

Development in AB Suite

Objective

By the end of this module, you'll be able to—

- Implement advanced concepts of AB Suite to develop an application

Module Topics

- Model Hierarchy
- Keys
- Framework Methods
- Dictionaries
- Designing Presentations
- Class Diagrams
- Reference Elements
- Developer Security
- Documentation

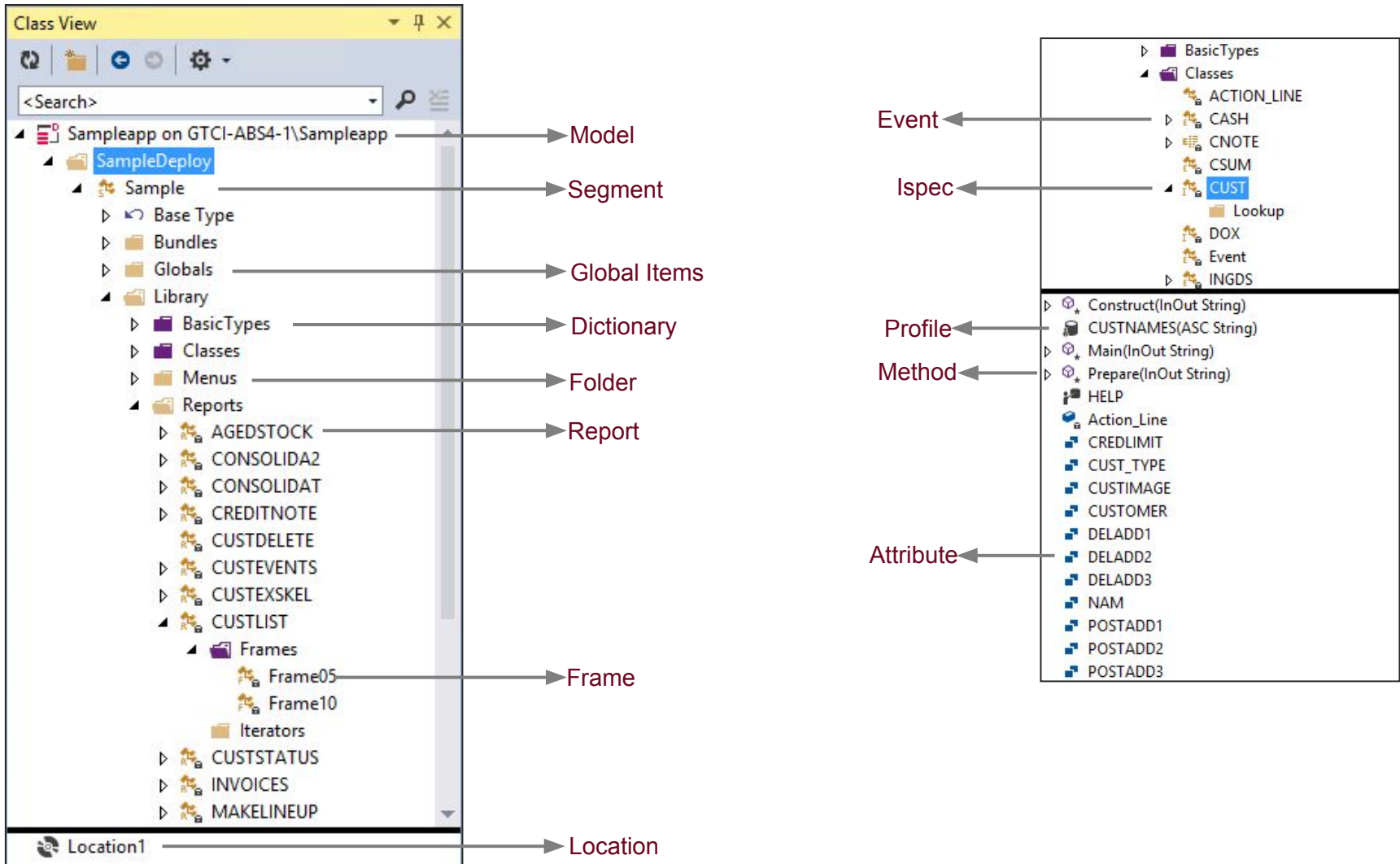
Model Hierarchy

Developing Model Hierarchy

When you develop the hierarchy, keep the following in mind:

- Define relationships through inheritance and dependencies
- Instantiate classes through inheritance and reduce redundancy
- Use dictionaries to create common classes that are used throughout the model
- Group classes logically within folders for better maintainability
- Encapsulate the model elements using the *Visibility* property
- Document the model for easy interpretation on the model-business mapping

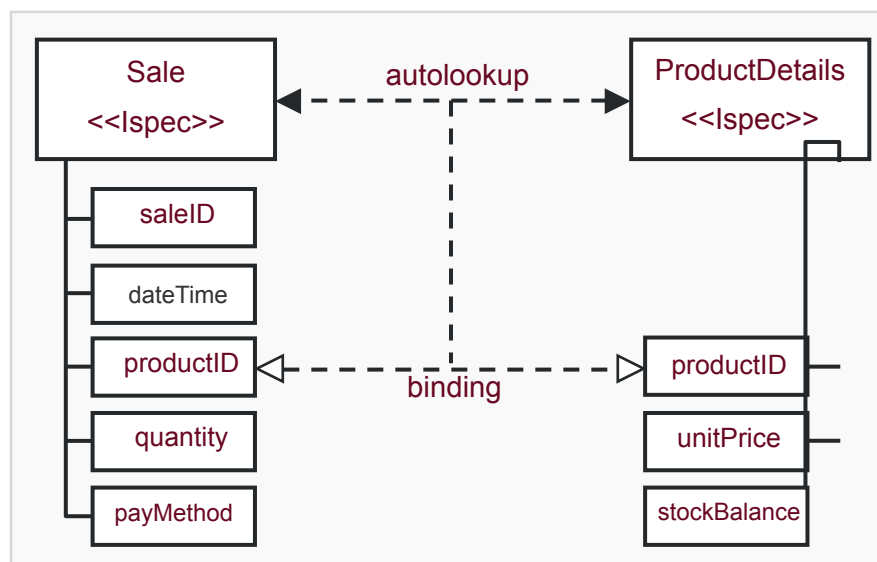
Developing Model Hierarchy



Keys

Profile Keys

- What are keys?
 - Keys provide an index to an ispec's database records
 - Keys help to uniquely identify individual database records
 - Keys help in establishing the autolookup dependency between two ispecs
 - Define the order that records are retrieved from the database.
- Only persistent attributes can be defined as keys



Multi Keyed Classes

- You can specify more than one attribute as keys in an ispec.
- If more than one attribute forms the key, the combination of their values must be unique for each record.
- You can define multi keyed classes by:
 - Using profiles and adding more than one key to a profile
 - Setting the *IsKey* property for more than one attribute

Multi Keyed Classes – Adding Keys to a Profile

To define multiple keys in an ispec:

1. Define attributes that you want to use as keys, but do not set the 'IsKey' property.
2. Define a profile and add the required persistent attributes as keys in the appropriate sequence.
3. You can specify the profile as the 'default profile' for an ispec.

The screenshot illustrates the configuration of a profile in SAP. The top part shows a tree view of the 'VENDPROD' ispec with 'PRODUCT' and 'VENDOR' attributes highlighted in red boxes. A yellow callout box points to these attributes, stating: "PRODUCT and VENDOR are the keys for the VPROD Ispec". Below this, a table shows the key sequence:

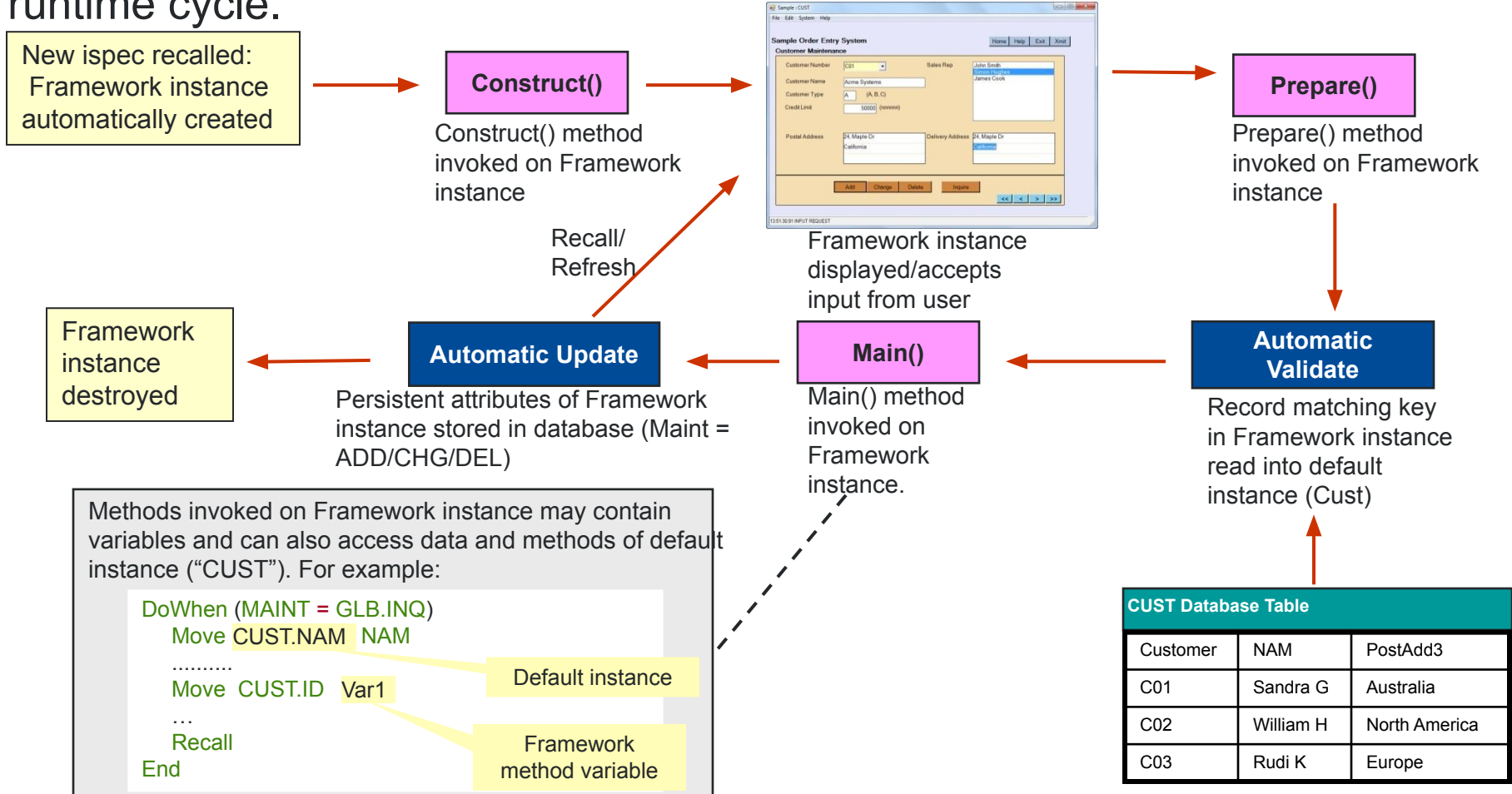
Name	Seq #	IfPresent	IsAscending
VENDOR	1	False	True
PRODUCT	2	False	True

The bottom part of the screenshot shows the 'Properties' dialog for 'VENDPROD'. The 'Owner' field is highlighted in red and set to 'VPROD'. A yellow callout box points to this field, stating: "Default Profile".

Framework Methods

Framework Methods

Framework methods are part of the runtime cycle of an element such as an ispec. The following flow shows how framework methods are invoked in the runtime cycle.



