



80436

C/SIDE Introduction in Microsoft
Dynamics® NAV 2013

Welcome!

Thank you for joining us today.

We've worked together with Microsoft Learning Partners and Microsoft IT Academies to bring you a world-class learning experience. This includes the following:

Microsoft Certified Trainers + Instructors. Your instructor is a premier technical and instructional expert who meets ongoing certification requirements.

Customer Satisfaction Guarantee. Our Partners offer a satisfaction guarantee and we hold them accountable for it. At the end of class, please complete an evaluation of

today's experience. We value your feedback!

Certification Benefits. After training, consider pursuing a Microsoft Certification to help distinguish your technical expertise and experience. Ask your instructor about available exam promotions and discounts.

We wish you a great learning experience and ongoing career success!

Module 1

Microsoft Dynamics NAV
Development Environment

Module Overview

- Present the basic object types in Microsoft Dynamics NAV 2013.
- Describe fundamental aspects of Microsoft Dynamics NAV Development Environment. This includes the user interface (UI), application objects, and basic metadata concepts.
- Explain the physical and logical database structure.
- Explain the features for multi-developer environments

Lesson 1: Basic Objects in Microsoft Dynamics NAV

7 Basic Objects

Object	Remarks
Table	Describes how data is stored in the database and how it is retrieved. Table is the most important object type, because other object types depend on them.
Page	Enables users to view, add, modify, or delete records in a table.
Report	Prints or previews the data from a Microsoft Dynamics NAV 2013 database.
Codeunit	Called from other objects to complete a specific task. A codeunit is an organized unit of programming code, typically for managing a single process.
Query	Defines a relational data model for direct and efficient querying of the underlying Microsoft SQL Server database.
XMLport	Imports or exports data to or exports data from a Microsoft Dynamics NAV 2013 database, in XML or text format.
MenuSuite	Describes menus that are displayed in the Departments section in the RoleTailored client.

Lesson 1: Basic Objects in Microsoft Dynamics NAV

- Object-oriented vs. object-based

Lesson 2: Object Designer Fundamentals

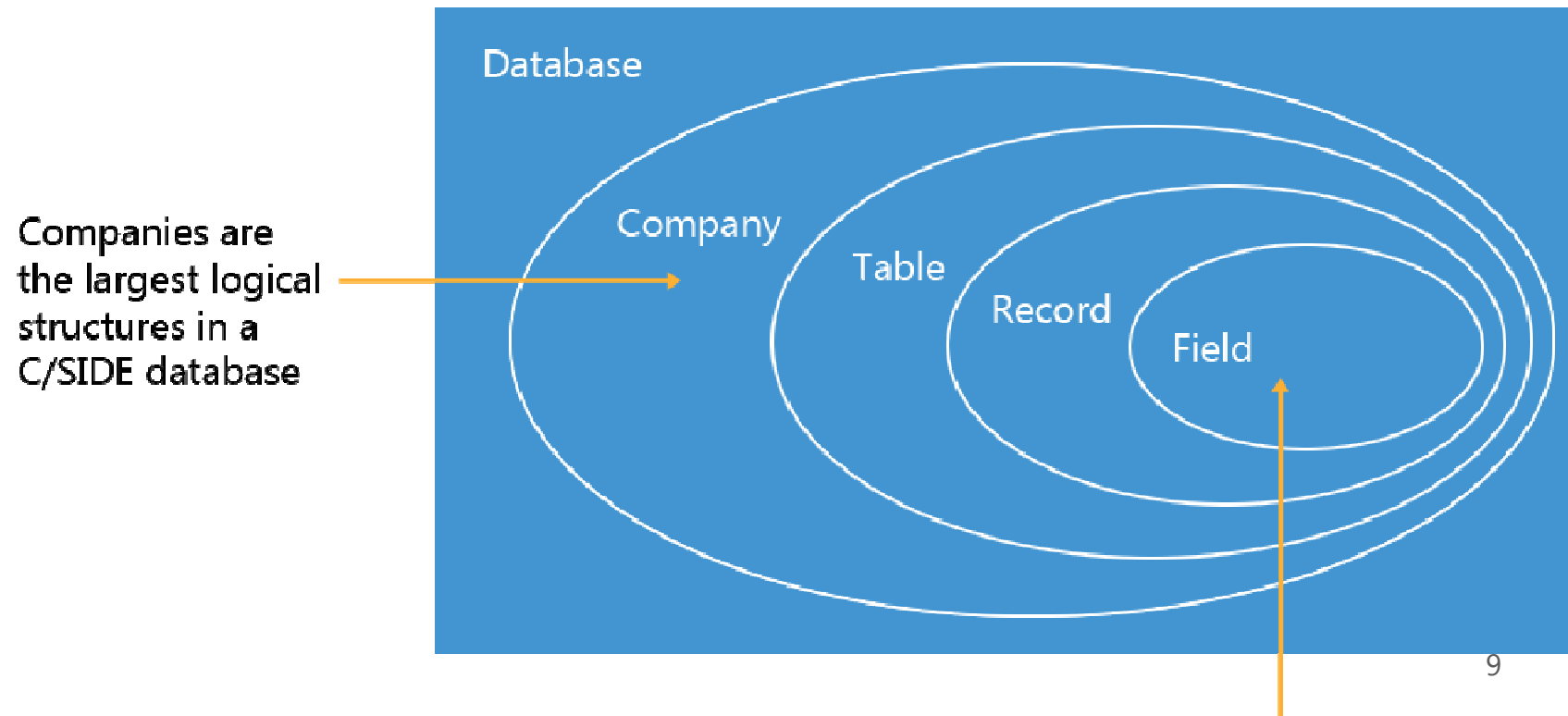
- Opening a database
- Managing objects
- Creating an object
- Editing an object
- Saving an object
- Compiling an object
- Properties
- Triggers
- Other Object Definition Elements

Lesson 3: Team Development Features

- Locking
- Unlocking
- Force Unlocking
- Auto-Locking

Lesson 4: Physical and the Logical Databases

- Fields
- Records
- Tables
- Companies



Lab 1.1: Designing and Running an Object

Scenario

Isaac is a developer for Cronus International Ltd. and is learning how to design objects in Microsoft Dynamics NAV Development Environment. He creates an empty page and saves it to test the functionality of the **Object Designer**.

This lab contains the following exercises:

- Accessing the Object Designer
- Creating an Object

Module 2

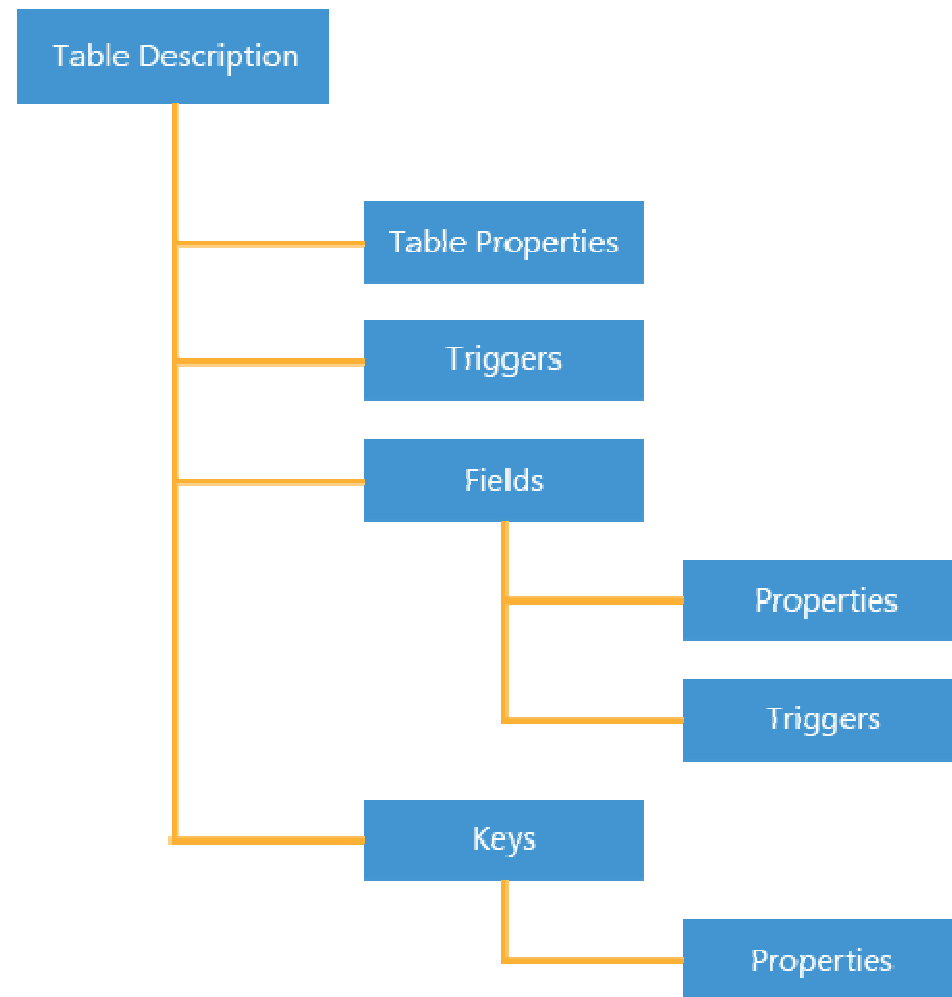
Tables

Module Overview

- Explain the concepts of tables and table components.
- Examine the concept behind primary and secondary keys, and explain how to set them.
- Create a simple table with primary and secondary keys, and add data to the table.
- Review the concept of table relation.
- Set table relations with a filter and condition.
- Describe the special table fields.
- Use special table fields to improve table features.

