MODULE 3: CODE UPGRADE

Module Overview

”Code Upgrade” covers the code upgrade phase of the upgrade project. It discusses the key development and testing activities that are included in the code upgrade.

Typically, customers want all the customizations that are implemented in their Microsoft Dynamics NAV 2009 SP1 (or later versions) databases to be migrated to their new Microsoft Dynamics NAV 2013 databases.

As the person doing the upgrade, you must decide between the following two courses of action:

- Implement new customizations in Microsoft Dynamics NAV 2013.
- Upgrade the application code by using a compare-and-merge process. This compare-and-merge process is known as a code upgrade. You perform the code upgrade before you perform the data upgrade.

The code upgrade is a sequence of development actions that are intended to fully transfer the functionality of the customer's solution to the new version of Microsoft Dynamics NAV. A code upgrade can also include add-on functionality that is developed by third parties or objects that are changed by the customer.

Objectives

- Review the code upgrade workflow and upgrade strategy.
- Examine upgrading customizations.
- Explain how to resolve conflicts that arise during the code upgrade procedure.
- Define the post-upgrade activities.
- Explain code upgrade testing.
- Describe the most important administration tools that are relevant for the upgrade process.

Code Upgrade Workflow

As part of the full upgrade process, code upgrade is a sequence of development actions that are intended to fully transfer the functionality of the customer's solution to the target version. During code upgrade, the project team performs the actual upgrade of the customized objects. This upgrades the add-on functionality and objects changed by the customer.

Moreover, the project team performs code upgrade testing to make sure that the customer's functionality is fully retained in the target version of the solution.

ISV Solutions Upgrade

Frequently, customers use Microsoft Dynamics solutions together with various Independent Software Vendor (ISV) add-ons that are typically custom Microsoft Dynamics NAV objects that are developed by a third-party. Therefore, the upgrade project may involve upgrading these solutions. You can do this in either of the following ways:

- If there is a version of the ISV add-on that corresponds to the target Microsoft Dynamics NAV, retrieve it from the vendor and install it on the new base version database before the code upgrade procedure.
- If the add-on does not have a release version that works with the target Microsoft Dynamics NAV version, you have the following options:
    - Postpone the whole upgrade until the required version of the add-on is released. This is the recommended approach.
    - Do not transfer the ISV solution to the target version.
    - Transfer the ISV solution to the target version as it is. This may require additional development effort. Verify that the ISV solution is compatible with the new version of Dynamics NAV.

Customization Upgrades

An upgrade customizations project is when we transfer the customizations that are implemented in the customer's solution to the target version of Microsoft Dynamics NAV. This activity includes the following steps:

1. Preparing and merging objects.
2. Implementing objects in the target Microsoft Dynamics NAV 2013 database. Additional effort may be required to adapt them.
3. Performing code optimization (optional).

When you work with code upgrade it is important to analyze and process the changes by comparing and evaluating the following three separate versions of the Microsoft Dynamics NAV database:

- **Old Base version** – This is a standard version of the Microsoft Dynamics NAV 2009 database.
- **Old Custom version** – This is the old base database plus the customer's changes and add-on solutions.
- **New Base version** – This is a standard version of the Microsoft Dynamics NAV 2013 database

Start by identifying any customer-specific changes that were made to the customer’s old custom version, and then implement those changes in the new base version. To identify the customizations that the customer has made, you must have an old base version to compare with the old custom version. This old base version must be the English (United States) version that you received from your local Microsoft country/region office.

The following procedure shows how to compare the objects in the customer’s old custom version with the old base version, and then implement the necessary changes in the new base version. The procedure assumes that you already have Microsoft Dynamics NAV 2009 installed.

1. Create three folders on the computer and name them: OldBaseVersion, OldCustomVersion, and NewBaseVersion.
2. Complete the following series of actions two times: one time for the old base version of the Microsoft Dynamics NAV 2009 database, and one time for the old custom version of the Microsoft Dynamics NAV 2009 database.
   a. Open the database by using the Microsoft Dynamics NAV 2009 Classic client.
   b. Delete all Dataports and forms from the database.
   c. Export all objects in txt format into the appropriate folder: OldBaseVersion or OldCustomVersion.
3. Uninstall Microsoft Dynamics NAV 2009.
4. Install Microsoft Dynamics NAV 2013. Select the Developer Option.
5. Complete the following series of actions two times: one time for the old base version, and one time for the old custom version.
   a. Run the TextFormatUpgrade2013.exe tool on the contents of the OldBaseVersion and OldCustomVersion folders to update all references from forms to pages.
   b. Create a new database in the Microsoft Dynamics NAV Development Environment.

c. Import all objects.
    i. The TextFormatUpgrade2013.exe tool saved the objects in a subdirectory of the original folder: **C:\OldBaseVersion\Converted\AllObjects.txt** or **C:\OldCustomVersion\Converted\AllObjects.txt**.

d. Compile all reports in the database.

e. Some reports do not compile because they use System Tables that have changed in Microsoft Dynamics NAV 2013. Correct this by doing the following:
    i. Fix the errors before you continue.
    ii. Delete any reports that are not required in Microsoft Dynamics NAV 2013.
    iii. Delete reports that are easy to re-create.

f. Upgrade all reports.

g. Export all database objects back to the respective directory: OldBaseVersion or OldCustomVersion.

You now can start to analyze and merge changes.

This table highlights some key code merging guidelines that you can apply. This depends on how a certain line or block of code that exists in the Old Base version is changed throughout the versions.

Old Base Version	Current Custom Version	New Base Version	New Custom Version	Comments
Exist	Not Modified	Deleted	Delete	
Exist	Not Modified	Modified	Copy from New Base version	
Exist	Not Modified	Not Modified	Copy from New Base version	
Exist	Modified	Deleted	Conflict	Determine whether the code is still required. If it is required, modify the New Custom version.

Old Base Version	Current Custom Version	New Base Version	New Custom Version	Comments
Exist	Modified	Modified	Conflict / Copy from New Base version	There is no conflict if the Current Custom version is modified in the same manner as the New Base version
Exist	Modified	Not Modified	Copy from Current Custom version	
Exist	Deleted	Deleted	Delete	
Exist	Deleted	Not Modified	Delete	
Exist	Deleted	Modified	Conflict	Determine whether the code is still required. If it is required, modify the New Custom version.
	Inserted		Copy from Current Custom version	
		Inserted	Copy from New Base version	
	Inserted	Inserted	Conflict / Copy from New Base version	There is a conflict if the inserted code is different in the Current Custom and in the New Base versions. Determine whether the code is still required. If it is still required, modify the New Custom version.

The following are examples of conflicts that must be resolved manually:

  - The object, field, variable, or function is added by using the existing ID in the New Base version. A typical solution is to assign a different ID to the field that the customer added.
  - The customization touches the functionality that was changed in the New Base version. A typical solution is to review the code and manually add changes to the New Custom version.

- The customization adds a function that is already present in the New Base version, such as the following:
  - **EXAMPLE** – In Table 37 Sales Line, the customer added Function 50057 SetHideValidationDialog(NewHideValidationDialog : Boolean). However, in the New Base version Function 57 SetHideValidationDialog(NewHideValidationDialog : Boolean) already exists with the same code.
  - **Solution** – Do not transfer Function 50057 SetHideValidationDialog(NewHideValidationDialog : Boolean) to the New Custom version. Instead, use the standard Function 57 SetHideValidationDialog(NewHideValidationDialog : Boolean).
  - The functionality that was introduced in the New Base Version resembles the functionality of the customizations. Typical solutions for this include the following:
    - Use the New Base version functionality without any changes.
    - Modify the New Base version functionality to adapt it to the customer's solution.
    - Replace the New Base version functionality with the customized functionality in the New Customer version.

***Note:** Generally, developers must always review and analyze the code, and use their best judgment to select the appropriate solution. However, when it is possible, we recommend that you try to prioritize standard code that is released with the new version of Microsoft Dynamics NAV instead of customization.*

If the difference in the functionality that is provided in the standard Microsoft Dynamics NAV version is insignificant compared to custom code, you should encourage the upgrade stakeholders to accept and use the standard functionality which might be an 80-90% fit. This helps customers avoid expensive support and upgrades in the future. Always analyze the true value of every customization before you apply it.

This stage of the upgrade is perfect for cleaning up the code and eliminating some customizations.

Before they start to upgrade customizations, developers must make sure that the following preconditions are met:

- The most recent backup of the customer's database is restored to a clean database of the source Microsoft Dynamics NAV version.
- When the ISV solution upgrade occurs, respective versions of the add-on are installed on the old base and new base version databases.

After merging customizations into the New Base version, you will have a New Custom version. This is the objective of the code upgrade. The product of the code upgrade is a .fob file that contains all Microsoft Dynamics NAV 2013 objects. This file includes customizations that you will use during the data upgrade.

When you upgrade the code, customers can continue to work with the Microsoft Dynamics NAV 2009 SP1 database (the old custom version) by using a copy of their database. However, we recommend that you avoid making any additional customizations to the old custom version while you work on the code upgrade.

Start by identifying any customer-specific changes that are made to the customer’s old custom version, and then implement those changes in the new base version. To identify the customizations that the customer made, you must have an old base version to compare with the old custom version.

Code Optimization

After transferring the customizations to the target version of the solution, you must make sure that the customizations do not adversely affect performance of the application. Code optimization is intended to improve the code of the upgraded customizations.

Code optimization consists of the following actions:

- **Code preparation for Microsoft SQL Server** – Edit the code to use the following C/AL functions that are designed specifically to optimize performance in the SQL Server architecture.
- FINDFIRST
- FINDLAST
- FINDSET

More information on this topic is available at the following website.

C/AL Database Functions and Performance on SQL Server

http://go.microsoft.com/fwlink/?LinkId=277023

- **Index optimization** – Create and maintain indexes to reduce load on the database when data is read or written into it. Use the following rules for optimal index use:
    - Do not create too many indexes, especially on frequently used tables. Each record update means an index update. This causes more disk I/O reads.
    - Do not maintain indexes on the SQL Server that are used only for sorting purposes.
    - If there are several indexes that begin with the same combination of keys, such as index fields, consider maintaining only one of them with the property MaintainSQLIndex.
    - Avoid making indexes with decimal or text fields to reduce the size of the index and number of disk I/O reads that are required to read the index.
    - Redesign indexes to increase query selectivity by doing the following:
        - Do not put Boolean and option fields at the beginning of an index.
        - Put date fields toward the end of the index.

Note: Selectivity *is several rows that are expected to be returned from the intended query.*

-
    - Do not create two detailed keys that produce small filtered sets, for example, "Posting Date, Customer No.". The index can become too large and can almost equal the size of the table itself. Therefore, it is rarely used by Microsoft SQL Server.
    - Do not maintain SIFT indexes on small and temporary tables, for example, Table 37 Sales Line, Purchase Line, Warehouse Activity Line, and similar tables, or if filtered sets are small. SQL Server can still retrieve sums directly from the source table.
    - Design tables to be smaller (contain less records) to guarantee faster data retrieval, better locking specificity, easier application upgrade, and better indexes.
- **Define and recode low-performance objects** – Improve performance of objects that have the most effect on the performance of the solution. You can use the following tools to define such objects:
    - SQL Server Dynamic Management Views and Functions
    - SQL Server Profiler
    - Microsoft Dynamics NAV Application Benchmark Tool

After you define low-performance objects, the following actions may be necessary:

- Rewrite low-performance code sections.
- Implement new code.
- Decrease SQL Server load by using temporary tables.

