Agile Business Suite

Developer User Guide

UNISYS

Release 6.1

December 2016 3826 5823-008

Contents

Section 1
Introduction

About This Guide

This document provides overview and description of AB Suite Developer 6.1 and its various components, such as System Modeler, Painter, Windows Presentation Foundation (WPF) Designer, Debugger, Builder, and so on. It also covers detail of Automated Test Tool (ATT) menu within System Modeler included in AB Suite Developer 6.1 and provides the functions to record test cases, play back the recorded test cases, and view the results of a playback operation for AB Suite transactions.

Audience

This document is primarily for those who use AB Suite Developer 6.1 to create model-driven applications, implement object oriented concepts in their applications, and test the functionality of their applications.

Documentation Update

This document contains all the information that was available at the time of publication. Changes identified after release of this document are included in problem list entry (PLE) 19158047. To obtain a copy of the PLE, contact your Unisys representative or access the current PLE from the Unisys Product Support website:

http://www.support.unisys.com/all/ple/19158047

Note: *If you are not logged into the Product Support site, you are asked to do so.*

Section 2
Getting Started

Agile Business Suite is a complete set of development tool for building ASP.NET Web applications, XML Web Services, desktop applications, mobile applications, and service based applications. It has a set of tools for the development, deployment, and runtime management of information systems. It provides the technology to create and maintain applications that are responsive to business change, whether small or very high volume. It is capable of generating complete software applications that can be run in a very large-scale mission critical environment.

AB Suite is a complete rapid application development and deployment environment. It:

- Defines, generates, and manages complete, highly-scalable, real-world composite applications that are easily adapted in changing business rules and processes, and can be deployed across multiple operating environments.
- Has model-based and model-driven development capabilities that allow users to define and modify applications at a higher level of abstraction.
- Enables enterprises to quickly implement new processes and encapsulate business knowledge as reusable components in a Service-Oriented Architecture (SOA) environment.

Agile Business Suite is integrated into the Microsoft Visual Studio environment and leverages the features of Visual Studio to provide a highly productive development environment. It is based on high-level specification of the business model, capability to deploy complete applications, and the flexibility to easily adapt the model. It helps in evolving business needs and changes in user environments. Agile Business Suite allows you to build business solutions that include both Agile Business Suite and other .NET components thus optimizing resource usage. An Agile Business Suite solution may, for example, comprise Agile Business Suite components, C#, Visual Basic .NET, and C++ components, or any type of project that Visual Studio is capable of generating. Through the use of standards such as SOAP, UDDI and WSDL, these solutions may work with other clients and services generated by other IDEs such as Web Sphere Application Developer (WSAD) or Visual Studio.

Overview and Features

AB Suite supports object oriented (OO) Fourth Generation Language (4GL) environment. It provides model-based and model-driven development capabilities. It integrates development, testing, versioning, and application deployment functions within the Visual Studio.NET IDE framework. It develops and builds applications in the Windows Visual Studio environment, and deploys them to supported Runtime System Environment.

For example, by building standard elements as classes and encapsulating them in a clean interface, you can customize standard applications to suit specific requirements. Classes can be included in a number of different components to allow for separate distribution thus minimizing development time. Specializing standard framework classes facilitates adding data or overriding behavior and classes generated as components can be specialized.

Agile Business Suite is comprised of two major components

- AB Suite Developer (Development Environment), which lets you define and then generate components for the Microsoft.Net Framework and ClearPath operating environments.
- AB Suite Runtime for each of the supported platforms.

Development Environment

AB Suite Developer or Development Environment is implemented within the Visual Studio Integrated Development Environment (IDE) framework. It includes tools for designing, developing, testing, generating, and deploying applications.

The AB Suite Development Environment includes the following:

- System Modeler – for modeling information systems
- Debugger – for testing systems modeled in System Modeler
- Builder – for generating and deploying these systems
- Version Control tools, a third-party software – for version management of models and model elements

AB Suite Runtime

AB Suite Runtime supports

- Windows® operating system
- ClearPath MCP operating system

AB Suite Runtime is installed on the target runtime platform, and provides an infrastructure to run the deployed AB Suite components. The runtime environment is different for each of the target platforms supported by AB Suite – Windows .NET and ClearPath MCP.

Features of Agile Business Suite

Agile Business Suite provides

- A higher level of definition
- Platform independence
- Ability to mask complexity
- Capturing the what without worrying about the how

It offers the following capabilities

- Stability and ease of migration to follow-on releases of the product.
- Builds and manages composite applications easily.
- Is able to use the concepts of component-based development.
- Provides the granularity for version control in the new architecture.
- Ability to deploy individual components of a single application to multiple platforms.
- Standards-based structures to provide both development and runtime interoperability
- Able to consistently and reliably generate very large and business-critical systems that can be used to run entire business operation.
- Deploy and manage components built with other component-based development tools and those built with the Developer.
- Provides an automatic mechanism to migrate data from existing systems. Migration scenarios that would require database reorganization must be kept to a minimum.
- Optionally provides a mechanism to take an existing system and transform it into a component-based application.
- Provides unprecedented flexibility, modern, and innovative development capabilities.
- Provides a thoroughly modern development environment with features that boost productivity and deliver high-quality applications faster.
- Allows reuse of existing EAE application assets in new ways to further maximize the value in the current systems.
- Built-in diagramming tool to enable you to visualize your application, or parts of it using the UML class diagram.

Development Environment

Development Environment is integrated within Visual Studio Integrated Development Environment (IDE) framework and includes the following:

System Modeler

System Modeler in AB Suite Developer functions as a ‘plug-in’ to Visual Studio and is a project in Visual Studio. It provides a model-based and OO development environment. As a model driven development environment, System Modeler allows you to focus on the logical requirements of a system without worrying about platform-specific implementation details. Because the models created in System Modeler are platform independent.

System Modeler enables you to model and build an application from the ground up, including the end-user presentation, classes, and objects that represent business logic, and automatic generation of the associated database tables.

In System Modeler, you define classes to represent logical entities. You define data by adding elements to a class. These elements can be simple data types, like numbers or text strings, or complex classes with members of their own. There are several ways to define the runtime behavior of a class. These include:

- Adding logic to the class
- Setting class properties
- Defining relationships between the class and other elements in the model

System Modeler also provides a set of stereotypes that you can apply to classes to implement processing behavior. A stereotype is a concept from Unified Modeling Language (UML). It indicates to the model how an element should behave. For example, ispecs and segments are both classes, but they have different stereotypes.

As an OO and model driven development environment, System Modeler enables you to extend and reuse the logical elements in your models, and to integrate these elements with components developed in other environments. In System Modeler, you can define a logical representation of your business using OO development techniques such as encapsulation, composition, inheritance, and polymorphism.

The System Modeler consists of the following screen panes:

- The **Solution Explorer** displays files of versionable elements of a project and project files of the AB Suite Client Framework applications such as, the DataModels project for all the client technologies, DataViewModels, Data Sources, and Views project for the WPF Client technology. Each versionable element in the Solution Explorer can be manipulated by source control in standard ways such as addition to a version control bank, checking in, and checking out.
- The **Class View**, displays the hierarchy of classes and their members, and can be used to add new elements to the model, move and rename them. Its main function, however, is to select elements to be displayed in the Properties window of the Developer System Modeler.
- The **Properties** window displays properties that are common to all the elements selected in the Class View or Solution Explorer. The properties in table format lists the property names and values. Changes in property values are applied to all selected elements.

The class view also shows a node for the project and each of the Segments it contains, but it shows all the elements in each of the classes in the project. As you select one or more elements in the class view or solution explorer, those properties that are common to those element types are displayed in the Properties window.

When you double-click any of the elements in the solution explorer or class view, the Developer System Modeler designer opens, with the tabbed pages appropriate to that kind of element. The Visual Studio searching mechanism is used to search the Agile Business Suite model too, as though the model is made up of files.

In addition to this, Developer System Modeler includes a Designer Window, which shows a number of views of an element in the model on a series of tabbed pages.