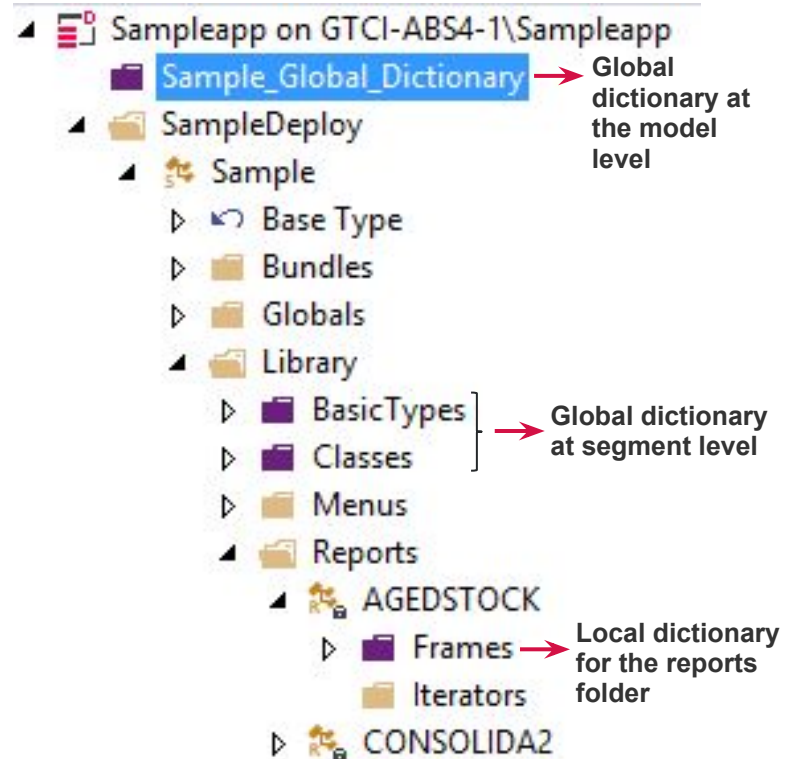
Framework Methods

- Framework methods are: Construct, Prepare, and Main.
- Depending on the stereotype, the methods are invoked during the runtime cycle of an element such as an ispec or a report.
- Framework methods of ispecs with presentation can be overridden. For example, you can perform a specific validation check in the ispec's Construct method.
- Framework methods might include variables to:
 - Temporarily hold values
 - Reference a specific instance of a class

Dictionaryes

Dictionaries

- Dictionaries are useful for organizing classes that define attributes and variables in a model.
- When you add an element to the model, System Modeler will look for a dictionary item with the same name. If a match is found, the element's *Inherits* property will be set automatically to the item in the dictionary.
- When searching for a matching dictionary item System Modeler searches all dictionaries within scope, starting with the most local dictionary.
- The element definitions can be inherited across the model depending on the relative hierarchy where the dictionary is created.



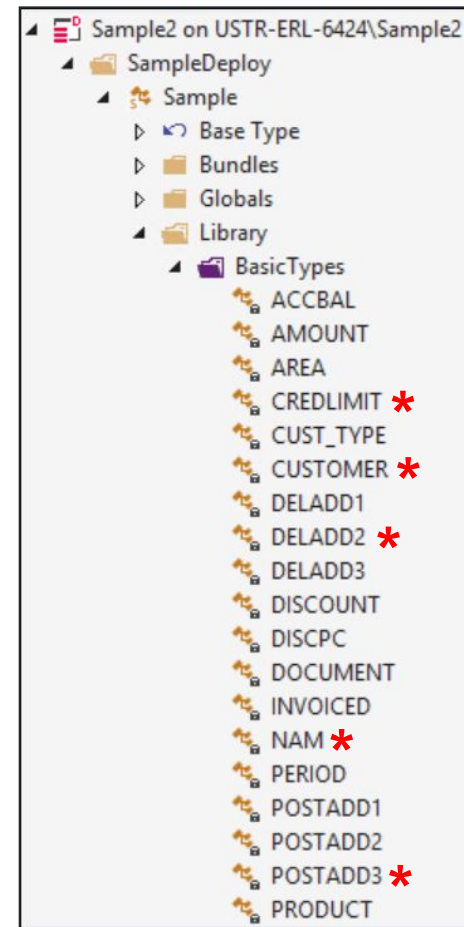
Dictionaries

- You can create dictionaries anywhere in your model.
- You can also create multiple dictionaries.
- A dictionary defined under a segment in System Modeler is available to all elements under the segment.

Dictionaries

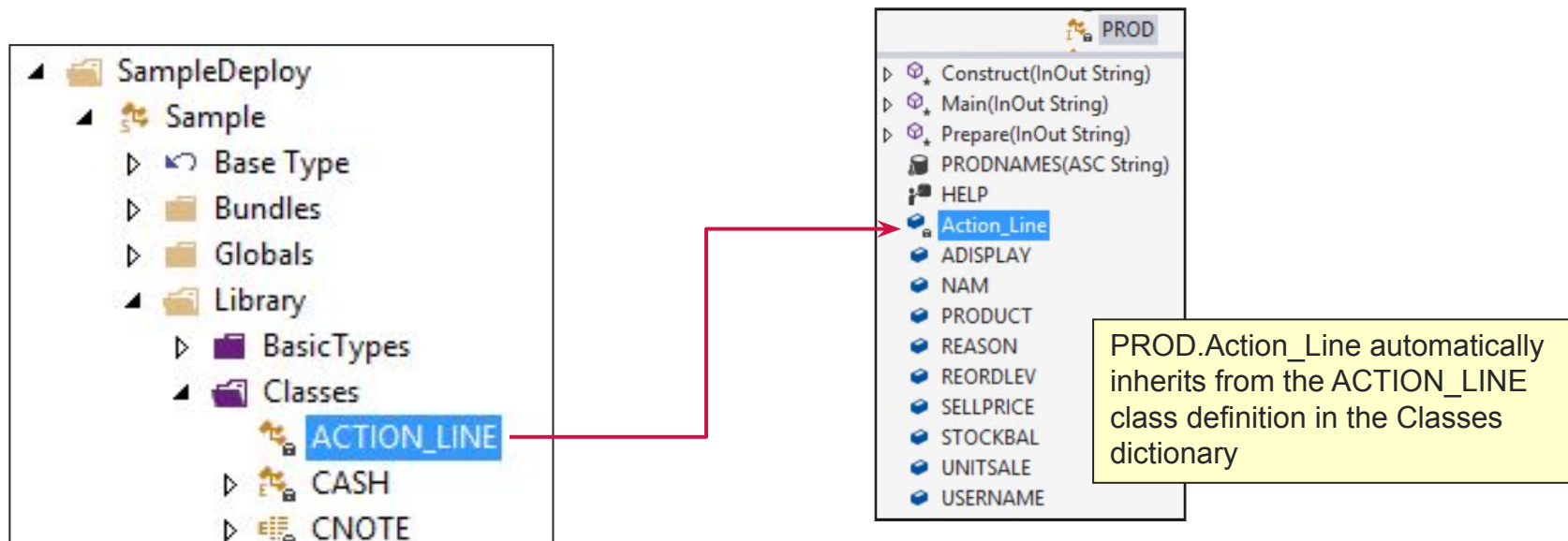
Example of members of CUST inheriting from the dictionary elements

Name	Kind	Inherits	Visibility
Action_Line	Attribute	Sample.ACTION_LINE	Private
Construct	Method		Protected
CREDLIMIT	Attribute	Sample.CREDLIMIT *	Private
CUST_TYPE	Attribute	Sample.CUST_TYPE	Public
CUSTIMAGE	Attribute		Public
CUSTNAMES	Profile		Public
CUSTOMER	Attribute	Sample.CUSTOMER *	Public
DELADD1	Attribute	Sample.DELADD1	Public
DELADD2	Attribute	Sample.DELADD2 *	Public
DELADD3	Attribute	Sample.DELADD3	Public
Get_Credlimit	Method		Private
HELP	Teach Screen		Public
Lookup	Folder		
Main	Method		Protected
NAM	Attribute	Sample.NAM *	Public
POSTADD1	Attribute	Sample.POSTADD1	Public
POSTADD2	Attribute	Sample.POSTADD2	Public
POSTADD3	Attribute	Sample.POSTADD3 *	Public
Prepare	Method		Protected
SALESREP	Attribute	Sample.SALESREP	Public
Set_Credlimit	Method		Private



Dictionaries

- A dictionary can also contain class definitions that may or may not have the *Multiplicity* set to = 0. This definition can then be inherited by an attribute in an Ispec or a report, or a variable under a method.
- In the following example, ACTION_LINE class is defined in the Classes dictionary. An attribute Action_Line within the PROD class automatically inherits from the dictionary definition to create an instance of that class.



Designing Presentations

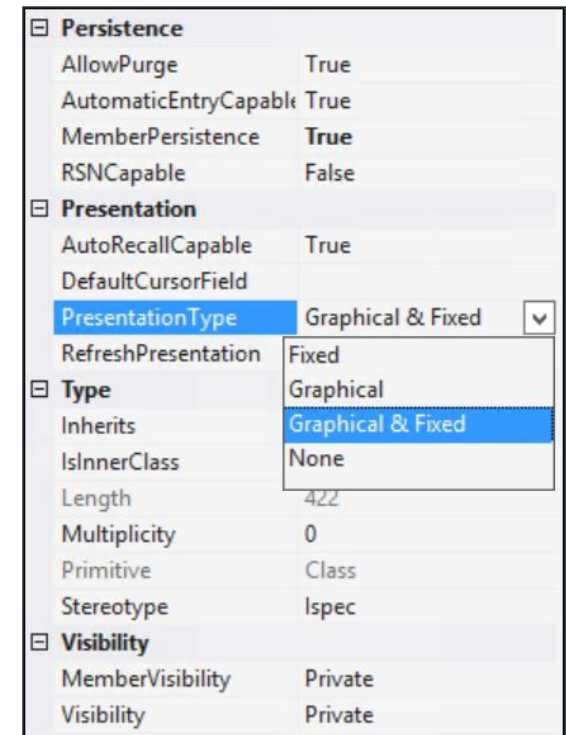
Designing Presentations

- The Painter is used to design presentations or forms.
- A class can have a graphical presentation, a character-based presentation, or both.
- Report frames can only have a character-based presentation.
- *PresentationType* property of a class determines its presentation type.

PresentationType Property

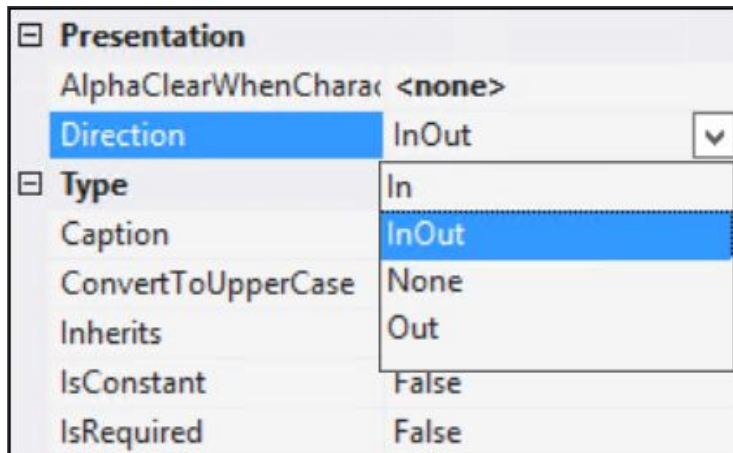
- The *PresentationType* property allows you to choose the type of presentation for a class.
- The choices are:

Graphical	The class has a graphical presentation only.
Fixed	The class has a character-based presentation only.
Graphical and Fixed	The class has graphical and character-based presentations.
None	The class does not have a presentation. Note: If a class has a <i>PresentationType</i> of None, then the <i>Direction</i> property is not available on its attributes.