Lesson 1: Table Fundamentals

- Properties
- Triggers
- Fields
- Field Properties
- Field Triggers
- Keys
- Field Groups

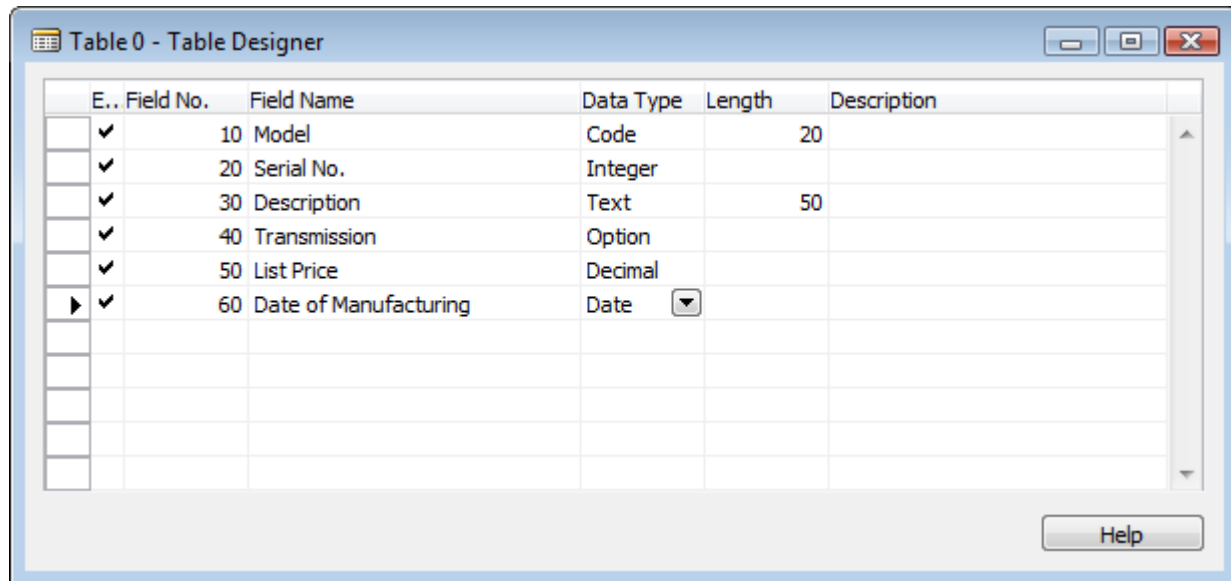


Lesson 2: Primary and Secondary Keys

- Primary Keys
- Secondary Keys
- A maximum of 40 keys can be associated to a table
- Primary Key = First key on the list
- Secondary Keys:
 - <> first key on the list
 - Optional

Lesson 2: Demonstration - Create a Simple Table

- Create table design
- Set the Primary Key
- Add data to the table
- Create a Secondary Key
- Use the Secondary Key



The screenshot shows a window titled "Table 0 - Table Designer". It contains a table with the following columns: "Field No.", "Field Name", "Data Type", "Length", and "Description". The table has 6 rows of data, each with a checkmark in the first column. The fields are: 10 Model (Code, Length 20), 20 Serial No. (Integer), 30 Description (Text, Length 50), 40 Transmission (Option), 50 List Price (Decimal), and 60 Date of Manufacturing (Date). A "Help" button is located at the bottom right of the window.

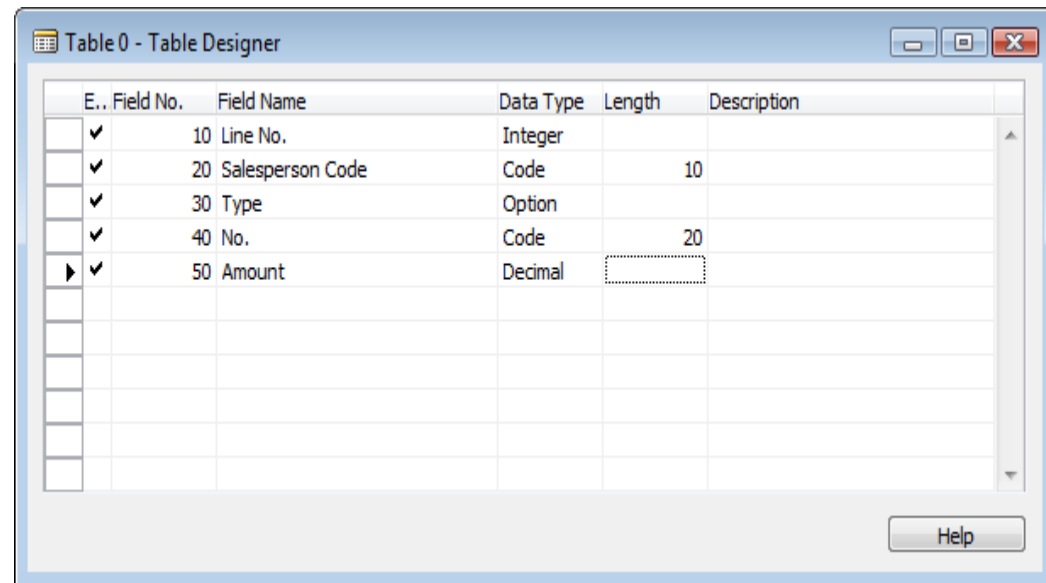
Field No.	Field Name	Data Type	Length	Description
✓ 10	Model	Code	20	
✓ 20	Serial No.	Integer		
✓ 30	Description	Text	50	
✓ 40	Transmission	Option		
✓ 50	List Price	Decimal		
▶ ✓ 60	Date of Manufacturing	Date		

Lesson 3: Table Relationships

- Three kinds of relationships
 - One-to-Many
 - Many-to-Many
 - One-to-One
- Table Relation Property
- Filter Table Relation
- Conditional Table Relation

Lesson 3: Demonstration - Set Table Relations

- Create a Table with Table Relations
- Set a Table Relation with a Filter
- Set a conditional Table Relation
- Test the Table Relation
- Test a Filter and a Conditional Table Relation



The screenshot shows a window titled "Table 0 - Table Designer". It contains a table with the following columns: "E.. Field No.", "Field Name", "Data Type", "Length", and "Description". The table has 5 rows of data, each with a checkmark in the "E.. Field No." column.

E.. Field No.	Field Name	Data Type	Length	Description
✓ 10	Line No.	Integer		
✓ 20	Salesperson Code	Code	10	
✓ 30	Type	Option		
✓ 40	No.	Code	20	
✓ 50	Amount	Decimal		

A "Help" button is located at the bottom right of the window.

Lesson 4: Special Table Fields

- Three kinds of specialized fields
 - SumIndexFields
 - FlowFields
 - FlowFilter Fields
- FlowFields
- Calculation Formulas and the CalcFormula property
- FlowFilter
- SumIndexFields
- Special Table Fields Example

Lesson 4: Demonstration - Use Special Table Fields

- Special Table Fields
 - Add Records
 - Create a FlowField
 - Create a SumindexField
 - Test the FlowField
 - Create a Flowfilter
 - Test the FlowFilter

Lab 2.1: Create a Table

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International Ltd. has decided to start selling Microsoft Dynamics NAV training courses as its business.

Simon must create a table to record course information and set several keys so that his users have the option for a different sorting sequence for the records in the table.

This lab contains the following exercises:

- Create Course Table
- Set OptionString property for the Type field
- Compile and Save
- Add records to the new table

Module 3

Pages

Module Overview

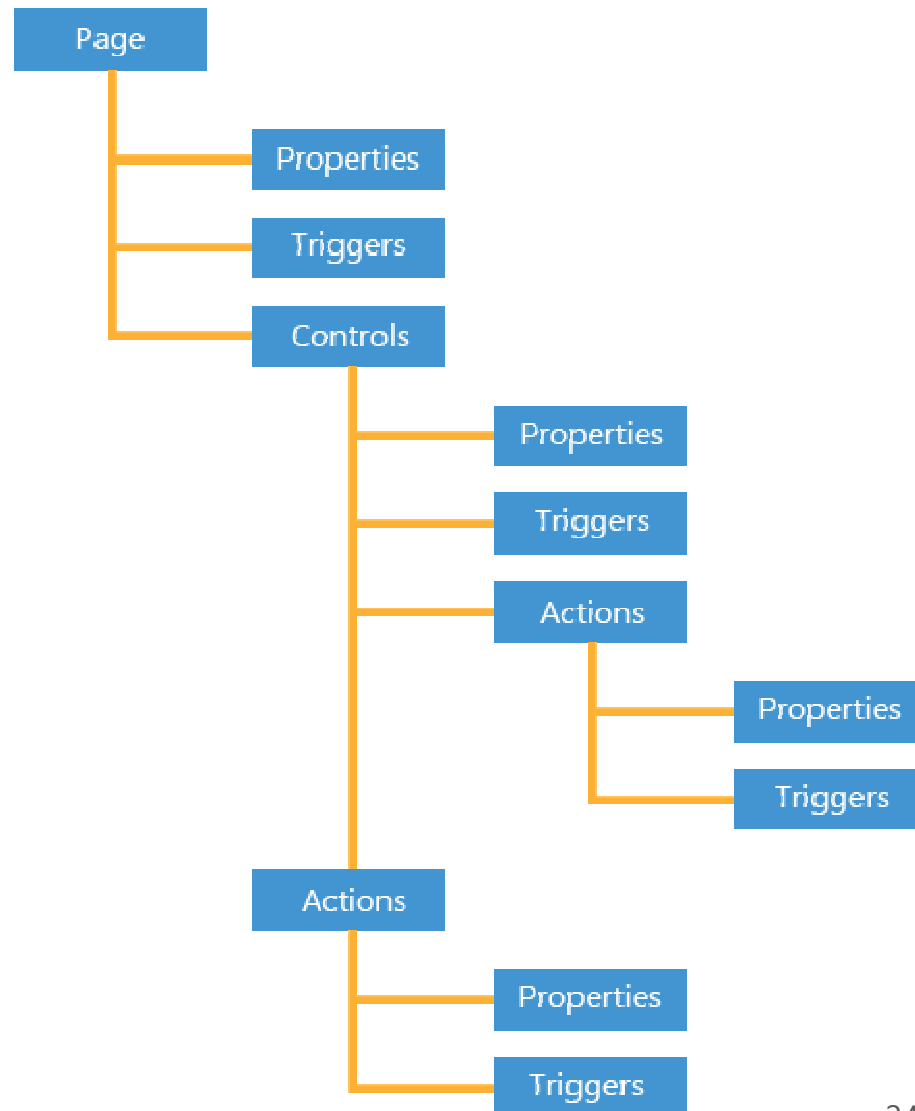
- Explain the concepts of pages and page components.
- Describe Page Designer and Action Designer.
- Create a simple page and add basic controls to the page.
- Provide an overview of different page types and their characteristics.
- Discuss best practices in designing pages.

Module Overview

- Create a Card page, add a container, FastTabs, and fields.
- Create a List page and link it to the Card page.
- Create a main page, a Part page, and link the two pages.

Lesson 1: Page Fundamentals

- Page
 - Properties
 - Triggers
 - Controls
- Controls
 - Properties
 - Triggers
 - Actions
- Actions
 - Properties
 - Triggers



Lesson 2: Page Designer

- Control Type and Subtype:

There are several types of controls that can be added to a page. The following list groups controls into several broad categories:

Type	Purpose
Container	Acts as a placeholder for other controls .
Group	Arranges other controls in a logical manner.
Field	Displays data from a table field, or a value of a variable, constant, or an expression.
Part	Displays another page, a chart, or a system-defined feature, such as Outlook or Record Links.

Lesson 2: Page Designer

- Linking Pages and Page Parts

Property	Purpose
SubPageView	Sets a specific table view that is applied to the page part. You can choose the key, the sorting order, and the table filter that are applied to the page part.
SubPageLink	Sets up a link between the main page and the page part. The link lets you filter the page part by field values on the main page, and to apply additional page filters.

Lesson 2: Page Designer

- Action Designer
 - Action Type
 - Action Container
 - Action
 - Action Group
- Running Objects
 - RunObject
 - RunPageView
 - RunPageLink
 - RunPageMode
- Preview Mode



Lesson 2: Demonstration - Explore the Page Designer

High Level Steps:

1. Create a blank page.
2. Select the card page type and specify **Item** as the source table.
3. Add a container to the page, and start the preview mode.
4. Add the group control for the **General** FastTab.
5. To the **General** FastTab, add field controls for **No.**, **Description**, **Costing Method**, **Profit %**, and **Picture** fields.
6. Set the properties on the **Profit %** field control to show the progress indicator.
7. Add a FixedLayout control that shows **Inventory**, **Qty. on Purch. Order**, **Qty. on Prod. Order**, **Qty. on Component Lines**, **Qty. on Sales Order**, and **Qty. on Service Order fields** in a single row.
8. Add an action to run the report 701, **Inventory – List**.
9. Add an action to run the page 158, **Item Turnover**.
10. Set the **Item Turnover** action property to show the turnover for the item that is currently selected in **Custom Page** page.
11. Compile, save and run the page 90001, **Custom Page**.

