MODULE 6: REPORTING

Module Overview

The Seminar module to this point includes the following:

- Master tables and pages.
- A way to create new seminar registrations.
- Routines to post the registrations.

These features are integrated into the standard application so that you can access them from a new **Seminar Management** menu in the Departments area. The next step is to create reports for the module.

Objectives

The objectives are:

- Use report event triggers.
- Use special report functions.
- Create reports for the RoleTailored client.
- Create a seminar participant list.
- Create a ProcessingOnly report that posts invoices.

Prerequisite Knowledge

Before analyzing and implementing the report functionality that is covered in this module, we will review the following concepts :

- Report request pages
- Report triggers
- Report functions
- ProcessingOnly reports

Lesson Objectives

- Use report event triggers.
- Apply report functions.

Report Request Pages

In Microsoft Dynamics NAV 2013, a report is initialized with a **Request** Page. A request page runs before a report executes. Request pages enable end-users to specify options and filters for a report.

Report Triggers

Each report object consists of several elements that can contain the following triggers:

- The report itself
- One or more data items
- A **Request** page that has an optional FastTab for each data item and an optional **Options** FastTab
- Columns and Labels that display data

Each of these elements has a fixed number of event triggers that execute during report execution. You must understand the order that some frequently used triggers execute. The following list details the order in which these common event triggers execute:

1. When a user starts the report, the OnInitReport trigger is called. This is the first trigger that runs. It performs processing that is required before any part of the report can run. If the **Request** page is needed, the OnInitReport trigger runs before the **Request** page is displayed. Use the OnInitReport trigger to initialize variables and to populate default values on the **Request** page.

2. If the OnInitReport trigger does not end the processing of the report, the **Request** page for the report runs if it is defined. The user can decide to cancel the report from the **Request** page.
3. If the user continues, the OnPreReport trigger is called. At this point, no data is processed. Similar to the OnInitReport trigger, the OnPreReport trigger ends report processing. Use the OnPreReport trigger to process values that the user entered on the **Request** page.
4. As long as the processing of the report is not ended in the OnPreReport trigger, the data items are processed. Each data item has its own OnPreDataItem**,** OnAfterGetRecord**,** and OnPostDataItem triggers.
5. Before any records are retrieved, the OnPreDataItem trigger is called. In the same manner, the OnPostDataItem trigger is called after the last record is processed.
6. Between the OnPreDataItem trigger and the OnPostDataItem trigger, the records of the data item process. Processing a record means executing the OnAfterGetRecord trigger for each record that the data item retrieves, and outputting the values of the records by using the report's layout.
7. If there is an indented data item, a data item run is initiated for this data item and for each record in its parent data item. You can nest data items up to ten levels deep.
8. When all records are processed in a data item, control returns to the point from which the processing initiates. For an indented data item this is the next record of the data item on the next higher level. If the data item is already on the highest level (indentation is zero), control returns to the report.
9. After the first data item at indentation level zero processes, the next data item at indentation level zero (if one exists) processes in the same manner.

When there are no more data items, the OnPostReport trigger is called. Use this trigger to do any necessary post processing, for example, cleaning up by removing temporary files.

The "Report Trigger Execution Flow" figure shows the trigger execution flow.

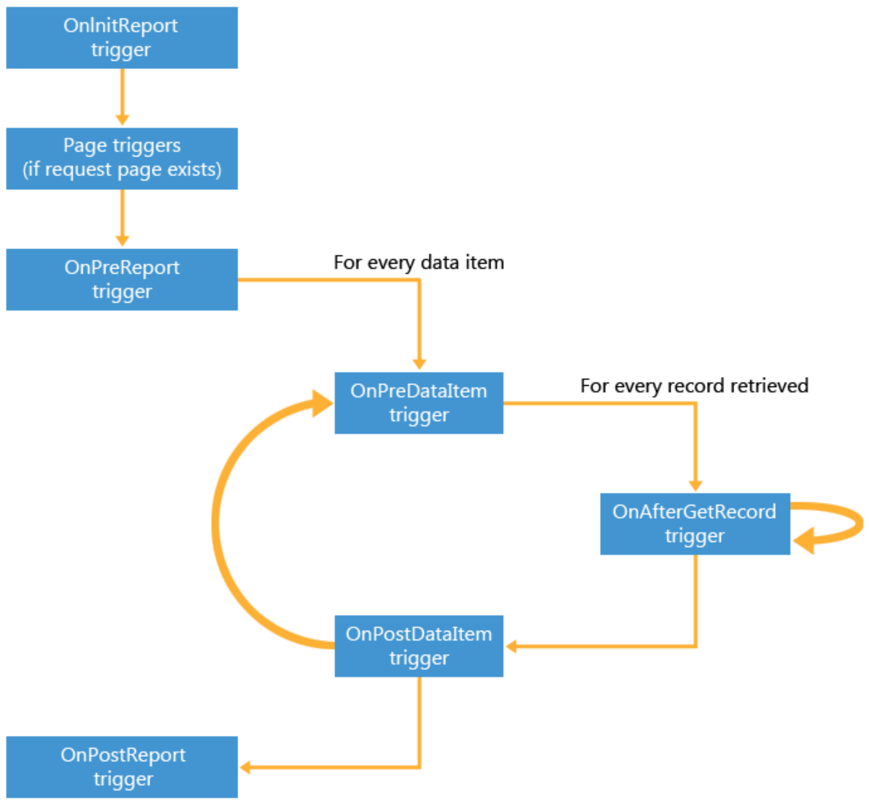


FIGURE 6.1: REPORT TRIGGER EXECUTION FLOW

Report Functions

You can use only certain functions in reports. Use these functions in complex reports. For a full listing of report functions, refer to the Microsoft Dynamics NAV 2013 Developer and IT Pro Help on the installation media or on MSDN. This developer Help file also contains useful walkthroughs to assist you in familiarizing yourself with Report Designer.

CurrReport.SKIP: Use this function to skip the current record of the current data item. If a record is skipped, it is not included in totals and it is not printed. Skipping a record in a report is much slower than never reading it at all. Therefore, use filters as much as you can.

A typical situation where you can use **SKIP** is to retrieve records from a related table by using values in the current record to form a filter. If the values in the current record already indicate that no records from the related table will be retrieved, you do not have to perform this processing and you can use **SKIP** to avoid the processing.

CurrReport.BREAK: Use this function to skip the rest of the processing of the data item that is currently processing. The report resumes processing the next data item. All indented data items under the data itemthat caused the break are also skipped.

CurrReport.QUIT: This function skips the rest of the report; however, it is not an error. It is a typical ending for a report. When you use the **QUIT** function, the report exits without committing any changes that were made to the database during the execution. The OnPostReport trigger will not be called.

CurrReport.PREVIEW: Use this function to determine whether a report is printing in preview mode.

Note: *If you run a report in preview and the **CurrReport.PREVIEW** function is called, then the **Print** and **Save As** functionality is not available in the Microsoft Dynamics NAV 2013 client for Windows, Microsoft Dynamics NAV Web client, or in an application that runs on Microsoft Dynamics NAV Portal Framework for Microsoft SharePoint 2010.*

*This makes sure that any functionality that depends on the **CurrReport.PREVIEW** function being FALSE is called correctly when doing the actual print. If you run a client report definition (RDLC) report layout in preview mode and do not call the **CurrReport.PREVIEW** function, then you can print from the **Print Preview** window.*

ProcessingOnly Report

A processing-only report is a report object that does not print, but only processes table data. Processing table data is not limited to processing-only reports. Reports that print can also change records. This section also applies to those reports. You can specify a report to be "Processing Only" by changing the ProcessingOnly property of the Report object. The report functions as it is supposed to (processing data items); however, it does not generate any printed output.