Direction in Presentation

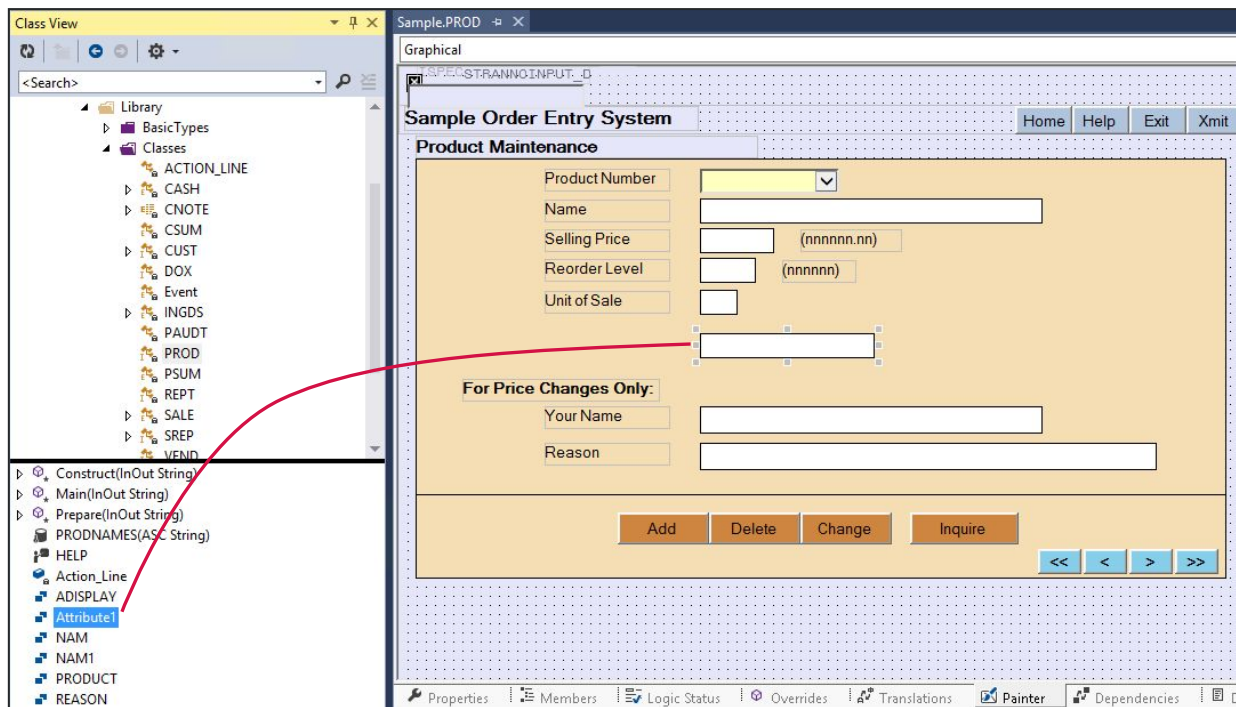
- *Direction* property of an attribute specifies how a control painted on a presentation passes data.
- Attributes with a *Direction* property value other than None are automatically painted on the presentation.



Direction options for an element

Adding an Attribute to the Presentation

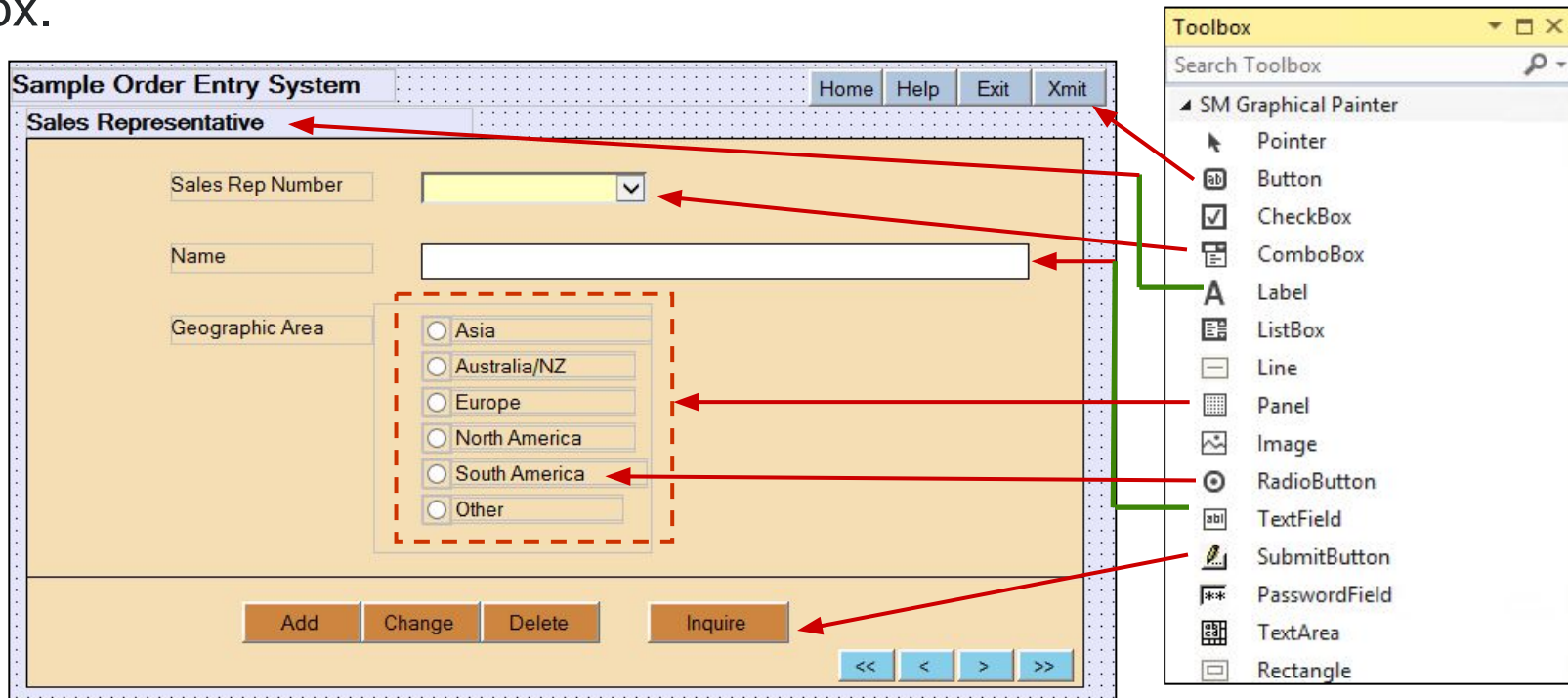
To add a new attribute to the presentation you can drag the attribute from the class view to the painter.



Save the presentation. The *Direction* property for the attribute is set to InOut that can be changed if required.

Adding New Controls to the Presentation

You can add controls to the presentation by dragging them from the tool box.



GridPanels are used to group controls in the presentation.

Group Controls: Graphical objects that are grouped together to represent one attribute are called Group Controls. Button, Checkbox, and Radio Button are called Group Controls.

You should use a panel to associate more than one RadioButton, CheckBox, or Button with a single attribute.

Working with the Grid Panel

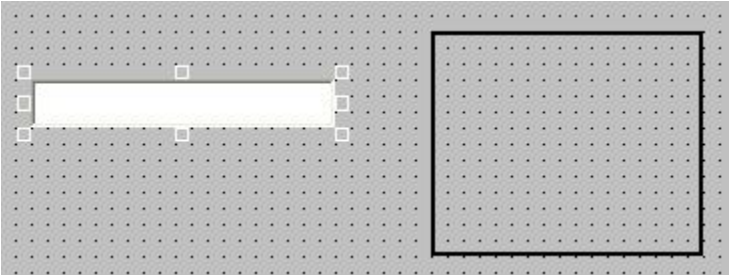
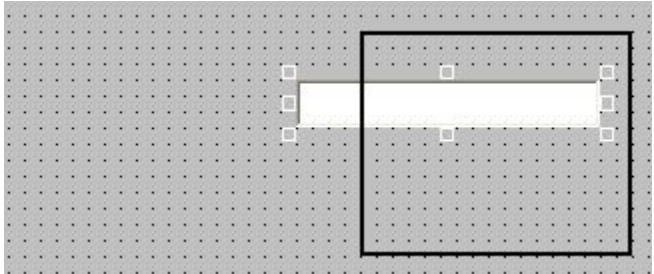
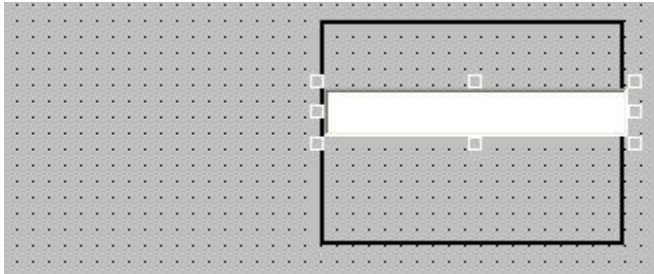
- Grid Panels allow you to associate a set of group controls with one attribute.
- To create a Group control:
 1. Drag a Grid Panel from the ToolBox to the Painter.
 2. Drag the Group control from the Toolbox into the Grid Panel in the Painter.
 3. Repeat the controls for the number of occurrences required.

For example, the following is a Grid Panel containing two Radio Buttons that represent one attribute - Question



Moving Controls into a Grid Panel

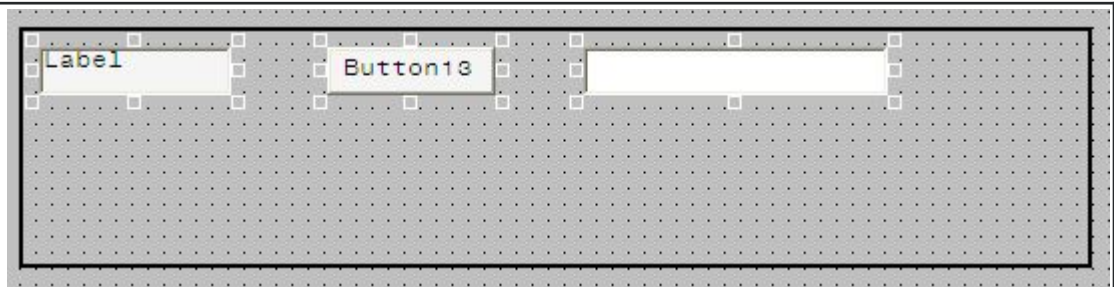
Using Ctrl key

Steps	
1. Select the controls that you want to move.	
2. Press and hold the left mouse button.	
3. Press and hold the Ctrl key and then move the controls in the Form.	