Lesson 3: Page Types and Characteristics

- Card Page
- List Page
- CardPart Page
- ListPart Page
- Worksheet Page
- Document Page
- RoleCenter Page
- ConfirmationDialog Page
- ListPlus Page
- StandardDialog Page

Lesson 3: Demonstration – Create a Simple Page

Demonstrations:

- Create a Card Page Using a Wizard
- Create a List Page Using a Wizard
- Create a Document Page Using a Wizard

Lesson 3: Page Types and Characteristics

- Page Design Best Practices:
 - Consider the types of users who use the new page and list the tasks that they perform.
 - Create a list of the fields, actions, and links that are needed for the page.
 - Select the page type that best matches the content that is displayed on the page.
 - Set the source table of the page to the table that contains the primary set of data that the page shows.
 - Design the page by using Page Designer to specify a page element. Determine the details of how data is displayed by adjusting the properties of each element on the page.

Lesson 3: Page Types and Characteristics

- Page Design Best Practices:
 - Consider any supplemental sources of information that add value to the page and add these as FactBoxes. Existing FactBoxes can support the design.
 - Simplify the user experience by streamlining the user's default view. This can be achieved by doing the following:
 - Promote actions to appropriate groups that have the appropriate sequence and size.
 - Set the Importance property to Additional for fields that are used less than 30 per cent of the time.
 - Display one to three FactBoxes. This is by default.

Lesson 3: Page Types and Characteristics

- Page Design Best Practices:
 - Add the new page to the menu suite so that it is included in the Departments location.
 - Consider specific user profiles if the page is a list location. These user profiles can have a link to the page from their role centers.

Lab 3.1: Create a Card and a List Page

This lab contains the following exercises:

Create a Card Page for the Course Table

Scenario

Simon wants to follow the standard recommendations for cards and lists. He first creates a card page, names it according to card page naming rules, and saves it.

Create a List Page for the Course Table

Scenario

Simon now creates the list page. He follows the list page naming standards, and guarantees that users cannot edit information in this page. He also makes sure that when a user double-clicks a course, the Course Card page displays the selected course. Simon also makes sure that **New**, **Edit**, and **View** actions work as expected in the client.

Module 4

Introduction to C/AL
Programming

Module Overview

- Describe the concepts and basic use of C/AL code elements.
- Describe the concepts of data types, simple data types and complex data types.
- Explain the concepts of identifiers, variables, and syntax.
- Explain the syntax of identifiers.
- Explain the scope of variables.

Module Overview

- Explain the initialization of variables.
- Create a simple codeunit to show how to define variables, assign data types, and investigate several default values that are initialized for several data types.

Lesson 1: C/AL Programming

- Code Statements and Triggers
 - Three types of triggers
 - Documentation triggers
 - Event triggers
 - Function triggers
- Accessing C/AL

Lesson 2: Intrinsic Data Types

- Simple Data Types
 - Numeric Data Types
 - Integer
 - BigInteger
 - Decimal
 - Option
 - Char
 - String Data Types
 - Text
 - Code
 - Boolean Data Types

Lesson 2: Intrinsic Data Types

- Date, Time, and DateTime Data Types
- Complex Data Types
 - BLOB
 - Record
 - Page
 - Codeunit
 - File
 - Dialog
 - Report
 - DateFormula
 - GUID

Lesson 2: Intrinsic Data Types

- TableFilter
- RecordRef
- RecordID
- FieldRef
- KeyRef
- Instream and Outstream
- Variant
- BigText

Lesson 3: Identifiers, Variables, and Syntax

- **Identifier:** name of a programming element
- **Variable:** location in memory where data is stored
- **Syntax:** grammatical rules for using identifiers

Lesson 4: Variable Scope

- **Variable Scope:** the context in a computer program in which a variable name or other identifier is valid and can be used.
 - Global Scope: available anywhere in an object.
 - Local Scope: only accessible in a single trigger in an object.
 - Variables cannot be accessed outside the object in which they are defined.
 - System defined: automatically defined and maintained by the system.
 - Variable initialization.

Lab 4.1: Investigate Data Types

Scenario

Isaac is a developer for CRONUS International Ltd. He has just learned how to use variables in C/AL, and now wants to practice declaring and using different types of variables. He also wants to learn how the variable values can be shown on screen, and what the initial (default) values are for various data types.

This lab contains the following exercises:

- Data Types
- Display the Variables

Module 5

Assignment Statements and
Expressions

Module Overview

- Explain the concepts of an assignment, a statement, and an assignment statement.
- Describe the syntax of statements and introduce the statement separator.
- Describe automatic type conversions for string, numeric, and other data types.
- Use assignment statements and the **Symbol** Menu.
- Understand the concepts of expressions, terms, and operators.

Module Overview

- Describe the syntax of an expression.
- Describe the string operator.
- Use the string operator.
- Describe the **MAXSTRLEN** and the **COPYSTR** functions.
- Use the **MAXSTRLEN** and the **COPYSTR** functions in an expression.
- Define numeric expressions, arithmetic operators, and operator precedence.
- Describe the arithmetic operators, and provide examples.

Module Overview

- Use the arithmetic operators and examine the operator precedence.
- Define relational and logical operators and expressions.
- Describe how to use relational expressions for comparison.
- Describe how to use relational expressions for set inclusion.
- Describe how to use logical expressions.
- Use logical and relational expressions in a page.

Lesson 1: Assignment Statements

- **Assignment Statement:** assigns a value to a variable.

Lesson 2: The Syntax of Statements

- Function Call
 - [return value :=] <Function Call>
- Assignment Statement
 - <variable> := <expression>
 - Assignment Operator: **:=**
 - Assignment Separator: **;**

Lesson 3: Automatic Type Conversions

- String Data Types
- Numeric Data Types
- Other Data Types

Lesson 4: Use Assignment and the Symbol Menu

- **C/AL Symbol** Menu: displays variables, functions, and objects that are defined in the **C/AL Globals** window. This menu provides information about the syntax and description of the variables, functions, and objects

Lesson 4: Demonstrations

- Create a Simple Assignment Statement
- Create Multiple Messages
- Use the Symbol Menu
- Set Char Constants
- Set Option Constants
- Compile-Time and Run-Time Errors

Lesson 5: Expressions, Terms, and Operators

- **Expression:** formula that tells the computer exactly how to generate a desired value.
- **Evaluation:** determine the expression's type and value
- **Term:** part of an expression that evaluates to a value
- **Operator:** part of an expression that acts upon:
 - Unary operator: the term directly following it
 - Binary operator: the terms on either side of it

Lesson 6: The String Operator

- String Operator:
 - Plus sign (+)
 - Concatenation
 - Binary

Lesson 7: Function Calls in Expressions

- MAXSTRLEN
- COPYSTR

Lesson 7: Demonstration - Use the MAXSTRLEN and COPYSTR functions

High Level Steps

1. Design codeunit 90001, **My Codeunit 2** from the Object Designer.
2. Define the following global variables: **MaxChar** of type Integer, and **Description** of type Text.
3. In the **OnRun** trigger, append the code that sets the **MaxChar** variable to the maximum length of the **Description** variable, and then shows the value of the **MaxChar** variable on the screen.
4. Under the last line of code, enter the code that does the following:
 - Assigns the result of the concatenation of the "The message is" and **CodeB** to the **Description** variable.
 - Displays the value of the **Description** variable on screen.
5. Compile, save, close, and run the codeunit.
6. Design codeunit 90001, **My Codeunit 2** from the Object Designer.
7. Change the line that assigns the **Description** variable so that it only assigns as many characters as **Description** can hold. Use the **COPYSTR** and **MAXSTRLEN** functions.
8. Compile, save, close, and run the codeunit.

Lesson 8: Numeric Expressions

- A *numeric expression* is an expression that results in a numeric value. The following are examples of numeric expressions:
 - $5 + 2 * 3$
 - $10 / 2$
- Operator Precedence

Lesson 9: Arithmetic Operators

- There are three levels of operator precedence used for arithmetic operators.
 - The highest level is the unary operator level. This includes both positive (+) and negative (-).
 - The second is the multiplicative operator level. This includes multiplication (*), both kinds of divides (/ , DIV) and modulus (MOD).
 - The lowest precedence level is the additive operator level. This includes both addition (+) and subtraction (-) binary operators.

Lesson 9: Arithmetic Operators

- Six arithmetic operators in C/AL:
 - Plus Operator (+)
 - Minus Operator (-)
 - Times Operator (*)
 - Divide Operator (/)
 - Integer Divide Operator (DIV)
 - Modulus Operator (MOD)

Lesson 9: Demonstrations

- Use the Arithmetic Operators
- Add Sub-Expressions

Lesson 10: Relational and Logical Expressions

- Relational Operator
 - Tests its relationship to the term before it, and to the term following it.
- The available relational operators are as follows:
 - = (equal to)
 - < (less than)
 - > (greater than)
 - <= (less than or equal to)
 - >= (greater than or equal to)
 - <> (not equal to)
 - **IN** (included in set) [12,15]

Lesson 11: Relational Expression for Comparison

- Numeric Comparisons
- String Comparisons
- Date and Time Comparisons
- Boolean Comparisons
- Relational Operator Precedence
 - The relational operators have the lowest precedence of any operator. Relational comparison is performed after the expressions on either side of the relational operator are evaluated.

Lesson 12: Relational Expression for Set Inclusion

- **Set**: a list of values
- The **IN** operator

Lesson 13: Logical Expressions

- Logical Operator Results

NOT	True	False
Results in:	False	True

OR	True	False
True	True	True
False	True	False

AND	True	False
True	True	False
False	False	False

XOR	True	False
True	False	True
False	True	False

Lesson 13: Logical Expressions

- Logical Operator Precedence

Type of Operator	Operator	Comments
Sub-expression or Terms	() [] . ::	Sub-expression in parentheses is evaluated first.
Unary	+ - NOT	Highest precedence in an expression.
Multiplicative	* / DIV MOD AND	
Additive	+ - OR XOR	
Relational	< <= = >= > <> IN	Lowest precedence in an expression.
Range	..	Used in Set Constants.

