

MODULE 5: FEATURE INTEGRATION

Module Overview

At this stage the Seminar Management application area is a combination of individual functions that CRONUS International Ltd. can use to input seminar master data, perform registrations, and post completed seminar registrations. The next step is to integrate these features with one another and with the standard application. This makes it easier for users to move through the application.

This module addresses the integration of solution functionality with the user interface (UI) of the application.

Objectives

- Integrate previously created Seminar Management features with one another.
- Explain the architecture of the Navigate feature.
- Enable easier searches by adding Navigate functionality to Seminar Management pages.
- Enable looking up Seminar Management information from standard application areas.

Prerequisite Knowledge

Making changes to standard or existing functionality frequently involves making structural changes to tables that already contain data. These structural changes may include any of the following:

- Adding new fields
- Deleting existing fields
- Changing the data type or length of existing fields
- Changing other table or field properties, such as TableRelation or DataPerCompany

Client data is valuable. Making any structural changes to that data requires planning and accuracy. Microsoft Dynamics NAV 2013 safeguards the data by only allowing specific types of changes to tables that contain data. This prevents any accidental data loss, corruption, or other kinds of problems that can arise from modifications.

By understanding the kind of changes that you can make to existing tables, you can plan implementation and development activities.

Changing Tables that Contain Data

When you developed the Seminar Management functionality, you created several tables, and changed several existing ones. Microsoft Dynamics NAV Development Environment makes sure that no data is lost when you change the structure of a table. This means that when you change a table that contains data, there are important guidelines that specify the changes that are allowed under certain conditions.

Changes to Fields

You can always make the following changes to table fields:

- Change the name.
- Change any properties that only control how data is displayed or formatted.
- Change the TableRelation, ValidateTableRelation, and TestTableRelation properties.
- Change a FlowField back to a regular field.
- Change the CalcFormula on a FlowField.
- Increase the length of a text or a code field.
- Add a new field.



Note: When you increase the length of a Text or a Code field, or add a new field, the only limitation is the maximum record size that is imposed by Microsoft SQL Server. This is 8,060 bytes per record.

You can make the following changes if the field does not contain data in any of the records for any companies in the database:

- Change the Data Type.
 - Change the Field No.
 - Change a normal field into a FlowField.
 - Disable a field by setting Enabled to **No**.
 - Delete the field.
-



Note: The system lets you delete a field even if there are remaining references to the field, such as from CalcFormula of a FlowField, TableRelation, page control, report column, a code reference, or any other type of reference. After you delete a field, you manually must remove all references to the field from other objects. When you run an object that has an invalid field reference, a run-time error occurs.



Best Practice: Changing the Field No. has the same effect on all field references as deleting the field completely. You should change the Field No. of an existing field only if it is necessary.

In addition to these rules, you can also decrease the length of a Text or a Code field as long as the length of all values in that field in all companies in the database is equal to or smaller than the target size.

If you try to make a prohibited change, Microsoft Dynamics NAV Development Environment warns you, and then prevents the change.

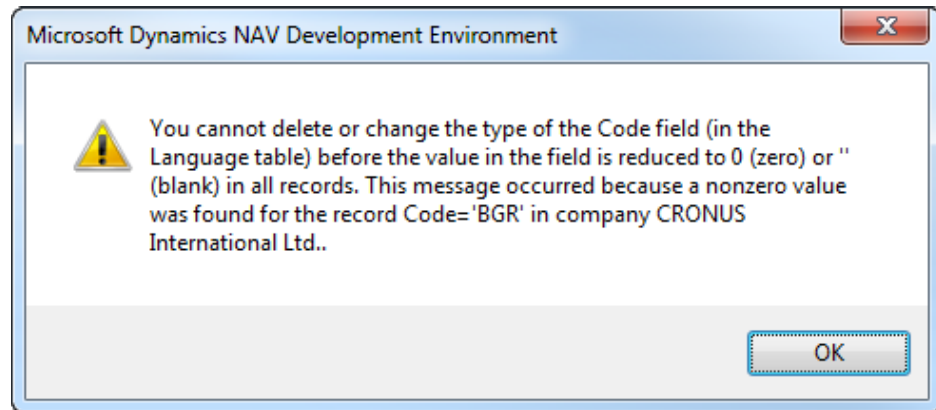


FIGURE 5.1: INVALID FIELD CHANGE WARNING WINDOW

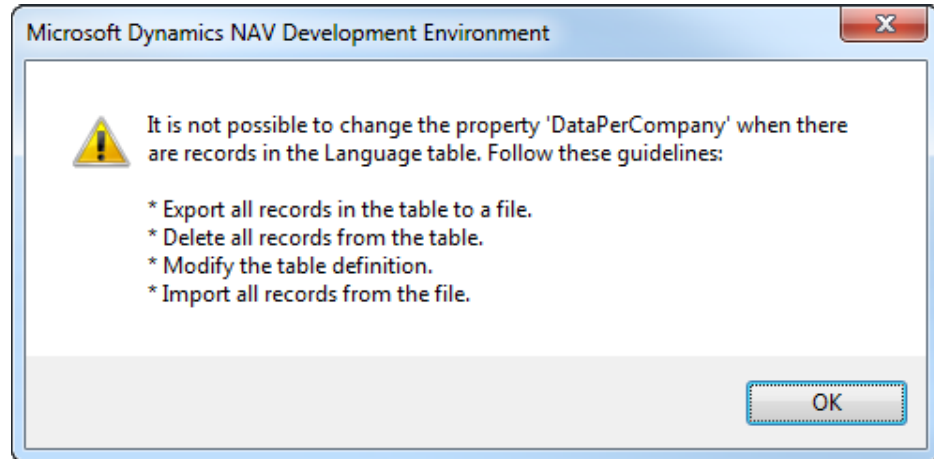
Changes to Tables

Even though changes to fields are the most common structural changes that you make to tables, there are certain changes that affect tables directly. Typically you can make most changes, including changing the Name, the ID, and most of the other properties.

The DataPerCompany table property is the only property that has specific rules for making changes. The following guidelines explain the conditions under which you can change this property:

- If there is only one company in the database, you can always change DataPerCompany to either value.
- If there are more companies in the database, you can change DataPerCompany to **Yes**. You can do this only if there is no data in the table in any of the companies, or if there is data in the table in only one of the companies. The table is empty in all other companies.
- If there are more companies in the database, and there is data in the table of any company, you cannot change this property to **No**.

If you try to change the DataPerCompany property in any of these prohibited situations, Microsoft Dynamics NAV Development Environment warns you and then prevents the change.