The Designer page includes the following:

- Properties page – It shows a summary of the information displayed in the Properties Window.
- Documentation page – It shows a WYSIWYG text editor (based on the standard RichEdit control) for text or embedded OLE objects describing the element.
- Members page – It shows the members visible to a namespace element. The list of parameters can be ordered to represent the order of parameters.
- Logic Status page – It shows the logic for a method, profile or SQL script.
- Dependencies page – It shows the relationships between this element and others. These relationships are actual (reflecting a dependency between this element and another).
- Class diagram page – It shows a UML class diagram of the element and its member classes.
- Painter page – It shows the design of the user interface, printed image, and teach screen for designing AB Suite applications.

  Note: *The Painter page is not available for Client Framework models. You must design the user interface separately from the Client Framework model by using the development tool of your chosen technology. For example, WPF Designer can be used for WPF/Extensible Application Markup Language (XAML) desktop applications. However, both the AB Suite and WPF projects can exist in the same solution to impart an integrated development experience.*

Debugger

Debugger runs within the Visual Studio IDE. It emulates the host runtime environment on the development workstation and enables to test the screens and logic before deploying the runtime system.

It performs Just-in-time (JIT) compilation of selected components that are debugged and does not require a full generate of the application. The reason a full generate is not required is because most of the components are never generated – they are interpreted.

It does not compile the LDL+ logic in the model. Instead, when logic is encountered in a debug session, it interprets it. So that, you can modify the logic during a debug session. For example, if you notice that a line of logic is incorrect, you can correct it and step through the logic again to test your changes, without closing the debug session.

The interpreted instance differs from the generated application only in that objects are constructed dynamically during interpretation and the LDL+ statements are executed by traversing the abstract syntax tree rather than the compiled code. However, the dynamically constructed objects inherit the standard behavior and elements from the same framework classes, use the same persistence implementation, and use the same classes representing the Agile Business Suite data types for their attributes, and the interpreted LDL+ executes the same methods of the class to implement the logic.

Builder

The Builder generates the database definition language, the program source code, and other files necessary to deploy the system represented by the model.

After you define a project using System Modeler, you can build and deploy it using the Builder. Builder provides a number of unique capabilities, including the ability to:

- Deploy to multiple operating environments from a single model.
- Generate the complete application code – including user interfaces, transaction control, and more.
- Automatically define a complete and enterprise-class database from the model.

You can generate or deploy an application in two ways:

- Using the Build commands within Visual Studio.
- Using the MSBuild.exe, a separate Builder application executable.

When you build an application, Builder Cache folder is created. This folder retains the generated files in an internal format and allows multiple users to share the generated output.

Version Control

In Agile Business Suite, multiple developers in a work group can share the same model, and multiple workgroups can share a common version repository, from which they check out model elements to modify them and then check them back in to publish the changes to other workgroups.

The Visual Studio framework provides an interface to external source control products, implemented as the SCC Manager component. The standard product comes with support for tools that support SCCAPI, and could be extended with implementations tailored for other products. Developer works with any supported source control system such as Clear Case, Borland StarTeam, SURE, StarTeam, Microsoft Visual SourceSafe, or Microsoft Team Foundation Server (TFS). Developer supports different versions of TFS, such as TFS 2012, TFS 2013, and TFS 2015 as source control system. Elements in the model are automatically protected by source control when it is active.

The standard source control infrastructure revolves around files. When a versionable element is added to source control its parent containers and its members are also added (if not already present) to define its position within the model hierarchy. For example, a segment can be added to the Source Control Bank along with its members such as Ispecs, reports, and attributes. When an Agile Business Suite Element is checked into source control, the XML export file is created and passed to the SCC Manager. The name of this file is recorded in the model. When an element is checked out, the XML file representing it is checked out. Where the project has been added to source control, elements must be checked out before they are modified.

Comparing and merging different versions of an element is implemented as a separate application, which can also be invoked directly from the source control system.

Access Control

Agile Business Suite is multi user environment. Access control for Developer and the generated application is implemented using the standard COM+ role based access control infrastructure for runtime. Individual users or user groups are added to these roles using Component Services in the Windows management console. By default, the 'Authenticated Users' group is member of the Developer and Generator roles, and the 'Administrators' user group is member of the 'Administrator' role.

Configurations

Configurations are a concept from Visual Studio. They are named as sets of compilation and deployment options. An element's configuration properties are inherited from those of its owner. Values for these properties are only stored if they are different from those inherited from the element's owner and different from the default.

Configuration Builder

Each project requires a Configuration Builder to manage the compilation of its various parts. In the case of Developer this includes the generation of intermediate source code, and the generation of multiple artifacts for elements in the model. The Visual Studio .NET Framework invokes the Configuration Builders for each project in the solution.

Application Building

In Agile Business Suite, after developing the application model, you can deploy it to one of the supported server platforms.

The Agile Business Suite Builder translates design information stored in the developer model into a running database application and clients to access that application. During this process, Builder uses input from the model, in the form of structural information, configuration information (properties), and logic.

The Builder stores the most recently generated files for each configuration. If the application has been generated previously, these files are retrieved from the Builder cache folder instead of regenerating them. The builder also has the capability to build changes that you have made since the last time you built the application.

The generated C# files, which make up a C# project, are compiled and linked using the pre-compiled libraries and input to the deployment project. Following compilation and linking of the C# project, the output files are stored in the model for each configuration, to be retained for future build/change analysis of the application.

Runtime Environment

Agile Business Suite enables flexible and rapid development of runtime solutions by supporting deployment of both its own components and those developed with other tools together with the capability of multiple platform deployment.

Agile Business Suite Runtime provides an infrastructure to run the deployed Agile Business Suite components. Your application is generated as a collection of C# files, which are compiled. The compiled results, together with database and other deployment information, is linked and assembled into a deployment package (MSI) for transfer to the deployment server and deployment on that server.

Windows® Runtime

Windows® Runtime consists of the following components:

- Runtime
- Protocol Adapters
- Database Server (Remote DB Server)
- GUI Server
- Client Container
- Administration Tool

Windows® Runtime supports deployment of components developed in Agile Business Suite and other tools and enables optimum, flexible, and rapid development of runtime solutions.

Windows® Runtime runs on .NET Framework Enterprise Services environment and includes a Microsoft Management Console (MMC) snap-in to administer the Runtime. The Administration Tool snap-in to MMC enables you to configure the Runtime environment. Using the Administration Tool, you can deploy systems and perform administration tasks required for deployed systems.

You can use the Runtime Administration Tool to create a test database to test your application using Debugger before you deploy it to a runtime host. It also enables you to configure and manage the runtime environment and deployed AB Suite applications for the Windows® operating system.

You can use the Runtime Administration Tool to perform the following tasks:

- Add a Runtime Database Server
- Add a Database Server Registration
- Add a New Database
- Prepare an Existing Database

Agile Business Suite Windows® runtime supports different protocols and transport layers as follows:

- **SOAP over HTTP (Web Services)**

  Simple Object Access Protocol (SOAP) is a protocol that has a predominant role in XML development and Web Services.

- **SOAP over MSMQ**

  This protocol adapter provides SOAP over Microsoft Message Queuing (MSMQ).

- **HUB**

  HUB is a proprietary protocol that allows Agile Business Suite systems to talk to each other.

- **NOF/OFF/USER/GLI**

  All of these are proprietary protocols that are mainly used to migrate applications from mainframe hosts (where these protocols were used more heavily) to Windows operating system hosts.

- **RATL over TCP/IP**

  A protocol that is used by Component Enabler.

- **RATL over MSMQ**

  A feature of Component Enabler.

MCP Runtime

AB Suite Runtime for MCP consists of the following components:

- **Runtime environment** – provides a runtime environment to deploy AB Suite applications.
- **Enterprise Database Server (DMS II)** – used as the database management system. AB Suite uses features of the standard DMSII product, with the facility to use DMS II Extended Edition features. Ispecs with persistent attributes are implemented as DATASETs. Event sets are also implemented as DATASETs. Profiles are implemented as SETs. Conditional profiles are implemented as SUBSETs.
- **Runtime Control and management utilities** – consists of two utility programs:
  - Database Management Utility (DMU) – used to provide direct support to be a Runtime database system — executing database garbage collection, upgrading DMS II release levels for a database and providing emergency database support for production systems such as fixing population limits.
  - System Management Utility (SMU) – used to reconfigure a Runtime system. For example, changing database and system names and family or pack settings.