When the ProcessingOnly property is set, the **Request** page for the report changes slightly, as the **Print** and **Preview** buttons are replaced with an **OK** button. The **Cancel** and **Help** buttons remain unchanged.

As soon as the ProcessingOnly property is set, use the following guidelines to develop processing-only reports:

- Decide which tables are read. These are the data items.
- Design the report so that most of the code goes into the OnAfterGetRecord trigger.
- Use the **INSERT** or **MODIFY** functions in the tables, as appropriate.
- Use a dialog to show the user the progress and let the user cancel the report.

There are advantages in using a report instead of a codeunit to process data:

- The **Request** page functionality that lets the user select options and filters for data items is easily available in a report, but it is difficult to program in a codeunit.
- The Report Dataset Designer helps you visualize the program execution flow.
- Instead of writing code to open tables and to retrieve records, use report data items to provide a declarative way to access data.

Note: *Processing Only reports have the advantage of built-in user interactivity by using the Request page designer. When the process requires user interactivity, choose a Processing Only report instead of a codeunit. Processing Only reports are also easier to maintain because of the way the data items visualize the flow of the code.*

Reporting Lab Overview

The customer's functional requirements describe the need for the following reports and statistics:

- A list of the participants registered for a seminar.
- The total costs for each seminar, segregated into chargeable and nonchargeable costs to the customer.
- Statistics for different time periods, such as a month, last year, this year, or to the current date.

You can create two main reports to fulfill these requirements:

- A participant list
- A processing-only report that posts invoices

Use a lab in this module to implement each of these reports.

Lesson Objectives

Create reports for the RoleTailored client.

Participant List Reporting

A review of the client's specifications shows that reporting is required by the Seminar Management module. Begin the Analysis and Design of the needed reports.

Lesson Objectives

Create a seminar participant list.

Solution Analysis

The client's functional specifications require a **Participant List** report. This is a list of enrolled participants for a seminar. This report should be available from both the main **Seminar** menu and the **Seminar Registration** page.

Solution Design

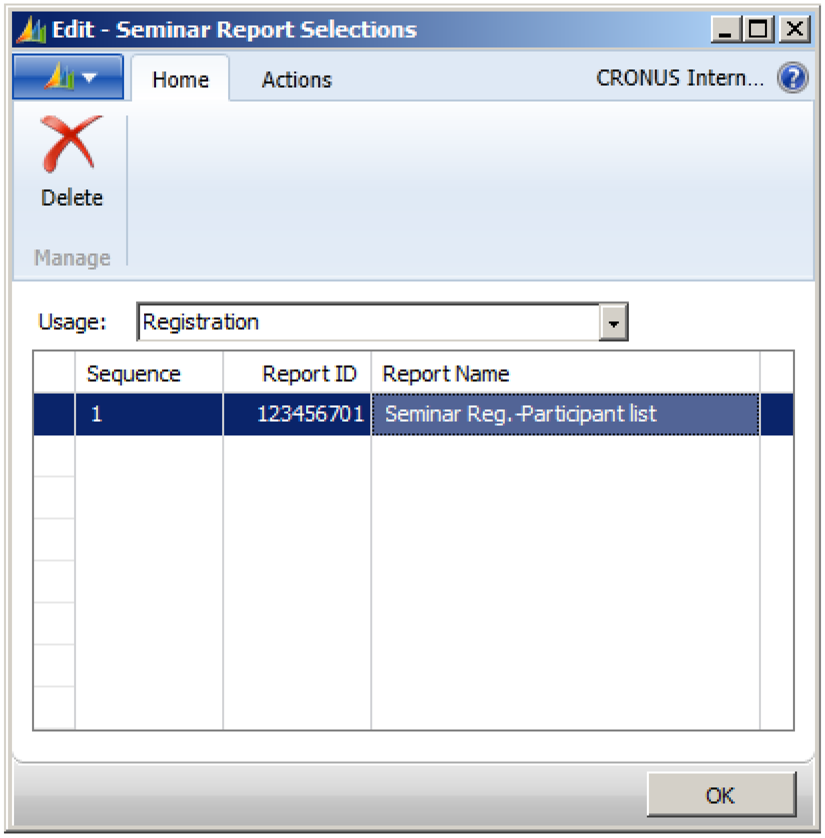
To implement this report, you must create the following items:

- The report itself
- The **Request** page to set the parameters of the report
- The controls to access the report from a page
- A page where you can select seminar reports

GUI Design

The **Seminar Report Selection** page displays the available seminar reports.

THE SEMINAR REGISTRATION PAGE (123456710)



Usage: Registration

Sequence	Report ID	Report Name
1	123456701	Seminar Reg.-Participant list

Seminar Menu: Modify this menu by adding the following menu item to the Reports Group (under Order Processing):

Menu Type	Menu Name	Group	Comment
Item	Seminar Reg.-Participant List	Reports	Opens report 123456701 Seminar Reg.-Participant List

Add the following menu item to the Setup Group:

Menu Type	Menu Name	Group	Comment
Item	Report Selections	Setup	Opens page 123456723 Seminar Report Selection

Change the **Seminar Registration** page by adding a promoted Print command to the **Actions** menu. This starts the **Participant List** report. See the "Seminar Registration Page" figure for an example.

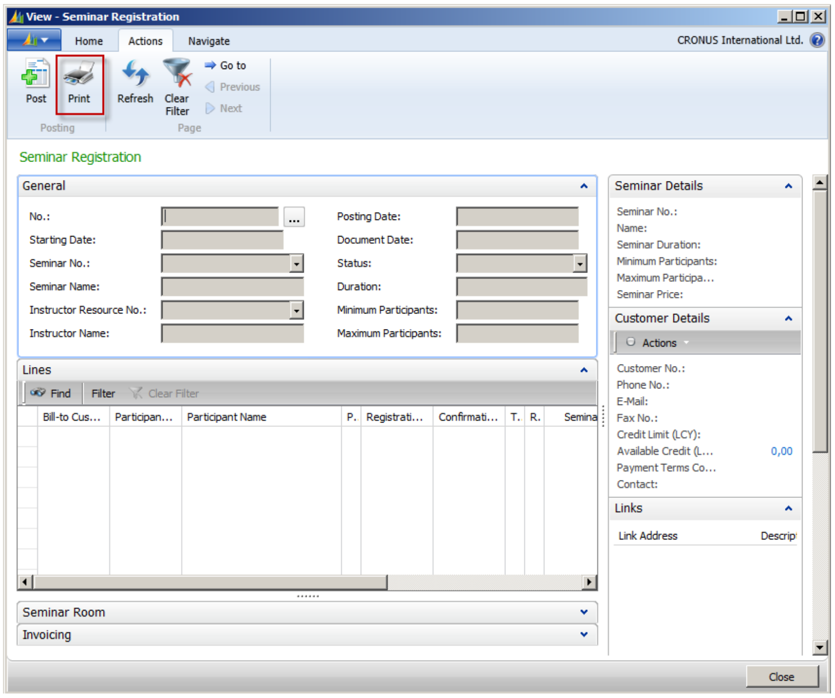


FIGURE 6.3: THE SEMINAR REGISTRATION PAGE (123456710)

Functional Design

The information for the report originates from the **Seminar Registration Header** table and the **Seminar Registration Line** table.

When users run this report, they select which **Seminar Registration Headers** to include. For each **Seminar Registration Header** table, the program then prints information from each corresponding **Seminar Registration Line** table.

Table Design

Implementation of the **Participant List** report requires the creation of one new table that is called the **Seminar Report Selections** table. You must also change the Seminar Registration Header table .