Seminar Feature Integration

You are now ready to integrate the Seminar Management features with the standard application and to one another. You have already integrated some features earlier, such as linking lists to card pages, or enabling users to call posting routines from document pages. Except for these simple and most common feature integration steps, you must provide deeper integration to maintain a consistent user experience across the application.

The following table summarizes the typical features that integrate different application functionalities.

Category	Features
Trail Record	<ul style="list-style-type: none"> Enables the Navigate feature to search for new ledger entry and posted document types.
Master Pages	<ul style="list-style-type: none"> Create new documents from master pages. Access open documents from master pages. Access ledger entries from master pages.
Transactions	<ul style="list-style-type: none"> Access Navigate from posted document pages. Access Navigate from ledger entries.

Integrating these features enables users to be more productive for the following reasons:

- It reduces the number of clicks for users to access related features that are available in all relevant pages.
- It reduces time that you spend on data entry and filtering by defaulting master data and document number fields.
- It reduces errors that are created by typing incorrect information by defaulting field values.

Solution Design

CRONUS International Ltd. provided no specific functional requirements about the feature integration. Most of the work that relates to feature integration belongs to the domain of Microsoft Dynamics NAV 2013 standards and best practices.

However, the following two nonfunctional requirements address customer productivity and ease-of-use issues:

- The solution must be consistent, user-friendly, and easy to learn and to use.
- Any custom-built functionality must follow the standards, principles, and best practices of Microsoft Dynamics NAV 2013, and must seamlessly integrate into the standard application. The solution must enable users to be productive and spend as little time as possible searching and filtering.

You can address these requirements by integrating features. Microsoft Dynamics NAV 2013 standard functionality provides many examples of integrating features when you develop your custom application functionality.

The three most common user tasks in Microsoft Dynamics NAV 2013 are as follows:

- Creating new transactions
- Maintaining and processing existing transactions
- Analyzing transaction history

The most common user task is entering transactional information. This includes documents and journals. Because documents and journals are always related to a master record, the majority of master pages in Microsoft Dynamics NAV 2013 provide a quick way to enter a document or a journal for a master record. For example, you can create a new quote or an invoice for a customer directly from the **Customer Card** or **Customer List**, or you can access the **Item Journal** directly from the **Item Card** or **Item List**.

In addition to creating new documents for a master record, you must quickly access any existing documents that are related to that master record. For example, from **Customer Card** or **Customer List**, you can quickly access any quotes, invoices, or other document types that are related to the selected customer. This provides an intuitive user experience, and makes it easy for users to quickly access related information.

Finally, users typically access transaction history for a master record. Therefore, users can click **Ledger Entries** on each master record card. This provides a standard and consistent way to access the ledger entries for the master record. **Ledger Entries** are a mandatory action on every master page. You can also press CTRL+F7. This is the system-wide shortcut for accessing related ledger entries from any master page. For example, to access **Customer Ledger Entries** from **Customer Card**, you can press CTRL+F7, or click **Ledger Entries**.

As a general principle, any information that is related to a record must be available from a page that displays that record. Therefore, to stay consistent with Microsoft Dynamics NAV 2013 standards, you must provide similar functionality for Seminar Management: table **Seminar** is the master record, **Seminar Registration** is the document, and Seminar Ledger Entries are ledger entries. Therefore, you must enable the following functionality from the **Seminar Card** and **Seminar List** pages:

- Creation of new documents
- Access to existing Seminar Registration documents
- Access to Seminar Ledger

Development

To integrate Seminar Management, you must change several existing pages and one table to be consistent with standard Microsoft Dynamics NAV 2013 functionality.

Creating New Documents from Master Pages

The standard way to create a new document from a master page is to click the appropriate action in the New group on the **Home** tab, or in the New Document group in the **Actions** tab. This action always runs the document page in the Create mode. It then sets the record link between the master record and the related field in the document table.

When a user inserts a new record, the code in the OnInsert trigger of the document header table checks whether there is a filter on the master record field that filters to a single master record value. If the code in the OnInsert trigger finds such a filter, it validates the master record field to the value in the filter. This makes sure that a new document is assigned automatically to the master record that the master page passed as the record link to the document page.

For example, when a user clicks **Sales Invoice** in the New group on the **Home** tab in the **Customer Card** page, the action runs the **Sales Invoice** page in Create mode. This sets the record link on the **Sell-to Document No.** field of the **Sales Header** table to the value of the **No.** field for the selected customer. Users finish creating the **Sales Invoice** page by either leaving the **No.** field, or selecting a number series by clicking the **AssistEdit** button. The code in the OnInsert trigger of the **Sales Header** table then checks whether there is a filter on the **Sell-to Customer No.** field for a single customer **No.** If there is such a filter, the code validates the **Sell-to Customer No.** field to the value in the filter.

The following code example from the OnInsert trigger in the **Sales Header** table is responsible for applying the record link filter from the **Customer Card** page to every new sales document.

Applying the Record Link Filter in the Sales Header Table

```
IF GETFILTER("Sell-to Customer No.") <> " THEN

  IF GETRANGEMIN("Sell-to Customer No.") = GETRANGEMAX("Sell-to Customer
  No.") THEN

    VALIDATE("Sell-to Customer No.",GETRANGEMIN("Sell-to Customer No."));
```

Tables

To enable creation of new Seminar Registration documents from the **Seminar Card** or **Seminar List** pages, add code to the OnInsert trigger of the **Seminar Registration Header** table. This code performs the following logic:

- Checks whether there is a filter on the **Seminar No.** field.
- If there is a filter on the **Seminar No.** field, the code checks whether the filter is to a single Seminar No. value.
- If the filter is to a single value, the code validates the **Seminar No.** field to the value in the filter on the **Seminar No.** field.

Pages

You must change the **Seminar Card** and **Seminar List** pages by adding the following action structure:

- Related Information (container)
 - Seminar (group)
 - Seminar Ledger Entries (action)
 - Registrations (group)
 - Registrations (action)

- New Document Items (container)
 - Seminar Registration (action)

The “Seminar Card Page (123456700) with the Home tab Selected” figure shows the **Seminar Card** page after customization, with the **Seminar Registration** action in the New group on the **Home** tab.