Exploring the Agile Business Suite Elements

This section describes the elements of the Agile Business Suite model. A model is a container for all the elements in the information system. The following elements are available in System Modeler:

- **Folder** – is a container for all the elements in a model and is used to classify the model elements into logical groups. Folders are also used to group elements for deployment. The configuration properties of folders define deployment options for the deployable elements they contain. Folders have no effect on the generated system.
- **Dictionary** – is a special kind of folder that contains a collection of element definitions. Elements in your model can inherit the properties of dictionary items, maintaining uniformity of related elements.
- **Objects** – An object is the UML way of representing an entity. Agile Business Suite objects comprises of classes, instances (variables, attributes, and parameters), and methods.
- **Profile** – is an index to your information. Using a profile, you get a functional view of your data — you only see records relevant to the function you are performing. For example, when you look at product information from a sales function, you need different information than you would from an inventory management function.
- **UML Diagram** – Uses standard Unified Modeling Language notation to represent the relationships between model elements. Like folders, UML diagrams do not own their members — an element can be a member of more than one UML diagram.

Objects

Objects are the basic building blocks for building software using object-oriented methodologies. Objects may contain data (attributes) and logic (methods). In System Modeler, many object characteristics are specified as properties on the object. AB Suite framework objects consist of built-in:

- **Classes** – Is the blueprint of an object. It is an abstract definition for creating other identical objects. Classes are abstract descriptions of objects. An object can inherit its definition from a class. Each element in a Class can be defined to be visible:
  - Only within its Class (private).
  - Within its Class and any class inheriting from it (protected).
  - Outside its Class (public).
- **Methods** – A method is a sequence of logic statements that apply to an object. It specifies the algorithm or procedure that affects the outcome of an operation. That is, it defines a behavior of the object to which it belongs. The model allows you to define the logic in a range of different languages. In Agile Business Suite this can be LDL+ or a dialect of SQL (as used in SQL scripts). The Agile Business Suite framework includes some built-in methods that form part of its processing cycles.

- **Attribute** – Is a member of an object that holds the state of a class. It is a specification that defines a property of an object, element, or a file. An attribute of an object usually consists of name and value of an element, type or class name of a file, and name and extension. Data used by an ispec, or other class, are called attributes. Attributes are specialized variables and can be persistent. The characteristics of an attribute are; it becomes 'output' by virtue of being persistent, and 'input' by virtue of appearing in a user interface, or the combination of the two.

  The properties of an attribute describe its role and the type of data that it needs to store. An attribute has properties such as Name, Caption, Description, and Direction. Direction property Indicate the direction that values can be passed between the attribute in memory and its presentation, that is input or output type.

- **Variable** – Is a temporary storage for information that is required during the lifespan of the application. Variables are a specialization of a Type, and they can be used anywhere a Type is used. This implements the 'same as' relationship, where another Variable acts as a Type.

  Variables can act as:

  – Local variables in a Method.

  – Attributes of a Class.

  – Parameters to a Method.

  An object can have member variables added to it. The members of a class are ordered, like the parameters of a Method.

- **Parameter** – Is a special variable used to pass data into and/or out of a method call.

Stereotypes

Stereotype indicates special behavior of a class. It indicates how an object or class fits into the AB Suite processing framework. The stereotype determines the class's behavior. It can define a number of built-in or framework methods, which are automatically executed at runtime. A stereotyped class or object inherits a set of built-in attributes and methods.

The following stereotypes are available in AB Suite:

- Segment
- Ispec
- Report
- Frame
- Group
- Insertable
- Event
- CopyIspec
- CopyEvent

- SQL Script
- Messenger

The stereotype that you apply to a class determines the properties that is displayed for the class. For example, a class with the Ispec or Event stereotype contains an *AutoEntryCapable* property while a class with the Report stereotype includes a *DefaultDevice* property.

Segment

Applications and application components are modeled using the segment stereotype. A segment is used at the top level in the model to contain all the elements that make up each application. A class with the Segment stereotype automatically includes the behavior and characteristics of a segment. For example, the built-in attributes and methods inherited by a segment class includes; an attribute called GLB and a built-in method called Startup().

Ispecs

Ispecs (interface specifications) are one of the basic stereotyped classes that make up a System Modeler model. An ispec is a definition of a business resource, such as a customer or product, and defines the user interface and methods for your user application. The graphical user interface formats, the data, and the logic defined for ispecs enable Builder to generate the runtime database structure and a major part of the runtime system. Ispec classes can contain methods. Presentation ispecs contain built-in framework methods (Construct, Prepare, and Main) that are automatically invoked during Ispec processing.

An ispec class can have a graphical user interface, or presentation. To create a presentation, set the PresentationType property to a value other than None. Once this is set, you can access the Painter window to design the appearance and layout of the interface.

Insertable, copyispec, copyevent, and SQL script are more specialized stereotypes.

All the elements within System Modeler, with the exception of the model itself, belong to another element that owns them. For example, a segment is owned by the model, an Ispec is owned by the segment in which it is contained. Elements are identified within their owner by name, and for this reason, their owner is called a ‘namespace’.

Using the Model

Getting Around System Modeler

System Modeler in Agile Business Suite Developer takes advantage of the Visual Studio functionality and interfaces to enable you to work with your application. It uses a combination of common Visual Studio and Developer specific windows and views. System Modeler models an application at the logical level.

It enables you to

- Focus efforts on describing the application behavior rather than details of implementation.
- Bring applications forward from one generation of technology to another, or from one platform to another, over the life of your application.

With this model System Modeler merges the best of the mainstream object and component concepts, including COM and UML, to provide a powerful development environment.

For users with existing applications this provides many benefits, without the need to perform extensive rework

- Greater modularity
  - Definitions can be more localized.
  - Definitions can be protected by encapsulation within well-defined interfaces.
- Improved simplicity
  - Uniform patterns applied to everything.
  - Fewer special cases and rules to remember.

New users benefit from System Modeler's high level model and business constructs to create more robust business components more quickly.

The model allows users to describe a change to the model as directly as possible, requiring the model to make whatever other consequential changes are required. For example, an ispec becomes output or IO if it has persistent attributes, rather than having to specify its kind and that of its attributes separately, and making sure they are compatible.

The model is stored in a transactional database, so that its integrity is protected and it can safely be shared between developers. Unlike 3GLs whose models are captured in text files, errors are actively prevented rather than having to be identified later.

Visual Studio Interfaces

This topic describes the common Visual Studio tool windows used by System Modeler. Refer to the *Visual Studio Online Help* for more information on the development environment.

Use the View menu to display these windows in your Visual Studio environment.

Solution Explorer

Solution Explorer displays a tree structure view of all the versionable files in your model and project files of the AB Suite Client Framework applications such as, the DataModels project for all the client technologies, DataViewModels, Data Sources, and Views project for the WPF Client technology.

When an element is versionable, it means that it can be stored under Source Control in the Source Control Bank.

Versionable elements are model elements that can be added to a Source Control Bank, and be maintained under source control. To use the Visual Studio Source Control Services, you must install a version control tool or use Team Foundation Server (TFS).

From Solution Explorer you can rename a version file and elements. Use the refresh button to refresh the state of the items in the selected project or solution.

Class View

Class View displays the elements in the application you are developing. You can open **Class View** from the View menu. There are two panes in the Class View; an upper Objects pane and a lower Members pane.

Objects pane: It contains an expandable tree of classes whose top-level node represents the model database. The Classes, Folders, Diagrams, Primitive Classes, and Dictionaries are displayed in the Objects Pane. To expand a node selected in the tree, click the plus (+) sign or press the plus (+) key on the keypad.

Members pane: It displays the other elements, Attributes, Profiles, Methods, on selection of the owner from the Objects pane.

Although Attributes appear in the Members pane, they can appear in the Object pane (as a Class), which have the Primitive property set as Class and when any of the following conditions are met:

- Attribute does not inherit from another class.
- Attribute inherits from another class and extends the definition.

The elements in the class view are sorted or grouped based on Name, Kind, and Visibility using the right-click shortcut menu options, **Sort Alphabetically**, **Sort By Object Type**, **Sort By Object Access**. The members of the classes can also be organized based on Name, Kind, or Visibility properties.