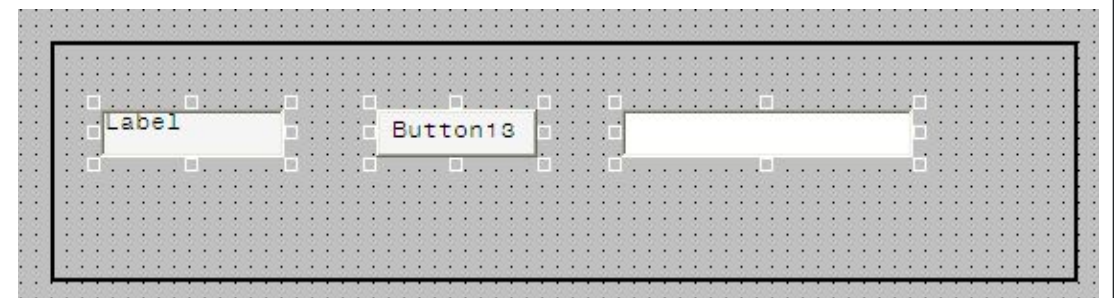
Moving Controls within a Grid Panel

Using Alt key

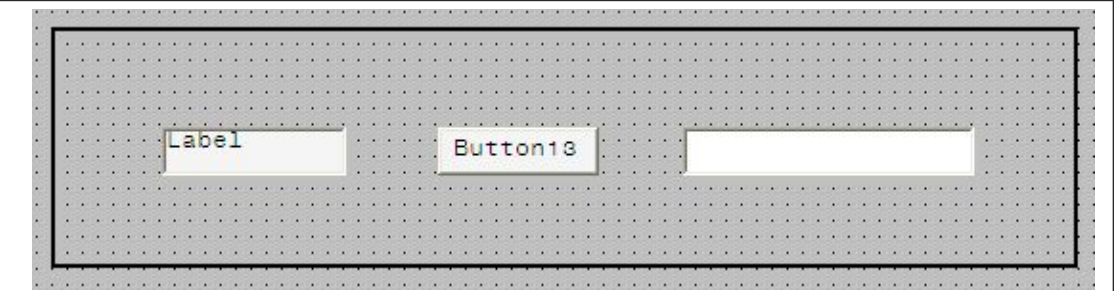
1. In the Grid Panel, select all the controls that you want to move



2. Press and hold the Alt key and then move the controls within the Grid Panel

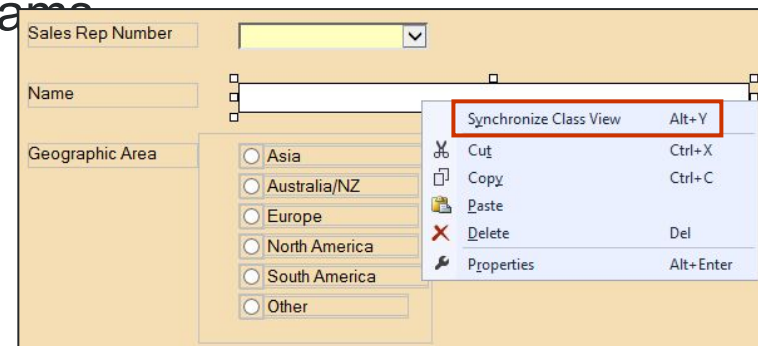


3. Place the controls in the required position



Synchronizing an Element With the Class View

- The Synchronize Class View option synchronizes the selected item with the Class View.
- You can also select Synchronize Class View option from the Edit menu.
- You can use this option to:
 - View some of the properties of the primitive data type associated with the selected form objects such as button, check box, radio button, text field, password field, or text area
 - View the properties of the element associated with the form
- Synchronize Class View can also be used in other views, such as the Members list, Search Results list, and Diagrams.
- To synchronize a selected element in the Painter, perform the following steps:
 1. Make sure that the form is displayed.
 2. Right-click a graphical object on the form.
 3. Select **Synchronize Class View** option.

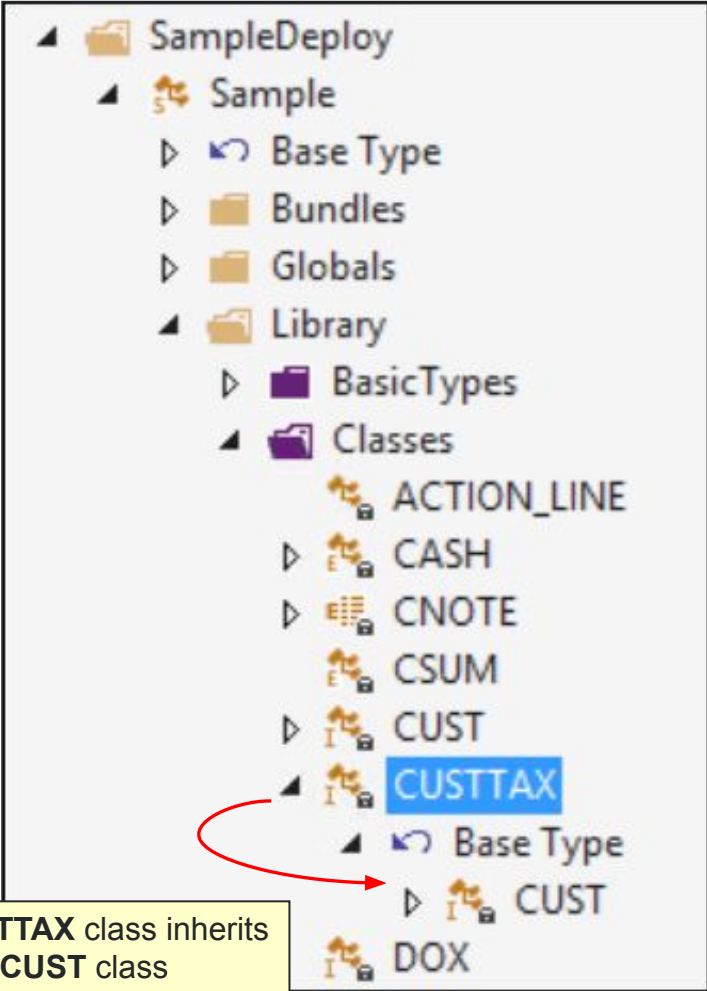


Generalization in Presentation

- Generalization is an Object-Oriented concept that describes “is a” relationship between two classes
- With a generalization relationship between two classes, the common structure and behaviour are used from the specialized class to the generalized class

Generalization in Presentation

Example: Consider a class CUSTTAX that represents customers who pay tax that inherits from CUST



CUSTTAX class inherits from CUST class

Generalization in Presentation

Sample.CUST - X

Graphical

SPECSTRANNOINPUT.D

Sample Order Entry System Home Help Exit Xmit

Customer Maintenance

Customer Number Sales Rep

Customer Name

Customer Type (A, B, C)

Credit Limit (nnnnn)

Postal Address

Delivery Address

Add Change Delete Inquire

<< < > >>

CUSTTAX class inherits presentation from the CUST class

Sample.CUST - X

Graphical

SPECSTRANNOINPUT.D

Sample Order Entry System Home Help Exit Xmit

Customer Maintenance

Customer Number Sales Rep

Customer Name

Customer Type (A, B, C)

Credit Limit (nnnnn)

Postal Address

Delivery Address

Add Change Delete Inquire

<< < > >>

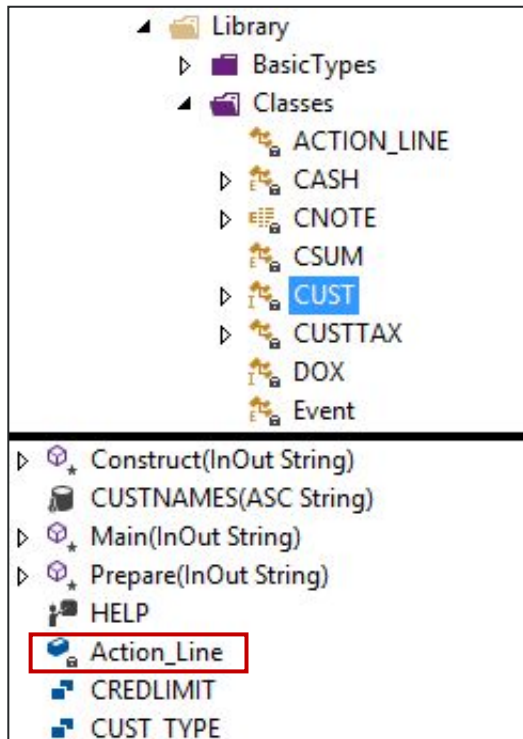
TAX

Composition in Presentation

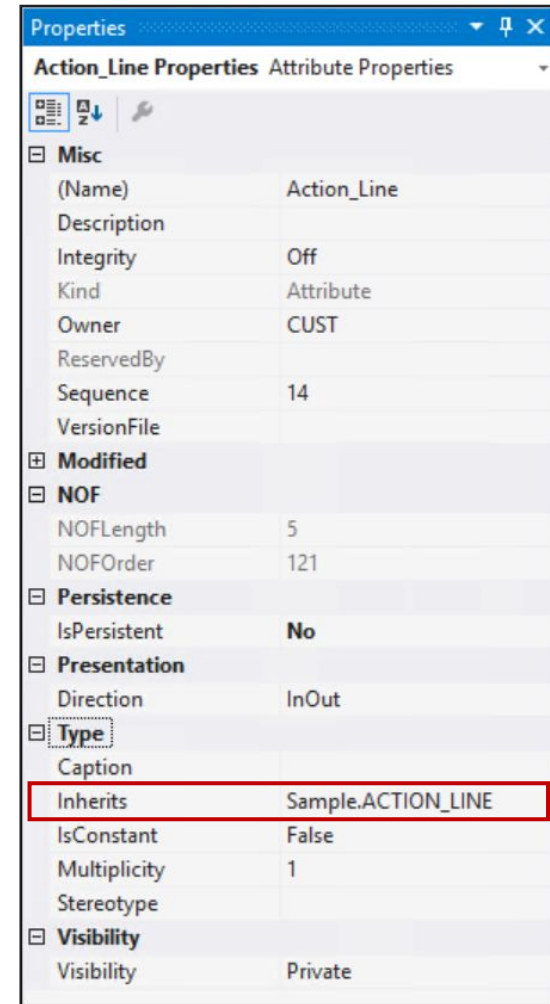
- Composition is an Object-Oriented concept that describes a relationship between two objects where one object owns, or is made up of, other objects.
- A class with a presentation can be inherited by an object belonging to another class. This way some of the common presentation elements such as headers and footers can be reused and extended within a form.

Composition in Presentation

Example: Consider the CUST class that has an attribute named Action_Line that inherits from the ACTION_LINE class



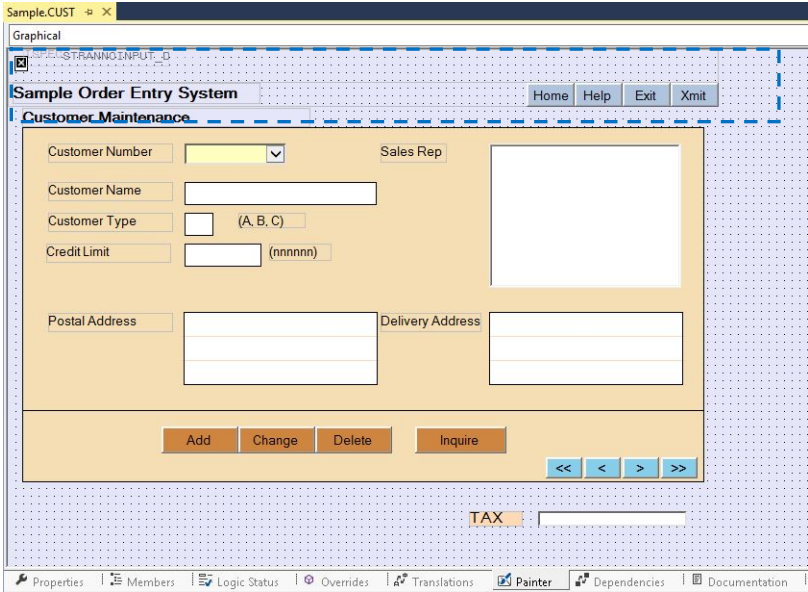
Action_Line attribute inherits from the ACTION_LINE class