Object Text Transformation

Use the TextFormatUpgrade2013 command-line tool to update Microsoft Dynamics NAV objects from Microsoft Dynamics NAV 2009 that were exported to .txt files. These conversions are part of the Microsoft Dynamics NAV 2013 upgrade process. The tool does not actually convert the objects but performs a smart find and replace for obsolete method calls and properties, where applicable.

Note: *Developers must run this tool on the OldBase.txt and OldCustom.txt files before they try to compare the code with NewBase.txt. Otherwise, too many objects will be read because of form-page differences.*

The Text Format Upgrade Tool

The TextFormatUpgrade2013 tool is provided in the UpgradeToolKit\Object Change Tools folder on the Microsoft Dynamics NAV 2013 installation media. The TextFormatUpgrade2013 tool replaces strings in code and object properties that were appropriate for Microsoft Dynamics NAV 2009, with strings that are appropriate for Microsoft Dynamics NAV 2013.

This tool takes a single argument that is either a .txt file to convert, or a folder that contains multiple .txt files to convert. The output is written to a subdirectory of the directory that contains the source text file. The names of the output files are the same as the input files. The subdirectory is named Converted. The tool creates this subdirectory if it does not exist, and overwrites existing files. For example, if the command that runs the tool resembles the following.

Code Example

```
TextFormatUpgrade2013.exe c:\OldBaseVersion\AllObjects.txt
```

The file that the tool creates is named C:\OldBaseVersion\Converted\AllObjects.txt.

The conversions that are performed by the TextFormatUpgrade2013 tool are specified by the TextFormatUpgrade2013.exe.config file. This file uses regular expressions to specify string replacements, and must be present in the same directory as the tool itself. Users can update the configuration file to add more conversions and patterns.

The following table contains some replacements that the tool performs.

Search for	Replaced by
FORM.RUN	PAGE.RUN
FORM::	PAGE::
UseRequestForm	UseRequestPage
UseReqForm	UseRequestPage
RunFormMode	RunPageMode
CardFormID	CardPageID
SubFormView	SubPageView
SubFormLink	SubPageLink
RunFormOnRec	RunPageOnRec
RunFormLink	RunPageLink
RunFormView	RunPageView
DrillDownFormID	DrillDownPageID
LookupFormID	LookupPageID
RequestOptionsForm	RequestOptionsPage

Demonstration: Use the Text Format Upgrade Tool

Viktor, the system developer, is transferring Codeunit 80 Sales Post from Dynamics NAV 2009 SP1 to Dynamics NAV 2013. To make sure that the file does not contain invalid references, he will use the Text Upgrade Toolkit to transform the object.

Demonstration Steps

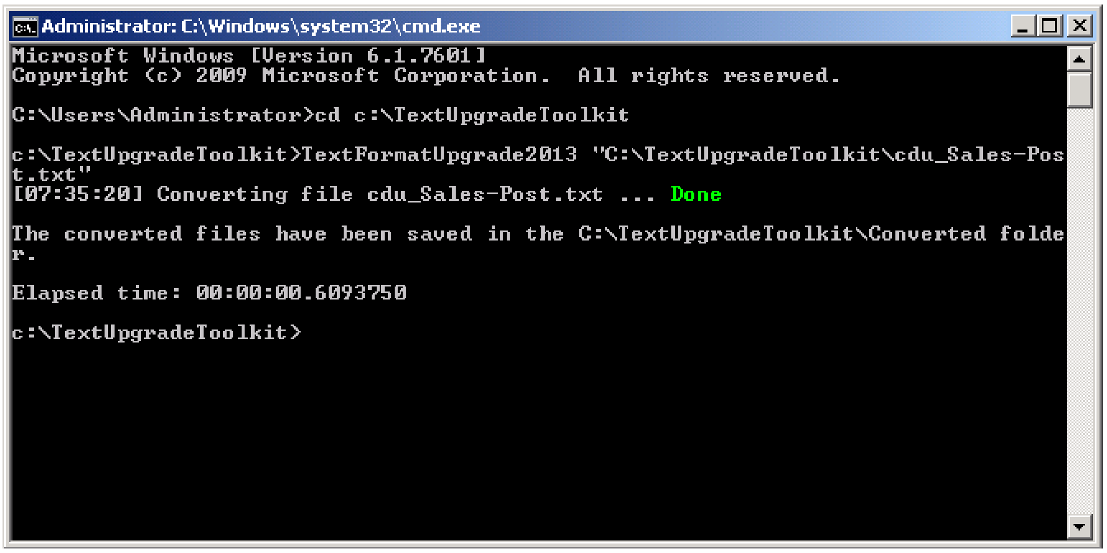
1. Create a new folder on the HyperV machine to work with the Text Upgrade Toolkit.
   a. On the HyperV, on the drive C, create a new folder named "TextUpgradeToolkit".
   b. On the X:\ drive, locate the folder in the following location: X:\NAV 2013 Setup files\DVD_Build33781\UpgradeToolkit\Object Change Tools**.**
   c. Copy and paste the two files (TextFormatUpgrade2013.exe and TextFormatUpgrade2013.exe.config) in this folder to the folder that you created named "C:\ TextUpgradeToolkit".

2. In the current custom version database, export Codeunit 80 Sales Post to a .txt file.
   a. On the HyperV, select the shortcut named “Microsoft Dynamics NAV 2009 Classic with Microsoft SQL Server”.
   b. Click **File > Database >Open** to open the **Open Database** window.
      i. If there is an error message “The Demo Database NAV (7-0) database on the NYC-SVR1\NAVDEMO server cannot be opened by this version of Microsoft Dynamics NAV Classic. The database has already been converted by a newer program version. You must upgrade Microsoft Dynamics NAV Classis to the latest version to open the database.”, click **OK** and continue.
   c. In the **Server Name** field, select: NYC-SVR1.
   d. In the **Database Name** field, select Demo Database NAV (6-0).
   e. In the **Authentication** field, select Windows Authentication.
   f. Click **OK.**
   g. Click **Tools > Object Designer** to open the **Object Designer** window.
   h. In the Object Designer, on the left toolbar, click **Codeunit.**
   i. Scroll to Codeunit 80 “Sales-Post”, and then select it in the list.
   j. Click **File > Export**. The **Export Objects** window opens.
   k. In the **File name** field, type: “C:\TextUpgradeToolkit\cdu_80_SalesPost.txt”.
   l. Click **Save**. The Codeunit is now exported.

3. Run the Text Upgrade Toolkit to convert the file
   a. On the HyperV, click **Start,** and then click **Run**. The **Run** window opens.
   b. In the **Open** field, type “cmd” to open a Command Prompt.
   c. At the command prompt, type “cd C:\TextUpgradeToolkit”, and then click **Enter**.
   d. At the command prompt, type the following command:

```
“TextFormatUpgrade2013 "C:\TextUpgradeToolkit\cdu_Sales-
Post.txt"”
```

   e. Click **Enter**. The Text Upgrade Toolkit executes and generates an output file in the folder named “Converted”, as shown in the “Text Upgrade Toolkit” figure.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>cd c:\TextUpgradeToolkit

c:\TextUpgradeToolkit>TextFormatUpgrade2013 "C:\TextUpgradeToolkit\cdu_Sales-Pos
t.txt"
[07:35:20] Converting file cdu_Sales-Post.txt ... Done

The converted files have been saved in the C:\TextUpgradeToolkit\Converted folde
r.