You can use **Class View** to add, delete, and open the elements in the editor and navigate to them directly.

Class View Toolbar Options

The **Class View** toolbar allows you to navigate within the Objects and Members panes. You can select a particular view of hierarchy tree and specify the elements that you want to display in Class View using **Class View Settings** menu. The following options are available in Class View Toolbar:

Refresh

This option ensures that the latest details of an element are retrieved from the model database.

New Folder

This option creates a new folder or subfolder into which you can drag other elements for easy access. It is useful for organizing frequently used elements. Refer to Exploring the Agile Business Suite Elements for more information on working with folders.

Back

This option allows you to navigate to the previously selected element. Keep clicking this button to navigate through previously selected elements until you reach the first element browsed. The Back button moves through a history list of previously browsed elements in Class View.

Forward

This option is activated when you click **Back**. It allows you to navigate to the next element selected. Keep clicking this button to return to the most recent element selected. The Forward button moves through a history list of previously browsed elements in Class View.

Class View Settings

The **Class View Settings** option on the Class View toolbar displays a menu from where you can choose a particular view of the hierarchy tree for the model database, and specify which of the available elements are displayed. Following options are available in Class View Settings. Some of these options are also available from shortcut menus in the Objects and Members panes.

- Show Base Types

  This option toggles the display of base classes in the Objects pane.

- Show Derived Types

  This option toggles the display of derived classes in the Objects pane.

- Show Hidden Types and Members

  This option toggles the display of hidden classes in the Objects pane and hidden members in the Members pane.

- Show Public Members

  This option displays the Members that are public for users who are using the classes.

  Note: *The visibility property of an element determines if the element is public, protected, or private.*

- Show Protected Members

  This option displays the Members that are public or protected for users who are extending the classes.

  Note: *The visibility property of an element determines if the element is public, protected, or private.*

- Show Private Members

  This option displays the Members of all visibility levels for users who are implementing and using the classes.

- Show Other Members

  This option displays the Members that do not belong to public, protected, private, or inherited category.

  Note: *The visibility property of an element determines if the element is public, protected, or private.*

- Show Inherited Members

  This option toggles the display of inherited members in the Members pane.

Class View Panes

The members of the class view are organized across two panes:

- Objects Pane
- Members Pane

Objects Pane

The Objects pane displays an expandable tree of classes whose top-level nodes represent the model database. If you select an element in the Objects pane, its members are displayed in the Members pane, and details of the element appear in the Description pane. Refer to Object Browser for more information on Description pane. Expanding a class node lists the classes that are defined within it.

The elements that are displayed in Objects pane are:

- Classes/Types
- Folders
- Dictionary
- Primitive Types
- Diagram

Objects Pane Shortcut Menu

If you right-click an element in the Class View, the Objects pane shortcut menu is displayed. In addition to the System Modeler options, the following options can appear on this menu, depending upon the element selected.

- **Sort Alphabetically** – Elements are listed alphabetically by their names in ascending order (a - z).
- **Sort By Object Type** – Elements are listed alphabetically by their names in ascending order (a - z).

- **Sort By Object Access** – Elements are listed in the order of their visibility, such as public, protected, or private.
- **Group By Object Type** – Elements are sorted into groups by kind.

Members Pane

Members Pane displays the members of the element selected in the Objects pane. The details of the elements selected in the Members pane appear in the Description pane. The elements that are displayed in the Members Pane are:

- Attribute
- Method
- Profile
- Teach
- Location

Members Pane Shortcut Menu

If you right-click a member in the Members pane, its shortcut menu is displayed. This menu allows you to search, sort, and copy members independently from their parent elements. Following are some of the options available in this shortcut menu.

- **Sort Alphabetically** – Members are listed alphabetically by their names in ascending order (a to z).
- **Sort by Member Type** – Members are listed in order of their kind, such as methods of base classes, followed by interface methods, and so forth.
- **Sort by Member Access** – Members are listed in order of their visibility, such as public or private.
- **Group By Member Type** – You can group methods, variables, and attributes within your project. Groups are displayed in the Members pane as an expandable list. Members can be organized by their kind in predefined folders, such as Methods, Variables, and Fields folder. This option is only available in the Members pane.

Working with Class View

All elements in the Class View are represented by an icon indicating its kind and visibility. To expand an element and display its members, click the plus (+) sign next to the icon. The elements, when selected display their properties in the **Properties** window. You can view or edit these properties.

Note: *To select multiple elements simultaneously with the Class View, press the SHIFT or CTRL keys as you click on them. This allows you to drag-and-drop groups of elements.*

Synchronizing an Element with the Class View

The Class View updates its contents automatically to reflect the changes. This enables you to select any element in the editor and locate it instantly in the Class View hierarchy.

The Class View is synchronized with a selected element in another view such as the Members list, Search result List, and Diagrams, which displays elements.

To synchronize a selected element, perform the following:

1. Right-click the selected element in the editor.
2. Select **Synchronize Class View** option.

The element is selected in the Class View.

You can also select Synchronize Class View option from the Edit menu.

Notes:

- *The **Synchronize Class View** option is also available in the Quick Navigator window when you right-click an element.*
- *If an element exists in multiple Folders or Dictionaries, then the instance of the element is highlighted in the folder in which it appears.*

Finding all Eeferences of an Element

The Find All References option enables you to find a list of the selected element that is used. You can also enable this option by using shift+F12. You can use this option for any elements, such as Ispec, Method, and other System Modeler elements. This option is also enabled from the Members pane, Class Diagram pane, and Logic Editor.

To find all references of a selected element, perform the following:

1. Right-click the selected element in the class view.
2. Select the **Find All References** option.

   The list of the possible result is displayed in the Find Symbol Results window.
3. From the Find Symbol Results window, you can view the following results:
   - Dependencies – Displays the elements that are dependent on the selected element.
   - Inheritance – Displays all the elements that are inherited from the selected element.
   - Logic Reference – Displays all the logic referenced to the selected element.

   When you click the item in the displayed list, it navigates to the desired location. For example:
   - Dependencies – Navigates to the dependencies tab of the selected element. Simultaneously the selected element is synchronized in the class view.
   - Inheritance – Navigates to default tab of the inherited element. The inherited element is simultaneously synchronized in the class view.
   - Logic Reference – Navigates to the logic editor and highlights the selected element.

Note: *The following restriction applies to the use of this feature:*

When you search for all the references of an instantiated class by selecting it in the Class View or logic editor, the number of records (result) displayed in the Find Symbol Results window is less than the actual number of occurrences that an instantiated class member is used in the logic editor. For example, if Segment1 contains Group2 with Multiplicity = 1, Group3 with Multiplicity =1, and has a method, Method1 added to it, the Find Symbol Results window displays one record when you include the following logic statements in Method1 and try to use the feature by selecting Group2:

```
Move Group2.Attribute1 Group2.Attribute2
Move Group2.Attribute1 Group3.Attribute1
```

But, if you select the instantiated class member, Attribute1 in the Class View or Group2.Attribute1 in the logic editor and then use the feature, the Find Symbol Results window displays both the records.

Searching an Element by Name

The Class View enables you to search an element by name.

To search by name, perform the following:

1. Select a previous search string or type full or part of the search string in the Search String field of the Class View toolbar (or Object Browser toolbar).
2. Click **Search**.

   The search result displays the elements whose names match the Search String. The search string is highlighted in each name where the name has matched.
3. Click **Clear Search** to clear the search or delete the string from the Search String field.

   This synchronizes the Class View with the selected string.

Displaying Inheritance Structure

Inheritance structure helps you to view, explore, and analyze the class and its generalization relationship. This feature is available through Class View and the Object Browser. It enables you to view the superclass and its derived classes of the currently selected class in the **Class View** tree or **Object Browser**. It also enables you to view the inherited members of the currently selected superclass.

To display superclass (base class), perform either of the following:

1. Open your model database in the integrated development environment (IDE).
2. In **Class View**, click the **Class View Settings** menu.
3. Select **Show Base Types**.

Or

1. Open your model database in the integrated development environment (IDE).
2. In the **Object Browser**, click the **Object Browser Settings** menu.
3. Select **Show Base Types**.

For each node in the tree that is a class, a Base Types sub node appears that contains a list of all its base classes.

