | |
| 6 | Document No. | 99000759<br>99000755 | 5<br>4 |
| 7 | Description | 99000759<br>99000755 | 11<br>10 |
| 8 | Operation No. | 99000794 | 3 |

| **ID** | **Field Name** | **Mapped from Table ID** | **Mapped from Field ID** |
|---|---|---|---|
| 9 | Work Center No. | 99000759<br>99000755 | 16<br>2 |
| 10 | Quantity | 99000794<br>99000755 | 11<br>11 |
| 11 | Setup Time | 99000794 | 34 |
| 12 | Run Time | 99000794 | 91 |
| 13 | Stop Time | | |
| 15 | Invoiced Quantity | 99000794 | 97 |
| 16 | Output Quantity | 99000794<br>99000759<br>99000755 | 35<br>35<br>35 |
| 17 | Scrap Quantity | 99000794<br>99000759<br>99000755 | 36<br>36<br>36 |
| 19 | Concurrent Capacity | 99000794<br>99000759<br>99000755 | 20<br>20<br>20 |
| 28 | Cap. Unit of Measure Code | 99000759<br>99000755 | 90<br>90 |
| 29 | Qty. per Capacity Unit of Measure | 99000759<br>99000755 | 92<br>92 |
| 33 | Global Dimension 1 Code | 99000759<br>99000755 | 31<br>31 |
| 34 | Global Dimension 2 Code | 99000759<br>99000755 | 32<br>32 |
| 39 | Last output Line | 99000794 | 102 |
| 42 | Completely Invoiced | 99000794 | 150 |
| 43 | Starting Time | 99000794<br>99000759<br>99000755 | 18<br>18<br>18 |
| 44 | Ending Time | 99000794<br>99000759<br>99000755 | 19<br>19<br>19 |
| 50 | Prod. Order No. | 99000759<br>99000755 | 23<br>23 |
| 51 | Prod. Order Line No. | 99000794 | 43 |
| 52 | Routing No. | 99000794 | 41 |
| 53 | Routing Reference No. | 99000794 | 42 |
| 56 | Item No. | 99000759<br>99000755 | 6<br>5 |

| ID | Field Name | Mapped from Table ID | Mapped from Field ID |
|---|---|---|---|
| 58 | Unit of Measure Code | 99000759<br>99000755 | 37<br>37 |
| 59 | Qty. per Unit of Measure | 99000759 | 93 |
| 60 | Document Date | 99000759<br>99000755 | 87<br>87 |
| 61 | External Document No. | 99000759<br>99000755 | 88<br>88 |
| 65 | Stop Code | 99000755 | 26 |
| 66 | Scrap Code | 99000759 | 27 |
| 68 | Work Center Group Code | 99000794<br>99000759<br>99000755 | 16<br>17<br>17 |
| 69 | Work Shift Code | 99000794<br>99000759<br>99000755 | 28<br>28<br>28 |
| 71 | Direct Cost | | |
| 72 | Overhead Cost | | |
| 76 | Direct Cost (ACY) | | |
| 77 | Overhead Cost (ACY) | | |
| 78 | Subcontracting | | |

# Appendix D
# Upgrading Item Tracking

This appendix provides some specification of the objects that take care of upgrading item tracking data from Navision Financials 2.60 and Navision Manufacturing 2.60 to Navision 4.0 and from Navision Attain 3.01 and 3.10 to Navision 4.0.

The appendix contains the following sections:

- D.1 Upgrading Item Tracking from NF/NM 2.60 to Navision 4.0
- D.2 Upgrading Item Tracking from NA 3.01/3.10 to Navision 4.0

## D.1 UPGRADING ITEM TRACKING FROM NF/NM 2.60 TO NAVISION 4.0

**Note**

Item Tracking in Navision Financials 2.60 and Navision Manufacturing 2.60 is very similar, and the upgrade tools are almost identical. The following therefore covers both paths.