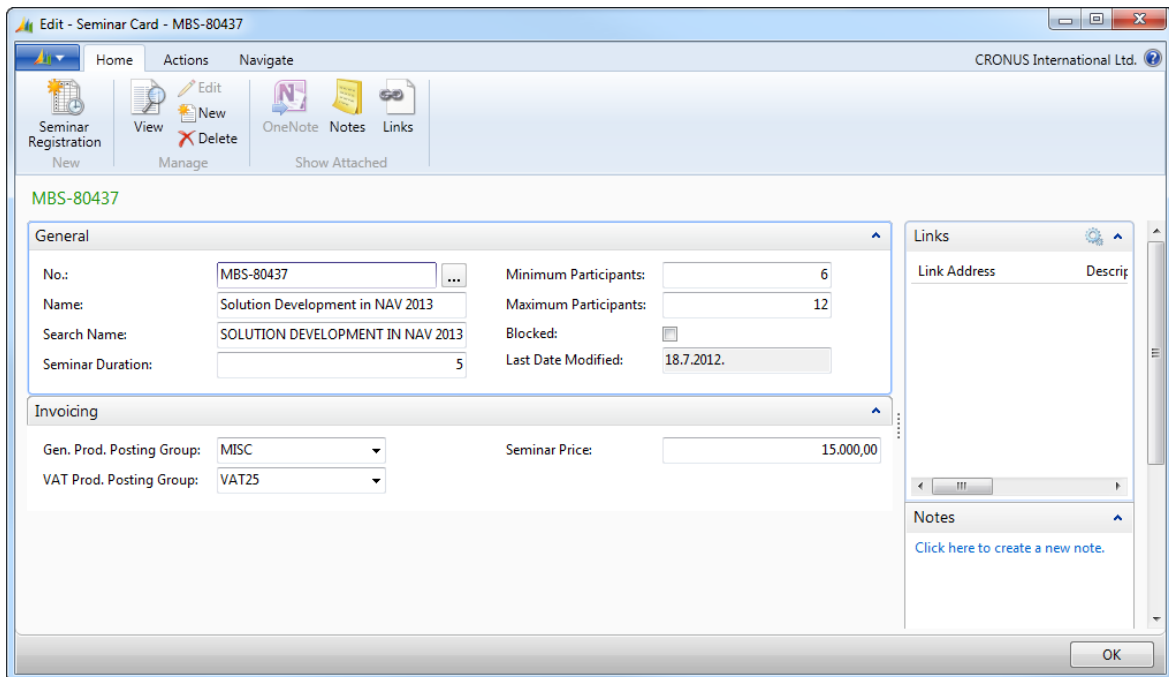


FIGURE 5.3: THE SEMINAR CARD PAGE (123456700) WITH THE HOME TAB SELECTED

The “Seminar List Page (1234565701) with the Navigate tab Selected” figure shows the **Seminar List** page after customization, with the **Ledger Entries** action in the **Seminar** group, and the **Registrations** action in the **Registrations** group on the **Navigate** tab.

C/SIDE Solution Development in Microsoft Dynamics® NAV 2013

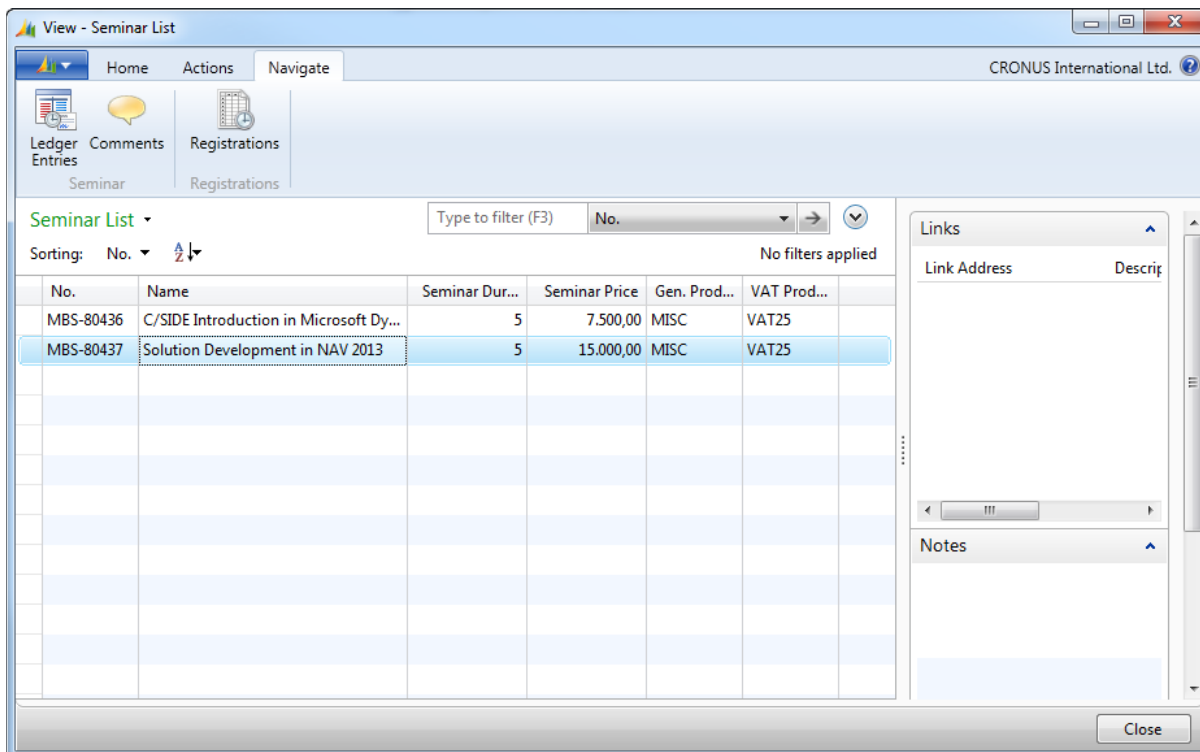


FIGURE 5.4: THE SEMINAR LIST PAGE (123456701) WITH THE NAVIGATE TAB SELECTED.

Lab 5.1: Integrating Seminar Features

Scenario

Isaac is a developer working on the implementation of Microsoft Dynamics NAV 2013 for CRONUS International Ltd. He is responsible for developing page customizations for Seminar Management integration.

He first must integrate Seminar Management features to enable standard navigation between master data, documents, and posted information.

Exercise 1: Customize Seminar Registration Master Pages

Exercise Scenario

Isaac must add actions to the **Seminar Registration Card** and **Seminar Registration List** pages to do the following:

- Enable creation of new Seminar Registrations.
- Enable access to existing Seminar Registrations.
- Enable access to **Seminar Ledger Entries** for the seminar that is shown in the card or selected in the list.

To create new seminar registrations for a specific seminar, Isaac also must customize the **Seminar** table. If there is a filter on the **Seminar No.** field, and the filter applies to a single **Seminar No.**, Isaac must make sure that the **Seminar No.** field validates to the value in the filter.