Elapsed time: 00:00:00.6093750

c:\TextUpgradeToolkit>
```

FIGURE 3.1:TEXT UPGRADE TOOLKIT RESULT

Object Transformation

When you upgrade to Dynamics NAV 2013, special attention is required for Report and Page Objects. When you upgrade a report, there are several possible procedures to follow , depending on the type of report.

New triggers and functionality are added to Page objects in Microsoft Dynamics NAV 2013. You should be aware of these new possibilities to improve merged and imported objects by using the new functionality where you can.

Reports Upgrade

If you want to use Microsoft Dynamics NAV 2009 SP1 reports in Microsoft Dynamics NAV 2013, then you must upgrade the reports before you can run or modify them. You can compile Microsoft Dynamics NAV 2009 SP1 reports in Microsoft Dynamics NAV 2013. However, you cannot run or design them.

Before you upgrade reports, we recommend that you create a backup of the reports. You cannot undo the upgrade after it is finished.

To back up reports, use one of the following methods:

- Export the reports to a file.
- Create a backup of the database.

For more information about how to export a report to a file, see the following website.

How to: Export Objects

http://go.microsoft.com/fwlink/?LinkId=277024

For more information about how to create a database backup, see the following website.

How to: Create Backups

http://go.microsoft.com/fwlink/?LinkId=277025

Upgrading reports is an automated process that extracts dataset information and deletes unnecessary Classic report components from the report object.

In Microsoft Dynamics NAV 2009 SP1, a report object could contain the following combinations of components:

- Classic report layout (sections) and request form
- Client report definition (RDLC) layout and request page
- Both Classic report layout (sections) and request form, and an RDLC report layout and request page.

In Microsoft Dynamics NAV 2013, a report object can only contain an RDLC layout and request page.

Upgrading Reports that Have Both Classic Report Layout and RDLC Layout

You must upgrade reports that you import from Microsoft Dynamics NAV 2009 SP1.

When you upgrade a report, the Classic report layout and the request form are deleted, and the field information from report sections is converted to a dataset definition that is valid for Microsoft Dynamics NAV 2013 reports.

Note: *Special consideration is required when you design these reports to avoid or minimize Out of Memory exceptions when running reports that process larger data volume.*

The "Upgrading Reports That Use Classic and RDLC Layout" figure shows how to upgrade a report that has both a classic report layout and a client report definition (RDLC) layout.

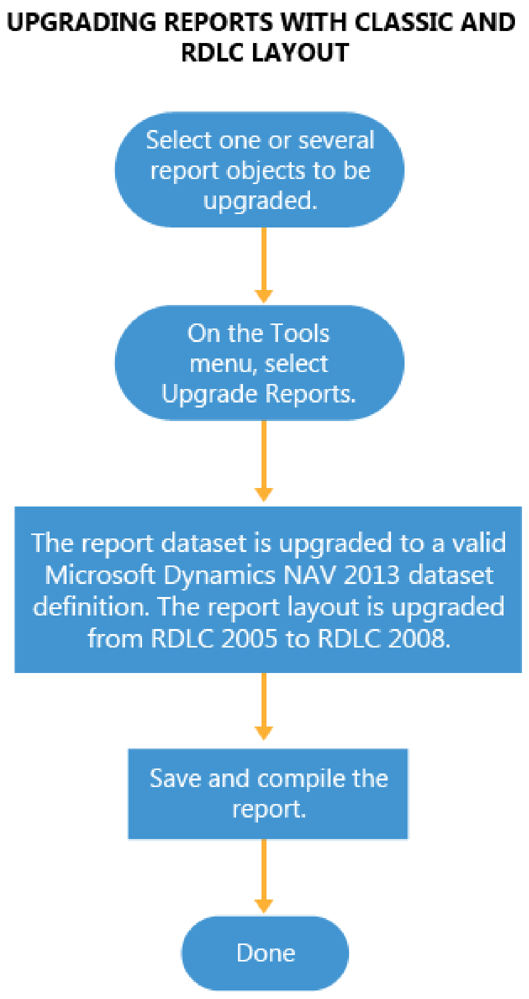

UPGRADING REPORTS WITH CLASSIC AND RDLC LAYOUT
Select one or several report objects to be upgraded.
On the Tools menu, select Upgrade Reports.
The report dataset is upgraded to a valid Microsoft Dynamics NAV 2013 dataset definition. The report layout is upgraded from RDLC 2005 to RDLC 2008.
Save and compile the report.
Done

FIGURE 3.2:UPGRADING REPORTS WITH CLASSIC AND RDLC LAYOUT

Upgrading Classic Reports

You may have Microsoft Dynamics NAV 2009 reports that do not have RDLC layouts and request pages.

For example, you may have deleted the RDLC layout from a Microsoft Dynamics NAV 2009 report so that you could start the Classic layout from the RoleTailored client. You may also have a report from a version of Microsoft Dynamics NAV that is earlier than Microsoft Dynamics NAV 2009 SP1, that does not contain an RDLC layout or a request page. To upgrade reports that do not contain an RDLC layout, upgrade the customized report as is by using the standard upgrade procedures. A layout is suggested automatically during upgrade. After you upgrade, you must manually fix any issues with the layout.

The "Upgrading a Classic Report" figure shows how to upgrade a classic report.

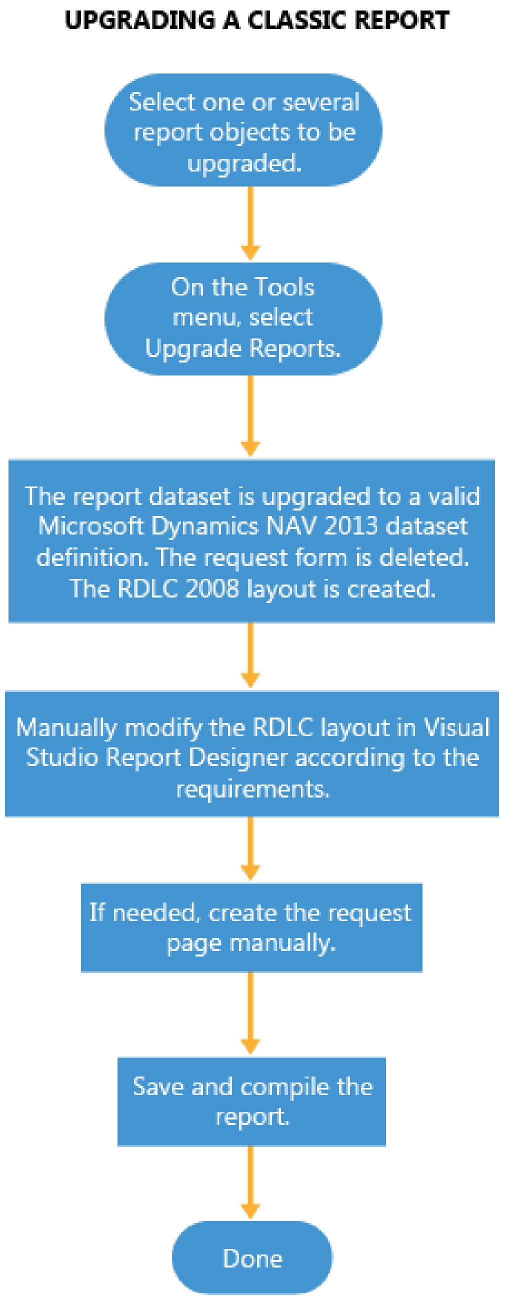

UPGRADING A CLASSIC REPORT
Select one or several report objects to be upgraded.
On the Tools menu, select Upgrade Reports.
The report dataset is upgraded to a valid Microsoft Dynamics NAV 2013 dataset definition. The request form is deleted. The RDLC 2008 layout is created.
Manually modify the RDLC layout in Visual Studio Report Designer according to the requirements.
If needed, create the request page manually.
Save and compile the report.
Done

FIGURE 3.3:UPGRADING A CLASSIC REPORT

Upgrading Processing-Only Reports

A processing-only report is a report that processes data, but produces no printed output.

The “Upgrading a Processing Only Report” figure shows how to upgrade a processing-only report.

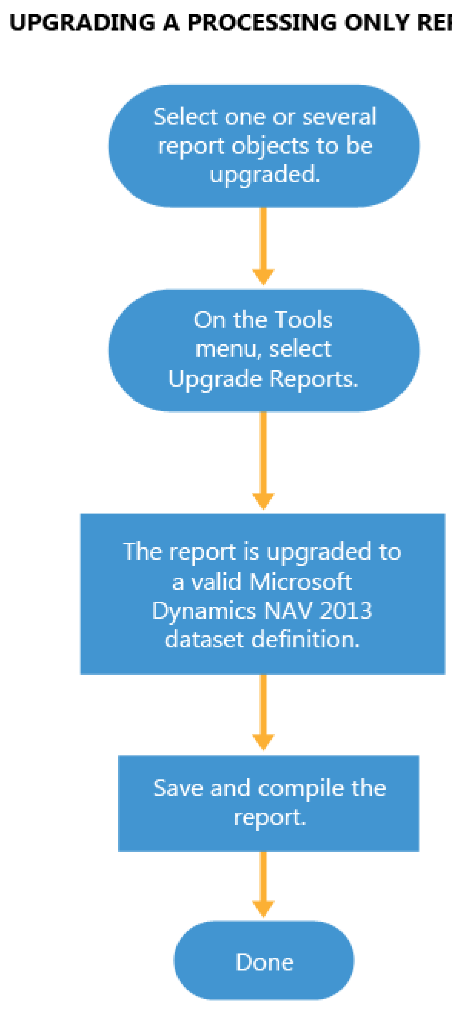

UPGRADING A PROCESSING ONLY REPORT
Select one or several report objects to be upgraded.
On the Tools menu, select Upgrade Reports.
The report is upgraded to a valid Microsoft Dynamics NAV 2013 dataset definition.
Save and compile the report.
Done

FIGURE 3.4:UPGRADING A PROCESSING ONLY REPORT

Demonstration: Upgrade a Report

This demonstration explains the steps to follow to upgrade a report after it is imported in Microsoft Dynamics NAV 2013.

Note: *A prerequisite for this demonstration is the completion of Lab A in the Project Analysis and Planning module, where Microsoft Dynamics NAV 2013 is installed.*

Demonstration Steps

1. To upgrade a report.
   a. Click **Start** and select the shortcut named Microsoft Dynamics NAV 2013 Development Environment.
   b. Click **File > Database > Open** to open the **Open Database** window.
   c. In the **Server Name** field, select NYC-SVR1\NAVDEMO.
   d. In the **Database Name** field, select Demo Database NAV (7-0).
   e. In the **Authentication** field, select Windows Authentication.
   f. Click **OK**.
   g. In the development environment, on the Tools menu, click Object Designer, in case the Object Designer is not opened. This is by default.
   h. In **Object Designer**, click **Report**, and then select the report that you want to upgrade, for example report Customer - List.
   i. On the **Tools** menu, click **Compile**.
   j. On the **Tools** menu, click **Upgrade Reports**.
   k. Click **Yes** to confirm that you want to upgrade the report.

      Notice that the Classic report layout and request form of the report is deleted during the upgrade.

   l. On the **Object Designer**, select the report, and then on the **Tools** menu, click **Compile**. In the dialog box, click **Yes** to confirm that you want to compile.

Note: *For any report in which the table data region spans more than one page, you must use the* ***SetData*** *and* ***GetData*** *functions so that the header information is displayed on the next pages. This process differs in Microsoft Dynamics NAV 2013 from Microsoft Dynamics NAV 2009 SP1. Update your reports by using the Microsoft Dynamics NAV 2013 procedure for using the* ***SetData*** *and* ***GetData*** *functions to correctly display header information.*

Verify that elements on the report are positioned correctly. During upgrade, elements may be repositioned because of rounding to the report height and width. You change the position of elements on a report by using Visual Studio Report Designer.

More information is available on the following website.

How to: Print Report Header Information on Multiple Pages

http://go.microsoft.com/fwlink/?LinkId=277032

Determine the Reports to Upgrade

You can use report usage logging to identify both frequently and rarely used reports in a customer installation. This information can be useful when you must determine which Classic reports should be transformed to RDLC reports. This only works in earlier version(s) of Dynamics NAV. A custom development could be an alternative to this solution.

To log report usage, you must download and install one of the following hot fixes

Microsoft Dynamics NAV 5.0: KB2575296

http://go.microsoft.com/fwlink/?LinkId=277026

Microsoft Dynamics NAV 2009: KB2558650

http://go.microsoft.com/fwlink/?LinkId=277027

After you install the required hot fix, you must set up report usage logging by following these steps:

1. Create a new table to log report usage.

Code Example

```
OBJECT Table 50000 Report Log

{

OBJECT-PROPERTIES

{

Date=11-06-12;

Time=12:36:58;

Modified=Yes;

Version List=NAV6;

}

PROPERTIES

{

}

FIELDS

{

{ 1 ; ;No. ;Integer ;AutoIncrement=Yes;

MinValue=1 }

{ 2 ; ;User ID ;Code50 ;TableRelation="User Role"."Role ID";

CaptionML=ENU=User ID }

{ 3 ; ;Report ID ;Integer ;CaptionML=ENU=Report ID }

{ 4 ; ;Report Name ;Text249 ;FieldClass=FlowField;

CalcFormula=Lookup(AllObjWithCaption."Object Caption" WHERE (Object
Type=CONST(Report),

Object ID=FIELD(Report ID)));

CaptionML=ENU=Report Name }

{ 5 ; ;Date Time ;DateTime }
}

KEYS

{

{ ;No. ;Clustered=Yes }

}

FIELDGROUPS

{

}

CODE

{

BEGIN

END.

}

}
```

2. Open Codeunit 1.
3. Open **C/AL Globals**.
4. Browse to **Functions**.
5. Create a new function named **OnReportRun**.
6. Open **Properties,** and then change the **ID** to "120".
7. Now select **View**, **Locals,** and then create a parameter named ReportId with a type Integer.
8. Select the **Variables** tab, and then create a variable named ReportLog.
9. Open **C/AL Editor** and scroll to the bottom to write the code for this new trigger.
10. In the **OnReportRun** trigger, write the following code.

Code Example

```
ReportLog."User ID" := USERID;

ReportLog."Report ID" := ReportId;

ReportLog."Date Time" := CURRENTDATETIME;