In Navision Financials/Navision Manufacturing 2.60, the item tracking number is carried directly on the lines. The task for the upgrade toolkit is for some of these lines to move the information down into the ***Reservation Entry*** table and for others to establish a relation to the item tracking information in the item ledger entries.

### Overview

Below is a list of tables in Navision Manufacturing 2.60 that carry the **Serial No.** or **Lot No.** fields:

| Table ID | Table Name | Lot No. | Serial No. | Update Action | Procedure |
|---|---|---|---|---|---|
| 32 | ***Item Ledger Entry*** | 5400 | 40 | Move values to 65xx fields | A |
| 37 | ***Sales Line*** | 5400 | 53 | Move values to T336/T337 | B |
| 39 | ***Purchase Line*** | 5400 | 53 | Move values to T336/T337 | B |
| 83 | ***Item Journal Line*** | 5400 | 38 | Must be empty | None |
| 89 | ***BOM Journal Line*** | 5400 |  | Must be empty | None |
| 111 | ***Sales Shipment Line*** | 5400 | 53 | Create relation to T32 via entry in T6507 | C |
| 113 | ***Sales Invoice Line*** | 5400 | 53 | The field is removed in NA 3.10 - no upgrade | E |
| 115 | ***Sales Cr. Memo Line*** | 5400 | 53 | The field is removed in NA 3.10 - no upgrade | E |
| 121 | ***Purch. Rcpt. Line*** | 5400 | 53 | Create relation to T32 via entry in T6507 | C |
| 123 | ***Purch. Inv. Line*** | 5400 | 53 | The field is removed in NA 3.10 - no upgrade | E |
| 125 | ***Purch. Cr. Memo Line*** | 5400 | 53 | The field is removed in NA 3.10 - no upgrade | E |
| 169 | ***Job Ledger Entry*** | 5400 |  | The field is removed in NA 3.10 - no upgrade | E |
| 210 | ***Job Journal Line*** | 5400 | 91 | Must be empty | None |
| 246 | ***Requisition Line*** | 5400 | 27 | Must be empty | None |
| 281 | ***Phys. Inventory Ledger Entry*** | 5400 | 40 | The field is removed in NA 3.10 - no upgrade | E |
| 337 | ***Reservation Entry*** | 5400 | 24 | None | None |
| 5214 | ***Misc. Article Information*** |  | 9 | None - not related | None |
| 5406 | ***Prod. Order Line*** | 24 |  | Move values to T336/T337 | B |
| 5407 | ***Prod. Order Component*** | 34 |  | Move values to T336/T337 | B |
| 5600 | ***Fixed Assets*** |  | 17 | None - not related | None |

| Table ID | Table Name | Lot No. | Serial No. | Update Action | Procedure |
|---|---|---|---|---|---|
| 99000755 | ***W. Center Ledger Entry*** | 24 | | The table is removed in NA 3.10 - no upgrade | F |
| 99000759 | ***M. Center Ledger Entry*** | 24 | | The table is removed in NA 3.10 - no upgrade | F |
| 99000767 | ***Capacity Journal Line*** | 24 | | Must be empty | None |
| 99000791 | ***Production Order*** | 34 | | The field is removed in NA 3.10 - no upgrade | E |
| 99000794 | ***Prod. O. Ledger Entry*** | 24 | 23 | The table is removed in NA 3.10 - no upgrade | F |
| 99000794 | ***P.O. Output Journal Line*** | 24 | 23 | Must be empty | None |
| 99000813 | ***P.O. Consump. Journal Line*** | 24 | 23 | Must be empty | None |
| 99000815 | ***Finished Production Order*** | 34 | | The field is removed in NA 4.0 - no upgrade | None |
| 99000817 | ***Fin. Prod. Order Ledg. Entry*** | 24 | 23 | The table is removed in NA 3.00 - no upgrade | F |
| 99000818 | ***Finished Prod. Order Line*** | 24 | | Create relation to T32 via entry in T6507 | C |
| 99000819 | ***Fin. Prod. Order Component*** | 34 | | Create relation to T32 via entry in T6507 | C |
| 99000828 | ***Planning Line*** | 24 | | The field is removed in NA 3.10 - no upgrade | E |
| 99000829 | ***Planning Component*** | 34 | | Move values to T336/T337 | B |
| 99000853 | ***Inventory Profile*** | 18 | 19 | Must be empty | None |

## General Procedure

### Step 0 - Cleaning the 2.60 database

Run a check function that will check whether journals are empty.