Composition in Presentation



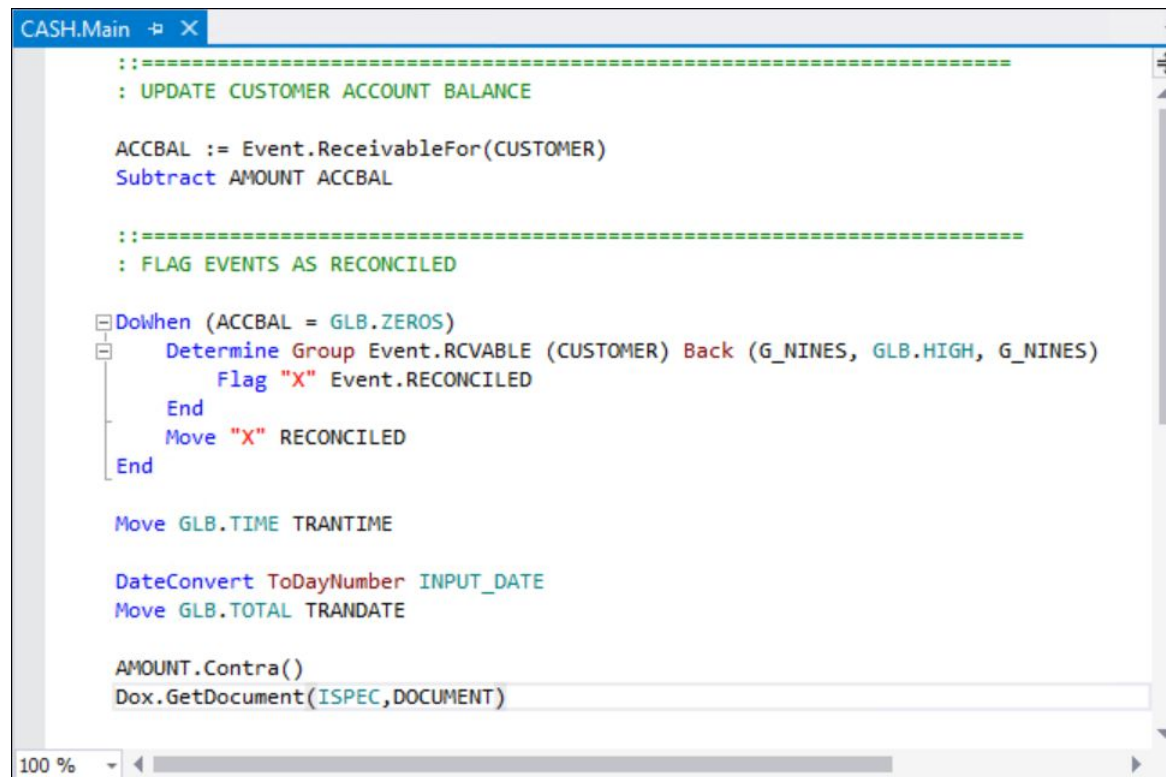
CUST class inherits presentation from the **ACTION_LINE** class



Logic Editor

Logic Editor

- The Logic Status tab displays the list of methods. Double click a method to invoke its Logic Editor.
- The Logic Editor is used to add, edit, save, and validate logic.



```
CASH.Main  [X]
-----
: UPDATE CUSTOMER ACCOUNT BALANCE

ACCBAL := Event.ReceivableFor(CUSTOMER)
Subtract AMOUNT ACCBAL

-----
: FLAG EVENTS AS RECONCILED

[ ] DoWhen (ACCBAL = GLB.ZEROS)
[ ]   Determine Group Event.RCVABLE (CUSTOMER) Back (G_NINES, GLB.HIGH, G_NINES)
      Flag "X" Event.RECONCILED
      End
      Move "X" RECONCILED
    End

Move GLB.TIME TRANTIME

DateConvert ToDayNumber INPUT_DATE
Move GLB.TOTAL TRANDATE

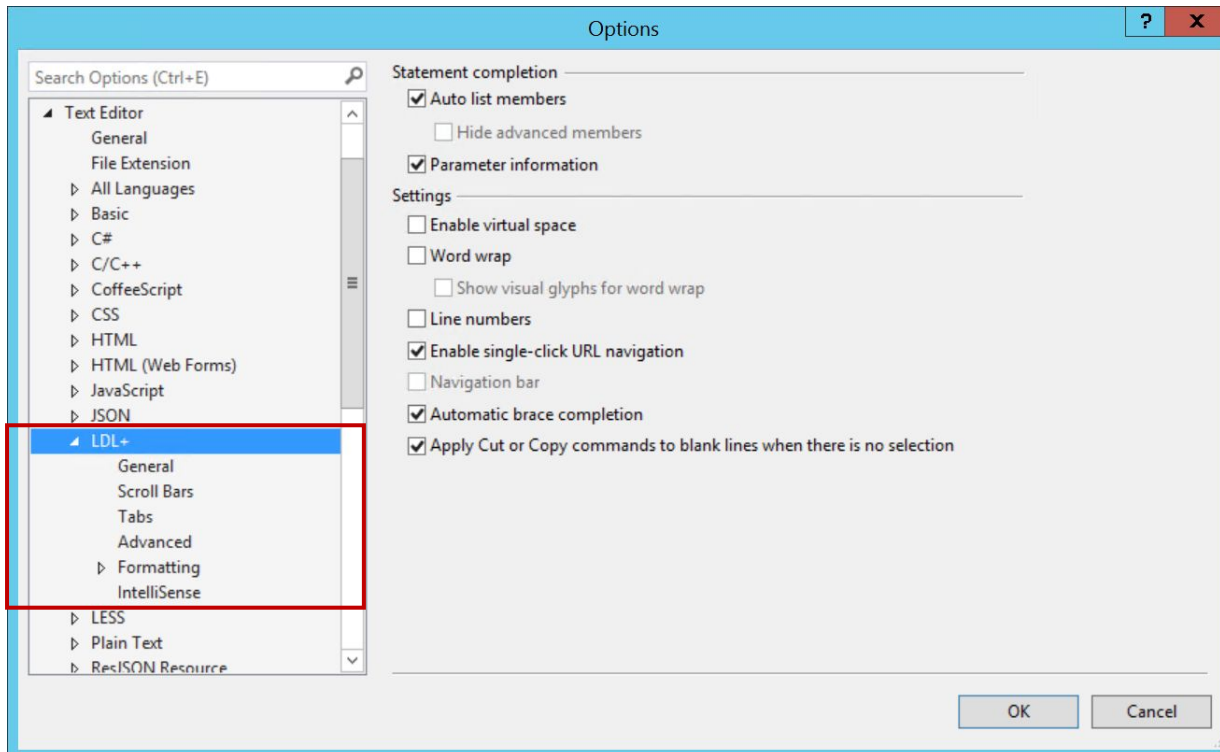
AMOUNT.Contra()
Dox.GetDocument(ISPEC, DOCUMENT)
```

Logic Editor Highlights

- Dynamic Validation
 - Immediately identify logic errors with the red squiggle
- Quick Actions
 - Quickly and easily resolve logic errors
- Code Definition and Peek Definition
 - View and edit logic without leaving the current method
- Personalize the experience
 - Customize the look, feel, and features to suit your development style
- Debugger Conditional Breakpoints
 - Break when logic conditions are met or hit counts are reached

Logic Editor Settings

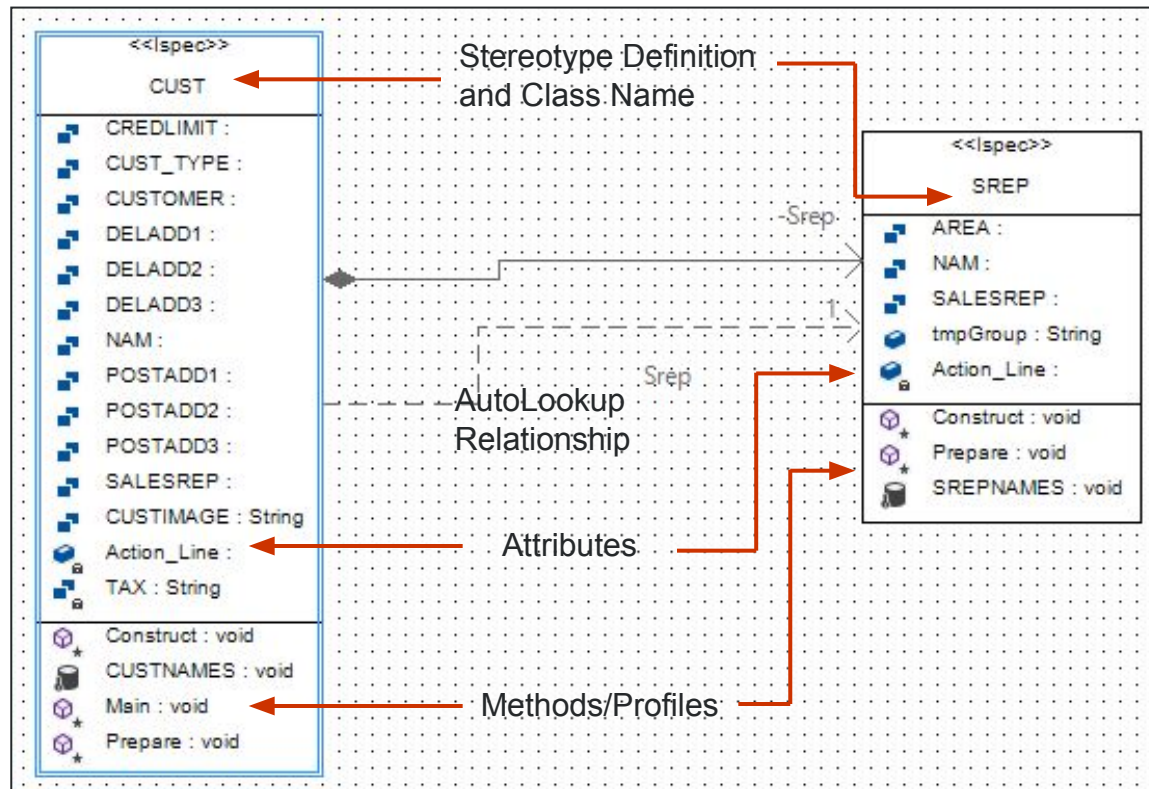
- The Logic editor options can be set from the Tools > Options menu.
- In the Options window, navigate to the LDL+ folder for setting Logic Editor features like Dynamic Validation, Quick Action, and Command Style.



Class Diagrams

Class Diagrams

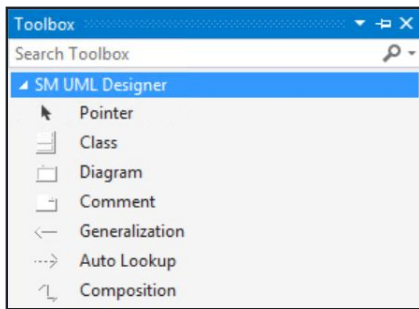
- A diagrammatic representation of classes and their relationships.
- In this class diagram, the CUST ispec has an AutoLookUp dependency defined on the SREP ispec.



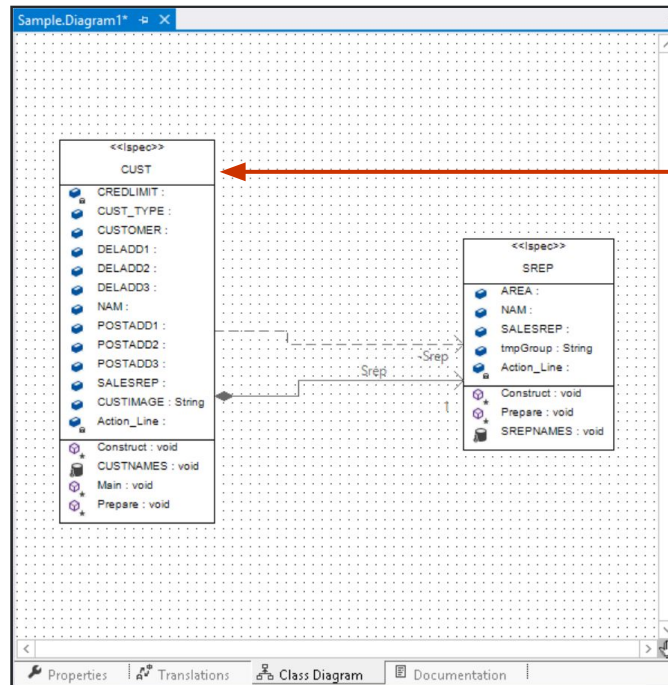
Class Diagram Editor

To create a class diagram:

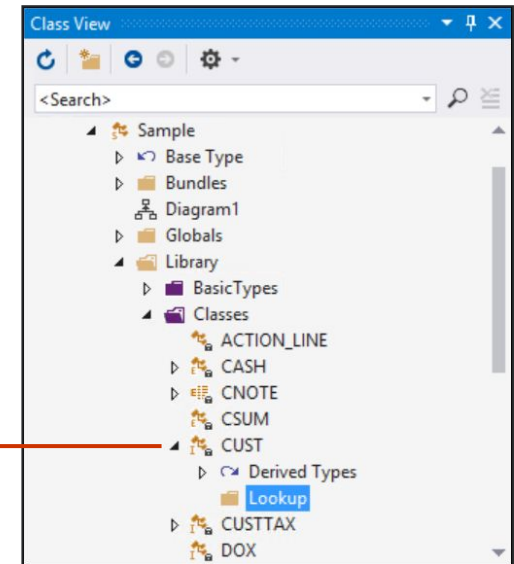
1. Add a diagram to an element in your model.
2. Drag elements from the Toolbox or the Class View on to the Class Diagram editor.