ReportLog.INSERT;
```

11. Restart the Classic client and run several reports.
12. Run the Report Log table to view the result.

Note: *The report logging tool only works for reports that have a classic layout. Reports that contain an RDLC layout can be logged by customizing the C/AL code in the OnInitReport or OnPreReport trigger.*

Upgrade Pages

In Microsoft Dynamics NAV 2013, improvements are made to page development. When you upgrade page objects, we recommend that you review the page object and analyze whether it is in the customer's best interest to use this new functionality.

Page Field Arrangement in a Grid

With the new GridLayout page control, you can lay out fields in rows and columns. You use the GridLayout control to span a field over rows or columns, and show or hide field labels.

By default, page fields are arranged automatically in two columns. You can use a GridLayout control or a FixedLayout control to arrange fields in rows and columns on a page.

More information about the GridLayout is available on the following websites.

Arranging Fields in Rows and Columns

http://go.microsoft.com/fwlink/?LinkId=277028

How to: Arrange Fields in Rows and Columns Using the GridLayout Control

http://go.microsoft.com/fwlink/?LinkId=277029

OnAfterGetCurrRecord Trigger

This trigger executes after the current record is retrieved from the table.

In a page that has a repeater, the trigger is only called when the current record in the repeater is updated. The OnAfterGetCurrRecord trigger is called directly after all OnAfterGetRecord triggers is called for each row in a list

If there is an error in the trigger code, then the page is closed.

In Microsoft Dynamics NAV 2013 there are several new, removed, or changed C/AL data types, functions, properties, and triggers since the earlier version of Microsoft Dynamics NAV. The Text Format Upgrade Tool resolves most of these changes.

Information about changes in C/AL behavior and support from earlier versions of Microsoft Dynamics NAV is available on the following website.

Changes in C/AL Behavior and Support from Earlier Versions of Microsoft Dynamics NAV

http://go.microsoft.com/fwlink/?LinkId=277030

Transformation Tool

To use the functionality and logic of forms in the RoleTailored client in Microsoft Dynamics NAV 2009 SP1, you must first transform the forms into pages. The scope of this course does not include details about how to transform forms to pages, because we assume that this transformation has happened already in NAV 2009 SP1.

Note: *The transformation tool is not included with Microsoft Dynamics NAV 2013. It is available in the product DVD of Microsoft Dynamics NAV 2009.*

Use the form transformation tool to automate transformation from forms to pages.

Form Transformation enables you to perform the following tasks:

- Transform forms to pages (basic transformation).
- Customize resulting pages for the RoleTailored client (UX transformation).

Form-to-page transformation is a basic transformation process that takes form objects from the Microsoft Dynamics NAV Classic application and creates page objects that you can use in the RoleTailored client.

The Transformation Tool converts form objects in the Classic client to page objects that you can use in the RoleTailored client. The tool uses a set of rules that map objects in the Classic client to objects in the RoleTailored client. These rules are contained both in the transformation input files and in the transformation tool source files. In addition to the transformation input files, which specify the modifiable rules for transformation, the transformation tool uses schema files, a configuration file, and a code rules files to transform the source forms in the forms.xml file into pages.

The transformation tool creates a pages.xml file. This file contains the pages that you can view in the RoleTailored client. The transformation also creates a log output file.

The "Transformation Tool Overview" figure shows the form transformation framework and how the transformation tool fits into that framework.

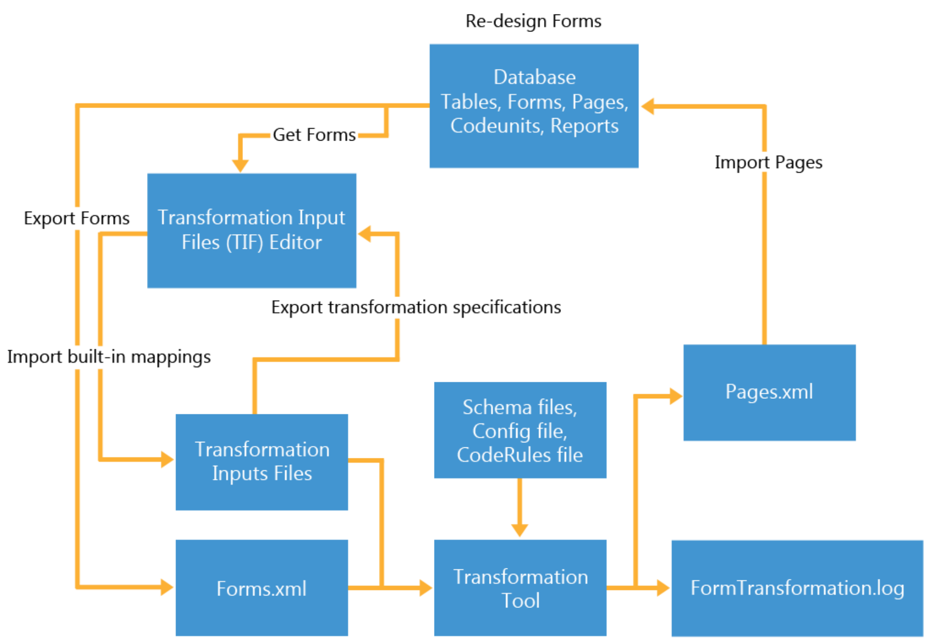

Re-design Forms
Database Tables, Forms, Pages, Codeunits, Reports
Get Forms
Import Pages
Export Forms
Transformation Input Files (TIF) Editor
Export transformation specifications
Import built-in mappings
Pages.xml
Schema files, Config file, CodeRules file
Transformation Inputs Files
Forms.xml
Transformation Tool
FormTransformation.log

FIGURE 3.5:TRANSFORMATION TOOL OVERVIEW

As this illustration shows, transforming forms is an iterative process. You may have to run the transformation tool multiple times on a given form to create a page that has the functionality that you want. After you run the tool and view the transformed page, you may want to do the following:

- Modify some rules that the tool uses.
- Produce different page functionality.
- Modify the user interface (UI) for the page.

You change the rules by modifying the transformation input files. You can modify these files directly or you can use the Transformation Input Files (TIF) Editor to modify input files.

More information about the transformation process from forms to pages is available on the following website.

Transforming Forms to Pages

http://go.microsoft.com/fwlink/?LinkID=277017&clcid=0x409

Upgrade of Dimensions

Dimension management functionality is redesigned in Microsoft Dynamics NAV 2013. It uses fewer tables, reduces the data that is stored in the database, and improves performance. The Data Upgrade process automatically translates the old dimension data structures to new data structures. However, understanding how dimensions are structured in Microsoft Dynamics NAV 2013 helps you customize the data upgrade procedures if the dimension management functionality of your customer's database is customized or if new, additional, or custom tables are integrated with it.

Dimension Sets

A *dimension set* is a unique combination of dimension values. It is stored as dimension set entries in the database. Each dimension set entry represents a single dimension value. The dimension set is identified by a common dimension set ID that is assigned to each dimension set entry that belongs to the dimension set.

The following example shows a dimension set that has three dimension set entries. The dimension set is identified by the dimension set ID 108.

Dimension Set ID	Dimension Code	Dimension Value Code	Dimension Value Name
108	AREA	70	America North
108	BUSINESSGROUP	HOME	Home
108	DEPARTMENT	SALES	Sales

Dimension Set Entries

Dimension sets are stored in the **Dimension Set Entry** table as dimension set entries with the same dimension set ID.

The "Dimension Set Entry" diagram shows how different tables use the same ID to relate to the same combination of dimension values.

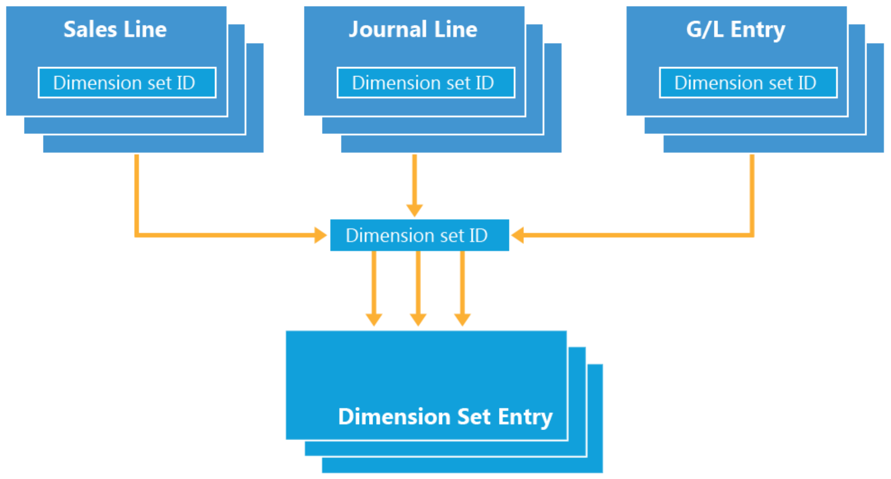


FIGURE 3.6:DIMENSION SET ENTRY

When you create a new journal line, document header, or document line, you can specify a combination of dimension values. Instead of explicitly storing each dimension value in the database, a dimension set ID is assigned to the journal line, document header, or document line to specify the dimension set.

When you edit and close the **Edit Dimension Set Entries** window, a check is performed to see whether the combination of dimension values exists as a dimension set in the table. If the combination occurs in the table, then the corresponding dimension set ID is assigned to the journal line, document header, or document line. Otherwise, a new dimension set is added to the table, and the new dimension set ID is assigned to the journal line, document header, or document line.

Table Structure

In Microsoft Dynamics NAV 2013, there are no specific tables to manage dimension information for documents, journals, and entries, as in earlier versions. Specific dimension tables are replaced with dimension set entries.

New Tables

Three new tables manage dimension set entries, as follows.

Table	Remarks
480, Dimension Set Entry	You cannot change this table. After data is written to the table, you cannot delete or edit it. Deleting data requires you to check against all occurrences of the dimension set ID in the whole database. This includes partner solutions.
481, Dimension Set Tree Node	You cannot change this table. It searches for a dimension set. If the dimension set is not found, a new set is created.
482, Reclas. Dimension Set Buffer	The table edits a dimension set ID. It is required when you edit a dimension value code and a new dimension value code, for example, in the **Item Reclas. Journal** table.

Changed Tables

A new field, 480 **Dimension Set ID**, is added to the following tables. For the tables that store posted data, the field only provides a noneditable display of dimensions. This is marked as Drill-down. For the tables that store working documents, the field is editable. The buffer tables that are used internally do not have to have editable or noneditable capabilities.

The 480 field is noneditable in these tables, as follows.

Table No.	Table Name
17	G/L Entry
21	Cust. Ledger Entry
25	Vendor Ledger Entry
32	Item Ledger Entry
110	Sales Shipment Header
111	Sales Shipment Line
112	Sales Invoice Header
113	Sales Invoice Line
114	Sales Cr.Memo Header

Table No.	Table Name
115	Sales Cr.Memo Line
120	Purch. Rcpt. Header
121	Purch. Rcpt. Line
122	Purch. Inv. Header
123	Purch. Inv. Line
124	Purch. Cr. Memo Hdr.
125	Purch. Cr. Memo Line
169	Job Ledger Entry
203	Res. Ledger Entry
271	Bank Account Ledger Entry
281	Phys. Inventory Ledger Entry
297	Issued Reminder Header
304	Issued Fin. Charge Memo Header
5107	Sales Header Archive
5108	Sales Line Archive
5109	Purchase Header Archive
5110	Purchase Line Archive
5601	FA Ledger Entry
5625	Maintenance Ledger Entry
5629	Ins. Coverage Ledger Entry
5744	Transfer Shipment Header
5745	Transfer Shipment Line
5746	Transfer Receipt Header
5747	Transfer Receipt Line
5802	Value Entry
5832	Capacity Ledger Entry
5907	Service Ledger Entry
5908	Service Header
5933	Service Order Posting Buffer
5970	Filed Service Contract Header
5990	Service Shipment Header

Table No.	Table Name
5991	Service Shipment Line
5992	Service Invoice Header
5993	Service Invoice Line
5994	Service Cr. Memo Header
5995	Service Cr. Memo Line
6650	Return Shipment Header
6651	Return Shipment Line
6660	Return Receipt Header
6661	Return Receipt Line

The 480 field is editable in these tables.

Table No.	Table Name
36	Sales Header
37	Sales Line
38	Purchase Header
39	Purchase Line
81	Gen. Journal Line
83	Item Journal Line
89	BOM Journal Line
96	G/L Budget Entry
207	Res. Journal Line
210	Job Journal Line
221	Gen. Jnl. Allocation
246	Requisition Line
295	Reminder Header
302	Finance Charge Memo Header
5405	Production Order
5406	Prod. Order Line
5407	Prod. Order Component
5615	FA Allocation
5621	FA Journal Line

Table No.	Table Name
5635	Insurance Journal Line
5740	Transfer Header
5741	Transfer Line
5900	Service Header
5901	Service Item Line
5902	Service Line
5965	Service Contract Header
5997	Standard Service Line
7134	Item Budget Entry
99000829	Planning Component

Note: *Table 83,* ***Item Journal Line*** *includes the field 481,* ***New Dimension Set ID****, to manage dimension reclassification by using the Item Reclassification Journal.*

The 480 field is added to the these buffer tables, as follows.

Table No.	Table Name
49	Invoice Post. Buffer
212	Job Posting Buffer
372	Payment Buffer
382	CV Ledger Entry Buffer
461	Prepayment Inv. Line Buffer
5637	FA G/L Posting Buffer
7136	Item Budget Buffer

To learn more about changes in dimension management in Microsoft Dynamics NAV 2013, refer to the Financial Management white paper at the following website.

White Paper - Dimension Set Entries

http://go.microsoft.com/fwlink/?LinkId=277031

Test of the Code Upgrade

After developers complete the code upgrade, you must make sure that the application functions and works correctly. A set of test data that is based on the customer's data can be created to make sure that basic tasks can be performed in the system.

To test whether the code upgrade to Microsoft Dynamics NAV 2013 succeeded, the following testing types must be used:

- Code review
- UI Testing
- Functional Testing

Code Review

Code review is performed to make sure that all code was correctly transferred and that no functionality was lost during the code upgrade. Code review is performed for all merged custom and upgraded standard objects.

After developers complete the code upgrade, the following versions of objects are available and must be compared to check the correctness of code merging:

- Old Base version
- Current Custom version
- New Base version
- New Custom version

Code review includes the following steps:

1. Define a list of objects for code review and distribute it to the team.
2. Compare versions between each object code by using special comparison tools.
3. Determine whether code was transferred correctly from the source solution to the target version. Developers or testers make this determination during code review.

The rules of code transfer are specified in the Upgrade Customizations topic of this module.

You can find the following four major bugs during code review:

- Code that the customer added in the old customized version was not transferred to the new customized version.
- Standard code of the new base version was deleted from the new customized version.

- Standard code that was removed in the new base version was transferred from the old customized version to the new customized version.
- Code that was deleted by the customer in the old customized version was not deleted in the new customized version.

UI Testing

UI testing verifies that the mainstream functionality (both standard and customer) executes without any unexpected errors, such as missing active keys, incorrect sorting order, and so on. In addition, UI testing determines whether the program starts and whether its interfaces are available and responsible. In other words, it touches all areas of the application that were changed by the customer without going into detail, and provides answers to the following basic questions:

- Is it possible to open the test item at all?
- Do the buttons and menu items on the forms trigger the correct actions or open the required windows?

Testing objectives for UI testing are shown in the following table for two object types:

- **Merged standard objects** – Standard objects that were changed by the customer and were merged by developers during code upgrade
- **New custom objects** – Objects that were added by the customer and upgraded by developers during code upgrade.

Testing objectives for UI testing are shown in the following table.

Object Type	Object Source	Testing Objectives (What is verified)
Page	Merged Standard	The page runs without producing error messages about missing active keys, incorrect sorting order, and so on. The records, lines, and fields are created (completed), modified, and deleted without unexpected error messages. All the necessary FastTabs, fields, and text boxes are present. Positions of actions, header and lines FastTabs, text boxes, and labels on FastTabs are correct. All the **AssistEdit(...)** buttons open the correct object, and you do not receive an error messages. In case new actions are added, standard shortcuts are not duplicated. If standard shortcuts are duplicated, communicate with the customer to define the correct priorities.
Page	New Custom	The page runs without producing error messages about missing active keys, incorrect sorting order, and so on. The records, lines, and fields can be created (completed), modified, and deleted without unexpected error messages.
Report	Merged Standard	The report runs and previews without producing error messages about missing active keys, incorrect sorting order, and so on. All the necessary tabs, fields, textboxes, and buttons are present on the report request page.
Report	New Custom	The report runs and previews without error messages about missing active keys, incorrect sorting order, and so on.
XMLPort	Merged Standard	The XMLport executes without producing error messages about missing active keys, incorrect sorting order, and so on. All the necessary tabs, fields, textboxes, and buttons are present on the request page.

Object Type	Object Source	Testing Objectives (What is verified)
XMLPort	New Custom	The XMLport executes without producing error messages about missing active keys, incorrect sorting order, and so on.
MenuSuite	Both merged standard and new custom	All required menu items are available and the behavior of menu items does not differ from that in the customer's solution.

SQL Server Administration

Microsoft Dynamics NAV 2013 only supports the Microsoft SQL Server database option. If the customer used Microsoft Dynamics NAV 2009 Database (native database option,) they must upgrade to Microsoft SQL Server.

Accordingly, you perform many administrative tasks in Microsoft Dynamics NAV 2013 by using Microsoft SQL Server management tools. The primary tool to manage Microsoft SQL Server is Microsoft SQL Server Management Studio.

This is a part of the standard SQL Server installation.

Backup Creation and Restoration

Microsoft SQL Server Management Studio lets you create and restore backups of SQL Server databases. This includes the databases that you use with Microsoft Dynamics NAV 2013. Even though it is still possible to use Microsoft Dynamics NAV 2013 Development Environment to back up and restore Microsoft Dynamics NAV databases, Microsoft SQL Server Management Studio is faster, and provides much more flexibility about backup tasks, such as the following:

- Create data and transaction log backups separately.
- Create incremental or differential backups.
- Define backup schedules.
- Restore a database to a point in time.

Note: *The Backup and Restore features of the Microsoft Dynamics NAV 2013 Development Environment are still the only option for backing up or restoring individual companies.*

Create a Backup