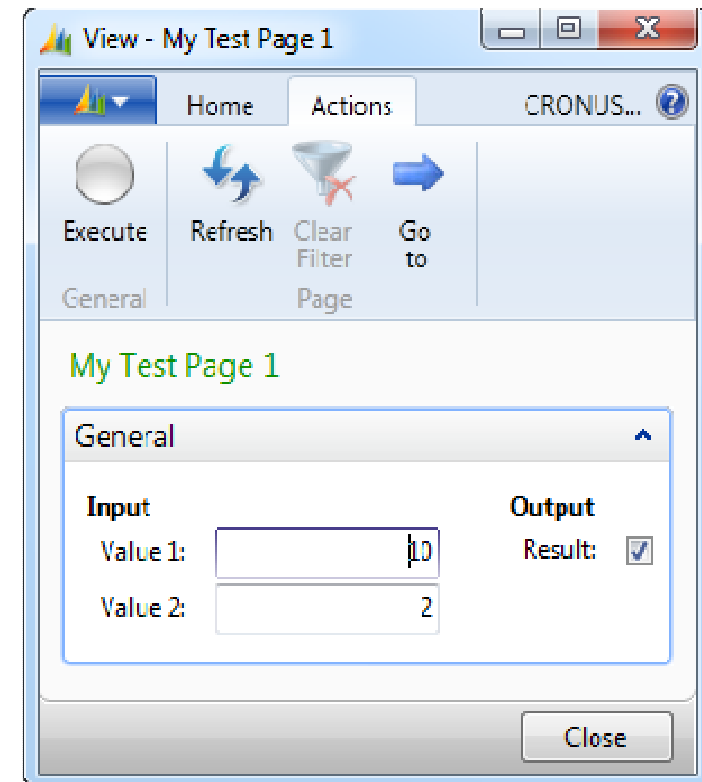
Lab 5.1: Use Logical and Relational Expressions in a Page

Scenario

Isaac is a developer at CRONUS International Ltd. He wants to test the newly acquired knowledge about logical operators. He wants to create a page, add several controls and an action to it, and write code that calculates a Boolean variable through a relational expression.

This lab contains the following exercise:

- Create a New Page



Module 6

C/AL Statements

Module Overview

- Define conditional statements and Boolean expressions.
- Describe the IF statement, the IF-THEN, and IF-THEN-ELSE syntax.
- Describe the EXIT statement and code indentation.
- Describe the CASE statement and the syntax of the CASE statement.
- Define compound statements and comments.
- Describe the syntax of compound statements with BEGIN and END.

Module Overview

- Understand the concepts of nested IF statements and the rule of correct indentation.
- Describe the syntax of comments.
- Use the IF, EXIT, CASE, and compound statements in a page.
- Test knowledge about C/AL statements.
- Define arrays and describe the components of arrays.
- Describe the syntax of arrays.
- Explain the power of arrays.

Module Overview

- Describe how to use strings as arrays of characters.
- Introduce repetitive statements that are available in C/AL.
- Use arrays and repetitive statements in a page.
- Describe the WITH statement, record variables, and the syntax of the WITH statement.

Lesson 1: Conditional Statements and Boolean Expressions

- **Conditional Statement**

A conditional statement tests a condition and executes one or several other statements based on the condition. The most frequently used conditional statement is the **IF** statement. Use the **IF** statement when there are only two possible values for the condition: **TRUE** or **FALSE**.

- **Boolean Expression**

A Boolean expression results in a Boolean value, such as a Boolean variable or constant, a relational expression, or a logical expression.

Lesson 2: The IF Statement

- Use the **IF** statement to execute one or several statements if the condition is **TRUE**, and one or several other statements if the condition is **FALSE**.
- IF <Boolean expression> THEN
 <statement>;
- IF <Boolean expression> THEN
 <statement 1>
ELSE
 <statement 2>;

Lesson 3: The EXIT Statement

- When an **EXIT** statement executes, the development environment exits the current trigger, back to the object that calls the trigger, if any, or back to the user.

Lesson 4: The CASE Statement

- The **CASE** statement is also a conditional statement. Use the **CASE** statement when there are more than two possible values for the condition.

Lesson 5: Compound Statements and Comments

- **A compound statement** consists of multiple statements. It improves the capabilities of the conditional statement.
- For example, in an **IF** statement, when a condition tests to **TRUE**, use a compound statement after the **THEN** to perform two or more assignments.
- Usually, after the **THEN** statement, only one statement is executed. However, if that statement is a compound statement, multiple assignment statements can be performed in it.

Lesson 5: Compound Statements and Comments

- A comment is a description that developers write in their code. Following are two ways to identify comments:
 - Two slash marks `//`
 - a set of braces `{}`

Lesson 6: The Syntax of Compound Statements

- Example

- `IF Quantity <> 0 THEN BEGIN`
 UnitPrice := ExtendedPrice / Quantity;
 TotalPrice := TotalPrice + ExtendedPrice;
 `END;`

The statements in a compound statement are indented two spaces from the word BEGIN and the word END. BEGIN and END are aligned with one another. See the following example.

- `IF Quantity <> 0 THEN`
 `BEGIN`
 UnitPrice := ExtendedPrice / Quantity;
 TotalPrice := TotalPrice + ExtendedPrice;
 `END;`

Lesson 7: Compound Statement by using Nested IF Statements

```
IF Amount <> 0 THEN
  IF Amount > 0 THEN
    Sales := Sales + Amount
  ELSE
    IF Reason = Reason::Return THEN
      IF ReasonForReturn = ReasonForReturn::Defective THEN
        Refund := Refund + Amount
      ELSE
        Credits := Credits + Amount
      ELSE
        Sales := Sales - Amount;
```

Lesson 7: Demonstrations

- Use the Conditional and Compound Statements in a Page.

The screenshot shows a software window titled "View - Test Statements Page" with a ribbon interface. The "Actions" tab is active, displaying buttons for "Execute IF", "Clear", "Refresh", "Clear Filter", and "Go to". Below the ribbon, the "Test Statements Page" section is visible, containing a "General" tab. Under the "General" tab, there are two columns: "Input" and "Output".

Input		Output	
Quantity:	5	Result:	1250
Unit Price:	250	Total Sales:	1250
		Total Credits:	0
		Grand Total:	1250

A "Close" button is located at the bottom right of the window.

Lesson 8: The Syntax of Comments

- Single-Line Comment
- Block of Comments
- Nested Comments

Lesson 9: Practice - Nested IF

- The IF and ELSE Statement

Lab 6.1: Use Conditional and Compound Statements

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International has decided to start selling Microsoft Dynamics NAV training courses as its business.

Simon has already created a **Course** table to store course information, a **Course Card** page, and a **Course List** page to enter and display course information. Now Simon must add several extension fields to make the list page easier to use.

Lesson 10: Arrays

- **Arrays** are *complex variables* that contain a group of variables, with the same data type. They are special variables that have more functionalities than other variables that we have discussed.
- **Components of Arrays:**
 - Identifier
 - Datatype
 - Elements
 - Index
 - Dimension

Lesson 10: Arrays

5	27	8	17	25	3	7	12
Box[1]	Box[2]	Box[3]	Box[4]	Box[5]	Box[6]	Box[7]	Box[8]

Box[1,c]	1	2	3	4	5	6
Box[2,c]	2	4	6	8	10	12
Box[3,c]	3	6	9	12	15	18
Box[4,c]	4	8	12	16	20	24
Box[5,c]	5	10	15	20	25	30
	Box[r,1]	Box[r,2]	Box[r,3]	Box[r,4]	Box[r,5]	Box[r,6]

Lesson 11: The Syntax of Arrays

- Variable Syntax
- Array Element Syntax
- How to think about arrays

Lesson 12: The Power of Arrays

- The true power of an array is that the index value can be an expression.
- ARRAYLEN

Lesson 13: Strings as Arrays of Characters

- A string variable, or variable of type *Text* or *Code*, can be thought of as an array of characters. Because of this, C/AL allows for access to each character as an element in an array.
- Think of a string variable, or variables of type *Text* or *Code* as an array of characters. Because of this character component, C/AL allows access to each character as an element in an array.

Lesson 14: Repetitive Statements

- A repetitive statement enables execution of one or more statements multiple times.

Lesson 14: Repetitive Statements

- Examples:
 - The FOR Statement
 - FOR ...TO
 - FOR... DOWNTO
 - The WHILE ... DO
 - The REPEAT...UNTIL

Lesson 14: Demonstration

- Use Arrays and Repetitive Statements

Lesson 15: The WITH Statement

- Use the **WITH** statement to ease coding with record variables.
- WITH <record variable> DO <statement>
- Implied WITH Statements
 - REC
 - XREC

Module 7

C/AL Functions

Module Overview

- Explain the concepts of functions and parameters.
- Explain the **C/AL Symbol** Menu.
- Describe the use and syntax of data access, filtering, and manipulation functions.
- Describe the use and syntax of user interaction functions.
- Describe the use and syntax of string functions.
- Describe the use and syntax of system functions.

Module Overview

- Describe the use and syntax of date functions.
- Describe the use and syntax of number functions.
- Describe the use and syntax of array functions.
- Describe the use and syntax of several other important functions.
- Provide an overview of the benefits of creating custom functions.

Module Overview

- Explain the concepts of local functions and local variables.
- Create custom functions in a page and call the functions from Actions.

Lesson 1: Functions and Parameters

- **Functions** are a fundamental programming concept.
- A function is a named part of a program, also known as a subprogram or a subroutine.
- When code that is running reaches a function, the main application pauses while the function code is handled.
- Functions call in an expression.
- Function call statements.

Lesson 1: Functions and Parameters

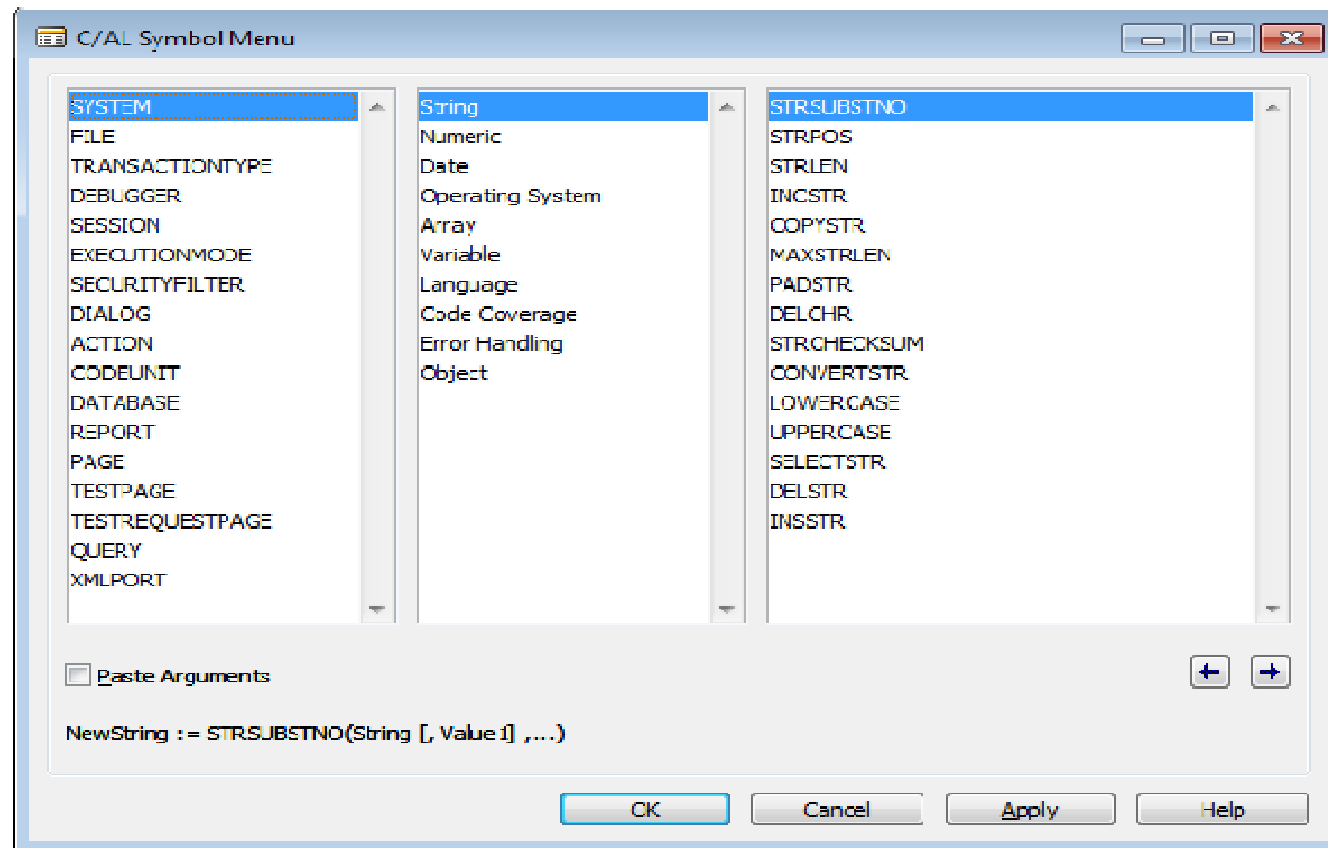
- **Parameters** are variables or expressions that are sent to the function through the function call.
- Parameters provide information to the function. The function can change that information.
- If a function has parameters, the function identifier has a set of parentheses that follows it. Within these parentheses are one or more parameters.
- If there is more than one parameter, the parameters are separated by commas.