### Step 1 - Preparing the 2.60 Database

The item tracking information in table 32 is stored in an upgrade table, and original fields are blanked.

The item tracking information regarding unposted items is transferred to table 337 records and item tracking information regarding posted items is transferred to table 336.

The **Serial/Lot No.** fields on the lines are blanked.

### Step 2 - Creating the NA 4.0 Database

Item tracking information in table 32 is recreated in the new fields based on the data in the upgrade table. Data is generated for table 336, table 6507 and table 6508 based on the item tracking information in table 32. When the update is completed, the upgrade data is deleted. The upgrade objects are deleted.

## Procedure A - Handling Item Tracking Fields in Table 32

The **Serial No.** and **Lot No.** fields in table 32 must be transferred to the corresponding fields in 4.0. However, as the fields have been moved number-wise we have to store the values temporarily when preparing the 2.60 database.

When creating the 4.0 database, the values will be written into the new fields on table 32. Furthermore, the related value entries will have to be updated, so that all VEs pointing at a specific ILE will inherit the item tracking information from that ILE.

## Procedure B - Moving Item Tracking information from Lines to Table 336 and 337

When preparing the 2.60 database, all lines carrying item tracking information are taken through the following procedure:

1. Find all outstanding document lines and create table 336 and table 337 records for them.
2. Blank the **Serial/Lot No.** fields on the line.

## Procedure C - Creating Relation to Item Ledger Entries

We must create a relation from posted document lines and partially posted ONEs to the related ILEs. The relation is made via the posted document line through the ILE reference field, e.g. on table 111, ***Sales Shipment Line,*** it would be the **Item Shpt. Entry No.** field.

1. Create records in table table 6507 that reflect the relation between the item ledger entries and the posted document lines.

2. Blank the **Serial/Lot No.** fields on the line.

### Procedure D - Creating Relation between Posted Invoices and Value Entries

In this section Invoice and Cr. Memo is treated as the same thing.

We must create a relation from posted invoice lines and partially posted ONEs to the related value entries. The relation is made via **Document No.** on the posted invoice line + the posting date from the invoice header; the VE will carry a reference to these.

From the posted document line, there is a reference to the order line, which will still be there if the line is not fully posted.

In step 2, table 5802 is run through and all entries carrying item tracking information are taken through the following procedure:

1. According to the type of the VE (sales, purchase,…), an attempt is made to find a related posted invoice line via the **Document No.** . Posting date is also checked.
2. If a line is found, a table 6508 record is created for the invoice line containing a reference (TextRowID) to the VE.

**Note**

Because the 2.60 products did not use the value entry principle, item tracking information that was posted and invoiced at the same time will not be upgraded to the posted document lines.

For the same reason, item tracking entries are never upgraded to invoiced document lines.

### Procedure E - Fields Removed

There are some item tracking fields that have simply disappeared between 2.60 and 3.00. These are not covered by the Item Tracking solution in 3.10 – they are just gone.

The field is blanked in step 1. Nothing further is done.

### Procedure F - Tables Removed

Some tables have been removed between version 2.60 and 3.00 because of the merging of different ledger entry tables related to manufacturing into Item Ledger Entry table.

## D.2 UPGRADING ITEM TRACKING FROM NA 3.01/3.10 TO NAVISION 4.0

**Note**

Item Tracking is practically unchanged between 3.01 and 3.10, and the upgrade tools are almost identical. The following therefore covers both paths.