Lab 6.1: Creating the Seminar Participant List

Scenario

A **Participant List** report is required for the RoleTailored client, along with a way to select and run seminar reports. Create a **Participant List** report that lists all registrations by seminar. Use the typical Header/Detail list format and logic. Then create a way to selected seminar reports from a page.

Exercise 1: Part A: The Report Dataset

Task 1: Add a field to the Seminar Registration Header table

High Level Steps

1. Add a field to the **Seminar Registration Header** table (123456710).

Detailed Steps

1. Add a field to the **Seminar Registration Header** table (123456710).

No.	Field Name	Type	Length	Comment
40	No. Printed	Integer		Must not be editable.

Task 2: Create a codeunit to increment the No. Printed field just added to the Seminar Registration Header table.

High Level Steps

1. Create the Seminar Registration-Printed codeunit (123456702).

Detailed Steps

1. Create the Seminar Registration-Printed codeunit (123456702).
    a. Set the property to specify the **Seminar Registration Header** table as the source table for this codeunit.
    b. Enter code in the appropriate trigger so that when the program runs this codeunit, it does the following tasks:
        - Finds the **Seminar Registration Header** record.
        - Increases the No. Printed by 1.
        - Changes and commits the table.

```
FIND;

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

MODIFY;

COMMIT;
```

Task 3: Create the actual dataset in Report Dataset Designer.

High Level Steps

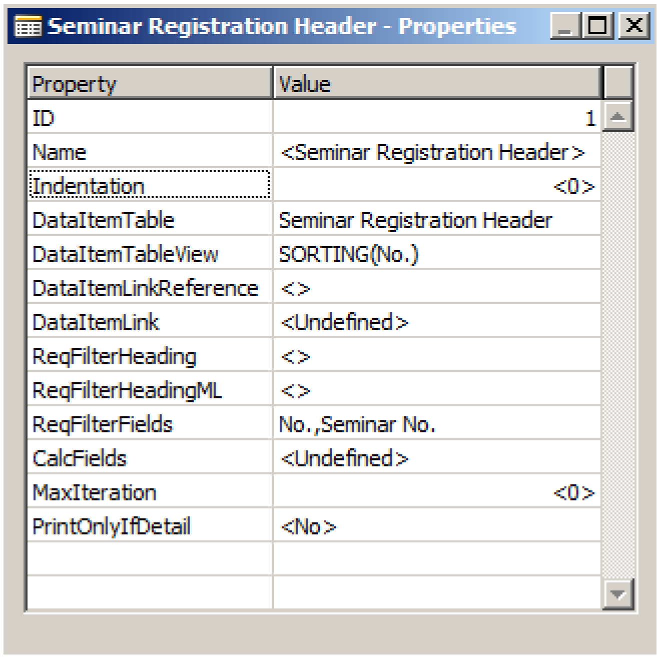
1. Create a new report.
2. Add a data item for the **Seminar Registration Header** (123456710) table.
3. Add sorting and filtering to the Seminar Registration Header data item.
4. Add a data itemfor the **Seminar Registration Line** table.
5. Add the following fields to the Seminar Registration Line data item.
6. Use the property of the columns so that captions are available for the fields of both data items.
7. Set the property for the report so that the caption is Seminar Reg**.-** Participant List**.**
8. Enter code in the appropriate trigger so that after the program gets the record of the **Seminar Registration Header** table, the program calculates the **Instructor Name** field.
9. Add a Label to the Report.

Detailed Steps

1. Create a new report.
   a. In the Object Designer, click New to create the Seminar Reg.-Participant List report (123456701).

2. Add a data item for the **Seminar Registration Header** (123456710) table.
   a. Add the following fields to the DataItem:
      - No.
      - Seminar No.
      - Seminar Name
      - Starting Date
      - Duration
      - Instructor Name
      - Room Name

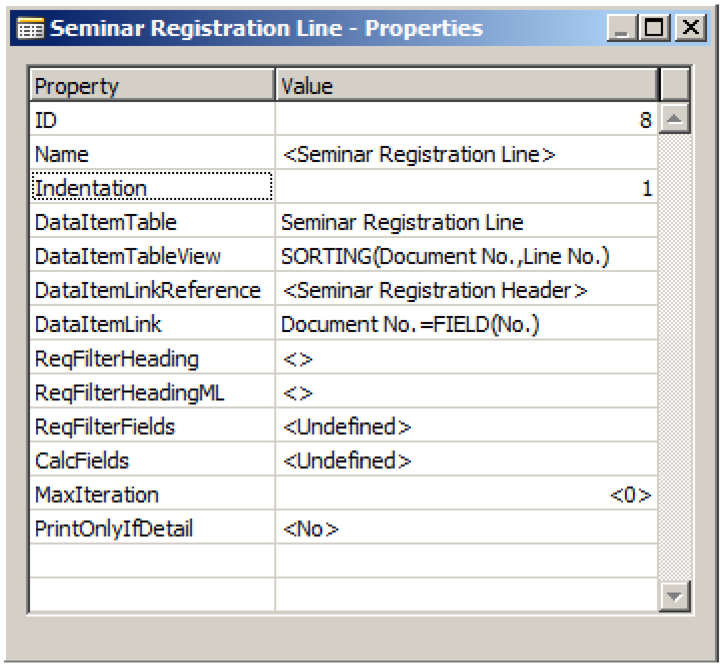
3. Add sorting and filtering to the Seminar Registration Header data item.
   a. Set the properties for the Seminar Registration Header data item so that it is sorted by No., and the filter fields are **No**., and **Seminar No**. as shown in the figure PROPERTIES FOR THE SEMINAR REGISTRATION HEADER DATA ITEM:



Property	Value
ID	1
Name	<Seminar Registration Header>
Indentation	<0>
DataItemTable	Seminar Registration Header
DataItemTableView	SORTING(No.)
DataItemLinkReference	<>
DataItemLink	<Undefined>
ReqFilterHeading	<>
ReqFilterHeadingML	<>
ReqFilterFields	No.,Seminar No.
CalcFields	<Undefined>
MaxIteration	<0>
PrintOnlyIfDetail	<No>

FIGURE 6.4: PROPERTIES FOR THE SEMINAR REGISTRATION HEADER DATA ITEM

4. Add a data itemfor the **Seminar Registration Line** table.
   a. Set the property so that the data item indents to the first level.
   b. Set the property so that the data item sorts by **Document No.** and **Line No.**.
   c. Set the DataItemLinkReference and the DataItemLink properties so that the data item links to the **Document No**.in the line data item is equal to the **No** .field in the header data item . as shown in the figure PROPERTIES FOR THE SEMINAR REGISTRATION LINE DATA ITEM:



Property	Value
ID	8
Name	<Seminar Registration Line>
Indentation	1
DataItemTable	Seminar Registration Line
DataItemTableView	SORTING(Document No.,Line No.)
DataItemLinkReference	<Seminar Registration Header>
DataItemLink	Document No.=FIELD(No.)
ReqFilterHeading	<>
ReqFilterHeadingML	<>
ReqFilterFields	<Undefined>
CalcFields	<Undefined>
MaxIteration	<0>
PrintOnlyIfDetail	<No>

FIGURE 6.5: PROPERTIES FOR THE SEMINAR REGISTRATION LINE DATA ITEM

5. Add the following fields to the Seminar Registration Line data item.
   a. Bill-to Customer No.
   b. Participant Contact No.
   c. Participant Name

6. Use the property of the columns so that captions are available for the fields of both data items.
   a. Set the Includecaption property to True for all columns.