UML Toolbox



Class Diagram Editor

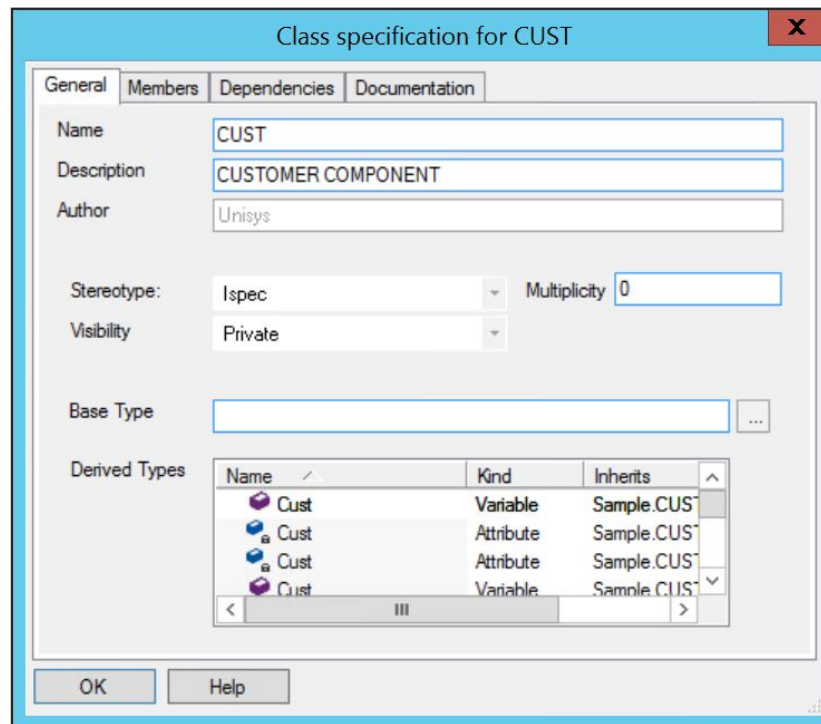


Class View

Modifying Class Specification

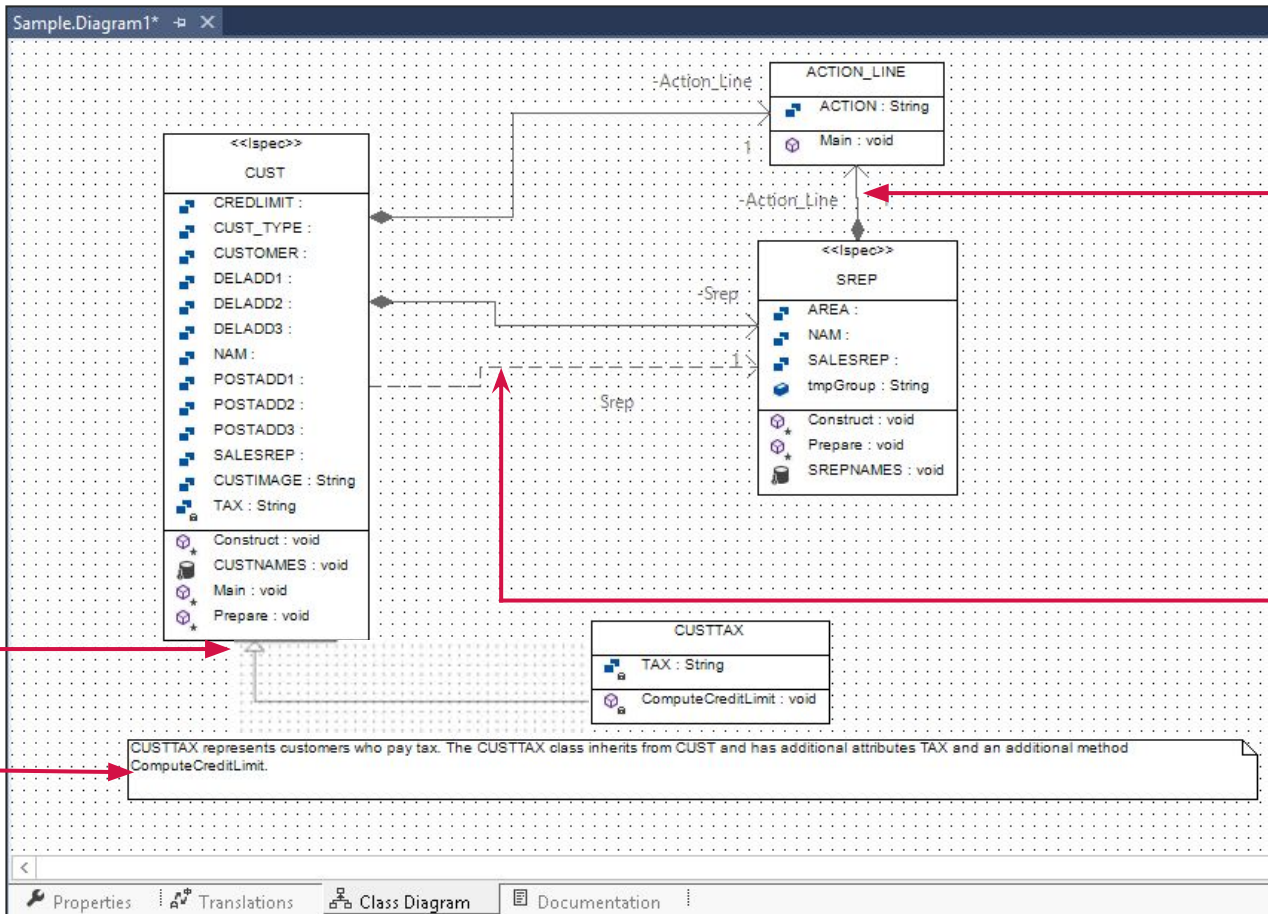
To modify class specifications:

1. In the Class Diagram editor, right-click the class entity and select Properties.
2. Modify the required class entity properties.



Defining Relationships

You can define all the relationships using the Toolbox.



← Generalization

↳ Composition

→ Auto Lookup

Comment

Reference Elements

Reference Elements

A reference is used:

- To display an attribute that cannot be directly dragged onto the painter. For example, arrays, and attributes of other classes.
- To avoid the need for additional attributes and supporting logic to move values around.

Reference Elements—Example

Library

- BasicTypes
- Classes
 - ACTION_LINE
 - CASH
 - CNOTE
 - CSUM
 - CUST**
 - CUSTTAX
 - DOX

CUSTOMER

DELADD1

DELADD2

DELADD3

NAM

POSTADD1

POSTADD2

POSTADD3

Reference1

SALESREP

TAX

1. Add a reference element to CUST Class

2. Set the *Constraint* property to SREP.NAM

3. Drag the reference element on to the CUST form. Save the Form.

4. Change the *Direction* property to "Out" from "None"

Properties Reference1 Properties

Misc

(Name)	Reference1
Constraint	SREP.NAM
Description	
Kind	Reference
Owner	CUST
Sequence	15
VersionFile	

Modified

Author	orchestrate
Created	10/23/2017 2:11 AM
Modified	10/23/2017 2:22 AM

NOF

NOFLength	30
NOFOrder	2147483647

Presentation

Direction	Out
-----------	-----

Type

Caption	
Inherits	Sample.SREP.NAM
Length	30
Primitive	String

Visibility

Visibility	Private
------------	---------

Sample Order Entry System

Customer Maintenance

Customer Number: [dropdown] Sales Rep: [dropdown]

Customer Name: [text box]

Customer Type: [dropdown] (A, B, C)

Credit Limit: [text box] (nnnnnn)

Postal Address: [text box] Delivery Address: [text box]

[Add] [Change] [Delete] [Inquire]

TAX: [text box]

Reference Elements—Example

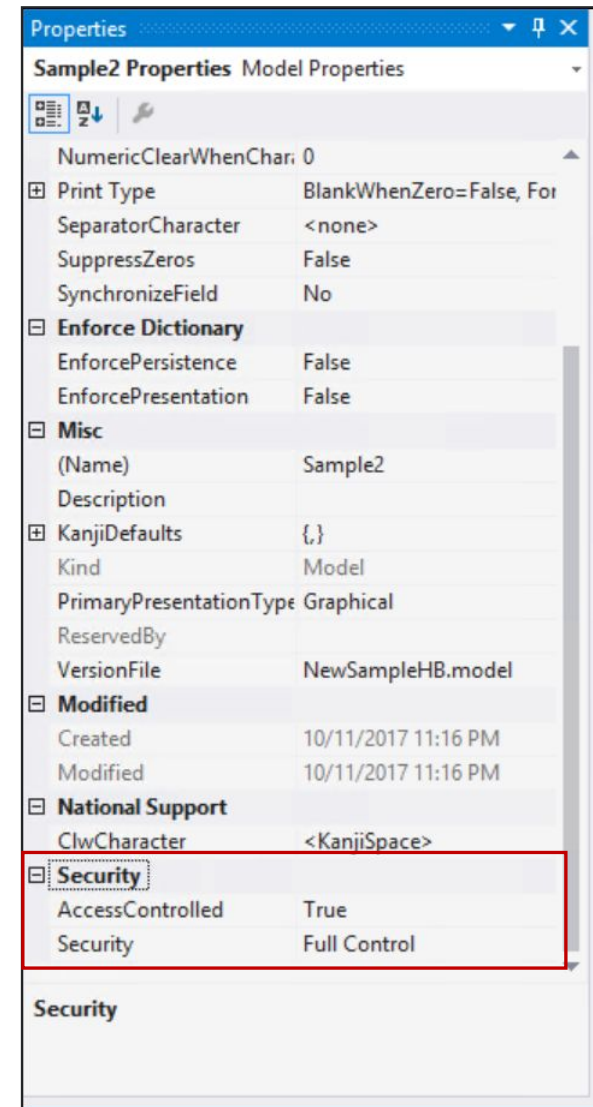
The screenshot shows a window titled "Sample : CUST" with a menu bar (File, Edit, System, Help) and a toolbar (Home, Help, Exit, Xmit). The main area is titled "Sample Order Entry System" and "Customer Maintenance". It contains several input fields: "Customer Number" (C003), "Customer Name" (Mark Taylor), "Customer Type" (A), "Credit Limit" (50000), "Postal Address" (56, Park Street, Devon County, San Jose, CA), and "Delivery Address" (56, Park Street, Devon County, San Jose, CA). A "Sales Rep" field is a list box with "John Smith" selected. A separate text box to the right contains "John Smith" and is highlighted with a red border. A callout box on the right explains that the SREP record is read and the SREP.NAM value is shown in that field. At the bottom, there are buttons for "Add", "Change", "Delete", and "Inquire", and a set of navigation arrows. The status bar at the bottom left shows "14:45:00:01 INPUT REQUEST".