Task 1: Customize the Seminar Registration Header Table

High Level Steps

1. Create code in the **Seminar Registration** table to apply the record link filter to the **Seminar No.** field.

Detailed Steps

1. Create code in the **Seminar Registration** table to apply the record link filter to the **Seminar No.** field.
 - a. Design the table 123456710, **Seminar Registration Header**.
 - b. In the **OnInsert** trigger, append the following code:

Appended code in the OnInsert Trigger

```
IF GETFILTER("Seminar No.") <> " THEN  
  
IF GETRANGEMIN("Seminar No.") = GETRANGEMAX("Seminar No.") THEN  
  
VALIDATE("Seminar No.",GETRANGEMIN("Seminar No."));
```

- c. Compile, save, and then close the table.

Task 2: Customize the Seminar Card Page

High Level Steps

1. Add an action to create a new **Seminar Registration** from the **Seminar Card** page.
2. Add a new action as the first action in the **Seminar** group, to run the **Seminar Ledger Entries** page for the current **Seminar** record.
3. Add a new group and an action to run the **Seminar Registration List** page for the current **Seminar** record.

Detailed Steps

1. Add an action to create a new **Seminar Registration** from the **Seminar Card** page.
 - a. Design the page 123456700, **Seminar Card**.
 - b. Open the **Action Designer** for **Page Actions**.
 - c. To the list of actions, append an ActionContainer of the NewDocumentItems **SubType**. Make sure that it has no indentation, by clicking the **Left** button as many times as needed.
 - d. Under the NewDocumentItems ActionContainer, add a new action, and set the **Caption** property to "Seminar Registration".
 - e. Set the following properties on the action.

Property	Value
RunPageMode	Create
Image	NewTimesheet
Promoted	Yes
PromotedCategory	New
PromotedIsBig	Yes
RunObject	Page Seminar Registration
RunPageLink	Seminar No.=FIELD(No.)

2. Add a new action as the first action in the **Seminar** group, to run the **Seminar Ledger Entries** page for the current **Seminar** record.
 - a. Under the **Seminar** ActionGroup, and above the Comments action, insert a new action, and set the Caption property to "Ledger Entries".

- b. Set the following properties on the action.

Property	Value
Image	WarrantyLedger
PromotedCategory	Process
PromotedIsBig	Yes
ShortCutKey	Ctrl+F7
RunObject	Page Seminar Ledger Entries
RunPageLink	Seminar No.=FIELD(No.)

3. Add a new group and an action to run the **Seminar Registration List** page for the current **Seminar** record.
- a. Under the RelatedInformation ActionContainer, and above the NewDocumentItems ActionContainer, add a new ActionGroup, and set its caption to "&Registrations". Make sure that it is the last group in the RelatedInformation ActionContainer, and that it is indented one level under the ActionContainer.
 - b. To the **Registrations** group, add a new action, and set the **Caption** to "&Registrations". Indent it one level under the **Registrations** group.
 - c. On the **Registrations** action, set the following properties.

Property	Value
Image	Timesheet
PromotedCategory	Process
RunObject	Page Seminar Registration List
RunPageLink	Seminar No.=FIELD(No.)

- d. Compile, save, and then close the page.

Task 3: Customize the Seminar List Page

High Level Steps

1. Make the same changes to the actions on the **Seminar List** page that you made to the **Seminar Card** page.

Detailed Steps

1. Make the same changes to the actions on the **Seminar List** page that you made to the **Seminar Card** page.
 - a. Design page 123456700, **Seminar Card**.
 - b. Open the **Action Designer** for **Page Actions**.
 - c. Select all actions (press CTRL+A), and then copy them to the clipboard (press CTRL+C).

- d. Close **Action Designer** and **Page Designer** for the **Seminar Card** page.
- e. Design page 123456701, **Seminar List**.
- f. Open the **Action Designer** for **Page Actions**.
- g. Select and delete any existing actions (press CTRL+A, then press F4, and then confirm the deletion).
- h. Paste the actions from the clipboard (press CTRL+V).
- i. Compile, save, and then close the page.



Note: You can repeat the same steps you did for the **Seminar Card** page.

Navigate Integration

The Navigate feature lets users view a summary of the number and type of entries with the same document number or posting date. This feature is very useful for finding the ledger entries or other types of posted information that result from certain transactions. The Navigate feature is one of the central traceability features in Microsoft Dynamics NAV 2013, and one of its most versatile features.

When users post a transaction, they rely on the Navigate feature when they must trace the results of the transaction. Users access the Navigate feature from any page that displays any type of posted entries or documents. The Navigate feature displays every resulting entry.

This makes it important to fully integrate any custom functionality that includes a posting process with the Navigate feature. Integrating the Navigate feature improves the traceability of your own transactions and the resulting posted documents and ledger entries.

Navigate Feature Architecture

The architecture of this feature is simple, even though the Navigate feature performs the complex task of searching and then displaying database records to the user in the appropriate page. When you search, the Navigate feature takes advantage of simple filtering mechanisms. When it shows pages, it uses the default lookup forms.

The Navigate feature is completely contained within the page 344, **Navigate**.

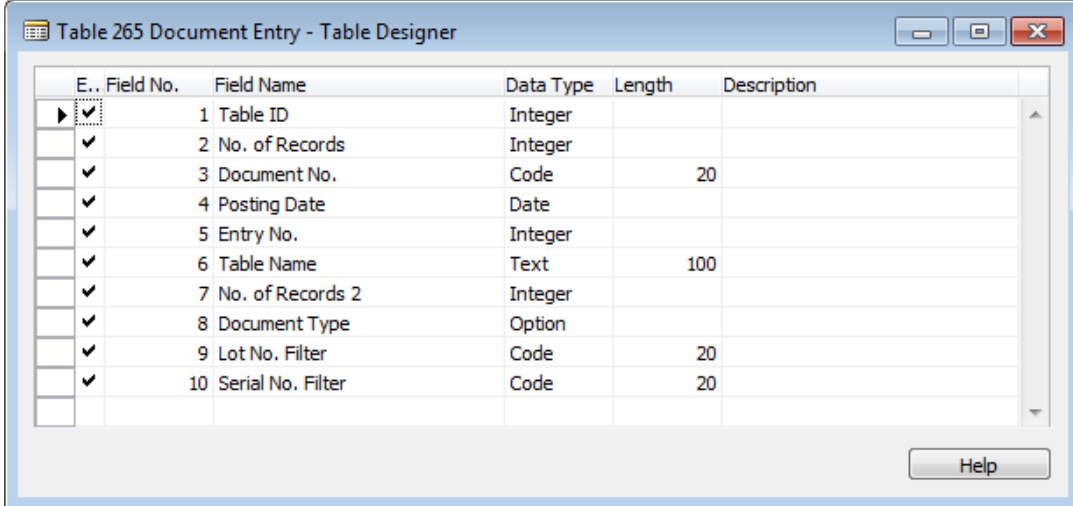
The "Navigate Page (344)" figure shows the **Navigate** page.