To create a backup of the database by using Microsoft SQL Server Management Studio, follow these steps:

1. Click **Start > All Programs > Microsoft SQL Server 2012 > SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, specify the server and the authentication options, and then click **Connect**.
3. In Object Explorer, expand the node that represents the server to which you are connected, and then expand **Databases**.
4. Right-click the database that you want to back up, and then click **Tasks > Back Up**.
5. Make sure that the **Backup type** is set to Full, and that **Destination** is set to Disk.
6. If the list of destinations is empty, click **Add** to open the **Select Backup Destination** dialog box.
7. Specify the file name, or browse to a specific file, and then click **OK**.
8. If the file that is specified in the list of destinations is wrong, select it, and then click **Remove.** Click **Add** to specify the destination file.

Note: *You can create multiple backup sets of the same database in a single backup operation. You can have multiple backup sets in the same backup file. You can restore a specific backup set later from a backup file that contains multiple backup sets.*

9. Click **OK** to create the backup.
10. When the backup is complete, it shows a success dialog box. Click **OK** to close it.

Restore a Backup

To restore a database from a backup file, follow these steps:

1. Do one of the following:
   - In SQL Server Management Studio, expand the node that represents the server to which you are connected, right-click **Databases**, and then click **Restore Database**.
   - Right-click the specific database that you want to restore to a previous state, and then click **Tasks > Restore > Database**.
2. In the **Restore Database** window, specify the source as either an existing database or device. When you select an existing database, the list of backup sets for that database is shown in the **Backup sets to restore** list. When you select a device as a source, then you can browse to the file that contains the backup.

3. In the **Backup sets to restore** list, select the backup set that you want to restore.
4. On the **Options** page, clear the Tail-Log Backup option.
5. Click **OK** to complete the restore operation.

Create a Copy of a Database by Using Restore

You can use the Restore feature to create a copy of an existing database. To do that, follow these steps:

1. Create a backup of the database by using the steps that were described earlier.
2. Right-click the database that you backed up, and then click **Tasks > Restore > Database**.
3. In the **Restore Database** window, under the Destination group, enter the new database name in the **Database** field. Make sure that you specify a unique name.
4. Click the **Files** page. This shows the list of data and transaction log files of the source database.
5. To specify a unique file name for each file, in the **Restore As** field, change the file name by either typing a unique file name, or by browsing to it. If you decide to type the file names, make sure that you keep the directory path, and only change the file name part.

Note: *The restore operation fails if any of the file names that you specify in the list is already being used by any of the files for any of the databases on the server.*

6. Click **OK** to complete the restore operation.

Copy of a Database Using a SQL Server Agent Job

You can use the SQL Server Agent feature to create a copy of an existing database. It is simpler than using a backup and restore process, and provides less room for error. It also lets you copy databases between servers.

To copy a database, follow these steps:

1. In SQL Server Management Studio, right-click the database that you want to copy, and then click **Tasks > Copy Database**. This starts the Copy Database Wizard.
2. Click **Next**.
3. On the **Select a Source Server** page, make sure that the **Source server** field contains the name of the server from which you are copying the database.

4. If it is necessary, specify the authentication options for the source database, and then click **Next**.
5. On the **Select a Destination Server** page, make sure that the **Destination server** field contains the name of the server to which you are copying the database.
6. If it is necessary, specify the authentication options for the destination server, and then click **Next**.
7. On the **Select the Transfer Method** page, select **Use the detach and attach method**.

Note: *If other users are connected to the database that you are copying, they must disconnect before you can create a copy of the database. If the database is a Microsoft Dynamics NAV database that has a Microsoft Dynamics NAV Server connected to it, you must first stop the Microsoft Dynamics NAV Server.*

8. Click **Next**.
9. On the **Select Databases** page, make sure that the database that you want to copy is selected, and then click **Next**.
10. On the **Configure Destination Database (1 of 1)** page, specify the new database name in the **Destination database** field. Make sure that the name is unique.
11. Click **Next**.
12. On the **Configure the Package** page, accept the default settings, and then click **Next**.
13. On the **Schedule the Package** page, accept the default settings, and then click **Next**.

Note: *Default settings cause the SQL Server Agent to immediately create a copy of the database. Or, you can define another schedule that SQL Server Agent uses to create the copy.*

14. On the **Complete the Wizard** page, click **Finish** to create a database copy job, and to start the job immediately.
15. The **Performing Operation** page shows the copying progress. After the process is complete, click **Close**.

User Management

Use SQL Server Management Studio to manage user accounts that have access to the server, and to grant access to user accounts to specific databases. Before a user can access a server, there must be a login for that user.

Determine whether a User Has Access to SQL Server

To determine whether a user has access to SQL Server, follow these steps:

1. In Object Explorer, expand **Security > Logins**.
2. In the list of logins under the **Logins** node, look up the user account that you are looking for. Windows logins are represented as their NTLM names, in the form DOMAIN\User or COMPUTER\User. SQL Server logins are represented only by their name.

Note: *A login can be disabled. A disabled login is represented by a small red downward arrow in the lower-right corner of the user icon next to the login name.*

Create a Login

To create a login for a user, follow these steps:

1. Expand the **Security** node, and then right-click **Logins**.
2. Click **New Login** to open the **Login – New** window.
3. If you are creating a new Windows login, make sure that **Window authentication** is selected, and then click **Search** to look up the login in the local computer or in the Active Directory.

Note: *If you know the exact NTLM name for the login, you can type it directly in the **Login name** field.*

4. If you are creating a new SQL Server login, select the **SQL Server Authentication** option, specify the **Login name**, and then define the password for the new login.
5. Click **OK** to create a new login.

Map a Login to a Database User

Creating a login for a user does not give that user implicit user rights to any of the databases on the server. A login must be mapped to a database user to access a database.

To determine database user rights for a login, follow these steps:

1. Under **Security > Logins**, double-click the login for which you want to determine database user rights.
2. In the **Login Properties** window, click **User Mapping**. Each database that the login has access to contains a check mark in the **Map** field in the **Users mapped to this login** list.
3. To see specific database roles that the user has, select the database for which you want to determine role membership. The **Database role membership** list contains the list of database roles. The roles that the user belongs to are selected.
4. Use the **Login Properties** window to change database access for a user. To grant access to a database to a login, select the **Map** field for that database, and then select the appropriate roles in that database to which you want the database user to belong.

Note: *When a login is mapped to a database user, at a minimum it belongs to the public role. You cannot remove a user from the public role.*

Determine Database Users

To determine which users have access to a database, follow these steps:

1. In Object Explorer, expand **Databases**, expand the database for which you are determining users, and then expand **Security > Users**.
2. To determine the login that maps to a specific database user, double-click that user.
3. In the **Database User** window, the **Login name** field contains the login name that maps to the database user.

Microsoft Dynamics NAV Administration Console

Microsoft Dynamics NAV 2013 includes an administration console that lets you manage Microsoft Dynamics NAV instances. By using Microsoft Dynamics NAV Administration console, you can perform the following actions:

- Start, stop, or restart a Microsoft Dynamics NAV 2013 server instance.
- Create a new Microsoft Dynamics NAV 2013 server instance.
- Remove a Microsoft Dynamics NAV 2013 server instance from a computer.
- Modify all configuration options for a Microsoft Dynamics NAV 2013 server instance.

To run the Microsoft Dynamics NAV Administration console, click **Start > Administration Tools > Microsoft Dynamics NAV Administration**.

Note: *Microsoft Dynamics NAV Administration console lists all Microsoft Dynamics NAV server instances. This includes Microsoft Dynamics NAV 2009. However, you cannot manage, create, stop, start, or remove Microsoft Dynamics NAV 2009 server instances by using the console.*

Additional Reading: *To learn more about the Microsoft Dynamics NAV Administration console, refer to the 80438 Microsoft Dynamics NAV 2013 Installation and Configuration training course.*

Instance Start, Stop, and Restart

To start, stop, or restart a Microsoft Dynamics NAV 2013 instance, follow these steps:

1. Click **Console Root > Microsoft Dynamics NAV (local)**.
2. In the **Microsoft Dynamics NAV Server Instance** list, select the instance that you want to start, stop, or restart.
3. In the **Actions** pane, under the action group for the selected server instance, click **Start**, **Stop**, or **Restart**. Or, right-click the instance, and then click the appropriate action.

Note: *If the instance is running, the **Start** action is not available. If the instance is stopped, the **Stop** action is not available.*

New Instance Creation

To create a new Microsoft Dynamics NAV 2013 server instance, follow this procedure:

1. Right-click the **Microsoft Dynamics NAV (local)** node.
2. Click **Add Instance**. The **Server Instance** page is displayed.
3. Specify the instance name and ports for management services, client services, SOAP services, and OData services.
4. Specify the Login Account information for the new Microsoft Dynamics NAV server instance.
5. Click **OK**.

Note: *The instance name and port numbers must be unique. No two instances that are created through the Microsoft Dynamics NAV Administration console can share the same instance name or port numbers. If you want several instances to share the same port numbers, you must create the instance manually at the command prompt, or by using PowerShell. Windows PowerShell® is a task-based command-line shell and scripting language designed especially for system administration.*

Instance Management

You can change all configuration settings for an instance at any time. To manage configuration settings for an instance, follow these steps:

1. Under the **Console Root**, click the arrow to the left of Microsoft Dynamics NAV (local) to open the list of instances. Then click the instance that you want to manage from the list in the left pane (do not use the instance name in the middle pane).
2. At the bottom of the central pane, click **Edit**.
3. Expand the FastTab that contains the option that you want to configure, and then change the value.
4. To accept and save the changes, click **Save**. To reject the changes, click **Cancel**.

Note: *You must restart the instance to apply any new configuration settings. Microsoft Dynamics NAV service tier only reads the configuration settings at the startup of the server instance.*

Lab A: Upgrade a Report

Scenario

Simon is a Microsoft Certified Partner who works for CRONUS International Ltd. CRONUS International Ltd. asked him to upgrade the Classic version of the Customer List report that CRONUS used in Microsoft Dynamics NAV 2009 SP1 to Microsoft Dynamics NAV 2013.

Note: *A prerequisite for this lab is the completion of Lab A in the Project Analysis and Planning module, where Microsoft Dynamics NAV 2013 is installed.*

Exercise 1: Export the Report

Exercise Scenario

Simon, the system developer, will export the report 101 "Customer – List" from Dynamics NAV 2009 SP1 database.

Task 1: Export the Report

High Level Steps

1. Export Report 101 Customer – List.

Detailed Steps

1. Export Report 101 Customer – List.
   a. Select the shortcut named "Microsoft Dynamics NAV 2009 Classic with Microsoft SQL Server".
   b. Click **File > Database > Open** to open the **Open Database** window.
   c. In the **Server Name** field, select NYC-SVR1.
   d. In the **Database Name** field, select Demo Database NAV (6-0).
   e. In the **Authentication** field, select Windows Authentication.
   f. Click **OK**.
   g. Click **Tools > Object Designer** to open the **Object Designer** window.
   h. In Object Designer**,** click **Report** (on the left toolbar).
   i. Scroll to Report 101 Customer – List, and then select it in the list.
   j. Click **File > Export**. The **Export Objects** window opens.
   k. In the **File name** field, type "C:\old_CustomerList_101.txt".
   l. In the **Save as Type** field, select Text Format (.txt).
   m. Click **Save**. The report is now exported.

Exercise 2: Import and Upgrade the Report

Exercise Scenario

Simon will upgrade the Classic report. This process deletes the request form, automatically creates an RDLC report layout, and upgrades the report to be valid for Microsoft Dynamics NAV 2013. Simon uses the following steps to import a Classic-only report and upgrade it.

Task 1: Import and Upgrade the Report

High Level Steps

1. Import report 101 Customer – List.
2. Upgrade the Classic Report.

Detailed Steps

1. Import report 101 Customer – List.
    a. Select **Start** and then the shortcut named Microsoft Dynamics NAV 2013 Development Environment.
        i. If there is an error message "The Demo Database NAV (6-0) database on the NYC-SVR1 server cannot be opened by this version of Microsoft Dynamics NAV Classic. The database has already been converted by a newer program version. You must upgrade Microsoft Dynamics NAV Classis to the latest version to open the database.", click **OK** and continue.
    b. Click **File > Database > Open** to open the **Open Database** window.
    c. In the **Server Name** field, select NYC-SVR1\NAVDEMO.
    d. In the **Database Name** field, select Demo Database NAV (7-0).
    e. In the **Authentication** field, select Windows Authentication.
    f. Click **OK**.
    g. In the development environment, on the Tools menu, click Object Designer, in case the Object Designer is not opened. This is by default.
    h. In Object Designer, on the **File** menu, click **Import**.
    i. In the **Import Objects** window, browse to the report object that contains only the Classic report layout (C:\old_CustomerList_101.txt), and then click **Open**. If you are prompted, click **OK** to open the Import worksheet to resolve any conflicts. Click **OK** to import the report.

2. Upgrade the Classic Report.
    a. In Object Designer, click **Report**, and select Report 101, Customer List. This is the list that you imported in the previous step.
    b. Select **Tools > Compile** and then click Yes, then click **Tools > Upgrade Reports**.
    c. In the dialog window, click **Yes** to confirm that you want to upgrade the report.
    d. In the **Tools** menu, click **Compile** to compile the report that you upgraded in the previous step.

Exercise 3: Modify the Layout of the Upgraded Report

Exercise Scenario

Hidden fields in the RDLC report layout can cause formatting problems when Simon prints the report because these fields are outside the width of standard paper. To eliminate formatting problems, Simon must move these hidden fields to another location on the report that is within the width of standard paper.

In the upgraded report, the captions are a part of the dataset. Simon will optimize the report to use the property Include Caption and labels.

Task 1: Open the Report in Visual Studio Report Designer

High Level Steps

1. Open the report in Visual Studio Report Designer.
2. Move the hidden fields.
3. Add the Functionality from the Section Trigger.
4. Enable the Include Caption property.
5. Remove the old caption fields from the dataset.
6. In Visual Studio Report Designer, replace the old caption fields by using the new captions.
7. In the Report Label Designer add labels.
8. In Visual Studio Report Designer, replace the old caption fields by using the new labels.

Detailed Steps

1. Open the report in Visual Studio Report Designer.
    a. In Object Designer, select report 101 Customer – List. Click **Design > View**, and then click **Layout**. The Visual Studio Report Designer opens.
        i. If it is the first time Visual Studio is opened the **Choose Default Environment Settings** window will open.
        ii. Select **Business Intelligence Settings** and then click **Start Visual Studio**.

2. Move the hidden fields.
   a. In Visual Studio, in the Report.rdlc file that is created in the Body section of the layout, select the following hidden fields on the right side of the table. (They are marked in Red)