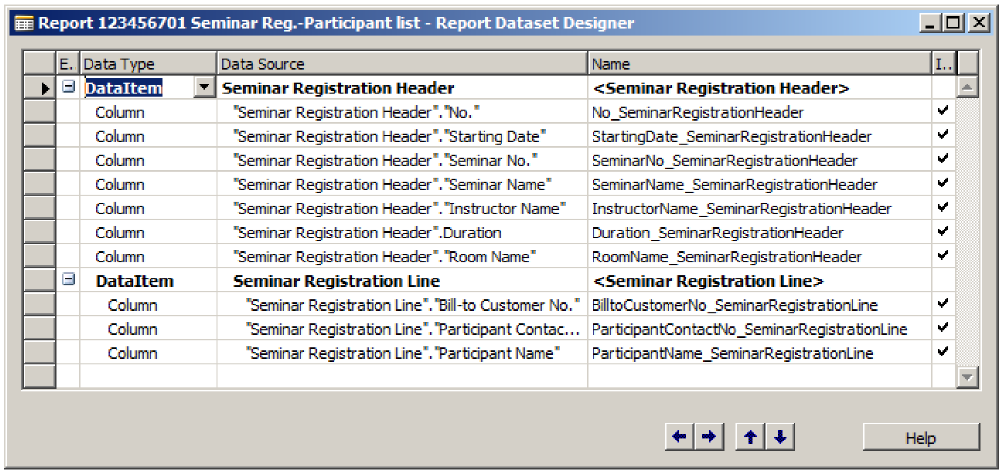
7. Set the property for the report so that the caption is Seminar Reg.- Participant List.
   a. In the Report Dataset Designer goto the last row and click the down arrow.
   b. Open the Properties window, the report properties are displayed.
   c. Set the Caption property to Seminar Reg.- Participant List.

8. Enter code in the appropriate trigger so that after the program gets the record of the **Seminar Registration Header** table, the program calculates the **Instructor Name** field.
   a. Select the Seminar Registration Header data item and click F9.
   b. In the OnafterGetRecord trigger type the following code:

```
CALCFIELDS(“Instructor Name”);
```

9. Add a Label to the Report.
   a. Open the Report Label Designer on the **View** menu.
   b. Add a label to the report with a caption of Seminar Registration Header:
      i. Select View, Labels.
      ii. In the Report Label Designer window on the first line type SeminarRegistrationHeader in the **Name** and in the **Caption**.
      iii. Close the Report Label Designer window.

   c. The Report Dataset now looks similar to the "Report Dataset Designer" figure.


Report 123456701 Seminar Reg.-Participant list - Report Dataset Designer

E.	Data Type	Data Source	Name	I..
⊟	DataItem	Seminar Registration Header	<Seminar Registration Header>	
	Column	"Seminar Registration Header"."No."	No_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Starting Date"	StartingDate_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Seminar No."	SeminarNo_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Seminar Name"	SeminarName_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Instructor Name"	InstructorName_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header".Duration	Duration_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Room Name"	RoomName_SeminarRegistrationHeader	✓
⊟	DataItem	Seminar Registration Line	<Seminar Registration Line>	
	Column	"Seminar Registration Line"."Bill-to Customer No."	BilltoCustomerNo_SeminarRegistrationLine	✓
	Column	"Seminar Registration Line"."Participant Contac...	ParticipantContactNo_SeminarRegistrationLine	✓
	Column	"Seminar Registration Line"."Participant Name"	ParticipantName_SeminarRegistrationLine	✓

FIGURE 6.6: REPORT DATASET DESIGNER

Exercise 2: Part B: The Report Layout

Exercise Scenario

To render a report from inside the RoleTailored client, you must create a Visual Studio client report definition (RDLC) report layout.

Task 1: Use Visual Studio Report Designer to create a layout for the report.

High Level Steps

1. Open Visual Studio Report Designer.
2. Add a table to the body of the report.
3. Add fields to the table.
4. Add group header rows to the table.
5. Add fields to the group header.
6. Format the table.

Detailed Steps

1. Open Visual Studio Report Designer.
   a. Select **View > Layout** on the Tools menu. This creates an empty RDLC layout and opens it in Visual Studio Report Designer.

2. Add a table to the body of the report.
   a. Select **View > Toolbox** in Visual Studio Report Designer if the toolbox is not visible by default.
   b. In the toolbox, select the **Table** control and drag it onto the body of the report.
   c. Position the table in the upper-left corner in the body of the report.

3. Add fields to the table.
   a. Select **View > Report Data** to show the report data window.
   b. In the report data window, open the Parameters folder.
   c. From the Parameters folder, select the field **[@BilltoCustomerNo_SeminarRegistrationLineCaption]**, and then drag it onto the table on the text box on the first row, first column (in the Header Row).
   d. Repeat this for the following fields:
      - **[@ParticipantContactNo_SeminarRegistrationLineCaption]**
      - **[@ParticipantName_SeminarRegistrationLineCaption]**

      The Header row in the table now contains the caption fields for the fields from the **Seminar Registration Line** table.

   e. In the second row of the table, in the first text box, right-click the text box, and then select the following field from the shortcut menu: [BilltoCustomerNo_SeminarRegistrationLine]
   f. Repeat this for the following fields:
      - **[ParticipantContactNo_SeminarRegistrationLine]**
      - **[ParticipantName_SeminarRegistrationLine]**

4. Add group header rows to the table.
   a. In the **Row Groups** window at the bottom of the screen, select **Details > Add Group > Parent Group**.
   b. In the Tablix group window, enter **[No_SeminarRegistrationHeader]** in the **Group By** field.
   c. Delete the first column of the table.
   d. Right-click the detail row of the table, and then select **Insert Row, Outside Group, Above**.

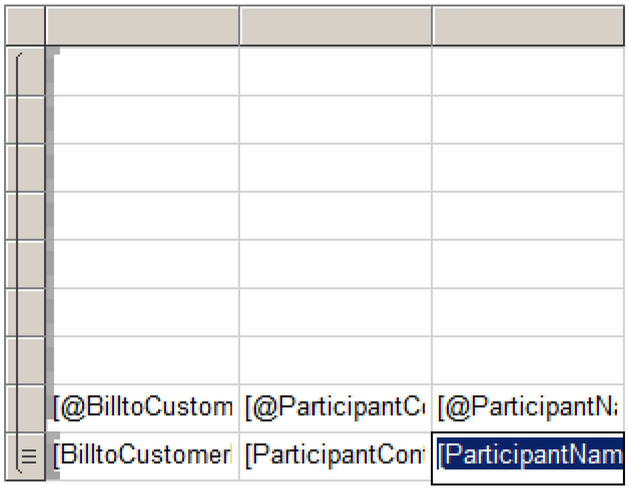
e. Repeat this step until there are seven empty group header rows in the table.

f. In the first row of the table, select the three text boxes.

g. Right-click, and then click **Cut**.

h. On the first text box on the header row just above the Details row, right-click, and then click **Paste**.

The table should look like the "Table with Group Header Rows" figure.



[@BilltoCustom	[@ParticipantC	[@ParticipantN
[BilltoCustomer	[ParticipantCon	[ParticipantNam

FIGURE 6.7: TABLE WITH GROUP HEADER ROWS

5. Add fields to the group header.
   a. In the first column of the table, in the empty group header rows, add the following fields:
      i. =Parameters!No_SeminarRegistrationHeaderCaption.Value
      ii. =Parameters!SeminarNo_SeminarRegistrationHeaderCaption.Value
      iii. =Parameters!SeminarName_SeminarRegistrationHeaderCaption.Value
      iv. =Parameters!StartingDate_SeminarRegistrationHeaderCaption.Value
      v. =Parameters!Duration_SeminarRegistrationHeaderCaption.Value
      vi. =Parameters!InstructorName_SeminarRegistrationHeaderCaption.Value
      vii. =Parameters!RoomName_SeminarRegistrationHeaderCaption.Value

   b. In the second column of the table, in the empty group header rows add the following fields:
      i. =Fields!No_SeminarRegistrationHeader.Value
      ii. =Fields!SeminarNo_SeminarRegistrationHeader.Value
      iii. =Fields!SeminarName_SeminarRegistrationHeader.Value
      iv. =Fields!StartingDate_SeminarRegistrationHeader.Value
      v. =Fields!Duration_SeminarRegistrationHeader.Value
      vi. =Fields!InstructorName_SeminarRegistrationHeader.Valu
e
      vii. =Fields!RoomName_SeminarRegistrationHeader.Value

6. Format the table.
   a. Increase the Width of the columns in the table.
   b. Change the Color property of the line captions to Blue.
   c. Change the FontSize property of the text boxes in the table to 8pt.