The screenshot shows the 'Edit - Navigate' window. The ribbon includes 'Home' and 'Actions' tabs. The 'Home' tab has 'Find', 'Show', and 'Print...' buttons. The 'Process' button is also visible. The 'General' section has 'Document No.' and 'Posting Date' fields. The 'Source' section has 'Document Type', 'Source Type', 'Source No.', and 'Source Name' fields. The 'Document Entry' section has a table with columns 'Table Name' and 'No. of Re...'. The bottom of the window has 'General', 'External', and 'Item Tra...' tabs, and a 'Close' button.

FIGURE 5.5: THE NAVIGATE PAGE (344)

The page uses table **265, Document Entry** as its source, and sets the property to **Yes**, to only work with temporary in-memory data.

The "The Document Entry Table (265)" figure shows the fields in the **Document Entry** table.



E..	Field No.	Field Name	Data Type	Length	Description
▶	1	Table ID	Integer		
✓	2	No. of Records	Integer		
✓	3	Document No.	Code	20	
✓	4	Posting Date	Date		
✓	5	Entry No.	Integer		
✓	6	Table Name	Text	100	
✓	7	No. of Records 2	Integer		
✓	8	Document Type	Option		
✓	9	Lot No. Filter	Code	20	
✓	10	Serial No. Filter	Code	20	

FIGURE 5.6: THE DOCUMENT ENTRY TABLE (265)

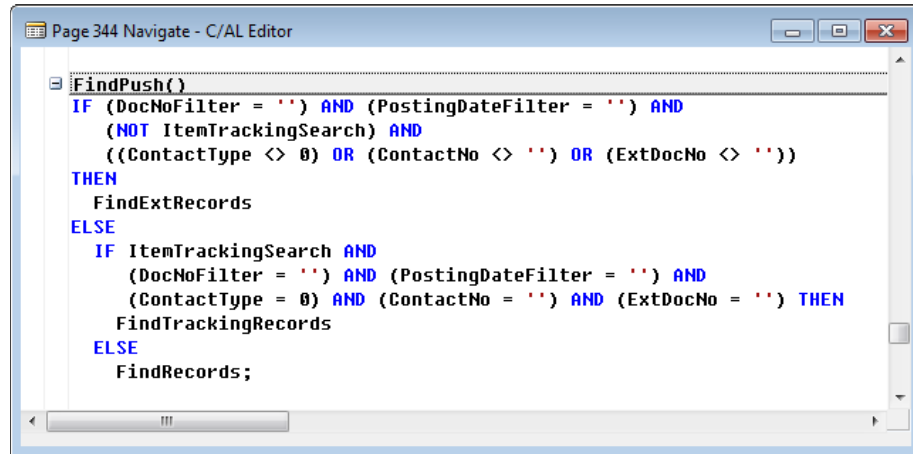
The following functions control the majority of the work in the **Navigate** feature:

- FindPush
- FindRecords and FindExtRecords
- InsertIntoDocEntry
- ShowRecords

FindPush

The most common way to use the **Navigate** page is to specify the **Document No.**, **Posting Date** or both, and then click **Find**. The field controls for the **Document No.** and Posting Date filters are bound to the DocNoFilter and PostingDateFilter variables. When the user clicks **Find**, it calls the **FindPush** function.

The following illustration shows the C/AL code in the FindPush function trigger.



```

Page 344 Navigate - C/AL Editor
FindPush()
IF (DocNoFilter = '') AND (PostingDateFilter = '') AND
  (NOT ItemTrackingSearch) AND
  ((ContactType <> 0) OR (ContactNo <> '') OR (ExtDocNo <> ''))
THEN
  FindExtRecords
ELSE
  IF ItemTrackingSearch AND
    (DocNoFilter = '') AND (PostingDateFilter = '') AND
    (ContactType = 0) AND (ContactNo = '') AND (ExtDocNo = '') THEN
    FindTrackingRecords
  ELSE
    FindRecords;
  
```

FIGURE 5.7: THE FINDPUSH FUNCTION TRIGGER

The **FindPush** function calls either the **FindExtRecords** or **FindRecords** function, depending on which filter fields the user populated with values.

Note: The **FindExtRecords** function looks for transactions that are based on the **External Document No.** field. This is the document number that is assigned by a third-party, such as a customer or a vendor. The **FindRecords** function looks for transactions that are based on the **Document No.** This is the document number that is assigned by Microsoft Dynamics NAV 2013, or by a Microsoft Dynamics NAV 2013 user. There is no basic difference in the C/AL code structure of either of these functions.

FindRecords

The **FindRecords** function follows this simple algorithm:

1. Empties the **Document Entry** temporary source table.
2. Repeats the following C/AL code pattern for each type of table.

The FindRecords Table Search Pattern

```

IF VendLedgEntry.READPERMISSION THEN BEGIN

  VendLedgEntry.RESET;

  VendLedgEntry.SETCURRENTKEY("Document No.");

  VendLedgEntry.SETFILTER("Document No.",DocNoFilter);

  VendLedgEntry.SETFILTER("Posting Date",PostingDateFilter);

  InsertIntoDocEntry(
  
```

```
DATABASE::"Vendor Ledger
Entry",0,VendLedgEntry.TABLECAPTION,VendLedgEntry.COUNT);
END;
```

- a. The READPERMISSION line checks whether the user has sufficient permissions to read from the table that is being searched. If this is the case, the search continues. Otherwise, it skips to the next table with the same pattern.
 - b. Resets the appropriate transaction (posted document or ledger entry) table.
 - c. Sets the appropriate table key to make the search easier.
 - d. Sets the filter to the **Document No.** and **Posting Date** fields to the values of the DocNoFilter and PostingDate variables. These variables are sources for the **Document No.** and **Posting Date** field controls in the page. The user enters the value into those variables directly.
 - e. Calls the **InsertIntoDocEntry** function by passing the table ID, document type (if relevant), the table caption, and the number of records in the filter.
3. Now all the necessary tables are searched. If any relevant posted document or ledger entry was found, the Rec variable is not empty. The following line of code checks for this condition.

```
DocExists := FINDFIRST;
```

4. If any records are found, then the source information is retrieved and shown in the page through several variables. Otherwise, the system informs the user that no records could be found.
5. Finally, the system updates the page.

Now the page shows the records it has found by using this algorithm.

InsertIntoDocEntry

The **InsertIntoDocEntry** function's task is to store information about any found records into the **Document Entry** temporary table.

The InsertIntoDocEntry function trigger contains the following C/AL code.