When you transmit the **CUST** screen, **SREP** record is read, and the appropriate **SREP.NAM** value is shown in that field on the screen without writing any logic.

Developer Security

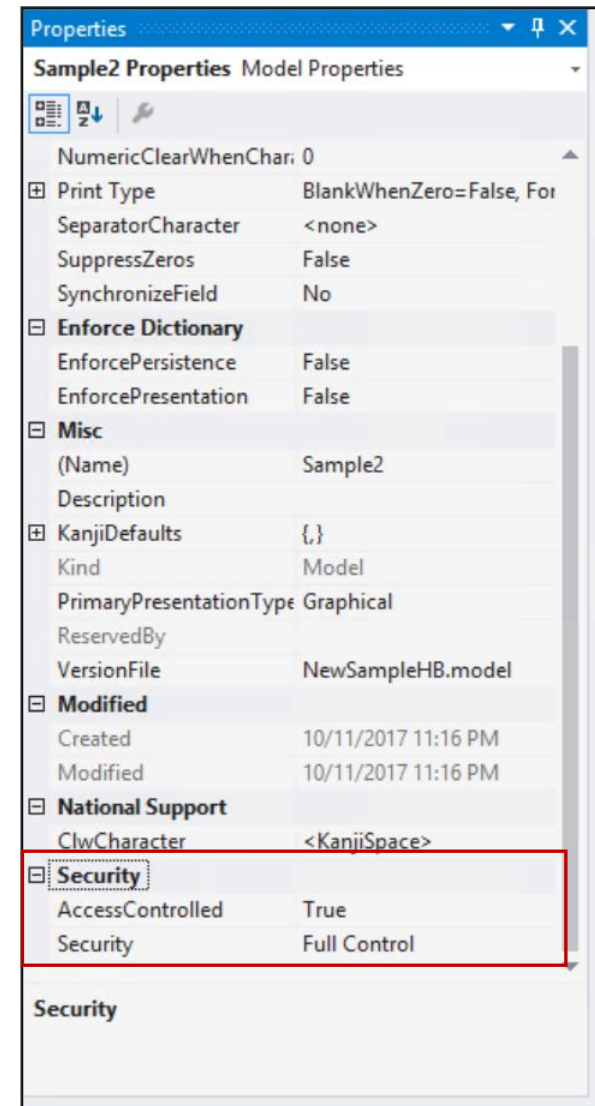
Developer Security

- Developer Security is similar to Windows Explorer Security
- To set the security in Developer, the *AccessControlled* property must be set to true at the model level
- All elements in the model and the model itself can have security privileges applied to it
- You can control the security for either the entire model or individual elements within the model



Developer Security

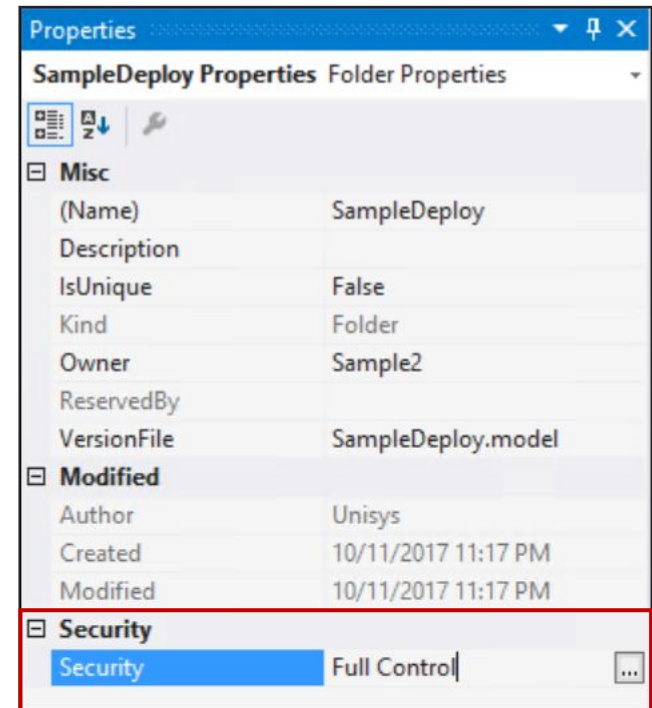
- *AccessControlled* property can be set by:
 - Administrative users
 - Model database owner (db_owner)
 - Security admin of the model database (db_securityadmin)
 - Security admin of the database server (securityadmin)



Developer Security

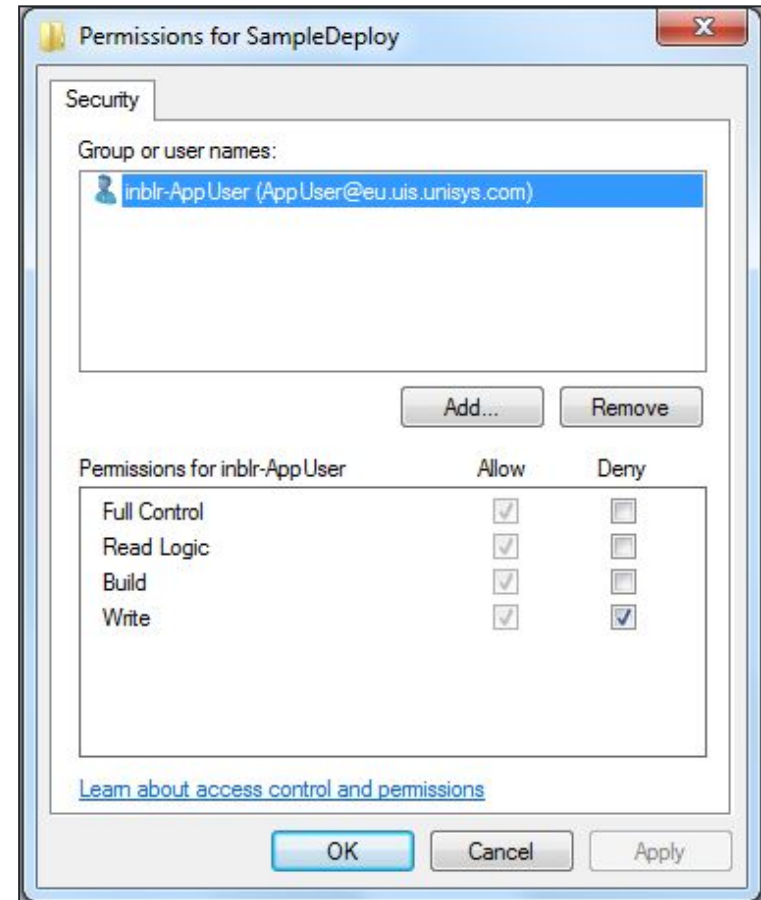
To set security for any element:

1. From the View menu, select Class View to open the Class View window.
2. Select an element in the Class View window.
3. From the View window, select Properties Window.
4. In the Properties window for the element, select the Security property.
5. Click the ellipses button to the right of the window, to open the Windows Security dialog box.

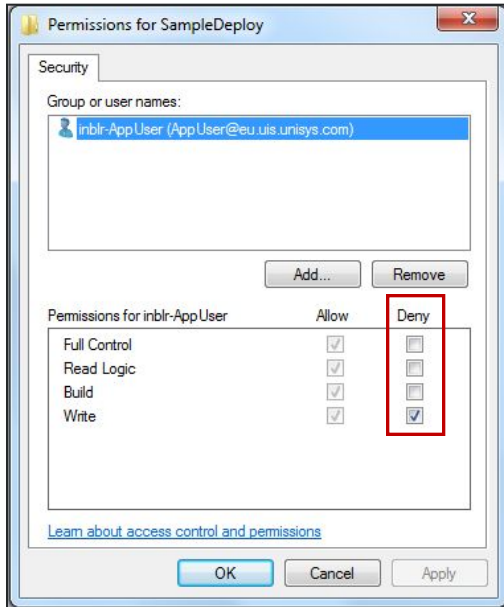


Developer Security

- In the Windows Security dialog box, you can add and remove users.
- Types of privileges you can set for the users:
 - **Full Control** – Allows or denies the user to set the security information
 - **Read Logic** – Allows or denies access to the logic of a method
 - **Build** – Allows or denies access to perform a model build
 - **Write** – Allows or denies access to modify any attributes of an element and all the privileges that applies to Read permission



Developer Security



- The application user is denied write permission to the SampleDeploy folder.
- Select the Security option in the **Build Comments Pages...** dialog box.

The Build Comments pages output displays the security options to help administer permissions to users.

- Sample - Solution

Sample Solution

Outer Classes	Kind	Description
Sample	Segment	Example Business Segment

Languages	Locale	Inherit	Description
Primary	1033 - English (United States)		

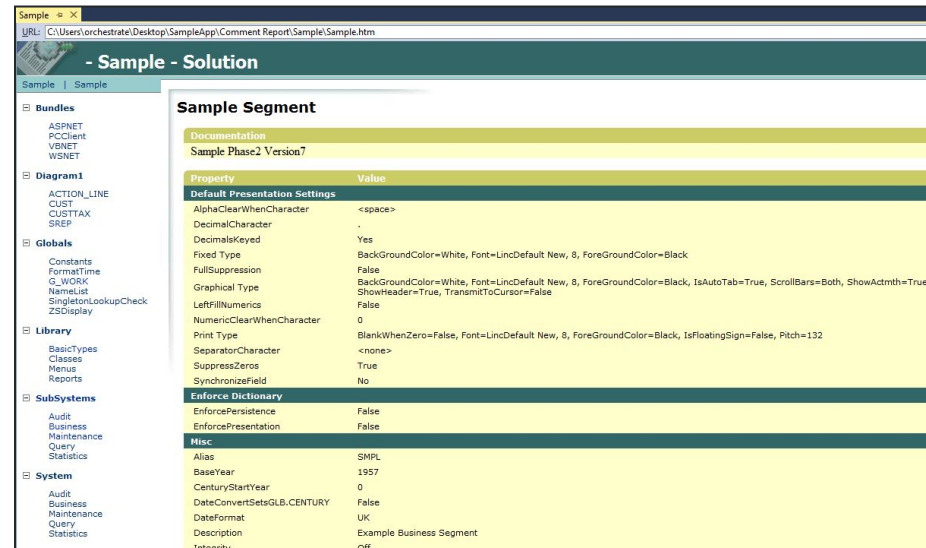
Element Security	User/Groups	Full Control	Read Logic	Build	Write
Sample	\Everyone	Allow	Allow	Allow	Allow
	GTCI-ABS4\admin	Allow	Allow	Allow	Allow
SampleDeploy	\Everyone	Allow	Allow	Allow	Deny
	GTCI-ABS4\Appuser	Allow	Allow	Allow	Deny
	GTCI-ABS4\Appadminuser	Allow	Allow	Allow	Allow

User Security	Element	Full Control	Read Logic	Build	Write
\Everyone	Sample	Allow	Allow	Allow	Allow
	SampleDeploy	Allow	Allow	Allow	Deny
GTCI-ABS4\admin	Sample	Allow	Allow	Allow	Allow
GTCI-ABS4\Appuser	SampleDeploy	Allow	Allow	Allow	Deny
GTCI-ABS4\Appadminuser	SampleDeploy	Allow	Allow	Allow	Allow

Documentation

Build Comment Pages

- Used to create a detailed report of your AB Suite project
 - Build the report of the entire model or
 - Build a report on individual elements or classes
- Creates a hierarchical written display of your AB Suite project
- The report
 - Lists the base class, all subclasses, and their members in the model.
 - Identifies the name of the element, any description that you have provided for it, its member visibility, and the base class to which it belongs.