A derived class inherits the members from the base class. To display derived classes, perform either of the following:

1. Open your model database in the IDE.
2. In **Class View**, click the **Class View Settings** menu.
3. Select **Show Derived Types**.

Or

1. Open your model database in the IDE.
2. In the **Object Browser**, click the **Object Browser Settings** menu.
3. Select **Show Derived Types**.

To display inherited members, perform either of the following:

1. Open your model database in the IDE.
2. In **Class View**, click the **Class View Settings** menu.
3. Select **Show Inherited Members**.

Or

1. Open your model database in the IDE.
2. In the **Object Browser**, click the **Object Browser Settings** menu.
3. Select **Show Inherited Members**.

When you select a class in the tree, the inherited members are displayed in the members pane along with the members defined within that class.

Displaying Inheritance Graphs

This feature is available through Class View and the Object Browser. It enables you to view the base types and derived types of the current type that is selected in the **Class View** tree or **Object Browser**. It also enables you to view inherited members of the currently selected type.

To display base types, perform the either of the following:

1. In Class View, click the **Class View Settings** menu.
2. Select **Show Base Types**.

Or

1. In the Object Browser, click the **Object Browser Settings** menu.
2. Select **Show Base Types**.

For each node in the tree that is a type, a Base Types sub node, that appears, contains a list of all the base types of a type.

To display derived types, perform the either of the following:

1. In **Class View**, click the **Class View Grouping** menu.
2. Select **Show Derived Types**.

Or

1. In the **Object Browser**, click the **Object Browser Settings** menu.
2. Select **Show Derived Types**.

For each node in the tree that is a type, a Derived Types sub node, that appears, contains a list of all the derived types of a type.

To display inherited members, perform the either of the following:

1. In **Class View**, click the **Class View Grouping** menu.
2. Select **Show Inherited Members**.

Or

1. In the **Object Browser**, click the **Object Browser Settings** menu.
2. Select **Show Inherited Members**.

When you select a type in the tree, inherited members are displayed in the members pane along with the members defined within that type.

Note: *The entries Show Project References and Show Hidden Types and Members are not applicable for an AB Suite project.*

Object Browser

Like Class View, the Object Browser enables you to view all the elements in your model. However, you cannot perform any updates. This browser has three panes:

- The Objects pane displays the container elements as a tree view.
- The Members pane displays certain contained members of a container element selected in the Objects pane.
- The Description pane displays details about an element selected in either of the other panes.

Properties Window

The Properties window displays the properties of a selected element and enables you to modify those properties, unless they are read-only. It also enables you to select multiple elements and displays the common properties. Refer to Setting Properties for more information on specific element properties.

Document Window

The document window hosts the various designers that make up the Developer specific interfaces. A document window is displayed for each open element. The available designers for the System Modeler specific interfaces change according to the kind of element that is opened, and its properties.

When you double-click an element in Class View, Object Browser, or the Members tab a Visual Studio document window is displayed for that element. At the bottom of this window is a number of tabs, which differ according to the kind of element selected. You can also display the window by right-clicking the element in the Class View or Members tab and then clicking **Open**. Both these methods open the element in the window which is set as the default for the particular type of element.

Alternatively you can right-click the element and click **Open With** to choose an available window in which to open the selected element.

To change the default window for a type of element, perform the following:

1. Right-click the element in **Class View**, or **Members** tab.
2. Select **Open With** and click **Set Default...**.
3. Select the required window from the list of windows in the dialog box and click **Set as Default**.

   All subsequently opened elements are displayed in the chosen default window. The list contains only windows that are appropriate for the selected element type. For a list of these windows, refer to the topic Document Windows in Developer.

Note: *In case of methods, the Logic Editor opens as a separate Microsoft Visual Studio document window and other tabs such as, Properties, Members, Translations, Dependencies, and Documentation open in another document window. You can open both these windows at the same time.*

The following are the various document windows in system modeller:

Logic Editor Window

The Logic Editor is used to create, edit, save, and validate logic for all methods within System Modeler.

Common Tabs

When you open an element, a document window is displayed for that element. At the bottom of this window are a number of tabs as listed below, which differ according to the kind of element selected.

Dependencies

This displays the dependencies of the selected element.

Documentation

The Documentation tab is common to all System Modeler elements. It is a rich text editor that enables you to document the elements in your system. It supports all the common text formatting and functions.

Member

The Member tab is displayed for those elements that contain child elements that are accessible from their owner. This tab displays all the children of the selected element. You can open the member elements, add new items, add existing items to folders, and delete member elements from this tab.

Properties

The Properties tab is common for all System Modeler elements. It displays the name, author, description, and in some cases the inheritance of the element.

Element Specific Tabs

Conditions

The Conditions tab is specific to profiles. It is a text editor used to edit, save, and validate conditions that determine the records selected by the profile.

Keys

The Keys tab is specific to profiles. It displays attributes that have been defined as keys for the selected profile.

Overrides

The Overrides tab is specific to classes. It displays the methods of the selected class that overrides methods in its super class (including framework methods).

Painter

The Painter tab is available for those elements whose PresentationType property is set to any value other than None. This tab enables you to design the user interface of an AB Suite application.

Note: *You cannot access the Painter tab when you are creating an AB Suite Client Framework model. You must design the user interface separately from the Client Framework model by using the development tool of your chosen technology. For example, you can use the WPF Designer for WPF/XAML desktop applications. However, both the AB Suite and WPF projects can exist in the same solution to impart an integrated development experience.*

Reservations

The Reservations tab displays a list of all the elements within the model that are reserved, and their corresponding users.

Translations

The Translations tab displays a split window. One view shows all the captions and string that are capable of being translated into other languages. The other view displays all the defined languages available, and allows you to add other languages.

Class Diagram

The Class Diagram tab is specific to diagrams. It allows you to view and manipulate the existence and nature of static relationships between member classes.

Value-Checking

The Value-Checking tab is specific to all primitive objects (attributes, variables and parameters) and classes. It is a text editor used to edit, save, and validate constraints that restrict the values that an attribute or type can have.

Conditions Tab

The Conditions tab is used to create, edit, save, and validate profile conditions. Profile conditions specify the criteria used to select a subset of the database records associated with a persistent class of which the profile is a member.

To open the Conditions tab for a profile, either:

- Double-click the profile in the Class View, Object Browser, or Members tab and then select the **Conditions** tab.
- Right-click the profile in the Class View or Object Browser, click **Go to Definition**, and then select the **Conditions** tab.

To set display, general, and tab options, refer to Setting Fonts and Colors.

Class Diagram Tab

The Class Diagram tab is specific to diagrams. It allows you to view and manipulate the existence and nature of static relationships between member classes. Any changes made to the corresponding fields in the Properties window are reflected on the Class Diagram tab, and vice versa.

Refer to Using Class Diagram Editor for more information on the Class Diagrams.

Dependencies Tab

The Dependencies tab displays the relationships of the selected element to other elements in the model. Dependencies can be created implicitly, for example, by referring to an element in logic. Alternatively, Lookup type dependencies can be created by adding elements directly to this tab.

Dependency relationships are described in terms of a client, the element defining the dependency, and a supplier, the element the client depends on.

Depends on

This list displays the elements on which the selected element depends.

The following table shows the dependency relationships that may be displayed for selected elements:

Dependency	Type	Client	Supplier	Description
Key		Profile Parameter	Attribute	Defines the ascending or descending order of persistent attributes that act as keys to a profile. Key dependencies become members of their client.
Auto	Lookup	Ispec	Attribute	Ispec classes can be defined to automatically look up other classes based on the values of one or more of their attributes. This dependency can be added manually on the Dependencies tab.
Auto	Persist	Event	Attribute	The client persistent attributes are made to persist in an attribute of another class. The dependency is valid when all persistent attributes of the client are keyed to persistent attributes of the supplier. A client can have only one Auto Persist dependency. Specifying the Auto Persist property for the selected event sets this dependency.

Dependency	Type	Client	Supplier	Description
Use		Method	Object	Identifies the usage of an object within a logic statement. A Use dependency becomes invalid when the supplier is modified after the dependency is created.