Task 2: Add a Page Header

High Level Steps

1. Add a page header to the report.
2. Format the title.

Detailed Steps

1. Add a page header to the report.
   a. In the Page Header, add four textboxes: one on the left side and three on the right side.
   b. In the text box on the left side of the page header, enter the following value expressions:

```
=Parameters!SeminarRegistrationHeader.Value
```

   c. In the three text boxes on the right side, add the following expressions:

```
=Globals!ExecutionTime

=Globals!PageNumber & " of " & Globals!TotalPages

=User!UserID
```

2. Format the title.
    a. In the text box on the Page Header that contains the title, set the Color to blue, and then make the text Bold.

Task 3: Add the Company Name to the Report

High Level Steps

1. To add the Company Name to the report, the dataset must be updated.
2. Add an integer data item to the bottom of the data item designer.
3. Add a Global Variable.
4. Add the Company Name as a column to the Integer DataItem.
5. Add the Company Name to the Page Header.

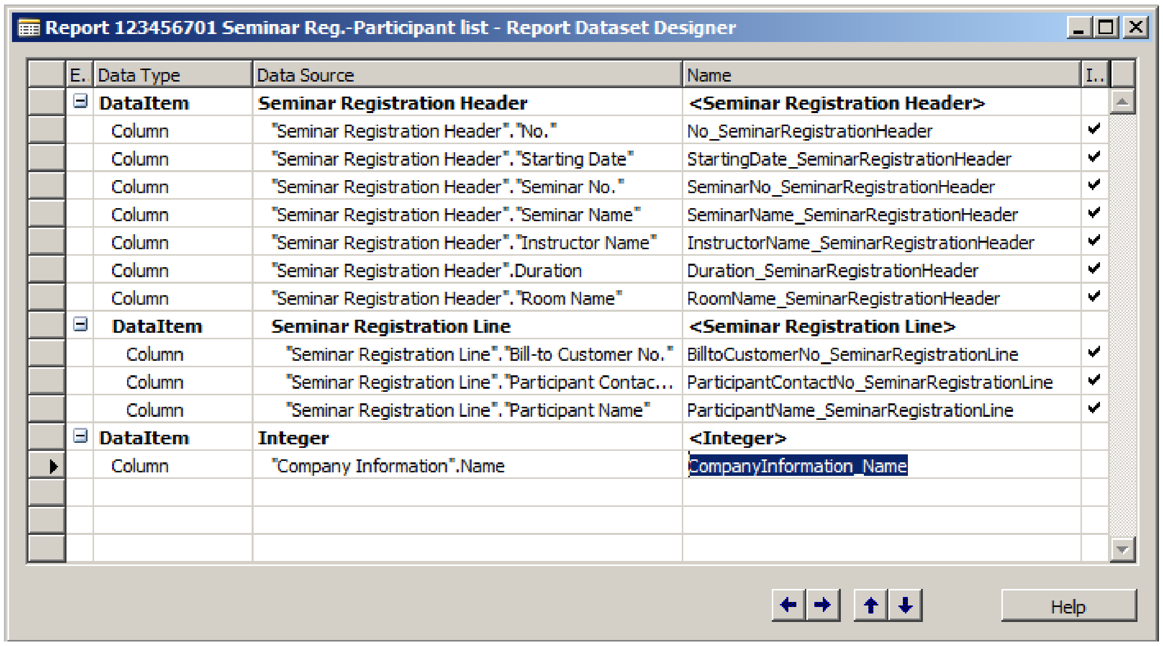
Detailed Steps

1. To add the Company Name to the report, the dataset must be updated. Add an integer data item to the bottom of the data item designer.
    a. Make sure that the Integer data item is not indented.
    b. Filter the Integer data item so that it only displays one record.
        i. In the DataItemTableView propertie of the Integer data item enter the following Value:

```
WHERE(Number=CONST(1))
```

2. Add a Global Variable.
    a. In the menu, select **View > Globals**, and then add a variable of type record from the **Company Information** (79) table.

3. Add the Company Name as a column to the Integer DataItem.
    a. Change the name of the column to CompanyInformation_Name. See the "Report Dataset with Company Name" figure.



Report 123456701 Seminar Reg.-Participant list - Report Dataset Designer

E.	Data Type	Data Source	Name	I..
⊟	**DataItem**	**Seminar Registration Header**	**<Seminar Registration Header>**	
	Column	"Seminar Registration Header"."No."	No_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Starting Date"	StartingDate_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Seminar No."	SeminarNo_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Seminar Name"	SeminarName_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Instructor Name"	InstructorName_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header".Duration	Duration_SeminarRegistrationHeader	✓
	Column	"Seminar Registration Header"."Room Name"	RoomName_SeminarRegistrationHeader	✓
⊟	**DataItem**	**Seminar Registration Line**	**<Seminar Registration Line>**	
	Column	"Seminar Registration Line"."Bill-to Customer No."	BilltoCustomerNo_SeminarRegistrationLine	✓
	Column	"Seminar Registration Line"."Participant Contac...	ParticipantContactNo_SeminarRegistrationLine	✓
	Column	"Seminar Registration Line"."Participant Name"	ParticipantName_SeminarRegistrationLine	✓
⊟	**DataItem**	**Integer**	**<Integer>**	
	Column	"Company Information".Name	CompanyInformation_Name	

FIGURE 6.8: REPORT DATASET WITH COMPANY NAME

4. Add the Company Name to the Page Header.
   a. Add a text box in the Page Header to display the Company Name. Use the following expression:

```
=Last(Fields!CompanyInformation_Name.Value, "DataSet_Result")
```

Task 4: Filter the table

High Level Steps

1. Because of the new integer data item the dataset has changed. It now has one additional row that contains the company name. Filter out this row in the table.