InsertIntoDocEntry Function

```
IF DocNoOfRecords = 0 THEN

    EXIT;

INIT;

"Entry No." := "Entry No." + 1;

"Table ID" := DocTableID;

"Document Type" := DocType;

"Table Name" := COPYSTR(DocTableName,1,MAXSTRLEN("Table Name"));

"No. of Records" := DocNoOfRecords;

INSERT;
```

The system first checks if there are any records. If there are none, the system exits. The system then performs the following tasks:

1. Initializes a new record.
2. Assigns the information that is passed as parameters into the fields of the **Document Entry** table by using the implicit Rec variable.
3. Inserts the new record.

ShowRecords

The power of the **Navigate** feature is not only its capability to search for records in the database, but its ability to show the relevant page for each record type that it finds. The **ShowRecords** function controls that part of the functionality.

Users can call the **ShowRecords** function by either clicking **Show**, or drilling down any of the rows that represent the found record types.

The **ShowRecords** function is large, however, it follows the same simple pattern to display records. This pattern consists of one large CASE block. Based on the table ID of the selected row, the default lookup page runs over the same table that was filtered earlier in the **FindRecords** or **FindExtRecords** function. The user can quickly access the details of any posted transaction by knowing only its posting date or document number.

Calling Navigate from Other Pages

When using **Navigate**, the users don't have to memorize the document numbers for the transactions. Instead, they can call **Navigate** from any posted document or ledger entry pages, and then **Navigate** automatically filters by the **Document No.** and the **Posting Date** of the transaction.

To enable this functionality, the **Navigate** page includes the **SetDoc** function. Other pages can call this function to set the filters before running the **Navigate** page. Then, when the **Navigate** page runs, it first checks if there are any filters already set by other objects. If there are, it immediately finds records depending on the type of filters that are set by other objects.

Solution Design

Users can access the **Navigate** feature from all ledger entry pages, posted documents, and from their Role Center. **Navigate** finds all posted entries and documents in Microsoft Dynamics NAV 2013 that have the same **Document No.**, **Posting Date**, or both, as specified by the user. Extend the **Navigate** feature to also look for records in the **Seminar Ledger Entry** and **Posted Seminar Reg. Header** tables.

Seminar managers can use the **Navigate** feature to view a complete summary of the ledger entries that are related to a Posted Seminar Registration or a Seminar Ledger Entry. Therefore, you must add an action to access the **Navigate** feature to all Seminar Management posted information pages. These pages are as follows:

- Posted Seminar Registration
- Posted Seminar Reg. List
- Seminar Ledger Entries

Development

Your primary task is to enable the **Navigate** feature to search for posted seminar registration documents and seminar ledger entries. This requires changes to the **Navigate** page. This page is responsible for searching through the tables and displaying the search results. The changes include appending the code to the functions that are responsible for searching for records, and displaying appropriate pages for each record type that the **Navigate** feature can find.

Tables

To maximize the table search performance, you must also add a secondary key to the **Seminar Ledger Entry** table and **Seminar Ledger Entry** table. The **Navigate** page searches for records by the **Document No.** field and the Posting Date field. Therefore, you must always add a key that includes these two fields to any ledger entry table that the Navigate feature uses.

Pages

You must add an action to run the **Navigate** page from the **Seminar Ledger Entries** page and the **Posted Seminar Registration** page. Change the **Seminar Ledger Entries** and **Posted Seminar Registration** pages by adding the **Navigate** action to the Actions tab.

Lab 5.2: Changing Objects to Integrate with Navigate

Scenario

After integrating Seminar Management features with one another, Isaac is now ready to integrate these features with standard application functionality. The most important standard feature that he must integrate is Navigate. He must enable Navigate to search for Seminar Management transaction records, and make Navigate available in Seminar Management pages.

Exercise 1: Customize Tables

Exercise Scenario

To enable **Navigate** to search for Seminar Management transactions in the most efficient way, Isaac must add a new key to the **Seminar Ledger Entry** table.

Task 1: Modify the Seminar Ledger Entry Table

High Level Steps

1. Add the key with **Document No.** and **Posting Date** fields to the **Seminar Ledger Entry** table.

Detailed Steps

1. Add the key with **Document No.** and **Posting Date** fields to the **Seminar Ledger Entry** table.
 - a. Design table 123456732, **Seminar Ledger Entry**.
 - b. Open the **Keys** window for the table.
 - c. In the first empty line enter "Document No.,Posting Date"
 - d. Close the **Keys** window.
 - e. Compile, save, and then close the table.

Exercise 2: Customize the Navigate Page

Exercise Scenario

Isaac changes the **Navigate** page so that it can search for **Posted Seminar Reg. Header** and **Seminar Ledger Entry** records.

Task 1: Add Global Variables

High Level Steps

1. In the **Navigate** page, add global variables for the **Posted Seminar Reg. Header** and **Seminar Ledger Entry** tables.

Detailed Steps

1. In the **Navigate** page, add global variables for the **Posted Seminar Reg. Header** and **Seminar Ledger Entry** tables.
 - a. Design page 344, **Navigate**.
 - b. Open the **C/AL Globals** window.
 - c. Add the following variables to the end of the **Variables** list.

Name	Data Type	Subtype
PostedSeminarRegHeader	Record	Posted Seminar Reg. Header
SeminarLedgEntry	Record	Seminar Ledger Entry



Best Practice: When customizing standard Microsoft Dynamics NAV 2013 objects, it is a best practice to always visibly separate your variables from existing variables. You do this by adding a separator variable, with a discernible name, such as ---CSD1.00 Variables---. Add your variables after this separator. Because there is no other way to clearly mark your custom variables, this is the only way to make your variables stand out. This simplifies object maintenance.

Task 2: Customize the FindRecords Function

High Level Steps

1. Add code to the **FindRecords** function to look for **Posted Seminar Reg. Header** and records similar to the way that the function searches for other entry types. The function must first check whether the user has permission to read the **Posted Seminar Reg. Header** table. The function then set filters for the **No.** field and the **Posting Date** field to the *DocNoFilter* variable values and the *PostingDateFilter* variable value. The **FindRecords** function finally calls the **InsertIntoDocEntry** function to store the information about any found records.
2. Repeat the same search algorithm for the **Seminar Ledger Entry** table. Make sure that you utilize the appropriate table key.