Lookup Dependencies

Lookup dependencies for ispecs are the only dependencies that can be manually added to this list:

1. Right-click anywhere in the list and select **Add Lookup**.
2. Select the required attribute in the Dependency Picker.
3. Click **OK**.

The unresolved keys are also displayed in the list for the Lookup dependencies.

To rectify the unresolved keys, perform the following:

1. Select the unresolved attribute in the Depends on list.
2. Select the required client attribute using the Element Picker in the Properties window.
3. Click **OK**.

AutoLookUp Properties

The table below lists all properties for the LookUp dependency:

Property	Function
Client	Read-only. Identifies the dependent element. Reflected as the 'Source' column on the Dependencies tab. The following elements can only be defined as Client and the elements must have their 'PresentationType' property set to a value other than None: • Ispec • CopyIspec • Event • CopyEvent
Description	Specifies a short description of the selected element.
IfPresent	Specifies whether to look up the object if the value for the key is defined. If set to True, the profile becomes conditional and returns a view instead of an index. ***Note:*** *This property becomes visible only when a key is selected.*
Kind	Read-only. Identifies the kind of element selected.

Property	Function
Supplier	Read-only. Identifies the element that satisfies the dependency. Reflected as the 'Target' column on the Dependencies tab. The following elements can only be defined as Supplier and the elements must contain attributes that are defined as keys: • Ispec • CopyIspec • Class ***Note:*** *The 'Event' and 'CopyEvent' elements cannot be defined as Supplier as these elements cannot contain a keyed attribute.*
Status	Specifies whether the dependency is valid.

Depended on by

This list displays the elements that are dependent on the selected element. It is for display purposes only. Elements cannot be added or deleted from this list.

Dependency Picker

The Dependency Picker enables you to define dependency relationships with selected elements.

The tree structure displays the elements defined in your model. All elements in the model are included, although only those that satisfies the requirements of the selected dependency relationship can be selected. Refer to Dependencies Tab for more information on dependency requirements.

Documentation Tab

The Documentation tab enables you to add text to any element in your application. You can use this tab to document the design and intent of your application, to list details specific to an element, or include any information relevant to your application.

The Documentation tab is a rich-text editor that supports all common text formatting and functions, including hyperlinks. The Rich Text toolbar is displayed above the window when the Documentation tab is opened. This toolbar contains all the formatting options for your text or OLE objects.

The Insert menu enables you to add page breaks also the date and time to your text.

You can also select the font options for text displayed in this window.

To set the default font options for this window, perform the following:

1. From the **Tools** menu, select **Options**.
2. In the Options dialog box, navigate to the Environment, Fonts and Colors folder.
3. From the **Show Settings for** list, select **System Modeler Documentation Editor**.

4. From the **Display Items** list, select **Documentation**.
5. Select the font type, size, and foreground color.

Note: *You cannot select the background color. The system adopts the Windows default color.*

Date and Time Dialog Box

The Date And Time dialog box enables you to select the format of the current date and time to be inserted in your text. To insert date and time:

1. In your text, click the position you want the date and time inserted.
2. From the **Insert** menu, select **Date and Time**.
3. Select a format from the list.
4. Click **OK**.

The date and time is inserted at the cursor position.

Inheritance Tab

The Inheritance tab is specific to the model. It displays the relationships of all classes in a tree structure. Superclasses are expandable, containing the defined subclasses.

Note: *Pasting and dropping to the Inheritance tab sets new subclass relationships, not ownerships.*

Keys Tab

The Keys tab is specific to profiles.

A key is an attribute of an ispec or event that acts as the unique identifier of one record from another. It is the access path to individual ispec or profile records.

To define an attribute as a key, perform the following:

1. Right-click in the tab and select **Add Key**. The **Select new key** dialog box contains the attributes of the element to which the profile belongs.
2. Select an attribute, and then click **OK**.
3. The attribute is added to the key list as the last in the sequence.

You may delete, replace or validate any key in the list, by right-clicking the key and selecting the appropriate action from the menu.

Logic Status Tab

The Logic Status tab is specific to elements that can contain methods. It displays a list of methods both directly and indirectly contained in the element, and their respective logic validation status.

Note: *Empty methods (those that do not contain any logic text) are not listed.*

To validate methods, perform the following:

1. Select the desired method(s) in the Logic Status tab.
2. From the **Build** menu, select **Validate**.

Note: *Methods can also be validated from the Logic Editor.*

Refer to Click Show potential fixes link. for more information on validation.

Filters

The Logic Status tab contains a Filter toolbar from which you can specify the members to be displayed. The Logic Status List filter is the default used for the Logic Status tab.

Note: *When the Logic Status List filter is applied, method members of insertable classes are not displayed, as they generally cannot be successfully validated in isolation from their inserted context. Refer to Validating Insertable Classes in Isolation for more information.*

Click the **Customize Filters** toolbar button to display the Filter Definitions dialog box and create customized filters.

Members Tab

The Members tab is displayed for those elements that contain child elements (or members), which are accessible from their owner. It displays all the children of the selected element. Using the Members tab, you can:

- Open member elements
- Add new elements
- Add existing elements to folders
- Delete member elements
- View a filtered list of members and their properties

The Members tab also displays the method members and their parameters. This enables you to view the structure of the method.

If you right-click anywhere on this tab you can add new items or add existing items to folders. You can also open, delete, and edit attributes of member items.

Note: *On selecting any element in the Members tab, ensure to highlight that element and then invoke the relevant online help for more information.*

Inherited Members

You can choose to show, or hide members that are inherited from other members. To change this option, from the Filter Definitions dialog box. When inherited, members are displayed in a different color to distinguish them from normal members.

Objects

Objects are displayed in the Members pane as an expandable list. Members of objects cannot be sorted, but can be rearranged by dragging and dropping, or by changing the sequence numbers in the property window.

To add an element to an object without opening it, right-click the element and select **Add New Item to <selected item name>**.

Filters

The Members tab contains a Filter toolbar from which you can specify the members to be displayed. The Default filter displays all members.

Click the **Customize Filters** toolbar button to display the Filter Definitions dialog box and create customized filters.

You can select the text foreground colors for the Members tab window.

To set the text color options for this window, perform the following:

1. From the **Tools** menu, select **Options**.
2. Navigate to the Environment, Fonts and Colors folder in the Options dialog box.
3. From the **Show Settings for** drop-down list box select Members Pane.
4. From the **Display Items** list box select the text item for which you want to specify color.
5. Select the foreground and back color of the selected item.

Filter Definitions Dialog Box

The Filter Definitions dialog box enables you to create and modify filters for the Members tabs. Click **Categorized or Alphabetic** to change the display order.

Filters List

Displays the currently defined filters. A number of preselected filters are included for you to use. These may be modified, deleted or added to.

Add Button

Adds a new filter to the Filters list with the default name Filter<x>. The name can be changed immediately in the list or later using either the Rename button or the Filter Name parameter. If the default name is not changed, <x> increments by one for each new filter added.

Make a Copy Button

Creates a copy of the selected filter's parameter settings. The new filter is added to the Filter list with the name of the original filter appended by a number, which is incremented as required to ensure uniqueness. You can enter a different name for the newly created filter.

Rename Button

Enables you to modify the name of the selected filter in the Filters list.

Delete Button

Deletes the selected filter.

Default Filter

Enables you to set the filter that the Members tab uses by default.

Parameters

To modify the values of a parameter, select the parameter and click the displayed button. This displays either a list of options or a dialog box.

The following table describes the parameters you can set for a filter:

Parameter		Description
Columns		Specifies the columns displayed on the tab, using the Select Columns Editor.
Filter Name		Specifies the name of the filter.
MembersSelection		Specifies the elements to be displayed in the tab, using the Select Members Editor.
PropertySelection		Specifies which elements are included in the filter based upon the properties selected.
	Direction	Specifies which elements with the specified Direction property are included in the filter. Select None, In, Out or IO.
	HasPresentation	Specifies elements with a PresentationType property other than 'None' are included in the filter. Select True to display elements which have a presentation.
	IsKey	Specifies which elements with the specified Is Key property are included in the filter. Select No, Descending or Ascending.