Lesson 1: Functions and Parameters

- **Parameter pass**
 - Pass by Value
 - Pass by Reference
- **Formal and Actual Parameters**

Lesson 2: Review Built-in Functions

- C/AL Symbol Menu



Lesson 2: Review Built-in Functions

- MESSAGE
- ERROR
- DATE2DMY

Lesson 2: Demonstration - Use the DATE2DMY Function

High Level Steps

1. Create a new codeunit, and save it as codeunit **90003, My Codeunit4**.
2. Define variables.
3. Type the code in the **OnRun** trigger that displays a message with the name of the current month.
4. Compile, save, and then close the codeunit.
5. Run the codeunit and examine the result.

Lesson 3: Data Access Functions

- GET
- FIND
- FINDFIRST
- FINDLAST
- FINDSET
- NEXT

Lesson 4: Sorting and Filtering Functions

- SETCURRENTKEY
- SETRANGE
- SETFILTER
- GETRANGEMIN
- GETRANGEMAX

Lesson 5: Data Manipulation Functions

- INSERT
- MODIFYALL
- RENAME
- DELETE
- DELETEALL

Lesson 6: Working with Fields

- CALCFIELDS
- SETAUTOCALCFIELDS
- CALCSUMS
- FIELDERROR
- FIELDCAPTION
- INIT
- TESTFIELD
- VALIDATE

Lesson 7: Using Interaction Functions

- MESSAGE
- CONFIRM
- STRMENU
- ERROR

Lesson 8: Other Common C/AL Functions

- **STRING FUNCTIONS**

- STRPOS
- COPYSTR
- STRLEN
- MAXSTRLEN
- LOWERCASE
- UPPERCASE
- INCSTR
- SELECTSTR

Lesson 8: Other Common C/AL Functions

- **Date Functions**
 - DATE2DMY
 - DATE2DWY
 - CALCDATE
 - NORMALDATE
 - CLOSINGDATE

Lesson 8: Other Common C/AL Functions

- **Numeric Functions**
 - ROUND
 - ABS
 - POWER
 - RANDOM
 - RANDOMIZE

Lesson 8: Other Common C/AL Functions

- **Array Functions**
 - ARRAYLEN
 - COMPRESSARRAY
 - COPYARRAY

Lesson 8: Other Common C/AL Functions

- **Stream Functions**
 - CREATEINSTREAM
 - CREATEOUTSTREAM
 - READ
 - READTEXT
 - WRITE
 - WRITETEXT
 - EOS
 - COPYSTREAM

Lesson 8: Other Common C/AL Functions

- **System Functions**
 - USERID
 - COMPANYNAME
 - TODAY
 - TIME
 - WORKDATE
 - TEMPORARYPATH
 - GUIALLOWED
 - GLOBALLANGUAGE

Lesson 8: Other Common C/AL Functions

- **Other Functions**
 - EXIT
 - CLEAR
 - CLEARALL
 - EVALUATE
 - FORMAT

Lesson 9: Create Custom Functions

- Reasons to create custom functions:
 - To organize the program.
 - To create self-explanatory code.
 - To simplify development process.
 - To make code reusable, which reduces work.
 - To reduce errors.
 - To make modifications easier.
 - To isolate data.
 - To reduce the size of the objects.

Lesson 10: Local Functions, Variables, and the EXIT Statement

- Local Function
- Local Variable
- The Exit Statement

Lab 7.1: Create Custom Functions

Scenario

Viktor is a business systems developer for Cronus International Ltd. He is learning how to develop customizations for Microsoft Dynamics NAV 2013, and how to program in C/AL to customize solutions for his customers.

He wants to customize the **Customer Card** page by adding an action that adjusts a customer's credit limit based on the customer's previous sales history, or to reset the limit to zero if there were no transactions.

The screenshot shows the 'Edit - Customer Card - 30000 - John Haddock Insurance Co.' window in Microsoft Dynamics NAV 2013. The window has a ribbon with tabs: Home, Actions, Navigate, and Report. The 'Home' tab is active, showing various icons for actions like Sales Invoice, Sales Order, Reminder, View, Edit, New, Delete, Update Credit Limit, Apply Template, Cash Receipt Journal, Sales Journal, Customer - Balance to Date Report, OneNote, Notes, and Links. The main area is divided into several sections:

- General**: Contains fields for No. (30000), Name (John Haddock Insurance Co.), Address (10 High Tower Green), Address 2, Post Code (M02 4RT), City (Manchester), Country/Region Code (GB), Phone No., Primary Contact No., Contact (Miss Patricia Doyle), Search Name (JOHN HADDOCK INSURAN...), Balance (LCY) (362,365.40), Credit Limit (LCY) (0.00), Salesperson Code (PS), Responsibility Center, Service Zone Code (N), Blocked, and Last Date Modified (30.5.2012).
- Communication**: Contains fields for Phone No., Fax No., E-Mail (john.haddock.insurance...), Home Page, and IC Partner Code.
- Invoicing**: Contains a dropdown menu with options NATIONAL, DOMESTIC, and LARGE ACC.
- Payments**: Contains a dropdown menu with options CM, DOMESTIC, and 1.5 DOM.
- Sell-to Customer Sales ...**: A summary section showing various statistics for the customer.
- Customer Statistics - Bill...**: A summary section showing various statistics for the customer.

The 'Sell-to Customer Sales ...' section shows the following data:

Customer No.:	30000
Quotes:	0
Blanket Orders:	0
Orders:	6
Invoices:	0
Return Orders:	0
Credit Memos:	0
Pstd. Shipments:	6
Pstd. Invoices:	3
Pstd. Return Rece...	0
Pstd. Credit Mem...	0

The 'Customer Statistics - Bill...' section shows the following data:

Customer No.:	30000
Balance (LCY):	362,365.40
Sales	
Outstanding Ord...	19,502.16
Shipped Not Inv...	1,996.90
Outstanding Inv...	0.00
Service	
Outstanding Ser...	10.65
Serv Shipped No...	0.00

The window also has an 'OK' button at the bottom right.

Lab 7.1: Create Custom Functions (cont.)

Scenario (cont.)

The requirements state the following:

- A customer's credit limit may not exceed 50% of total sales revenue for the customer in the past twelve months.
- Customer's credit limit must always be rounded to the nearest 10,000.
- If the credit limit is rounded during the process, the application must send notification to the user.
- If the new credit limit does not differ from the old one, the application must send notification to the user.
- If the customer has no sales history over the past twelve months, the credit limit must be reset to zero.
- The function that sets the credit limit must be available to other objects.
- The **Customer Card** page must include an action that calls the function.
- If the function is called from the page action, and credit limit would be increased, the function must ask the user for confirmation before actually updating the credit limit.

Module 08

Reports

Module Overview

- Explain the concepts of reports and report components.
- Provide an overview of different report types and their characteristics.
- Describe the difference between the logical and the visual design of reports and introduce Report Designer.
- Describe the logical design of a report.
- Create the data model for a new report by defining data items in the Report Dataset Designer.

Module Overview

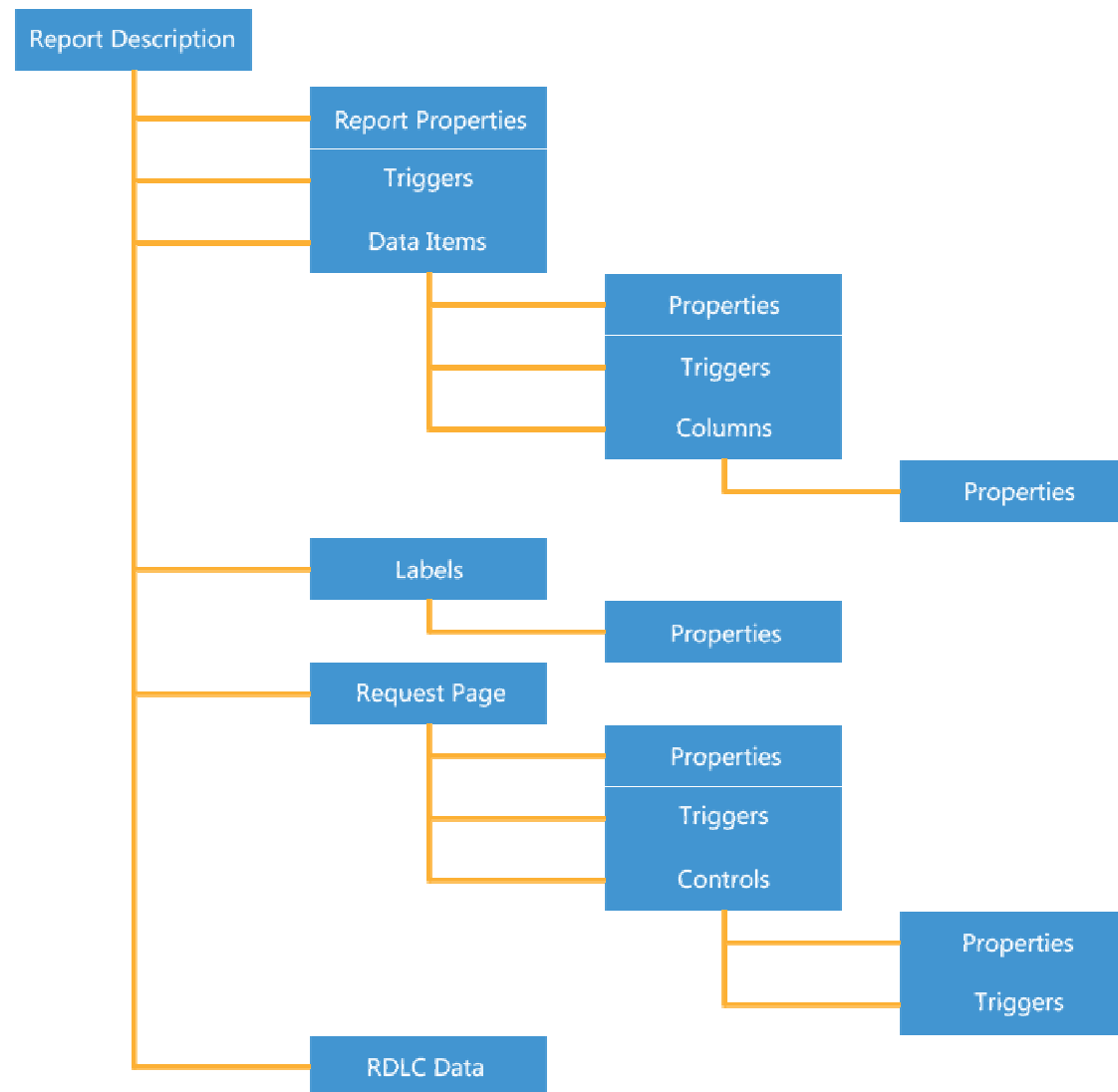
- Describe the visual design of a report and introduce Microsoft Visual Studio Report Designer.
- Design the report layout.
- Introduce Request Page Designer.
- Design the **Request Options** page.
- Explain the concepts of grouping and totaling in a report.
- Create a grouping and totaling for a report.
- Add advanced features to a report.