Detailed Steps

1. Because of the new integer data item the dataset has changed. It now has one additional row that contains the company name. Filter out this row in the table.
   a. Add a Filter to the Group in the Tablix to make sure that only the following lines are allowed:

```
=Len(Fields!No_SeminarRegistrationHeader.Value) > 0
```

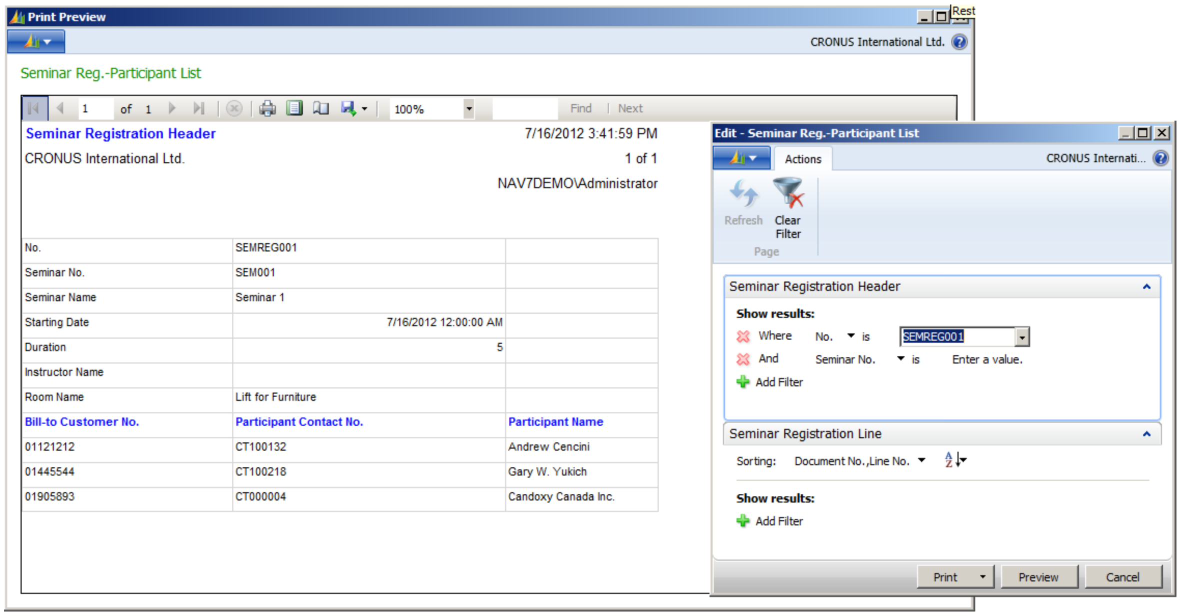
Task 5: Run the Report

High Level Steps

1. Save and Run the Report.

Detailed Steps

1. Save and Run the Report.
   a. Save the report layout and close Visual Studio.
   b. Save the report in the NAV Report Designer, and then accept the changed layout. See the "Report 123456701 Rendered in RoleTailored client" figure for an example.



Seminar Registration Header 7/16/2012 3:41:59 PM
CRONUS International Ltd. 1 of 1
NAV7DEMO\Administrator

No.	SEMREG001	
Seminar No.	SEM001	
Seminar Name	Seminar 1	
Starting Date	7/16/2012 12:00:00 AM	
Duration	5	
Instructor Name		
Room Name	Lift for Furniture	
Bill-to Customer No.	**Participant Contact No.**	**Participant Name**
01121212	CT100132	Andrew Cencini
01445544	CT100218	Gary W. Yukich
01905893	CT000004	Candoxy Canada Inc.

Seminar Registration Header
Show results:
Where No. is SEMREG001
And Seminar No. is Enter a value.
Add Filter

Seminar Registration Line
Sorting: Document No.,Line No.
Show results:
Add Filter

FIGURE 6.9: REPORT 123456701 RENDERED IN ROLETAILORED CLIENT

Exercise 3: Part C: Report Selections Table and Page

Exercise Scenario

Import the report selection objects.

Task 1: Import the Seminar Report Selections Table and Page

High Level Steps

1. Import the starter object.

Detailed Steps

1. Import the starter object.
   a. In Object Designer, click **File** > **Import**.
   b. Browse to the file Mod06\Labfiles\Lab 6.B - Starter - Report Selection.fob.

c. Click **Open**.

d. Click **Yes**, and then click **OK** to complete the import.

Exercise 4: Part D: Testing

Task 1: Test Report Selections

High Level Steps

1. Test the application.

Detailed Steps

1. Test the application.
    a. If there are no Seminar Registration records, create one by selecting **Seminars > Order Processing > Registrations** from the main menu. Complete all fields in the General and Seminar Rooms FastTabs and add at least two participants.
    b. Use the **Seminar Report Selection** window to set up the Seminar Reg.-Participant List as the report to run for Registration. To do this, select **Seminars** > **Setup > Report Selections** from the main menu. Select **Registration** as the Usage and 123456701 on the first line for the **Report ID**. The report name should be filled in automatically.
    c. Open the **Seminar Registration** page again, and then view the record prepared in Step 1. Click **Print**.
    d. The **Request** page should open with the No. set to the current registration. Select **Print** or **Preview** to print and check the report.
    e. Try to print the report by using different parameters and different registration data to verify that it works correctly.

Invoice Posting Batch Job

When a seminar concludes, customers are invoiced for their registered participants. These invoices are implemented as Batch Jobs. Batch jobs should allow for multiple participants and can include project and resource information.

Lesson Objectives

Create a Processing Only report that posts invoices.

Solution Analysis

The process to create invoices is implemented as a ProcessingOnly Report. Use this type of report object to process and modify data without rendering any report for display.

This ProcessingOnly Report creates an invoice for each customer in a filter that is specified by the user. The report has lines for each participant in a registered seminar that falls within the filter set by the user. The user should also have the option of either creating the invoices, or creating and posting them at the same time.
This type of report is known in Microsoft Dynamics NAV as a batch job. A batch job in Microsoft Dynamics NAV is a routine that processes data in batches, such as the Adjust Exchange Rates Batch Job, or Batch Posting Sales Orders, Invoices, and Credit Memos.

If you set up numerous orders, invoices, or credit memos, you can post them with a batch job. You can post them at night or at another convenient time. Batch posting invoices and credit memos functions exactly like orders.

Solution Design

Use the **Seminar Ledger Entries** to access all charges to invoice for posted seminars. The report creates Sales Headers and Sales Lines as appropriate. If the user selected the option to also post the invoices, the report runs the posting process for sales invoices.

Lab 6.2: Creating the Invoice Posting Batch Job

Scenario

As soon as a seminar concludes, the registrations must be posted in preparation for billing. Create a ProcessingOnly report to generate Microsoft Dynamics NAV standard format invoices that are ready for billing and posting.

Exercise 1: Create the Invoice Posting Batch Job

Exercise Scenario

This report differs from the previous two reports because it is not a print report, but is a ProcessingOnly report.

Task 1: Import the Report

High Level Steps

1. Import the report object.

Detailed Steps

1. Import the report object.
   a. In the development environment, open the Object Designer using **Tools > Object Designer**.
   b. Locate the Mod06\LabfilesLab 6.B - Starter.fob.
   c. Select **File > Import** to import the report.

Task 2: Complete the C/AL code of the imported report

High Level Steps

1. In Object Designer, locate report **Create Seminar Invoices** (123456700), and then complete the C/AL code according to these guidelines.
2. Verify that these global variables are defined for the report.
3. Verify that these text constants are defined.
4. Enter code in the FinalizeSalesInvHeaderfunction trigger so that the function performs these tasks.
5. Enter code into the InsertSalesInvHeader function trigger so that the trigger performs these tasks.
6. Enter code into the appropriate trigger so that just before the data item is run, the program performs these tasks.

7. Enter code in the appropriate trigger so that after the program gets a record for the DataItem, the program performs these tasks.
8. Enter code in the appropriate trigger so that when the program posts the DataItem, it performs these tasks.
9. Access the Request Page Designer by clicking **View > Request Page**.

Detailed Steps

1. In Object Designer, locate report **Create Seminar Invoices** (123456700), and then complete the C/AL code according to these guidelines.
   a. Verify the property to specify that this report is for processing-only.
   b. Verify a data item for the **Seminar Ledger Entry** table and set the properties for the data item as follows:
      i. Specify the Key in DataItemTableView property of the data item to sort by the following fields:
         - **Bill-to Customer No**.
         - **Document No**.
         - **Charge Type**
         - **Participant Contact No**.
      ii. Specify that the user can filter on the **Bill-to Customer No**., **Seminar No.**, and **Posting Date**.
      iii. Specify the variable name of the data item as SeminarLedgerEntry.