The upgrade from 3.01 or 3.10 to the new Item Tracking structure of 4.0 is a rather complex task.

### Overview

The item tracking solution in 3.01 or 3.10 uses the following tables:

| TableID | Name | Update Action | Procedure |
|---|---|---|---|
| 6500 | ***Item Tracking Line*** | Transferred to T336/T337 | A |
| 6501 | ***Item Tracking Entry*** | Transferred to T32 (split) | B |
| 6502 | ***Item Tracking Code*** | None | X |
| 6503 | ***Item Tracking Setup*** | Empty table and remove object | C |
| 6504 | ***Serial No. Information*** | None | X |
| 6505 | ***Lot No. Information*** | None | X |
| 6506 | ***Item Tracking Comment*** | None | X |
| 6521 | ***Sales Shipment Tracking Line*** | Replaced by relation to T32 via 6507 | D |
| 6523 | ***Sales Invoice Tracking Line*** | Replaced by relation to T5802 via 6508 | E |
| 6525 | ***Sales Cr. Memo Tracking Line*** | Replaced by relation to T5802 via 6508 | E |
| 6526 | ***Transfer Shipm. Tracking Line*** | Replaced by relation to T32 via 6507 | D |
| 6527 | ***Transfer Receipt Tracking Line*** | Replaced by relation to T32 via 6507 | D |
| 6531 | ***Purch. Receipt Tracking Line*** | Replaced by relation to T32 via 6507 | D |
| 6533 | ***Purch. Inv. Tracking Line*** | Replaced by relation to T5802 via 6508 | E |
| 6535 | ***Purch Cr. Memo Tracking Line*** | Replaced by relation to T5802 via 6508 | E |
| 6536 | ***Posted Whse. Activ. Trkg. Line*** | Replaced by relation to T5802 via 6508 | E |
| 6537 | ***Posted Service Inv. Trkg. Line*** | Replaced by relation to T32 via 6507 | D |
| 6656 | ***Return Shipment Tracking Line*** | Replaced by relation to T32 via 6507 | D |

| TableID | Name | Update Action | Procedure |
|---|---|---|---|
| 6666 | ***Return Receipt Tracking Line*** | Replaced by relation to T32 via 6507 | D |

Table 6521...6666 have exactly the same structure and are used for carrying item tracking information for different tables.

Table 6502 is used for setup information.

Table 6500 carries actual lot and serial numbers related to the specific ONE by an **Item Tracking No.**

Table 6501 is used for information regarding posted items (item ledger entries). The relation is created by field 18, **Item Ledger Entry No.**

Table 6503 is used solely for saving the last item tracking number.

## General Procedure

### Step 0 - Cleaning the 3.01/3.10 database

Run a check function that will check whether the assumptions for the upgrade are fulfilled. The check function might also itself make some initial clean-up of reservations and tracking.

### Step 1 - Preparing the 3.01/3.10 database

The item tracking information is stored in upgrade tables that are carried to the new database.

The data in the original tables is then deleted.

### Step 2 - Creating the 4.0 database

Procedures are run that will use the data in the upgrade tables for splitting ILEs and value entries and put data in table 336, 337, 6507 and 6508.

When the update is completed, the upgrade data is deleted.

The specific Item Tracking objects that have been removed from the 4.0 solution are deleted from the database in case they are still there in step 2.

## Procedure A - Transferring Item Tracking Lines

The data that was placed in table 6500, ***Item Tracking Line***, must be transferred to table 337 as far as items not handled are concerned, and table 336 for items already handled - that is, sales or purchase lines that have been partially posted.

The data that should be placed in table 336 is available more directly through the relation to the split ILE. Hence we will not use the content of table 6500 for this purpose – that data is redundant.

This means that we will concentrate on creating records in table 337. The tool demands that no reservations exist for items carrying item tracking information. For the same reason, we will delete all order tracking entries before making the transfer of item tracking numbers.