Lesson 1: Report Fundamentals

- In Microsoft Dynamics® NAV, reports have several purposes. They are as follows:
 - Structure and print information from a database.
 - Print and display documents in the application.
 - Automate many recurring tasks.

Lesson 1: Report Fundamentals

- Report Components



Lesson 2: Report Design Process

- The report design process is divided into two distinct phases that reflect different aspects of creating a report. They are as follows:
 - Defining the logical structure (the data model)
 - Designing the visual element

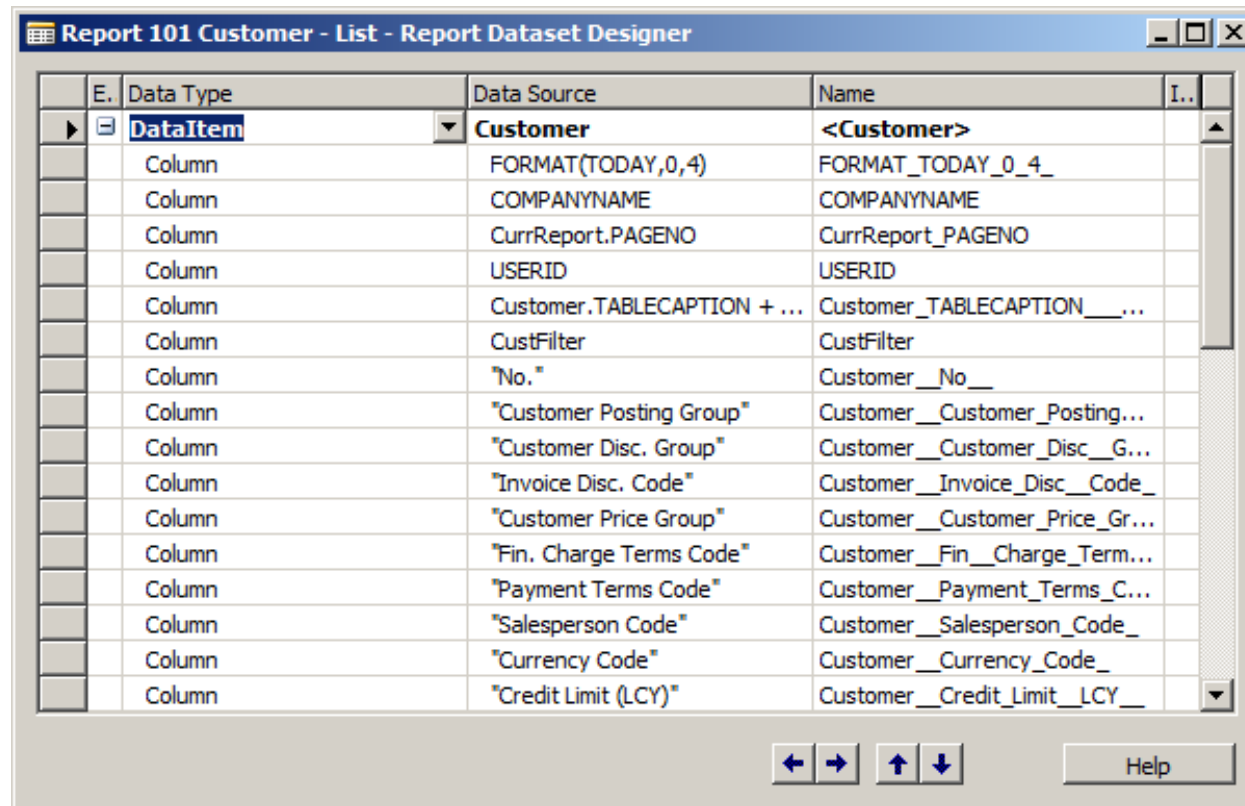
Lesson 2: Demonstration - Report Dataset Designer

High Level Step

1. Open Report Dataset Designer

Lesson 3: Design the Data Model

- Report Dataset Designer



Lesson 3: Design the Data Model

- Ordering and Indenting Data Items
- Linking a Data Item
- Adding columns

Lesson 4: Demonstration - Creating a Data Model

- Demonstrations:
 - Prepare the Customer Table.
 - Create the Report Data Model.
 - Add columns and captions to the data model.

Lesson 5: Designing the Layout

- **Microsoft Visual Studio Report Designer**
Microsoft Visual Studio Report Designer is an external designer that is used to design the visual element (the RDL data) of a report. It is accessed from the Report Dataset Designer.

Lesson 5: Demonstration - Design the Layout

High Level Steps

1. Design the Layout in Microsoft Visual Studio Report Designer
2. Add Column Headers
3. Add a Grouping:
A dataset result in **Visual Studio Report Designer** is flattened data.
4. Add Fields to the Report Layout.
5. Run the Report in the RoleTailored Client
6. Print
7. Print Header on every Page
8. Show New Record in a New Page

Lesson 6: The Request Page Designer

- The **Request** page is displayed when a report runs. The **Request** page lets the user provide additional input before previewing or printing a report.

Lesson 7: Demonstration - Design the Request Options Page

- Demonstrations:
 - Add and Remove FastTabs on the Request Page.
 - Create Options FastTab and Add a Check Box
 - Skip Lines in a Report

Lesson 8: Demonstration - Grouping and Totaling

- **Totaling the report** usually requires adding one or more fields that are displayed or read by the report, and then displaying the sum.
- **Subtotals** are shown after a logical grouping of records.
- **Grand totals** are shown at the end of the report.
- Subtotals sum the group; grand totals sum the whole report. This means grand totals sum all subtotals.

Lesson 9: Demonstration - Add Advanced Features

- Demonstrations
 - Using FlowFilters
 - Adding Interactive Sorting
 - Adding Hide/Collapse
 - Adding a Chart

Lab 8.1: Creating a Basic Report

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International has decided to sell Microsoft Dynamics NAV training courses as its business.

Simon has already created a **Course** table to record course information. Now, Simon must create a report to list all available courses.

This lab contains the following exercise:

- Build the report

Module 9

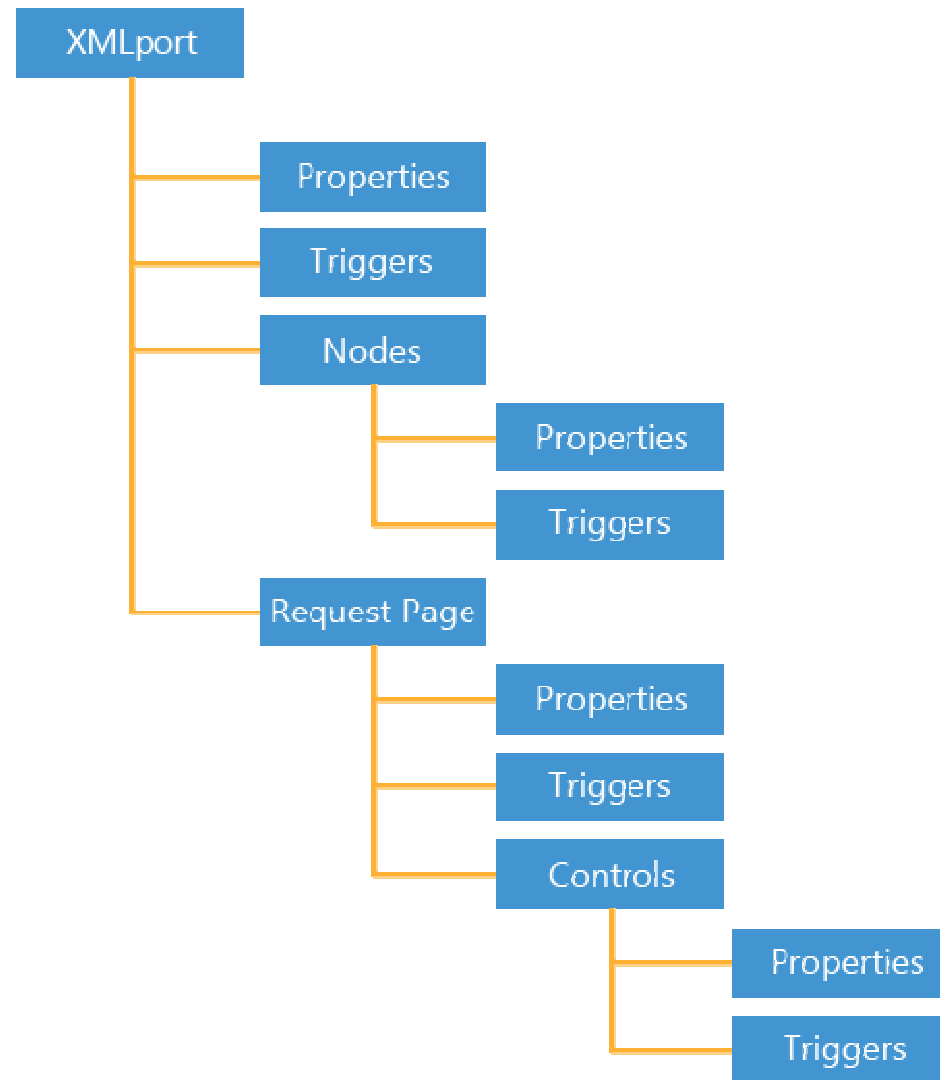
XMLPorts

Module Overview

- Describe the fundamentals of an XMLport and its components.
- Review how to design XMLports.
- Explain the Request page functionality.
- Describe how to use XMLports from C/AL code.
- Create XMLports for export and import with XML format.
- Create XMLports for export and import with fixed and a variable text format.