2. Verify that these global variables are defined for the report.

Name	Data Type	Subtype	Length
SalesHeader	Record	Sales Header	
SalesLine	Record	Sales Line	
SalesSetup	Record	Sales & Receivables Setup	
GLSetup	Record	General Ledger Setup	
Customer	Record	Customer	
CurrencyExchRate	Record	Currency Exchange Rate	
SalesCalcDiscount	Codeunit	Sales-Calc. Discount	
SalesPost	Codeunit	Sales-Post	

Name	Data Type	Subtype	Length
Window	Dialog		
PostingDateReq	Date		
DocDateReq	Date		
CalcInvoiceDiscount	Boolean		
PostInvoices	Boolean		
NextLineNo	Integer		
NoOfSalesInvErrors	Integer		
NoOfSalesInv	Integer		

3. Verify that these text constants are defined.

Name	ConstValue
Text000	Please enter the posting date.
Text001	Please enter the document date.
Text002	Creating Seminar Invoices...\\
Text003	Customer No. #1##########\
Text004	Registration No. #2##########\
Text005	The number of invoice(s) created is %1.
Text006	Not all the invoices are posted. A total of %1 invoices are not posted.
Text007	There is nothing to invoice.

4. Enter code in the FinalizeSalesInvHeaderfunction trigger so that the function performs these tasks.
   a. If the CalcInvoiceDisc variable is TRUE (which means the user selected this option), the function runs the Sales-Calc. Discount codeunit with the Sales Line record and finds the Sales Header record.
   b. The function then performs a commit.
   c. The function clears the SalesCalcDiscount and SalesPost variables and increments the NoOfSalesInvoice by one.
   d. If the PostInvoice variable is TRUE (which means the user selected this option), the function clears the SalesPost variable.
   e. If running the Sales-Post codeunit with the SalesHeader returns FALSE, the function should increment the NoOfSalesInvErrors by one.

Code Example

```
WITH SalesHeader DO BEGIN

  IF CalcInvoiceDiscount THEN

    SalesCalcDiscount.RUN(SalesLine);

  GET("Document Type","No.");

  COMMIT;

  CLEAR(SalesCalcDiscount);

  CLEAR(SalesPost);

  NoOfSalesInv := NoOfSalesInv + 1;

  IF PostInvoices THEN BEGIN

    CLEAR(SalesPost);

    IF NOT SalesPost.RUN(SalesHeader) THEN

      NoOfSalesInvErrors := NoOfSalesInvErrors + 1;

  END;

END;
```

5. Enter code into the InsertSalesInvHeader function trigger so that the trigger performs these tasks.
   a. Initializes a new Sales Header record with a Document Type of Invoice and a blank No**.** and inserts it into the database.
   b. Validates that the Sell-To Customer No**.** of the new record equals the Bill-to Customer No**.** of the ledger entry.
   c. If the Bill-to Customer No**.** value differs from that of the Sell-To Customer No., the program validates that the Bill-to Customer No**.** of the new record equals the Bill-to Customer No. of the ledger entry.
   d. Validates that the Posting Date of the new record is the PostingDateReq**.**
   e. Validates that the Document Date of the new record is the DocDateReq**.**
   f. Validates that the Currency Code of the new record is blank.
   g. Changes and commits the new record.
   h. Sets the NextLineNo variable to 10000.

Code Example

```
WITH SalesHeader DO BEGIN

  INIT;

  "Document Type" := "Document Type"::Invoice;

  "No." := '';

  INSERT(TRUE);

  VALIDATE("Sell-to Customer No.","Seminar Ledger Entry"."Bill-to Customer No.");

  IF "Bill-to Customer No." <> "Sell-to Customer No." THEN

    VALIDATE("Bill-to Customer No.","Seminar Ledger Entry"."Bill-to Customer
No.");

  VALIDATE("Posting Date",PostingDateReq);

  VALIDATE("Document Date",DocDateReq);

  VALIDATE("Currency Code",'');

  MODIFY;

  COMMIT;

  NextLineNo := 10000;

END;
```

6. Enter code into the appropriate trigger so that just before the data item is run, the program performs these tasks.
    a. In the OnPreDataItem trigger enter code that:
        i. Shows an error if the PostingDateReq or DocDateReq is empty by using the text constants Text000 and Text001.
        ii. Opens a dialog window with the text constants Text002, Text003, and Text004.

Code Example

```
IF PostingDateReq = 0D THEN

  ERROR(Text000);

IF DocDateReq = 0D THEN
```

```
  ERROR(Text001);

Window.OPEN(

  Text002 +

  Text003 +

  Text004);
```

7. Enter code in the appropriate trigger so that after the program gets a record for the DataItem, the program performs these tasks.
   a. In the OnAfterGetRecord trigger enter code that:
      i. If the No. of the Customer record differs from the Bill-to Customer No., the program gets the Customer record that corresponds to the Bill-to Customer No.
      ii. If the Customer record is blocked for All or Invoice, the NoOfSalesInvErrors increments by 1 (one).
      iii. If the Customer is not blocked for All or is only blocked for Invoice, the program proceeds with the following tasks:
         - If the Bill-to Customer No. differs from that on the current Sales Header record, the program updates the dialog window with the new Bill-to Customer No.
         - If the No. on the Sales Header is not blank, it runs the **FinalizeSalesInvoiceHeader** function.
      iv. The program then runs the **InsertSalesInvoiceHeader** function and sets the **Document Type** field and the **Document No**. field on the Sales Line to be those of the Sales Header.
      v. Updates the dialog window with the Sales Registration No.
      vi. Sets the No. of the Sales Line depending on whether the Charge Type that is being posted is for an instructor, room, or participant.
      vii. Completes the following Sales Line fields as follows:
         - **Document Type** and **Document N**o. from the **Sales Header**
         - **Line No**. from the NextLineNo variable
         - **Description** from the **Seminar Ledger Entry**

- Unit Price from the Seminar
- If the **Currency Code** on the **Sales Header** is not blank, the program test that the **Currency Factor** is blank and calculates the **Unit Price** for the **Sales** Line by running the **ExchangeAmtLCYToFCY** function of the **Currency Exchange Rate** table.

viii. Completed the **Sales Lin**e fields as follows: Quantity from the Quantity of the Seminar Ledger Entry.

ix. Inserts the Sales Line record.

x. Increments the **NextLineNo** by **10000**.

Code Example

```
IF "Bill-to Customer No." <> Customer."No." THEN

  Customer.GET("Bill-to Customer No.");

IF Customer.Blocked IN [Customer.Blocked::All,Customer.Blocked::Invoice] THEN
BEGIN

  NoOfSalesInvErrors := NoOfSalesInvErrors + 1;

END ELSE BEGIN

  IF "Seminar Ledger Entry"."Bill-to Customer No." <> SalesHeader."Bill-to
Customer No." THEN BEGIN

    Window.UPDATE(1,"Bill-to Customer No.");

    IF SalesHeader."No." <> '' THEN

      FinalizeSalesInvoiceHeader;

    InsertSalesInvoiceHeader;

  END;

  Window.UPDATE(2,"Seminar Registration No.");

  CASE Type OF

    Type::Resource:

    BEGIN

      SalesLine.Type := SalesLine.Type::Resource;

      CASE "Charge Type" OF
      "Charge Type"::Instructor:
        SalesLine."No." := "Instructor Resource No.";
      "Charge Type"::Room:
        SalesLine."No." := "Room Resource No.";
      "Charge Type"::Participant:
        SalesLine."No." := "Instructor Resource No.";
    END;
  END;
END;
SalesLine."Document Type" := SalesHeader."Document Type";
SalesLine."Document No." := SalesHeader."No.";
SalesLine."Line No." := NextLineNo;
SalesLine.VALIDATE("No.");
Seminar.GET("Seminar No.");
IF "Seminar Ledger Entry".Description <> '' THEN
  SalesLine.Description := "Seminar Ledger Entry".Description
ELSE
  SalesLine.Description := Seminar.Name;
SalesLine."Unit Price" := "Unit Price";
IF SalesHeader."Currency Code" <> '' THEN BEGIN
  SalesHeader.TESTFIELD("Currency Factor");
  SalesLine."Unit Price" :=
    ROUND(
      CurrencyExchRate.ExchangeAmtLCYToFCY(
      WORKDATE,SalesHeader."Currency Code",
      SalesLine."Unit Price",SalesHeader."Currency Factor"));

  END;

  SalesLine.VALIDATE(Quantity,Quantity);

  SalesLine.INSERT;

  NextLineNo := NextLineNo + 10000;

END;
```