```
=FIELDS!COMPANYNAME.VALUE

=FIELDS!Customer__ListCaption.VALUE

=FIELDS!CurrReport_PAGENOCaption.Value
```

   b. Right-click the selected fields, and then click **Cut**.
   c. Right-click the row handle of the last row, and then click **Insert row below** to insert a new row in the table.
   d. Right-click the first field in the new row, and then click **Paste**.
   e. On the **View** menu, click the **Properties** window.
   f. Select the first text box in the new row. In the **Properties** window, under Size, change the value of the Height property to 0.2 cm. The height 0.2 cm is sufficient to view the row in the layout, and it does not cause formatting problems when you print the report.
   g. This report has unsupported code on the section trigger for Customer, Header (3). You must add the functionality for this code elsewhere in the report so that it will run in Microsoft Dynamics NAV 2013. This is because code in a section trigger is not supported and will not be migrated.

3. Add the Functionality from the Section Trigger.
   a. In Visual Studio, on the **View** menu, click the **Properties** window.
   b. In the Report.rdlc file, in the Body section of the layout, select the field at the top of the layout with the following value.

```
=First(Fields!Customer_TABLECAPTION__________CustFilter.Value)
```

   c. In the drop-down list for the **Hidden** propertie, select <Expression>.
   d. In the Expression window, type the following expression (if it is not already present).

```
=IIF(Fields!CustFilter.Value<>"",False,True)
```

   e. Click **OK**.
   f. Save and close Visual Studio.
   g. Click the Report Dataset Designer and, when you are asked to load the changes, click **Yes**.

4. Enable the Include Caption property.
   a. In the Report Dataset Designer, select the field that has the name Customer__No__, and then click **Include Caption** to enable it.
      i. **Include Caption** is the last column in the Report Dataset Designer.

   b. Repeat step a for the following fields:
      i. Customer__Customer_Posting_Group_
      ii. Customer__Customer_Disc__Group_
      iii. Customer__Invoice_Disc__Code_
      iv. Customer__Customer_Price_Group_
      v. Customer__Fin__Charge_Terms_Code_
      vi. Customer__Payment_Terms_Code_
      vii. Customer__Salesperson_Code_
      viii. Customer__Currency_Code_
      ix. Customer__Credit_Limit__LCY__
      x. Customer__Balance__LCY__
      xi. Customer_Contact
      xii. Customer__Phone_No__
      xiii. Customer__Balance__LCY___Control42

5. Remove the old caption fields from the dataset.
   a. In the Report Dataset Designer, remove the following lines, by selecting them and clicking F4:
      i. Customer__No__Caption
      ii. Customer__Customer_Posting_Group_Caption
      iii. Customer__Customer_Disc__Group_Caption
      iv. Customer__Invoice_Disc__Code_Caption
      v. Customer__Customer_Price_Group_Caption
      vi. Customer__Fin__Charge_Terms_Code_Caption
      vii. Customer__Payment_Terms_Code_Caption
      viii. Customer__Salesperson_Code_Caption
      ix. Customer__Currency_Code_Caption
      x. Customer__Credit_Limit__LCY__Caption
      xi. Customer__Balance__LCY__Caption
      xii. Customer_ContactCaption
      xiii. Customer__Phone_No__Caption

6. In Visual Studio Report Designer, replace the old caption fields by using the new captions.
   a. Click **View, Layout** to open Visual Studio Report Designer.
   b. In Visual Studio Report Designer, in the first row of the table, right-click the text box that contains the following expression and select Expression to open the Expression window.