Lesson 1: XMLPort Fundamentals

- Components of an XMLport



Lesson 2: Design XMLports

- Designing an XMLport is actually designing the structure of the external file, whether it is an XML document or a text file.
- Data Model
- External File

Lesson 2: Design XMLports

- XML Properties
 - Direction
 - DefaultFieldsValidation
 - FileName
 - UseRequestPage
- Node Properties
 - Indentation
 - NodeName
 - NodeType
 - SourceType
 - SourceTable

Lesson 2: Design XMLports

- SourceField
- TextType
- FieldValidate
- VariableName
- SourceTableView
- CalcFields
- AutoCalcField
- LinkTable
- LinkTableForceInsert
- LinkFields
- Temporary

Lesson 2: Design XMLports

- MinOccurs
- MaxOccurs
- Occurrence

Lesson 2: Design XMLports

- Node Properties for Import

Property Name	Remarks
AutoSave	Specifies whether imported records are automatically written to the table.
AutoUpdate	Specifies whether a record in the database that has the same primary key as a record in the import file is updated with values from the imported record. Values of database fields that are not present in the import file will not be modified by the XMLport.
AutoReplace	Sets whether imported records automatically replace existing records with the same primary key. Values of database fields that are not present in the import file will be reset to their initial value, according to their InitValue property.

Lesson 2: Demonstrations

- Create an XMLport to Export to XML Documents.
- Create an XMLport to Import from XML Documents.

Lab 9.1: Create an XMLport to Export XML Data

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International has decided to start selling Microsoft Dynamics NAV training courses as its business. Simon has already created a Course table to record course information and Card and List pages to interface with the Course table. Now, Simon must create an XMLport to export all the course details from the **Course List** page in the XML format.

This lab contains the following exercise:

- Create an XMLport for export to the XML document.

Lesson 3: Importing and Exporting Plain Text

- Variable Text and Fixed Width Formats
- Properties for Handling Text Files:
 - Format
 - Width
 - FieldDelimiter
 - FieldSeparator
 - RecordSeparator
 - TableSeparator

Lesson 3: Demonstrations

- Create XMLports to Export Variable Text.
- Run an XMLport from a Page.

Lab 9.2: Create an XMLport to Export Variable Text

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International has decided to start selling Microsoft Dynamics NAV training courses as its business.

Simon has already created a Course table to record course information, Card and List pages to interface with the Course table, and an XMLport to export the course details in XML format. Some customers of CRONUS International are using legacy systems that cannot handle XML data. Simon must create an XMLport to export all the course details from the **Course List** page in the comma-separated text format.

Lesson 4: Using XMLports in C/AL Code

- Importing and Exporting Data through C/AL Code

Function or Property	Remark
IMPORTFILE	Specifies whether the XMLport will import the data or export the data.
FILENAME	Specifies the source or the destination file for the XMLport.
RUN	Runs the XMLport object.

Lesson 4: Using XMLports in C/AL Code

- Streaming Data by Using XMLports

Function	Remarks
SETSOURCE	Specifies the inStream variable to be used as the source for the data import.
SETDESTINATION	Specifies the OutStream variable to be used as the destination for the data export.
IMPORT	Imports the data from the designated variable by using the SETSOURCE function.
EXPORT	Exports the data into the designated variable by using the SETDESTINATION function.

Lesson 4: Demonstrations

- Create a Codeunit to Run the XMLport for Export.
- Create an XML Input File.
- Create a Codeunit to Run an XMLport for Import.

Module 10

Codeunits

Module Overview

- Explain the concepts of codeunits.
- Provide an overview of designing codeunits.
- Provide an overview by using codeunits.
- Define variables and functions in a codeunit.
- Use the SMTP Mail codeunit.

Lesson 1: Codeunit Fundamentals

- Functions
- Local Variables
- Global Variables
- Temporary Tables
- Triggers
- A codeunit can be run by using `<CodeunitVariableName>.RUN`.
- When a codeunit is run, the code that is written in the OnRun trigger is executed.

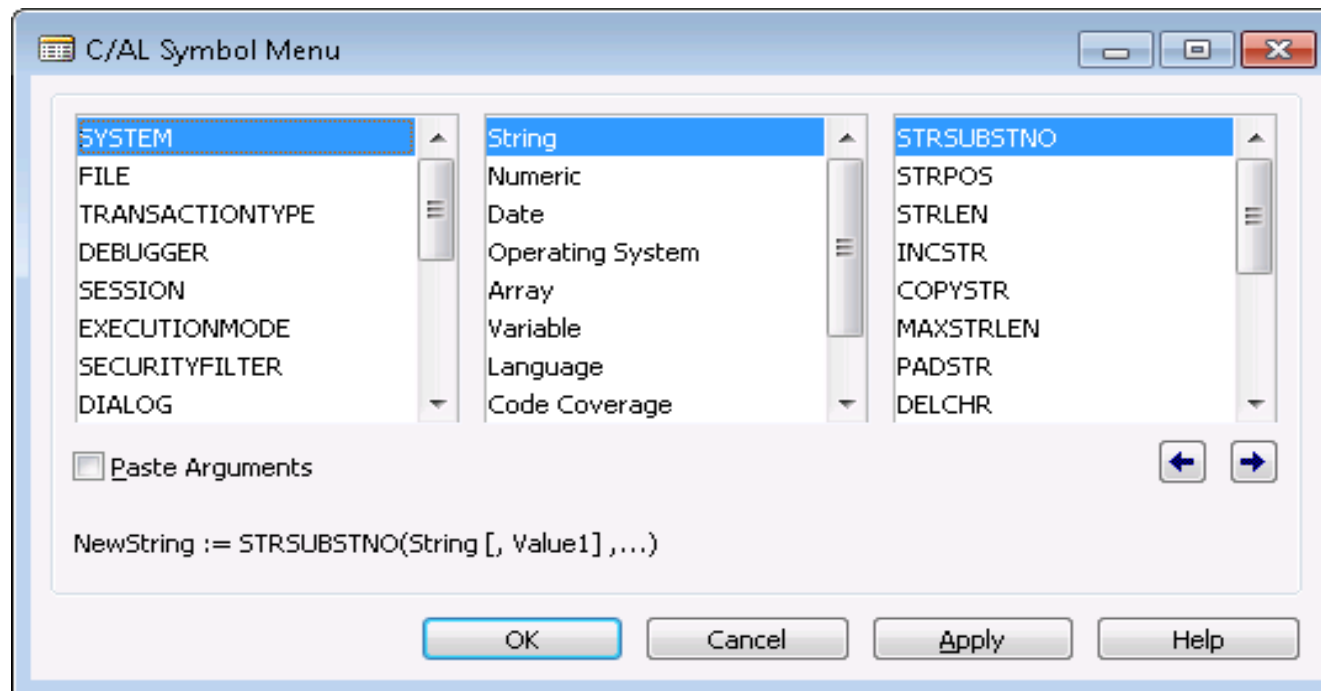
Lesson 2: Design Codeunits

- Use the C/AL Editor

Shortcut Key	Description
CTRL + X	Cut the selected text to the clipboard.
CTRL + C	Copy the selected text to the clipboard.
CTRL + V	Paste the text on the clipboard into the codeunit in the cursor position.
CTRL + F	Open the Find window and search for text.

Lesson 2: Design Codeunits

- Use the C/AL Symbol Menu

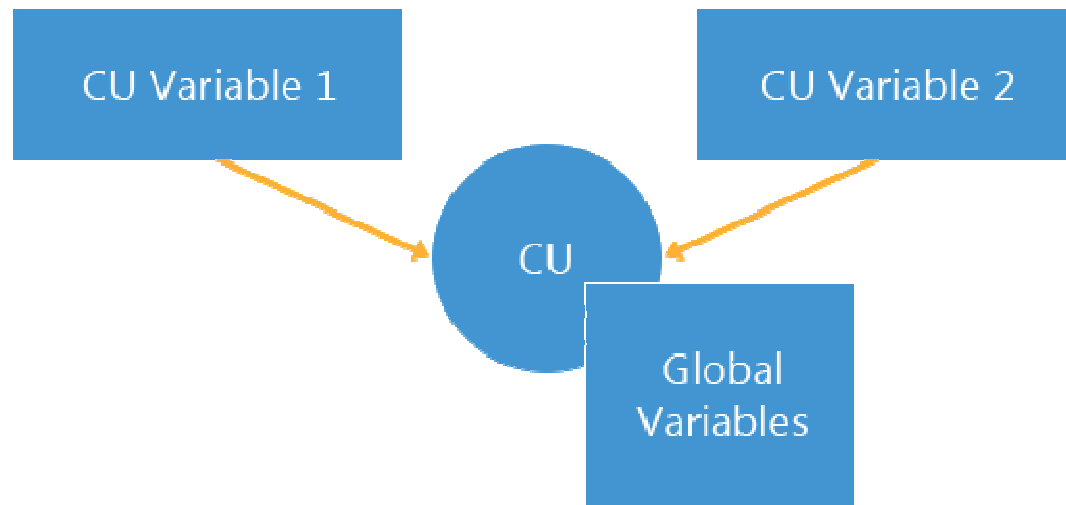


Lesson 3: Use Codeunits

- `<CodeunitVariableName>.<FunctionName>`
- `[{Ok} := CODEUNIT.RUN(Number [, Record])`

Lesson 3: Use Codeunits

- Accessing Codeunits through a Variable
- Codeunit Variable Assignment



Lesson 3: Use Codeunits

- CLEAR
- CLEARALL
- Single Instance Codeunits
- Limitations

Lesson 3: Demonstration - Define and Use a Codeunit from C/AL Code

Lesson 4: SMTP

- One of the features of Microsoft Dynamics NAV 2013 is the ability to send email by using Simple Mail Transfer Protocol (SMTP).

Lesson 4: SMTP

- SMTP Mail Setup

Edit - SMTP Mail Setup

Home Actions CRONUS International...

View Edit OneNote Notes Links

Manage Show Attached

SMTP Mail Setup

General

SMTP Server: smtp.cronus.net UserID:

SMTP Server Port: 25 Password:

Authentication: Anonymous Secure Connection: ☐

OK

Lesson 4: SMTP

- The **SMTP Mail** codeunit functions:
 - CreateMessage
 - TrySend
 - Send
 - AddRecipients
 - AddCC
 - AddBCC
 - AppendBody
 - AddAttachment
 - AddAttachmentStream
 - CheckValidEmailAddresses
 - CheckValidEmailAddress
 - GetLastSendMailErrorText
 - GetSMTPMessage

Module 11

Microsoft .NET Framework
Interoperability

Module Overview

- Explain the .NET Interoperability features.
- Describe the concept of constructors.
- Explain the communication between client-side and server-side objects.
- Describe how to respond to events that are raised by .NET objects.
- Examine mapping between C/AL and .NET data types.