Parameter		Description
	IsPersistent	Specifies which elements with the specified Is Persistent property are included in the filter. Select No, Yes or From Owner.
	Type	Specifies which elements with the specified Type property are included in the filter. Select Class, Number, Signed Number, String, Boolean, Date, Mixed String or Wide String.
	Visibility	Specifies which elements with the specified Visibility property are included in the filter. Select Private, Protected or Public.
ShowInherited		Specifies whether the elements to be displayed should include elements inherited from other elements. Select True to display inherited elements.
ShowWithinFolders		Specifies whether elements contained in folders are displayed. Select True to display elements contained in folders.
InitialSortOrder		Displays the default sort order of the tab, which is defined using the Column and Direction sub-parameters.
	Column	Specifies the default column on which to sort the tab.
	Direction	Specifies the default direction sort order of the tab.
SecondarySortOrder		Displays the secondary sort order of the tab, which is defined using the Column and Direction sub-parameters.
	Column	Specifies the default column on which to sort the tab.
	Direction	Specifies the default direction sort order of the tab.

Select Members Editor

Use the Select Members Editor to specify the elements to be displayed in the Members tab.

The dialog contains a list of all the available System Modeler element kinds and stereotypes.

Select the check box for each element to be displayed in the Members tab when the filter is used.

Select Columns Editor

Use the Select Columns Editor to select and order the columns displayed on the tab.

To add a column, perform the following:

1. Select the required column from the **Available Columns** list.
2. Click **Add**.

3. Use the **Move Up** and **Move Down** buttons to position the column in the required position in the Selected Columns list.

If a column is added that is not appropriate for the selected members displayed in the window, a blank column is added.

Overrides Tab

The Overrides tab is specific to classes. You can use this tab to override methods inherited from the superclass of the selected element.

When a class is defined as a subclass it automatically inherits the methods of its superclass. By overriding a method you are providing a new implementation of the base method with the same name and parameters as the overridden one.

The **Add New Override** list displays the methods that are available to be overridden. Selecting a method adds it to the list window, to the Members tab, and to the Class View.

Note: *Cut, copy, paste, and drag and drop have not been implemented in this tab.*

Painter Tab

Painter is used to design and construct screen interfaces, with which your user interacts with your deployed application. It allows you to add simple graphical objects, such as lines and images ,and add graphical objects, such as edit boxes and radio buttons, which are bound to an attribute as its data source.

Any named element can have a form created for it when its ‘PresentationType’ property is set to a value other than None, with the exception of a model or folder.

A form grid provides horizontal and vertical guidelines to help you align objects on your form.

To select the default form grid settings, perform the following:

1. From the **Tools** Menu, select **Options**.
2. Navigate to the Windows Forms Designer, General folder in the Options dialog box.
3. If it is not already expanded, click the plus (+) symbol next to the Grid Setting.
4. If it is not already expanded, click the plus (+) symbol next to the Grid Size enter the appropriate values.
    - **Width** – Sets the distance between horizontal grid lines. The default value is 8 pixels with a setting in the range of 2 to 200.
    - **Height** – Sets the distance between vertical grid lines. The default value is 8 pixels with a setting in the range of 2 to 200.

5. Set **Showgrid** to True, so that by default when a form is opened the grid is displayed, unless the form has been saved with a different setting.
6. Set **Snap to grid** to True so that snap to grid is enabled by default when a form is opened, unless the form has been saved with a different setting.

Or

WPF Designer

WPF Designer is used to design user interfaces for the WPF Client applications by using the AB Suite Access Layer. It allows you to add attributes and graphical objects, such as text boxes, combo boxes, and list boxes from the Data Source window and WPF toolbox onto the WPF Designer. An automatic binding is established between the control and the attribute in the ViewModel. The binding mechanism in WPF allows data to be exchanged automatically between a control in the user interface and the corresponding attribute in the ViewModel.

You can also modify the generated XAML code manually in the XAML view to manipulate the behavior of the graphical controls according to your requirements.

Reports and Teach Screens

The painter is also used to create teach screens or reports. In these cases there are limitations on the type of graphical objects and fonts types, which can be used on these types of forms.

To set the default font for reports, perform the following:

1. From the **Tools** menu, select **Options**.
2. Navigate to the Environment, Fonts and Colors folder in the Options dialog box.
3. From the **Show Settings for** drop-down list box select System Modeler Report Painter.
4. From the **Display Items** list box select **Text**.
5. Select the Font type, size, and foreground color.

Note: *The font type in reports is restricted to fixed width, if any other font is selected a warning is given so that you can select the correct type, otherwise the last valid selection is retained.*

A similar restriction exists for the font foreground color which can only be Black(Automatic), White, Blue, Yellow, Red, Cyan, Green or Magenta. If you select any other color a warning is given and the last valid color selection is retained.

Refer to the *Agile Business Suite Developer Online Help* for more information.

Profile Data Tab

The Profile Data tab is specific to profiles. A Profile Data element is one which is physically stored in a Profile and its associated Ispec Class.

Through this tab you can create or maintain Profile Data elements.

To add a profile Attribute, perform the following:

1. Right-click in the Profile Data tab and select **Add Data**. The **Select new data** dialog box contains the attributes of the element to which the profile belongs.
2. Select an attribute, and then click **OK**.
3. The attribute is added to the data list as the last in the sequence.

   You may delete, replace or validate any attribute in the list, by right-clicking the item and selecting the appropriate action from the menu.

Properties Tab

Each model entity has a Properties tab. For all entities this tab displays a Name and Description field, and in some cases an Author field. Any changes made to the corresponding fields in the Properties window are reflected on the Properties tab, and vice versa.

For classes, the Properties tab also displays the following fields:

- Superclass
- Subclass

Refer to Setting Properties for more information on these fields.

Subclasses

The Subclasses list displays the classes that inherit from the selected class. It also displays nested subclasses.

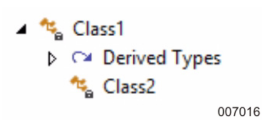
Add new or existing classes directly to this list by:

- Right-clicking and selecting Add New Item or Add Existing Item as appropriate.
- From the File menu, selecting Add New Item or Add Existing Item as appropriate.

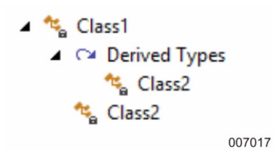
To add a subclass to a class displayed in the list, select a class and use one of the above processes.

Displaying Superclasses and Subclasses

Subclasses display their superclasses in Class View and the Object Browser as part of the tree hierarchy, as shown in the following examples:



Class1 expands to identify that it has a superclass defined.



Further expansion shows that Class2 is defined as a superclass to Class1.

Quick Navigator Window

The Quick Navigator tool window is an editor in which you can type in a qualified element name and navigate to this element.

The qualified element name is relative to the scope of the currently selected element in the Class View, or the Designer Editor, whichever has focus, within the Quick Navigator Window.

In the case of multiple selected items, the scope element defaults to the last item selected. The scope element can only be one element.

You can open the Quick Navigator Window either by the **View Menu** > **Other Windows** > **Quick Navigator Window**, or using the Alt+G keys.

To perform a quick navigation, perform the following:

1. Type the qualified name of the required element in the Quick Navigator Window (Quick Navigator supports autocomplete).
2. Right-click the qualified name of the element and select **Go To**.

This opens the Designer Editor on the default tab for the element. The scope name is displayed in the window title to easily identify the scope of the element.

The window supports auto-completion. Auto completion and navigation use the scope element to retrieve the required information. Auto-completion is initiated when a '.' (period) is entered or the Ctrl+Space key combination is used.

You can also supply the name or a parameter to open the element in a specific tab. For this you must enter a '/ ' (forward slash) followed by a two letter identifier as shown in the table below. If there is no parameter supplied, the element is opened in its default tab.

Identifier	Parameter
pr	Properties
me	Members
in	Inheritance
de	Dependencies
la	Language
lo	Locks

Identifier	Parameter
lg	Logic
ls	Logic Status
mp	Parameters
mo	Overrides
uml	Class Diagram
pa	Painter
pc	ProfileConditions
pk	ProfileKeys
vl	ValueChecking
do	Documentation
un	Unresolved

Auto-completion is also available to assist in entering the parameters. When the forward slash is typed, the auto-completion list opens with the list of parameters. The list includes the name of the tab that is displayed.