The place where this will cause problems is in connection with existing MTO structures, because MTO connections are registered in reservation records.

The records in table 6500 are run through. For every record with remaining quantity to handle, we will create and insert a surplus/prospect record in table 337. The values for the pointer fields on the table 337 record will be retrieved via the **Item Tracking No.**, looking up the related record in the database.

Transfer orders are special in this relation. In the 4.0 solution, we keep track of the ILEs created during posting and use this relation for retrieving item tracking information later on.

Therefore, item tracking information is retrieved from the ILE directly, looking for ILEs with type Transfer, filtering on document no etc. For transfers we do not create table 336 records.

Table 337 records will be created for remaining quantity to handle/ship and put on the transfer line. The quantities already shipped must be created as table 337 entries pointing at the transfer receipt, that is the derived lines.

### Procedure B - Splitting Item Ledger Entries

Run through table 6501, group by ILE entry number, and split the ILE according to the records in table 6501. If there are *n* records in table 6501 within the group, we will generate *n-1* new ILEs, which will get new entry numbers based on the last existing ILE in the database. The original ILE will be modified according to table 6501 record *n*.

For every group, we have to find out whether we have relations to the original ILE. This will be the case on posted document lines, e.g. ***Posted Shipment***. On the posted document lines we must clear the reference field, e.g. table 111 **Item Shpt. Entry No.**, and instead create entries in table 6507 linking the line with all the ILEs in the group.

At the same time we must split the value entries related to the ILE, which is new since 3.01/3.10. Records must be stored in table 6508 to save the information about which records the VEs are split into.

To find out which specific value entries are related to a posted invoice, we will have to use some secondary indications, as we do not have any strict relation between the records. Instead we will have to make use of several fields on the value entry and set up the link that way:

**Item No.** : Must match

**Posting Date**: Must match the posting date on the document header

**Item Ledger Entry Type**: Sales, Purchase - Depending on what type of document we’re dealing with

**Source No.**: Line No. of the posted document line

**Document No.**: Document No. of the posted document line

Concerning Item Registers, we will create new entries in the item register for the split entries, ILEs and VEs. As the original record will remain with the same primary key, existing item registers do not have to be modified.

### Procedure C - Removing Table 6503, Item Tracking Setup

Table 6503 was used solely for keeping the last entry number for the purpose of the **Item Tracking No.** field, which was used all over to link item tracking information to all kinds of lines. In the 4.0 solution, the system uses pointers to the primary key to make this kind of relation. Therefore, the table and all the related fields are removed.

In order to be able to build up relations properly in step 2, we will set up a special table to carry information about the primary key(s) related to each **Item Tracking No.** The table will consist of the 6 pointer fields (primary key) and the **Item Tracking No.** The table must have a key on **Item Tracking No.** The **Item Tracking No.** field must be blanked in all records in the following tables:

| TableID | Table Name | Fields No. |
|---|---|---|
| 37 | ***Sales Line*** | 6500 |
| 39 | ***Purchase Line*** | 6500 |
| 83 | ***Item Journal Line*** | 6500 |
| 111 | ***Sales Shipment Line*** | 6500 |
| 113 | ***Sales Invoice Line*** | 6500 |
| 115 | ***Sales Cr. Memo Line*** | 6500 |
| 121 | ***Purch. Rcpt. Line*** | 6500 |
| 123 | ***Purch. Inv. Line*** | 6500 |
| 125 | ***Purch. Cr. Memo Line*** | 6500 |
| 246 | ***Requisition Line*** | 6500 |
| 5108 | ***Sales Line Archive*** | 6500 |
| 5110 | ***Purchase Line Archive*** | 6500 |
| 5406 | ***Prod. Order Line*** | 6500 |
| 5407 | ***Prod. Order Component*** | 6500 |
| 5741 | ***Transfer Line*** | 6500 |
| 5745 | ***Transfer Shipment Line*** | 6500 |
| 5747 | ***Transfer Receipt Line*** | 6500 |
| 5767 | ***Warehouse Activity Line*** | 6500 |
| 5773 | ***Posted Whse. Activity Line*** | 6500 |
| 5902 | ***Service Invoice Line*** | 94 |
| 5932 | ***Posted Service Invoice Line*** | 94 |

| TableID | Table Name | Fields No. |
|---|---|---|
| 6651 | ***Return Shipment Line*** | 6500 |
| 6666 | ***Return Receipt Line*** | 6500 |
| 99000829 | ***Planning Component*** | 6500 |