Module Overview

- Review the most important C/AL functions for managing .NET objects.
- Use arrays, collections, and enumerations.
- Explain how to stream data between C/AL and .NET objects.

Lesson 1: The DotNet Data Type

- The .NET Framework interoperability is achieved through the DotNet C/AL data type.
- By declaring a variable of DotNet data type, and subtyping it to a specific .NET Framework class, you can access all functionality of the referenced class.

Lesson 1: The DotNet Data Type

- The DotNet variable enables you to do the following:
 - Access a specific .NET Framework class and its members.
 - Respond to events that are raised by the referenced .NET Framework class.
 - Target the Microsoft Dynamics NAV 2013 Server or the RoleTailored client.

Lesson 1: The DotNet Data Type

- Declaring A DotNet Variable
- Assembly List

Tab	Description
Dynamics NAV	Assemblies located in the local Add-ins folder of your Microsoft Dynamics NAV 2013 RoleTailored Client installation are listed in the Dynamics NAV tab.
.NET	Assemblies installed in the Global Assembly Cache (GAC) are listed in the .NET tab.

Lesson 1: The DotNet Data Type

- Deployment Options
 - Global Assembly Cache (GAC)
 - Local Application Folder
 - Resolution Priority
- Client-side and Server-side Execution
- Events
 - Event publisher
 - Event subscriber
 - WithEvents property
 - Client-side and Server-side Events
 - Synchronous and Asynchronous Events

Lesson 1: The DotNet Data Type

- Constructors
- Static Classes and Members
- Comparing Values

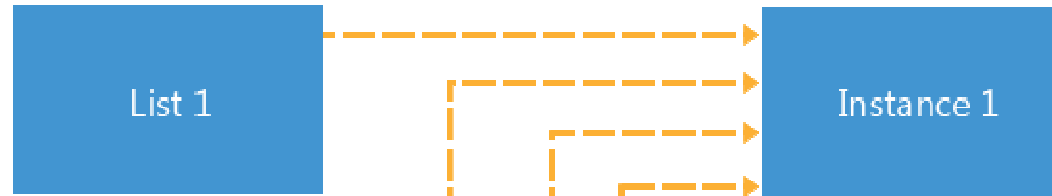
Lesson 1: Demonstration - Declaring a DotNet Variable and Subscribing to Events

Lesson 2: DataType Mapping and Assignment

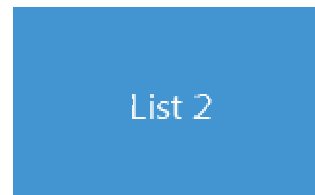
- Right-to-left Mapping
- Limited Mapping
- Mapping of System.String and System.DateTime
- Assigning Instance References

Lesson 2: DataType Mapping and Assignment

List1 := List1.List;



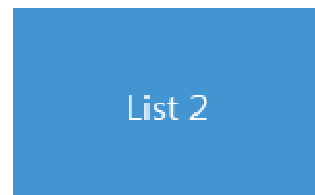
List2 := List1;



List1.Add('First');

List2.Add('Second');

List1 := List1.List;



List2.Add('Second');



Lesson 2: DataType Mapping and Assignment

- Assignment Between DotNet Subtypes
 - Assignment between types
 - System.Object
- Local and Global Scope
- Serializing Data Between Client And Server

Lesson 3: .NET Framework interoperability C/AL Functions

- Null and ISNULL function
- System.Type.Net Framework Type
- GETDOTNETTYPE
- CANLOADTYPE

Lab 11.1: Use a Dictionary Object

Scenario

Isaac, a developer at Cronus International Ltd., is developing a report that shows the profit and loss statement for a specific customer. He has to calculate the sum of all transactions for each General Ledger (G/L) account where **Income/Balance** is **Income Statement**, and where the G/L entries are related to transactions of the specified customer. He decides to use the `System.Collections.Generic.Dictionary` class because of its fast performance for random data access to develop the algorithm that is used by the required report.

This lab contains the following exercises:

- Declare and Instantiate a Dictionary
- Populate the dictionary

Lesson 4: Streaming

- Types of Streams in .NET

.NET Framework Type	Description
System.IO.MemoryStream	Manages streams whose backing data store is memory.
System.IO.FileStream	Exposes a stream programming interface around files, allowing stream-based read and write access to files stored in the file system.

Lesson 4: Streaming

- Passing Streams between C/AL and .NET
 - Using System.IO.Stream with XMLports
 - Streaming to file from XMLport through .NET Framework
 - Passing C/AL Streams to .NET Framework Objects
 - Using Instream instead of System.IO.Stream in Xdocument.Load method

Lesson 4: Demonstration - Use Streams to Import through XMLports

High Level Steps

1. Create a new codeunit and save it as codeunit 90015, **Import Fault Codes From String**.
2. Declare the variables.
3. In OnRun trigger, write code that assigns the following values: 9 to Tab, 13 to Cr and 15 to Lf. Then write code that stores three lines of plain text data, according to the format that is defined in XMLport 5901, **Import IRIS to Fault Codes**.
4. Append code to OnRun trigger, to convert the string to a System.IO.MemoryString, by using the System.Text.Encoding.AsciiEncoding type. Then import the resulting stream into Microsoft Dynamics NAV 2013 database through XMLport 5901, **Import IRIS to Fault Codes**.
5. Run the codeunit, and verify that the data was imported into table **5918, Fault Code**.

Module 12

Queries

Module Overview

- Present the Query Designer and its features
- Explain the principles of the query design process
- Show how to select, join, filter, aggregate, and order data
- Demonstrate how to access queries from C/AL code
- Explain how to export data from queries

Lesson 1: Query Design

- Query Designer
 - Data Items
 - Columns
 - Filters

Lesson 1: Query Design

- Selecting Data
 - Defining data items
 - Set Type to DataItem
 - Set Data Source to a table
 - Defining columns
 - Set Type to Column
 - Set Data Source to a field

Lesson 1: Demonstration - Creating and Running a Simple Query

High Level Steps

1. Create a new query.
2. Add a data item for the **Item** table.
3. Add columns for the **No.**, **Description**, **Base Unit of Measure**, and **Unit Cost** fields.
4. Save the **Query** object.
5. Run the **Query**, and then close the **Query Designer**.

Lesson 1: Query Design

- Joining Data
 - DataItemLink
 - DataItemLinkType

Lesson 1: Demonstration – Joining Data Items

High Level Steps

1. Add a data item for the **Vendor** table to the **Simple Item Query** query.
2. Add columns for **Name** and **City** to the **Vendor** data item.
3. Add a data item for the **Purchase Price** table.
4. Add columns for the **Currency Code** and **Direct Unit Cost** fields.
5. Join the **Vendor** data item to the **Item** data item.
6. Join the **Purchase Price** data item to the **Item** and **Vendor** data items.
7. Save, and run the query, then observe the results.
8. Configure the join on the **Vendor** data item to only show those rows from the **Item** table that have a matching row in the **Vendor** table, and then run the query.
9. Configure the join on the **Purchase Price** data item to only show those rows from the **Item** table that have a matching row in the **Purchase Price** table.
10. Save and run the query, then observe the results.
11. Reconfigure the join on the **Purchase Price** data item to show default values if there is no match. This will show the purchase price list with all items that have a vendor, but without any price information if a purchase price is not defined.
12. Save and run the query, and then observe the results.

Lesson 1: Query Design

- Filtering Data
 - DataItemTableFilter property
 - ColumnFilter property
 - Filter rows
 - Set Data Source to a field
 - ColumnFilter property
 - SETRANGE and SETFILTER functions
 - OnBeforeOpen trigger

Lesson 1: Demonstration - Defining Filters in a Query

High Level Steps

1. Filter the **Simple Item Query** to only show items that use the purchase replenishment system by defining the **DataItemTableFilter** property on a data item.
2. Filter the query to only show those items that have unit cost defined by setting the **ColumnFilter** property on a column.
3. Allow users to dynamically filter the query on the **Vendor No.** and **Vendor Posting Group** fields, without including the fields in the resulting dataset.
4. Compile, save, close, and run the query. Then, observe the results.

Lesson 1: Query Design

- Aggregating Data
 - Sum, Average, Min, Max, and Count
- Ordering Data
 - OrderBy property
- Date Methods
 - Day, Month, Year

Lesson 1: Demonstration - Aggregate and Order the Data

High Level Steps

1. Order the **Simple Item Query** by vendor name in alphabetical order, and by price from highest to lowest.
2. Configure the query to only show the average price for each item.
3. Save, compile, close, and run the query. Then, observe the results.

Lab 12.1: Using a Query from a Chart

Scenario

Susan, the sales order processor at CRONUS International Ltd. needs a chart which shows top ten customers by revenue. She wants to access this chart from her Role Center.

Isaac and Simon, the consultants at the ISV company that implements Microsoft Dynamics NAV 2013 for CRONUS International Ltd., will help Susan by creating and customizing the necessary objects, and configuring Microsoft Dynamics NAV 2013 according to Susan's requirements.

This lab contains the following exercises:

- Creating a Query
- Creating a Chart
- Adding the chart to the Role Center

Lesson 2: Accessing Queries from C/AL

- Running Queries
 - OPEN, READ, and CLOSE functions
- Accessing Columns
 - Syntax: QueryName.ColumnName
- Filtering Queries
 - SETRANGE and SETFILTER
- Limiting the number of rows returned
 - TOPNUMBEROFROWS
- Saving the result sets
 - SAVEASXML and SAVEASCSV

Lab 12.2: Using Queries in C/AL

Scenario

Julia, the marketing executive at CRONUS International Ltd., as a part of the new sales and marketing strategy, wants to be able to automatically remove any credit limits from the most valuable customers. She wants the function to automatically select the top ten customers and show them in a modal list page, and if the modal list page is confirmed, to reset the **Credit Limit (LCY)** field to zero for all the customers shown. She wants this function to be available from her Role Center.

Isaac, the business application developer at the ISV company that implements Microsoft Dynamics NAV 2013 for CRONUS International Ltd. will create and customize the necessary objects to meet Julia's requirements.

This lab contains the following exercise:

- Create a Codeunit which Uses a Query

Lesson 3: Advanced Query Concepts

- Queries and Records

Goal	Record	Query
Initiate iteration through a dataset	FIND('-') FINDSET	OPEN
Retrieve the next record from a dataset	NEXT	READ
Closing the dataset	Not needed	CLOSE

Lesson 3: Advanced Query Concepts

- Mapping Queries to T-SQL

T-SQL	Query Feature
SELECT	Row of Type Column in the Query Designer window
FROM	Row of Type Dataltem in the Query Designer window
JOIN type	DataltemLinkType and SQLJoinType query properties
ON	DataltemLink data item property
WHERE	DataltemTableFilter data item property ColumnFilter property of columns and filters Row of Type Filter
HAVING	ColumnFilter property of columns and filters, when aggregation is used
GROUP BY	Automatically switched on for each row of Type Column, when aggregation is used
ORDER BY	OrderBy query property
TOP	TopNumberOfRows query property

Thank you

To learn more about Microsoft Certifications and to find additional training, books, and resources for your career, go to the following Microsoft website: microsoft.com/learning

Microsoft®