8. Enter code in the appropriate trigger so that when the program posts the DataItem, it performs these tasks.
   a. In the OnPostDataItem trigger enter code that:
      i. Closes the dialog window.
      ii. If the No. of the Sales Header is blank, the program displays a message that states there is nothing to invoice.
      iii. If the No. of the Sales Header is not blank, runs the **FinalizeSalesInvoiceHeader** function.
      iv. The program displays a message that displays the number of errors that occurred, if any, or the number of invoices that were created.

Code Example

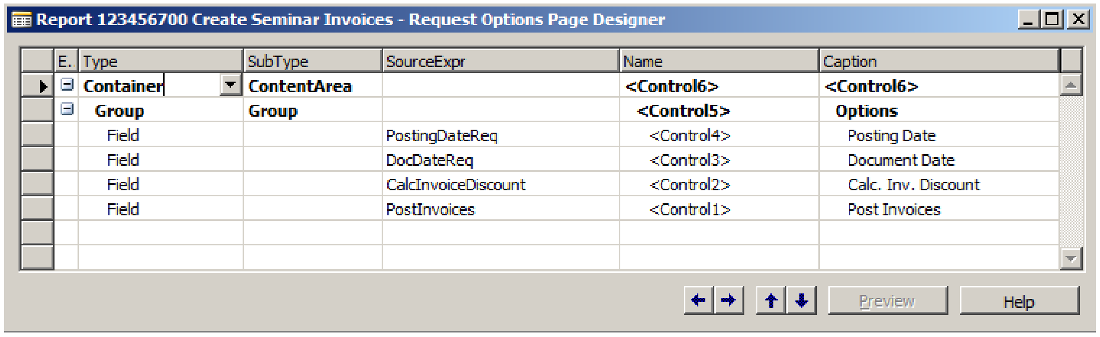
```
Window.CLOSE;

IF SalesHeader."No." = '' THEN BEGIN

  MESSAGE(Text007);

END ELSE BEGIN

  FinalizeSalesInvoiceHeader;

  IF NoOfSalesInvErrors = 0 THEN

    MESSAGE(

      Text005,

      NoOfSalesInv)

  ELSE

    MESSAGE(
    Text006,

    NoOfSalesInvErrors)

END;
```

9. Access the Request Page Designer by clicking **View > Request Page**.
    a. Add an Options group to the **Request** page and include these variables:
        - PostingDateReq
        - DocDateReq
        - CalcInvoiceDiscount
        - PostInvoices
    b. Set the property for the **Request** page so that the page saves the values after the page is closed.
    c. Enter code in the appropriate trigger so that when the **Request** page is opened, if the **PostingDateReq** and **DocDateReq** variables are not filled, they are set to the work date and set the CalcInvDisc value to the value of the **Calc. Inv. Discount** field in the **Sales Setup**.
        i. In the OnOpenPage trigger of the Request Page enter the following code:

Code Example

```
IF PostingDateReq = 0D THEN

  PostingDateReq := WORKDATE;

IF DocDateReq = 0D THEN

  DocDateReq := WORKDATE;

SalesSetup.GET;

CalcInvoiceDiscount := SalesSetup."Calc. Inv. Discount";
```

    d. The Request Page will now resemble the Figure: REQUEST PAGE OF CREATE SEMINAR INVOICES


Report 123456700 Create Seminar Invoices - Request Options Page Designer
Preview
Help

E.	Type	SubType	SourceExpr	Name	Caption
⊟	Container	ContentArea		<Control6>	<Control6>
⊟	Group	Group		<Control5>	Options
	Field		PostingDateReq	<Control4>	Posting Date
	Field		DocDateReq	<Control3>	Document Date
	Field		CalcInvoiceDiscount	<Control2>	Calc. Inv. Discount
	Field		PostInvoices	<Control1>	Post Invoices

FIGURE 6.10: REQUEST PAGE OF CREATE SEMINAR INVOICES

Task 3: Testing

High Level Steps

1. To test this part of the seminar module, complete the steps that are described in the previous exercises. In particular, make sure that the Customer who is associated with the registrations that are being tested is set up correctly.

Detailed Steps

1. To test this part of the seminar module, complete the steps that are described in the previous exercises. In particular, make sure that the Customer who is associated with the registrations that are being tested is set up correctly.
   a. Click **Seminars** > **Periodic Activities** > **Periodic Activities** from the main menu. The **Create Seminar Invoices** Request page opens.
   b. Enter data so that one or more posted registrations meet the criteria. If there are no posted registrations, create one.
   c. Click **OK** to run the ProcessingOnly Report.
   d. Look at the Sales Invoices that were created. See that the header and lines were created correctly based on the Seminar information.
   e. Expect to have one invoice per Bill-to Customer with a line for each contact registered.
   f. Check that the data flowed correctly from the registration to the invoice.
   g. Try running the Create Seminar Invoices process with the Post Invoices Option marked and unmarked and verify that the Sales Invoices are posted as indicated.

Module Review

Module Review and Takeaways

In this module, you created two reports. The first, Participant List, was a typical report. The second report was a processing-only report that enabled you to create invoices for customers with participants in completed seminars.

In the next module, you will add statistics to the Seminar module.

Test Your Knowledge

Test your knowledge with the following questions.

Sequencing Activity

Put the following steps in order by numbering each to indicate the correct order.

	Steps
	OnPreDataItem
	OnPreDataItem
	OnPostDataItem
	OnPostReport
	OnPreReport
	OnAfterGetRecord
	OnAfterGetRecord
	OnInitReport
	OnInitReport
	OnPostReport
	OnPreReport
	OnPostDataItem

1. What report function would you use to skip the processing of one record in a DataItem?

2. What are the advantages of using processing-only reports in some situations instead of codeunits?

3. In what trigger do you first have access to the values that the user entered on the **Request** page?

Test Your Knowledge Solutions

Module Review and Takeaways

Sequencing Activity

	Steps
5	OnPreDataItem
5	OnPreDataItem
9	OnPostDataItem
11	OnPostReport
3	OnPreReport
7	OnAfterGetRecord
7	OnAfterGetRecord
1	OnInitReport
1	OnInitReport
11	OnPostReport
3	OnPreReport
9	OnPostDataItem

1. What report function would you use to skip the processing of one record in a DataItem?

   MODEL ANSWER:

   The CorrReport.SKIP function is used to skip one record in a DataItem.

2. What are the advantages of using processing-only reports in some situations instead of codeunits?

   MODEL ANSWER:

   In Codeunits, you must write all of the code to build the data record looping mechanism. Report objects have this mechanism built in by default. It is much easier to obtain user input by using a report object.

3. In what trigger do you first have access to the values that the user entered on the **Request** page?

   MODEL ANSWER:

   The first trigger that runs after the user has completed the **Request** page is the OnPreReport trigger. This trigger is used to evaluate the user's input before it continues with the data items.