## NA 3.01/3.10 Objects that Should not Exist after the Upgrade

The following objects are strictly from NA 3.01/3.10 and should not remain in the NA 4.0 database after all upgrade steps are completed:

| Type | No. | Name |
|---|---|---|
| Table | 6500 | ***Item Tracking Line*** |
| Table | 6503 | ***Item Tracking Setup*** |
| Table | 6521 | ***Sales Shipment Tracking Line*** |
| Table | 6523 | ***Sales Invoice Tracking Line*** |
| Table | 6525 | ***Sales Cr. Memo Tracking Line*** |
| Table | 6526 | ***Transfer Shipm. Tracking Line*** |
| Table | 6527 | ***Transfer Receipt Tracking Line*** |
| Table | 6531 | ***Purch. Receipt Tracking Line*** |
| Table | 6533 | ***Purch. Inv. Tracking Line*** |
| Table | 6535 | ***Purch Cr. Memo Tracking Line*** |
| Table | 6536 | ***Posted Whse. Activ. Trkg. Line*** |
| Table | 6537 | ***Posted Service Inv. Trkg. Line*** |
| Table | 6656 | ***Return Shipment Tracking Line*** |
| Table | 6666 | ***Return Receipt Tracking LIne*** |
| Form | 6509 | ***Item Trkg. Transfer Shipment*** |
| Form | 6514 | ***Item Trkg. Lines Prod. Order*** |
| Form | 6516 | ***Item Trkg. Lines Whse. Activ.*** |
| Form | 6517 | ***Item Tracking Line Lookup*** |
| Form | 6521 | ***Sales Shipment Tracking Lines*** |
| Form | 6523 | ***Sales Invoice Tracking Lines*** |
| Form | 6525 | ***Sales Cr. Memo Tracking Lines*** |
| Form | 6526 | ***Transfer Shpt. Tracking Lines*** |
| Form | 6527 | ***Transfer Rcpt. Tracking Lines*** |
| Form | 6531 | ***Purch. Rcpt. Tracking Lines*** |
| Form | 6533 | ***Purch. Inv. Tracking Lines*** |
| Form | 6535 | ***Purch. Cr. Memo Tracking Lines*** |
| Form | 6536 | ***Purch. Whse. Activ. Trkg. Lines*** |

| | | |
|---|---|---|
| Form | 6537 | ***Posted Service Inv. Trkg. Lines*** |
| Codeunit | 6501 | ***Item Reg. Show Item Tracking*** |
| Codeunit | 6511 | ***Item Tracking Mgt. Sales Line*** |
| Codeunit | 6512 | ***Item Tracking Mgt. Req. Line*** |
| Codeunit | 6513 | ***Item Tracking Mgt. Purch. Line*** |
| Codeunit | 6514 | ***Item Trkg. Mgt. Item Jnl. Line*** |
| Codeunit | 6517 | ***Item Trkg. Mgt. Prod. Order Line*** |
| Codeunit | 6518 | ***Item Tracking Mgt. Prod. Comp*** |
| Codeunit | 6519 | ***Item Tracking Mgt. Transfer*** |
| Codeunit | 6521 | ***Item Tracking Mgt. Plng. Comp.*** |
| Codeunit | 6522 | ***Item Trkg. Mgt. Whse. Activ. Line*** |
| Codeunit | 6523 | ***Item Trkg. Mgt. Serv. Inv. Line*** |