Build Comment Pages

- Is displayed in the HTML Browser within Visual Studio after the report is generated as a file in a folder of your choice.
- Includes the security details of an element in the generated output

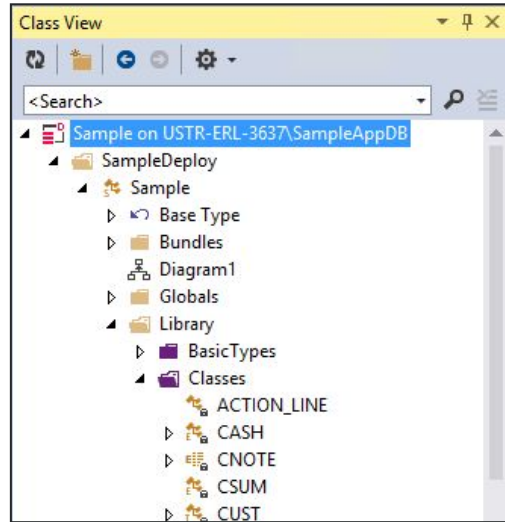
Sample Segment

Documentation

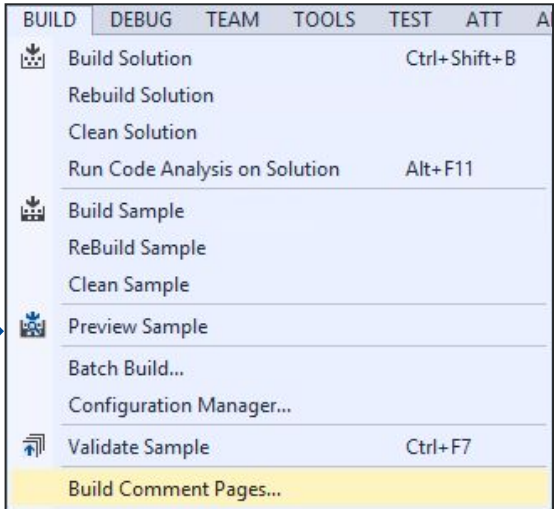
Sample Phase2 Version7

Property	Value
Default Presentation Settings	
AlphaClearWhenCharacter	<space>
DecimalCharacter	
DecimalsKeyed	Yes
Fixed Type	BackColor=White, Font=LinkDefault New, 8, ForegroundColor=Black
FullSuppression	False
Graphical Type	BackColor=White, Font=LinkDefault New, 8, ForegroundColor=Black, IsAutoTab=True, ScrollBars=Both, ShowActmth=True, ShowHeader=True, TransmittToCursor=False
LeftFillNumerics	False
NumericClearWhenCharacter	0
Print Type	BlankWhenZero=False, Font=LinkDefault New, 8, ForegroundColor=Black, IsFloatingSign=False, Pitch=132
SeparatorCharacter	<none>
SuppressZeros	True
SynchronizeField	No
Enforce Dictionary	
EnforcePersistence	False
EnforcePresentation	False
Misc	
Alias	SMPL
BaseYear	1957
CenturyStartYear	0
DateConvertSetsGLB.CENTURY	False
DateFormat	UK
Description	Example Business Segment
Integrity	Off

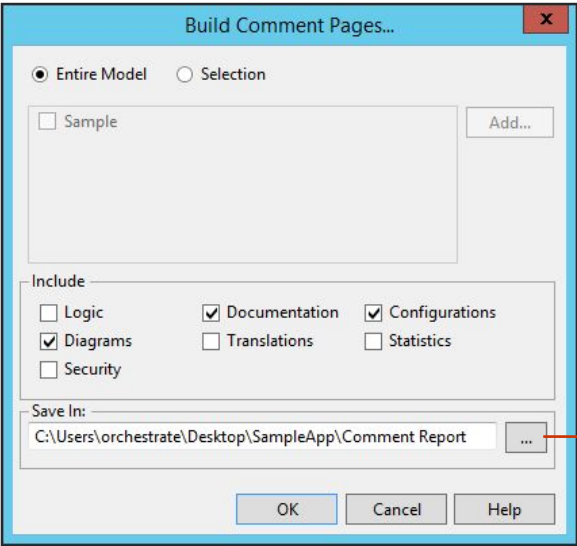
Build Comment Pages



1. Select the Model in Class View.

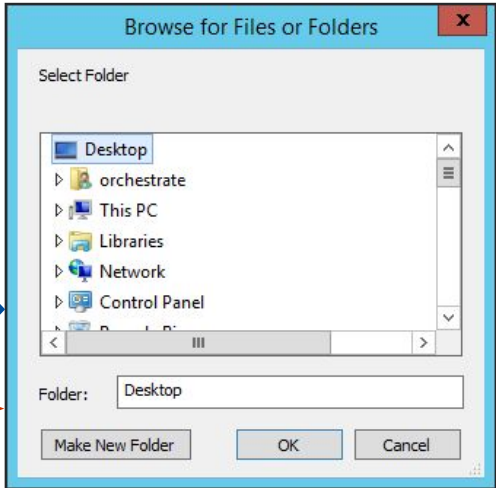


2. Select Build Comment Pages from the **Build** menu. The Build Comment Pages... dialog box appears.



3. To build a report on the entire model, select the **Entire Model** option. Select the **Selection** option to build a report only on selected classes or to include individual elements in the report.

4. Browse for the location to save the HTML file of the report. Click OK to build the report.



Product Documentation

Product Information

- Developer CD
- PI Only CD
- Online Help
- Unisys Support Web Site

Product Documentation – PI Only CD

- All User Guides are published for a release
- Except Generator Customization Kit document
- Acrobat Reader display external to browser
- Search function of Reader
- Infopack

**Unisys Agile Business Suite
Product Information Contents**

**Release 6.1
December 2016**

This page contains links to the Agile Business Suite Product Information documentation library. If you want to browse the library by title, use the first list. If you know the part number of the document you want, use the second list.

Please make a selection from the options below:

1. Select document by title

Select document...

- Access Layer Application Programming Interface
- Business Integrator - Getting Started
- Client Framework Programming Reference Manual
- Component Enabler Class Reference
- Component Enabler Generator Customization Kit
- Component Enabler User Guide
- Developer User Guide
- Installation and Configuration Guide
- Programming Reference Manual
- Migration Guide
- Migration Reference Guide

2. Select document by part number

Select document...

button on the Acrobat Reader toolbar to perform full-text searches across the whole product

at Reader with Search" to perform this function. See the Acroread.txt file on this CD for

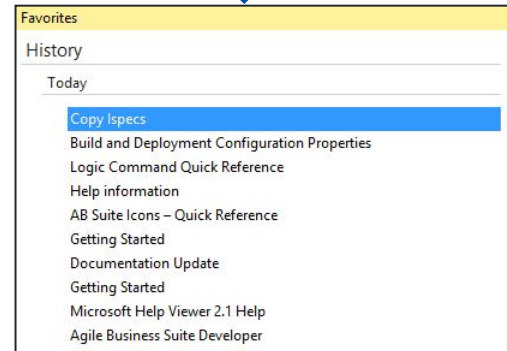
fill out the online Information Questionnaire by clicking [here](#).

Product Documentation – Online Help

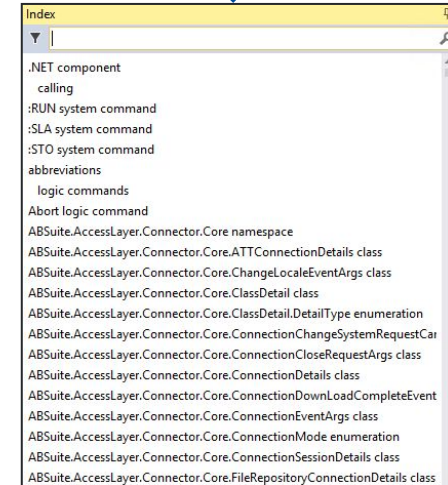
Functions available for Online help:

- Contents
- Index
- Favorites
- Find
- Context-sensitive help (F1)

Favorites



Index



Find



Contents

Readme Document

- AB Suite Readme document is supplied with every release of the product and provides information related to -
 - New features and changes in the respective release
 - Product limitations and known issues
 - Installation specific instructions
- Readme document is updated for every Interim Correction (IC)

Product Documentation – Unisys Support Web Site

Documentation section

- <https://www.support.unisys.com/common/epa/DocumentationLibraries.aspx?PLA=ABS&NAV=ABS>

The screenshot displays the Unisys Product Support website interface. The header includes the Unisys logo with the tagline "Securing Your Tomorrow" and the text "Product Support". A search bar is located in the top right corner. The main content area is titled "Agile Business Suite (AB Suite) Documentation Libraries". On the left side, there is a sidebar menu with various support options. The "Documentation" link in this sidebar is circled in red. The main content area lists several documentation libraries, including Agile Business Suite GCA versions, Infopack versions, and Software Qualification and Support Matrix documents.

Support Options

- Search
- Recent Alerts
- FAQs
- Support Database
- Fixes
- Documentation**
- Products
- Releases
- Localization Kits
- ClearPath ePortal
- Create a Service Incident

Employee Options

- Training Materials for Customers
- Training Materials/Webcasts for Employees
- Evaluation Software
- UCF Submission Requirements
- View Downloaded Files
- Entitled Users
- Engineering Only

Agile Business Suite (AB Suite) Documentation Libraries

- Agile Business Suite 6.1 GCA
- Agile Business Suite 5.0 GCA
- Agile Business Suite 4.0 GCA
- Agile Business Suite 3.0/IC 3.0.1300
- Agile Business Suite 3.0 GCA
- Infopack 6.1
- Infopack 5.0
- Infopack 4.0
- Infopack 3.0
- Agile Business Suite Demonstrations
- Agile Business Suite HowTo
- Agile Business Suite Quick Start Tutorials
- Agile Business Suite White Papers
- Agile Business Suite Utility
- AB Suite 6.1 Software Qualification and Support Matrix
- AB Suite 5.0 Software Qualification and Support Matrix
- AB Suite 4.0 Software Qualification and Support Matrix
- AB Suite 3.0 Software Qualification and Support Matrix

Working Efficiently with Your Model

Working Efficiently with Your Model

- Better ways to define:
 - Insertables
 - Group Attributes
 - Ispecs
- Some points to remember

Better Way to Define Insertables

In AB Suite, an insertable stereotype is not required to define a common part of a screen.

- A class can have fixed, graphical, and print presentations.
- Presentation can be placed on ispec screens or frame layouts.
- Achieved by instantiating the class under the ispec or frame, and dragging the attribute onto the Painter.

Better Way to Define Group Attributes

- A group stereotype is not required to visually group attributes.
 - Attributes can be modelled under any class.
 - Assignment between instances of the same class is allowed – this will perform a deep copy of all members.
- Groups are only required when moving an entire group to or from a string or another group with a different structure.
- Avoiding the group stereotype avoids generated code and memory used to express the whole group as a string.

Better Way to Define Ispecs

- In AB Suite, it is possible to create a class with no stereotype which only has persistent members. The behavior will be identical to an ispec so long as no keys are defined.
 - Ispec keys add a 1 character MAINT field to the table.
 - Profiles can be added to any class with persistent members.

Better Way to Define Global Attributes

- In AB Suite, you can avoid unnecessary declaration of global attributes. Alternatives are:
 - You can declare parameters and variables to use with a method
 - You can declare local attributes to ispecs and reports rather than declaring global attributes to be used in the application
 - With the above two considerations, methods become 'self-contained' and do not depend on global items, enhancing the maintainability
- You can use global attributes where it makes sense and is necessary, for example, Latest_currency_value (banking application), Today's_discount (retail store application).

Points to remember...

- Understand the structure of your model and identify the elements from an object-oriented design perspective.
- Use the UML diagrams to create class diagrams.
- Use inheritance and dictionary elements to create new ispecs and attributes instead of copying and pasting them to reduce redundancy in your model.
- Use folders to logically group elements or build (generate) smaller sets of elements.
- Familiarize with the Painter tab, Toolbox, and various controls in Toolbox.
- Understand how attribute properties of the attributes influence the rendering on Painter.

Summary

In this module, you have learned to:

- Implement advanced AB Suite concepts in application development.