```
=First(Fields!Customer__No__Caption.Value)
```

   c. Replace it with the following expression.

```
=Parameters!Customer__No__Caption.Value
```

   d. Repeat step a. to c. for all text boxes in the first row of the label by using the corresponding parameter.
   e. Repeat step a. to c. for the text box in the first column that contains the following field.

```
=Fields!Customer_ContactCaption.Value
```

   f. Replace it with the following expression.

```
=Parameters!Customer_ContactCaption.Value
```

   g. Repeat step a for the text box in the first column that contains the following field.

```
=Fields!Customer__Phone_No__Caption.Value
```

   h. Replace it with the following expression.

```
=Parameters!Customer__Phone_No__Caption.Value
```

   i. Save and close Visual Studio.
   j. Click in the **Report Dataset Designer** and, when you are asked to load the changes, click **Yes**.

7. In the Report Label Designer add labels.
   a. Click **View**, and then **Labels** to open Report Label Designer.
   b. In Report Label Designer, click the first empty line.
   c. On the newly added line, set **Name** to Total__LCY_CaptionLbl, and **Caption** to Total (LCY).

   d. Close the Report Label Designer.
   e. In the Report Dataset Designer, select the last line. This line has **Name** set to Total__LCY_Caption.
   f. Press F4 to delete the line, and then click **Yes** to confirm.

8. In Visual Studio Report Designer, replace the old caption fields by using the new labels.
   a. Click **View**, and then **Layout** to open Visual Studio Report Designer.
   b. In Visual Studio Report Designer, right-click the following text box that contains the expression:

```
=First(Fields!Total__LCY_Caption.Value)
```

   c. Change it to the following expression.

```
=Parameters!Total__LCY_CaptionLbl.Value
```

Exercise 4: Save the RDLC Report Layout

Exercise Scenario

Simon must save the RDLC layout. He must also save and compile the changes in the Microsoft Dynamics NAV 2013 development environment.

Task 1: Save the RDLC Layout

High Level Steps

1. Save the report.

Detailed Steps

1. Save the report.
   a. In Visual Studio, on the **File** menu, click **Save report.rdlc**.
   b. Close Visual Studio.
   c. In the Microsoft Dynamics NAV 2013 development environment, select an empty line in the Report Dataset Designer. You receive a message that the layout of the report id changed.
   d. Click **Yes** to load the changes that you made to the RDLC report layout.
   e. On the **File** menu, click **Save**.
   f. In the **Save** dialog box, make sure **Compiled** is selected, and then click **OK**.
   g. Close the report in Object Designer.

Lab B: Customize the Upgrade

Objectives

In this lab, you will practice resolving code conflicts and comparing objects in WinDiff.

Exercise 1: Open WinDiff and Import the Objects

Exercise Scenario

In this exercise, Prakash will locate the objects and import them in WinDiff.

Task 1: Locate the Objects to Compare, Open WinDiff, and Import the Objects

High Level Steps

1. Locate the objects on the Allfiles drive of the HyperV.
2. Start WinDiff.
3. Import the Objects to compare.

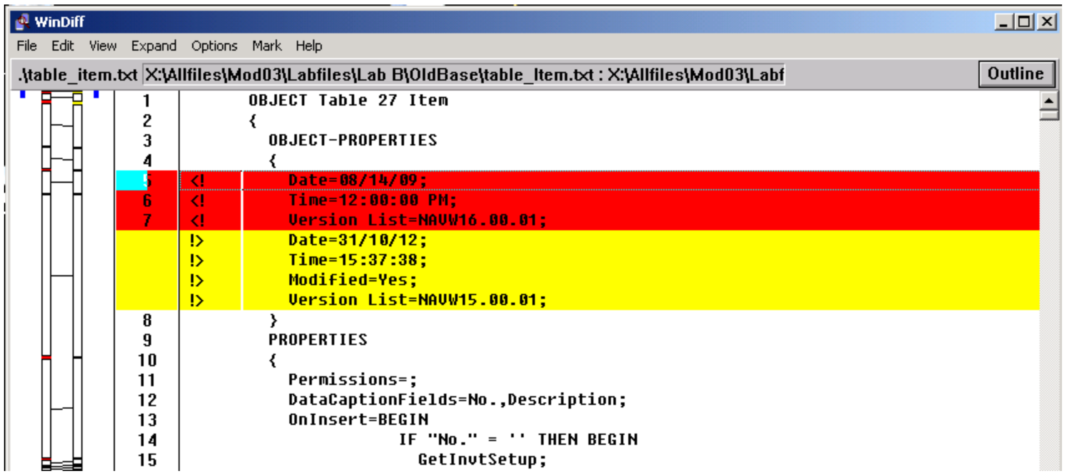
Detailed Steps

1. Locate the objects on the Allfiles drive of the HyperV.
   a. On the HyperV, locate the following files:
      i. X:Allfiles\Mod03\Labfiles\Lab B\OldCustom\table_Item.txt
      ii. X:Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt

2. Start WinDiff.
   a. Click **Start**, type "WinDiff" in the **Search programs and files** text box, and then click **Enter**. The **WinDiff** window opens.

3. Import the Objects to compare.
   a. In WinDiff, click **File,** and then **Compare Files**. The **Select First File** window opens.
   b. Browse to the folder where you find the first file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt, and then click **Open**. The file is imported and now the **Select Second File** window opens.
   c. Browse to the folder where you find the file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldCustom\table_Item.txt.., and then click **Open**.
   d. In the WinDiff application, select the first line, and then click **Expand**.

e. WinDiff now displays the two files that have the differences as shown in "Item Table Differences in WinDiff" figure.


WinDiff
File Edit View Expand Options Mark Help
.\table_item.txt X:\Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt : X:\Allfiles\Mod03\Labf
Outline

```
        OBJECT Table 27 Item
        {
          OBJECT-PROPERTIES
          {
  <!        Date=08/14/09;
  <!        Time=12:00:00 PM;
  <!        Version List=NAVW16.00.01;
  !>        Date=31/10/12;
  !>        Time=15:37:38;
  !>        Modified=Yes;
  !>        Version List=NAVW15.00.01;
          }
          PROPERTIES
          {
            Permissions=;
            DataCaptionFields=No.,Description;
            OnInsert=BEGIN
                       IF "No." = '' THEN BEGIN
                         GetInvtSetup;
```

FIGURE 3.1:ITEM TABLE DIFFERENCESIN WINDIFF

Exercise 2: Locate Code Conflicts

Exercise Scenario

In this exercise, Prakash will compare two objects to detect possible conflicts.

Task 1: Locate Code Conflicts in WinDiff

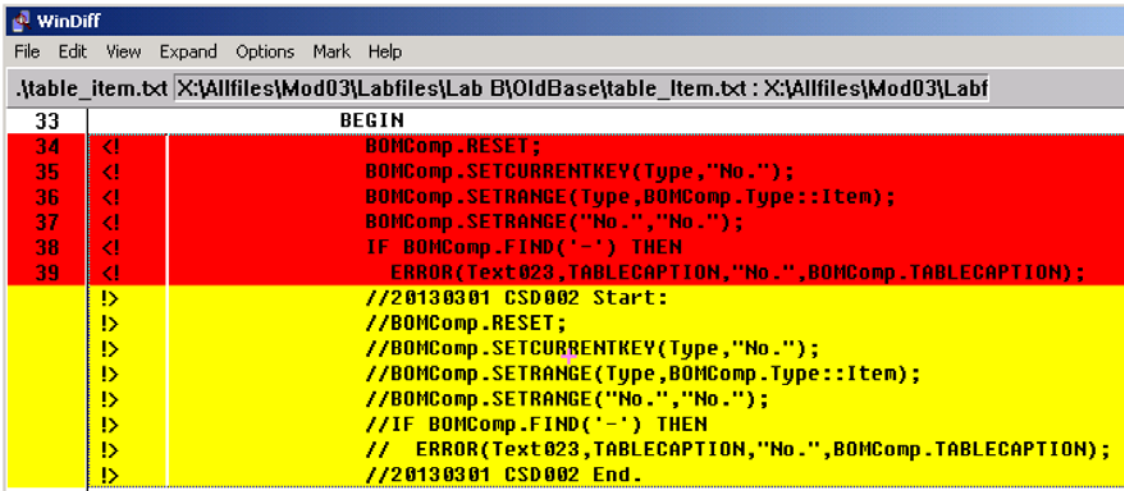
High Level Steps

1. Review and compare the objects and locate conflicts.
2. In WinDiff, locate the changed marked with //CSD002.
3. In WinDiff, find the difference that is marked with CSD001.
4. In WinDiff, insert a comment for this conflict.

Detailed Steps

1. Review and compare the objects and locate conflicts.
   a. In WinDiff, colors indicate differences between two files. In this case, the color red is used for the file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt. The color yellow is used for the file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldCustom\table_Item.txt. The lines that are different are also marked with "<!" and "!>" correspondingly.
   b. Click **View > Next Change** to move to the next change (or press F8).

2. In WinDiff, locate the changed marked with //CSD002.
   a. Click **View > Next Change** (or press F8) until you see the difference that is shown in figure "CSD002."

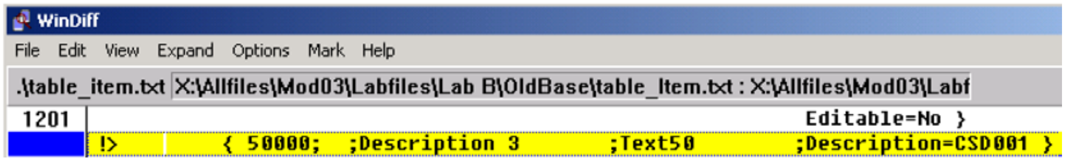

WinDiff
File Edit View Expand Options Mark Help
.\table_item.txt X:\Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt : X:\Allfiles\Mod03\Labf