Detailed Steps

1. Add code to the **FindRecords** function to look for **Posted Seminar Reg. Header** and records similar to the way that the function searches for other entry types. The function must first check whether the user has permission to read the **Posted Seminar Reg. Header** table. The function then set filters for the **No.** field and the **Posting Date** field to the *DocNoFilter* variable values and the *PostingDateFilter* variable value. The **FindRecords** function finally calls the **InsertIntoDocEntry** function to store the information about any found records.
 - a. Show the **C/AL Editor** window for the **Navigate** page.
 - b. Locate the **FindRecords** function trigger.
 - c. In the **FindRecords** function trigger, before the DocExists := FINDFIRST line, enter the following code.

```
//CSD1.00>  
  
IF PostedSeminarRegHeader.READPERMISSION THEN BEGIN  
  
    PostedSeminarRegHeader.RESET;  
  
    PostedSeminarRegHeader.SETFILTER("No.",DocNoFilter);  
  
    PostedSeminarRegHeader.SETFILTER("Posting Date",PostingDateFilter);  
  
    InsertIntoDocEntry(  
  
        DATABASE::"Posted Seminar Reg.  
Header",0,PostedSeminarRegHeader.TABLECAPTION,  
  
        PostedSeminarRegHeader.COUNT);  
  
END;  
  
//CSD1.00<
```

2. Repeat the same search algorithm for the **Seminar Ledger Entry** table. Make sure that you utilize the appropriate table key.
 - a. In the FindRecords function trigger, enter the following code immediately after the code that you added in the previous step, but still within the comment block.


```

IF SeminarLedgEntry.READPERMISSION THEN BEGIN

SeminarLedgEntry.RESET;

SeminarLedgEntry.SETCURRENTKEY("Document No.,"Posting Date");

SeminarLedgEntry.SETFILTER("Document No.",DocNoFilter);

SeminarLedgEntry.SETFILTER("Posting Date",PostingDateFilter);

InsertIntoDocEntry(

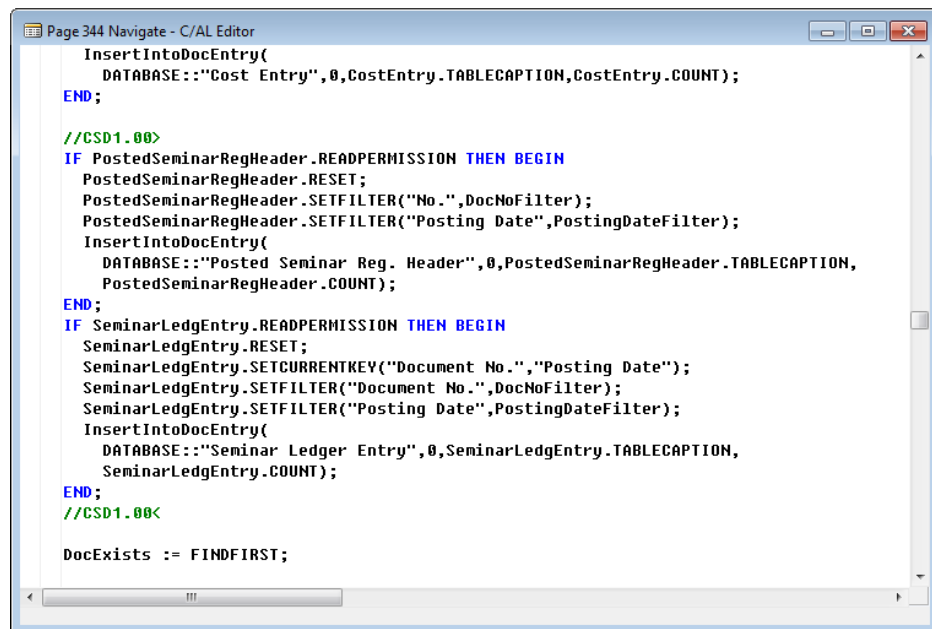
    DATABASE::"Seminar Ledger Entry",0,SeminarLedgEntry.TABLECAPTION,

    SeminarLedgEntry.COUNT);

END;

```

The “C/AL Logic for Finding the Posted Seminar Reg. Header and Seminar Ledger Entry Records” figure shows the completed code in the context of the **FindRecords** function trigger.



```

Page 344 Navigate - C/AL Editor
InsertIntoDocEntry(
    DATABASE::"Cost Entry",0,CostEntry.TABLECAPTION,CostEntry.COUNT);
END;
//CSD1.00>
IF PostedSeminarRegHeader.READPERMISSION THEN BEGIN
    PostedSeminarRegHeader.RESET;
    PostedSeminarRegHeader.SETFILTER("No.",DocNoFilter);
    PostedSeminarRegHeader.SETFILTER("Posting Date",PostingDateFilter);
    InsertIntoDocEntry(
        DATABASE::"Posted Seminar Reg. Header",0,PostedSeminarRegHeader.TABLECAPTION,
        PostedSeminarRegHeader.COUNT);
END;
IF SeminarLedgEntry.READPERMISSION THEN BEGIN
    SeminarLedgEntry.RESET;
    SeminarLedgEntry.SETCURRENTKEY("Document No.,"Posting Date");
    SeminarLedgEntry.SETFILTER("Document No.",DocNoFilter);
    SeminarLedgEntry.SETFILTER("Posting Date",PostingDateFilter);
    InsertIntoDocEntry(
        DATABASE::"Seminar Ledger Entry",0,SeminarLedgEntry.TABLECAPTION,
        SeminarLedgEntry.COUNT);
END;
//CSD1.00<
DocExists := FINDFIRST;

```

FIGURE 5.8: C/AL LOGIC FOR FINDING THE POSTED SEMINAR REG. HEADER AND SEMINAR LEDGER ENTRY RECORDS

Task 3: Customize the ShowRecords Function

High Level Steps

1. Add code to the **ShowRecords** function to include two more case switches, for the **Posted Seminar Reg. Header** and **Seminar Ledger Entry** tables. For each case switch, run the lookup page for the appropriate record variable.

Detailed Steps

1. Add code to the **ShowRecords** function to include two more case switches, for the **Posted Seminar Reg. Header** and **Seminar Ledger Entry** tables. For each case switch, run the lookup page for the appropriate record variable.
 - a. Locate the **ShowRecords** function trigger.
 - b. In the **ShowRecords** function trigger, at the end of the CASE block, and immediately after the DATABASE::"Cost Entry": case switch, enter the following code:

```
//CSD1.00>  
  
DATABASE::"Posted Seminar Reg. Header":  
  
    PAGE.RUN(0,PostedSeminarRegHeader);  
  
DATABASE::"Seminar Ledger Entry":  
  
    PAGE.RUN(0,SeminarLedgEntry);  
  
//CSD1.00<
```

The "ShowRecords Function Trigger Modification" figure shows the completed code in the context of the ShowRecords function trigger.

```

Page 344 Navigate - C/AL Editor
DATABASE::"Service Ledger Entry":
PAGE.RUN(0,ServLedgerEntry);
DATABASE::"Warranty Ledger Entry":
PAGE.RUN(0,WarrantyLedgerEntry);
DATABASE::"Cost Entry":
PAGE.RUN(0,CostEntry);
//CSD1.00>
DATABASE::"Posted Seminar Reg. Header":
PAGE.RUN(0,PostedSeminarRegHeader);
DATABASE::"Seminar Ledger Entry":
PAGE.RUN(0,SeminarLedgEntry);
//CSD1.00<
END;
END;

```

FIGURE 5.9: SHOWRECORDS FUNCTION TRIGGER MODIFICATION



Note: The call to open the page passes the value 0 (zero) into the **PAGE.RUN** function. This causes the system to open the default lookup page for the table. Always make sure that the table that is included in the Navigate feature has the lookup page defined, or explicitly define the page to run.