```
33                             BEGIN
34  <!                           BOMComp.RESET;
35  <!                           BOMComp.SETCURRENTKEY(Type,"No.");
36  <!                           BOMComp.SETRANGE(Type,BOMComp.Type::Item);
37  <!                           BOMComp.SETRANGE("No.","No.");
38  <!                           IF BOMComp.FIND('-') THEN
39  <!                             ERROR(Text023,TABLECAPTION,"No.",BOMComp.TABLECAPTION);
    !>                           //20130301 CSD002 Start:
    !>                           //BOMComp.RESET;
    !>                           //BOMComp.SETCURRENTKEY(Type,"No.");
    !>                           //BOMComp.SETRANGE(Type,BOMComp.Type::Item);
    !>                           //BOMComp.SETRANGE("No.","No.");
    !>                           //IF BOMComp.FIND('-') THEN
    !>                           //  ERROR(Text023,TABLECAPTION,"No.",BOMComp.TABLECAPTION);
    !>                           //20130301 CSD002 End.
```

FIGURE 3.2:CSD002

As you can see, in the Old Custom version a segment of code is commented. In the Old Base version it is not. To determine whether this is a conflict and if this change must be migrated to the New Custom version, talk to the Business Consultant.

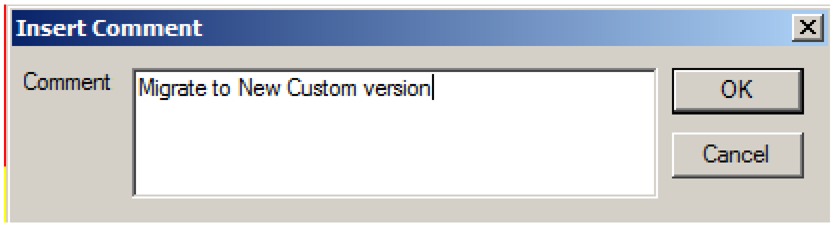
3. In WinDiff, find the difference that is marked with CSD001.
   a. Click **View Next Change** (or press F8) to move to the next conflict as shown in figure "CSD001," or you can scroll down in WinDiff until you see line 1201. This is an example of a field named **Description 3** that was added in the Current Custom version. The line is in red, and there is no corresponding line in yellow. If the field is still required in the New Custom version, you must mark it as a conflict. It will have to be migrated to the New Custom version.


WinDiff
File Edit View Expand Options Mark Help
.\table_item.txt X:\Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt : X:\Allfiles\Mod03\Labf

```
1201                                                       Editable=No }
      !>       { 50000;  ;Description 3        ;Text50       ;Description=CSD001 }
```

FIGURE 3.3:CSD001

4. In WinDiff, insert a comment for this conflict.
   a. Click **Edit > Insert Comment** to open the **Insert Comment** window as shown in the "WinDiff Insert Comment" figure.

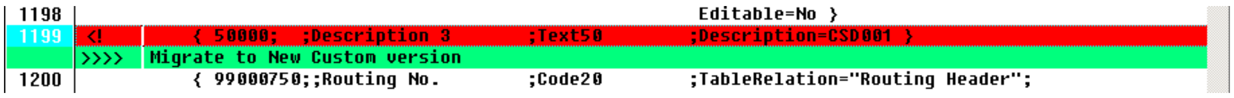


Migrate to New Custom version

FIGURE 3.4:WINDIFF INSERT COMMENT

b. In the **Comment** field, type “Migrate to New Custom version”.

c. The comment is now added in WinDiff as shown in the “WinDiff Comment Added” figure. Notice the comment in WinDiff is marked with a green color.



```
1198                                                   Editable=No }
1199 <!         { 50000;  ;Description 3       ;Text50       ;Description=CSD001 }
     >>>>  Migrate to New Custom version
1200            { 99000750;;Routing No.        ;Code20       ;TableRelation="Routing Header";
```

FIGURE 3.5:WINDIFF COMMENT ADDED

Exercise 3: Correctly Identify the Conflicting Code

Exercise Scenario

In this exercise, Prakash will enter a comment in WinDiff for every detected conflict and export the comments to a text file for analysis.

Task 1: Export the Comments from WinDiff

High Level Steps

1. Export the comments from WinDiff to a text file.

Detailed Steps

1. Export the comments from WinDiff to a text file.
   a. In WinDiff, click **File > Save Comments List**.
   b. The **Choose file name to save comments to** window opens.
   c. In the **File Name** field, type “X:Allfiles\Mod03\Labfiles\Lab B \table_Item_Comments.txt”.
   d. Click **Save** and **OK**. The comments are now exported and saved to the file that is named table_Item_Comments.txt.

Task 2: Open the comments file and verify the comments

High Level Steps

1. Open the comments file.

Detailed Steps

1. Open the comments file.
    a. Browse to the folder "X:Allfiles\Mod03\Labfiles\Lab B".
    b. Open the file that is named "table_Item_Comments.txt". The information in the comments file will resemble the following.

```
==============================

File:.\table_item_5sp1.txt : .\table_item_2013.txt, Total comments:1

==============================

Comment #1, L1199 of left file<1395>

Reviewer comment:

    Migrate to New Custom version

Source line:

        { 50000;  ;Description 3      ;Text50       ;Description=CSD001 }
```

The comments file is now available as a source of information for the developer who is responsible to customize objects in the New Custom version.

Exercise 4: Compare the OldBase and NewBase objects

Exercise Scenario

Prakash now has an indication about the difference between the OldBase and OldCustom objects. Now he must compare the OldBase and NewBase objects to determine whether the changes that were implemented by Microsoft in the new base version conflict with the customizations that were made for the customer.

Task 1: Locate Code Conflicts

High Level Steps

1. Locate the objects on the Allfiles drive of the HyperV.
2. Start WinDiff.
3. Import the Objects to compare.
4. In WinDiff, locate the next difference.

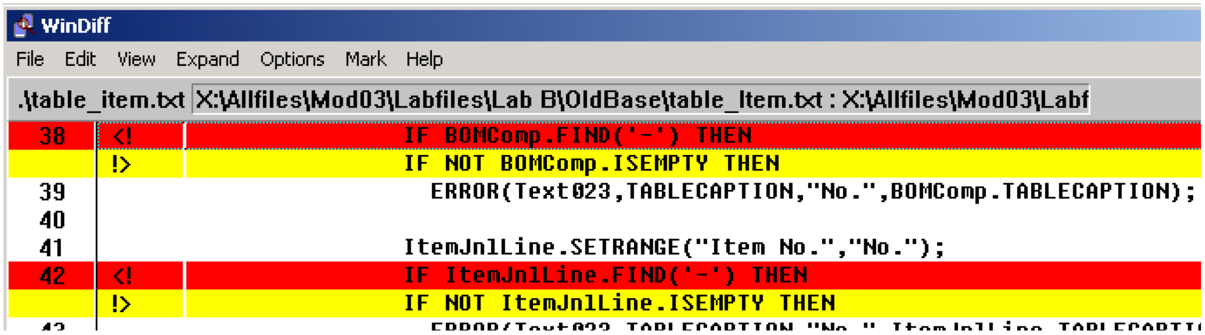
Detailed Steps

1. Locate the objects on the Allfiles drive of the HyperV.
   a. On the HyperV, locate the file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt and X:Allfiles\Mod03\Labfiles\Lab B\NewBase\table_Item.txt.

2. Start WinDiff.
   a. Click **Start**.
   b. Type "WinDiff" in the **Search programs and files** text box, and then click **Enter.** The **WinDiff** window opens.

3. Import the Objects to compare.
   a. In WinDiff, click **File > Compare Files**. The **Select First File** window opens.
   b. Browse to the folder where you find the first file that is named X:Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt., and then click **Open**. The file is imported and now the **Select Second File** window opens.
   c. Browse to the folder where you find the file that is named X:Allfiles\Mod03\Labfiles\Lab B\NewBase\table_Item.txt., and then click **Open**.
   d. In the WinDiff application, select the first line and then click **Expand**. WinDiff now displays the two files that have the differences.

4. In WinDiff, locate the next difference.
   a. Click **View > Next Change** (or press F8) to move to the conflict that is shown in the "Code Optimization" figure (line 38). The selected difference is an example of a code optimization that is implemented in the New Base version. The **FIND**('-') function is replaced with an **ISEMPTY** function. Because this is a code optimization in the New Base version, it is not a conflict. If you decide to keep the old functionality, then mark the change as a conflict.



```
.\table_item.txt  X:\Allfiles\Mod03\Labfiles\Lab B\OldBase\table_Item.txt : X:\Allfiles\Mod03\Labf
38  <!        IF BOMComp.FIND('-') THEN
    !>        IF NOT BOMComp.ISEMPTY THEN
39              ERROR(Text023,TABLECAPTION,"No.",BOMComp.TABLECAPTION);
40
41            ItemJnlLine.SETRANGE("Item No.","No.");
42  <!        IF ItemJnlLine.FIND('-') THEN
    !>        IF NOT ItemJnlLine.ISEMPTY THEN
```

FIGURE 3.6:CODE OPTIMIZATION

Summary

The Code Upgrade phase of the upgrade project is the first phase that includes actual development activities, upgrade of application modules, ISV solutions, and customizations.

You can use the following means of code optimization to guarantee optimal performance of the upgraded solution:

- Use C/AL functions that are designed specifically for Microsoft SQL Server.
- Maintain indexes correctly.

Code upgrade testing is an important part of a successful solution upgrade. It includes the following types of testing:

- Code review
- UI testing
- Functional testing.

Use the TextFormatUpgrade2013 command-line tool to replace strings in code and object properties that were appropriate for Microsoft Dynamics NAV 2009, with strings that are appropriate for Microsoft Dynamics NAV 2013.

When you upgrade reports, determine which reports to upgrade. The following reports follow the upgrade procedure:

- Reports that have both Classic Report Layout and RDLC Layout
- Classic Reports
- Processing-Only Reports

Module Review

Test Your Knowledge

Test your knowledge with the following questions.

1. UI testing determines whether the application can be started and if its interfaces are available.

   ( ) True

   ( ) False

2. What is the TextFormatUpgrade2013 tool?

   ( ) The TextFormatUpgrade2013 replace strings that were appropriate in Microsoft Dynamics NAV 2009 with their substitutes in Microsoft Dynamics NAV 2013.

   ( ) The TextFormatUpgrade2013 tool upgrade reports to RDLC 2008 format, and remove references from RDLC 2005.

   ( ) The TextFormatUpgrade2013 tool transforms code to support 3-tier architecture.

   ( ) The TextFormatUpgrade2013 tool is not used during code upgrade.

3. Functional testing is performed after code review, UI testing is completed, and all the reported bugs and defects are fixed, verified, and closed. Functional testing at this point of the upgrade is intended to verify that upgraded code is functioning as expected against basic processes.

   ( ) True

   ( ) False

4. Before you can upgrade a report it should first be compiled.

   ( ) True

   ( ) False

5. In Microsoft Dynamics NAV 2013, a report object can only contain an RDLC layout and request page.

   ( ) True

   ( ) False

Test Your Knowledge Solutions

Module Review and Takeaways

1. UI testing determines whether the application can be started and if its interfaces are available.

    (√) True

    ( ) False

2. What is the TextFormatUpgrade2013 tool?

    (√) The TextFormatUpgrade2013 replace strings that were appropriate in Microsoft Dynamics NAV 2009 with their substitutes in Microsoft Dynamics NAV 2013.

    ( ) The TextFormatUpgrade2013 tool upgrade reports to RDLC 2008 format, and remove references from RDLC 2005.

    ( ) The TextFormatUpgrade2013 tool transforms code to support 3-tier architecture.

    ( ) The TextFormatUpgrade2013 tool is not used during code upgrade.

3. Functional testing is performed after code review, UI testing is completed, and all the reported bugs and defects are fixed, verified, and closed. Functional testing at this point of the upgrade is intended to verify that upgraded code is functioning as expected against basic processes.

    (√) True

    ( ) False

4. Before you can upgrade a report it should first be compiled.

    (√) True

    ( ) False

5. In Microsoft Dynamics NAV 2013, a report object can only contain an RDLC layout and request page.

    (√) True

    ( ) False