- c. Compile, save, and then close the page.

Task 4: Define Default Lookup and Drilldown Pages for Tables

High Level Steps

1. Define the default lookup and drilldown pages for the **Posted Seminar Reg. Header** table.
2. Define the default lookup and drilldown pages for the **Seminar Ledger Entries** table.

Detailed Steps

1. Define the default lookup and drilldown pages for the **Posted Seminar Reg. Header** table.
 - a. Design table 123456718, Posted Seminar Reg. Header.
 - b. In the **Table – Properties** window, set the **LookupPageID** property to "Posted Seminar Registration".
 - c. Set the **DrillDownPageID** property to the same value.
 - d. Compile, save, and then close the table.
2. Define the default lookup and drilldown pages for the **Seminar Ledger Entries** table.
 - a. Design table 123456732, **Seminar Ledger Entry**.
 - b. In the **Table – Properties** window, set the **LookupPageID** property to "Seminar Ledger Entries".
 - c. Set the **DrillDownPageID** property to the same value.
 - d. Compile, save, and then close the table.

Exercise 3: Customize Pages

Exercise Scenario

Finally, Isaac changes the relevant Seminar Management pages, by adding the **Navigate** action to run the **Navigate** page to each of them.

Task 1: Customize the Posted Seminar Registration Page

High Level Steps

1. Add the **Navigate** action to the Actions group on the **Posted Seminar Registration** page.
2. Add code to the **Navigate** action **OnAction** trigger, to set the default document and posting date filters in the **Navigate** page, and run the **Navigate** page.

Detailed Steps

1. Add the **Navigate** action to the Actions group on the **Posted Seminar Registration** page.
 - a. Design page 123456734, **Posted Seminar Registration**.
 - b. Show the Action Designer for Page Actions.
 - c. Define the ActionItems ActionContainer, and make sure that you remove all indentation.
 - d. Under the ActionItems ActionContainer, add a new action, and set the Caption property to "&Navigate".
 - e. Set the following properties on the **Navigate** action.

Property	Value
Image	Navigate
Promoted	Yes
PromotedCategory	Process

2. Add code to the **Navigate** action **OnAction** trigger, to set the default document and posting date filters in the **Navigate** page, and run the **Navigate** page.
 - a. Define a global variable for page 344, **Navigate**, and name it **Navigate**.
 - b. In the OnAction trigger for the **Navigate** action, enter the following code.

```
Navigate.SetDoc("Posting Date","No.");  
  
Navigate.RUN;
```

- c. Compile, save, and then close the page.



Note: Repeat the same steps for the **Posted Seminar Reg. List** page.

Task 2: Customize the Seminar Ledger Entries Page

High Level Steps

1. Add the **Navigate** action to the Actions group on the **Posted Seminar Registration** page.
2. Change code in the **Navigate** action **OnAction** trigger, to set the default document and posting date filters in the **Navigate** page. Run the **Navigate** page.

Detailed Steps

1. Add the **Navigate** action to the Actions group on the **Posted Seminar Registration** page.
 - a. Design page 123456734, **Posted Seminar Registration**.
 - b. Show the **Action Designer** for **Page Actions**.
 - c. Select the ActionItems ActionContainer and Navigate action, and copy them to clipboard (CTRL+C).
 - d. Close Action Designer, and then close the page.
 - e. Design page 123456721, **Seminar Ledger Entries**.
 - f. Show the **Action Designer** for **Page Actions**.
 - g. Paste the actions from the clipboard.
2. Change code in the **Navigate** action **OnAction** trigger, to set the default document and posting date filters in the **Navigate** page. Run the **Navigate** page.
 - a. Define a global variable for page 344, **Navigate**, and name it *Navigate*.
 - b. In the OnAction trigger for the **Navigate** action, replace the existing code with the following code.

```
Navigate.SetDoc("Posting Date","Document No.");  
  
Navigate.RUN;
```

- c. Compile, save, and then close the page.

Module Review

Module Review and Takeaways

Seminar Management individual features are now working together. Users can easily move through pages by clicking actions. To do this, you added appropriate actions to Seminar Management pages that you created earlier.

You also extended the functionality of pages that display the posted transaction data for Seminar Management, by letting users call Navigate from those pages.

Finally, you extended the Navigate feature to search for posted seminar registration information and seminar ledger entries. Navigate is a central traceability feature of Microsoft Dynamics NAV 2013. Users frequently depend on this feature to search for specific transactions, or analyze the results of a single transaction.

Test Your Knowledge

Test your knowledge with the following questions.

1. Which of the following action fails if there is data in the table or a field?

- () Changing table name.
- () Changing the properties that control how data is displayed or formatted.
- () Changing TableRelation property of a field.
- () Change the type of a field from Code to Text.
- () Increase the length of a Text or a Code field.

2. Which conditions must be met if you want to change a normal field to a FlowField?

3. Which object contains the complete Navigate functionality?

- Page 344, Navigate
- Page 433, Navigate
- Codeunit 344, NavigateManagement
- Codeunit 433, Navigate
- Report 344, Navigate

4. Which functions in the Navigate page do you have to customize to enable searching and showing new documents or ledger entries?

Test Your Knowledge Solutions

Module Review and Takeaways

1. Which of the following action fails if there is data in the table or a field?

- Changing table name.
- Changing the properties that control how data is displayed or formatted.
- Changing TableRelation property of a field.
- Change the type of a field from Code to Text.
- Increase the length of a Text or a Code field.

2. Which conditions must be met if you want to change a normal field to a FlowField?

MODEL ANSWER:

The field must be of the data type compatible with the FlowField calculation type, and there must be no data in the field in any of the companies in the database.

3. Which object contains the complete Navigate functionality?

- Page 344, Navigate
- Page 433, Navigate
- Codeunit 344, NavigateManagement
- Codeunit 433, Navigate
- Report 344, Navigate

4. Which functions in the Navigate page do you have to customize to enable searching and showing new documents or ledger entries?

MODEL ANSWER:

FindRecords and ShowRecords