Within the window, there is a context menu, which contains the following items:

- Go to – execute the current line.
- Set Scope to – change the scope for the selected item.
- Synchronize class view – synchronizes the selected item with the class view.
- Cut, Copy, Paste – standard clipboard operations.
- Select All – select all the text in the window.

The window saves its contents to the .suo file when the window is closed and the contents restored when it is reopened.

Reservations Tab

To allow an Administrator or a user, to see which elements are reserved and by whom, a Reserved List is available when a Model is selected. This list shows elements that have been reserved within the model, the type of element, and the name of the user (or the system) that has reserved it.

By selecting an element in this list, you can unreserve or remove the reservation on the element. The Unreserve operation is performed on elements that are not under source control and the Remove Reservation operation is performed on elements that are under source control.

The unreserve operation can be carried out by the current user who has reserved the element, or by an Administrator who can unreserve elements reserved by any user. The remove reservation operation can be performed by the administrators only.

To display the Reservations tab, perform the following:

1. From the Class View window, select a Model.
2. From the right-click context menu, select the **Open** menu item and select **Reservations**, or select **Open**.
3. Click the **Reservations** tab, which is now displayed at the bottom of the document window to display the list of reserved elements.

Note: *If you reserve an element while the Reservations list is displayed, you must select the View, Refresh menu item to update the list.*

To unreserve an element that is not under source control, perform the following:

1. Select an element, or multiple elements, in the list.
2. Right-click the selection to display the context menu.
3. Select **Unreserve** or **Unreserve Selections** as appropriate, to unreserve the selection.

To remove reservation on an element that is under source control, perform the following:

1. Ensure that the TFS Administrator has performed an undo operation on the element that has been checked out by another user in the Source Control Explorer. Refer to Checking Out an Element for more information.
2. Open an AB Suite project and ensure that the user has the database db_owner privilege.
3. Double-click the Model node.

   The document window appears.
4. Click **Reservations** tab, at the bottom of the documentation window.

   A list of locked element appears in the documentation window.
5. Right-click the element and select **Remove Reservation…** from the context menu.

Refer to Reservations Tab and Element Reservation for more information on element reservation.

Translations Tab

The Translations tab is available for all namespaces from the Model level down to an attribute and displays a split window of two panes.

Upper Pane

The upper pane shows all the captions and strings that can be translated into other languages, and a column for each language defined by you. The columns can be re-ordered, hidden, or added. To translate a value for a language, simply click in the appropriate column for a language and enter the translation in the entry field.

To remove a translation for all elements for a given language, select any element, right-click in the column for the language to be removed, and then select **Clear Translations** for <language> from the context menu.

Lower Pane

The lower pane displays all the defined languages available and the inheritance relations between languages. In this panel, languages can be added, deleted, or renamed and their relation to another language can be changed. When a language has an inheritance relation with another language (a Derived Language), it means that any value that is not translated for that language inherits the value from the primary language in the tree. The model creates and defines a default primary language that cannot be deleted, though it can be renamed.

To add a derived language, perform the following:

1. Right-click the primary language in the lower pane, and then select **Add Language** from the context menu.

   The **Add Language** dialog box appears.

2. From the **Name** list, select a language.

   The corresponding locale code and name appears in the **Locale** list.

   This list includes all the languages installed in the system. Note that this list includes languages that are additional to the ones available in the Windows Regional Options.

3. Click **OK**.

   The new language is added under the primary language.

   You can double-click the resources.<LocaleName>.resx file to view the labels and the translated value of the label that would appear in the user interface.

Session Language

You can set a **Session Language** through the combo box **Language** on the toolbar. The session language is used by all views that display language depended information, such as values and captions.

The Painter window uses the session language to display the presentation for that language, and ‘remembers’ this language. When the language changes, the Painter prompts you to save any changes. Painter remembers the language. You can select different language for different Painters for elements.

When a window becomes active, it updates the Language combo box with either the language remembered by the window (in the case of the Painter window) or it sets it to the session language.

Unresolved Tab

The Unresolved tab is displayed in the model whenever there is an element in the model with its IsResolved property set to False. Refer to Unresolved Elements for more information.

All Unresolved elements are listed in the window. To resolve elements, right-click a single element or multiple elements and choose Resolve. Once an element is resolved it is no longer displayed on this tab, and cannot be made unresolved again.

You can select the text foreground colors for the Members tab window.

To set the text color options for this window, perform the following:

1. From the **Tools** menu, select **Options**.
2. Navigate to the Environment, Fonts and Colors folder in the Options dialog box.
3. From the **Show Settings for** drop-down list box, select **Unresolved Pane**.
4. From the **Display Items** list box, select the text item to specify color.
5. Select the foreground and background color of the selected item.

Value-Checking Tab

The Value-checking tab is used to create, edit, save, and validate value-checking logic. Value-checking logic is used to constrain the value that an attribute (or type) can have. Although it is available for all attributes and types, it only applies to input from the user interface – attributes with their Direction property set to In or IO, and correspondingly, their owning classes must also have their PresentationType property set a value other than None.

Value-checking logic is evaluated as part of the automatic validation stage of the segment cycle, and an error is returned to the application client if the input value of the attribute is not validated successfully.

Value-checking logic is defined as conditional expressions evaluating the current value of the attribute (dynamically qualified as ‘this’) against another value, which can be a set expression.

To open the Value-checking tab for an attribute or type, either:

- Double-click the attribute or type in the Class View, Object Browser, or Members tab and then select the **Value-checking** tab.
- Right-click the attribute or type in the Class View or Object Browser, click **Go to Definition**, and then select the **Value-checking** tab.

To set display, general, and tab options, refer to Setting Fonts and Colors. However, the only settings that affect the Value-checking tab are:

- Line number color
- Line number display

- Tab size
- Insert spaces and Keep tabs

Value-checking logic examples:

Example 1

This value-checking logic specifies that the value of the attribute must not be equal to 45, nor can it be greater than 90 (its value multiplied by 2 must be less than 180).

```
this NOT= 45 && this * 2 < 180
```

Example 2

This value-checking logic specifies that the value of the attribute must be greater than 0.

```
this > 0
```

Example 3

This value-checking logic specifies that the value of the attribute must be one of 'A', 'C', or 'Z'.

```
this in {"A", "C", "Z"}
```

Example 4

This value-checking logic specifies that the value of the attribute must not be one of 'A' or 'B'.

```
this NOT in {"A","B"}
```

Example 5

This value-checking logic specifies that the value of the attribute must be in the range 'A' to 'Z', but not one of 'M', 'N', or 'O'.

```
this in {"A" .. "Z"}-{"M" .. "O"}
```

Example 6

This value-checking logic specifies that the value of the attribute must be in the combined range of 'A' to 'Z' and 0 to 9.

```
this in {"A" .. "Z"} + {0 .. 9}
```

Customizing Document Windows

Document windows that display their contents in a list such as the Members, Keys, and Dependencies windows can be customized to sort and display their contents selectively.

To customize the window display, right-click a column heading to display a context menu. The following table identifies the display options on the menu. Any display option is restricted to the column that you selected.

Menu	Description
Best Fit	Sets the width of the column to the widest text displayed.
Column Chooser	Displays the Selected Columns dialog box in the Members window only.
Customize Current View	Displays the Filter Definitions dialog box.
Filtered	Displays the Filter Bar.
Member Chooser	Displays the Selected Members dialog box in the Members window only.
Remove This Column	Removes the selected column header.
Sort Ascending	Sorts the selected column in Ascending. (Select this + holding the Ctrl-Key, performs multiple sorting).
Sort Descending	Sorts the selected column in Descending. (Select this + holding the Ctrl-Key, performs multiple sorting).

Filter Bar

The Filter Bar allows the entry of text in one or more columns for which the entries in the columns are matched. Click the button adjacent to each text box to display a menu which allows the filter for that column to be cleared, or allows to specify if the selection should match the case of the text entered. The following operands !, <>, <, <=, >, >=, = precedes the text that we need to filter on in the Filter Bar.

Further customization of the window contents can be achieved using the Filter Definitions dialog box.

Setting Fonts and Colors

The following options control the look and feel of all LDL+ logic editors including Profile conditional logic, Value-checking logic, and Method logic.

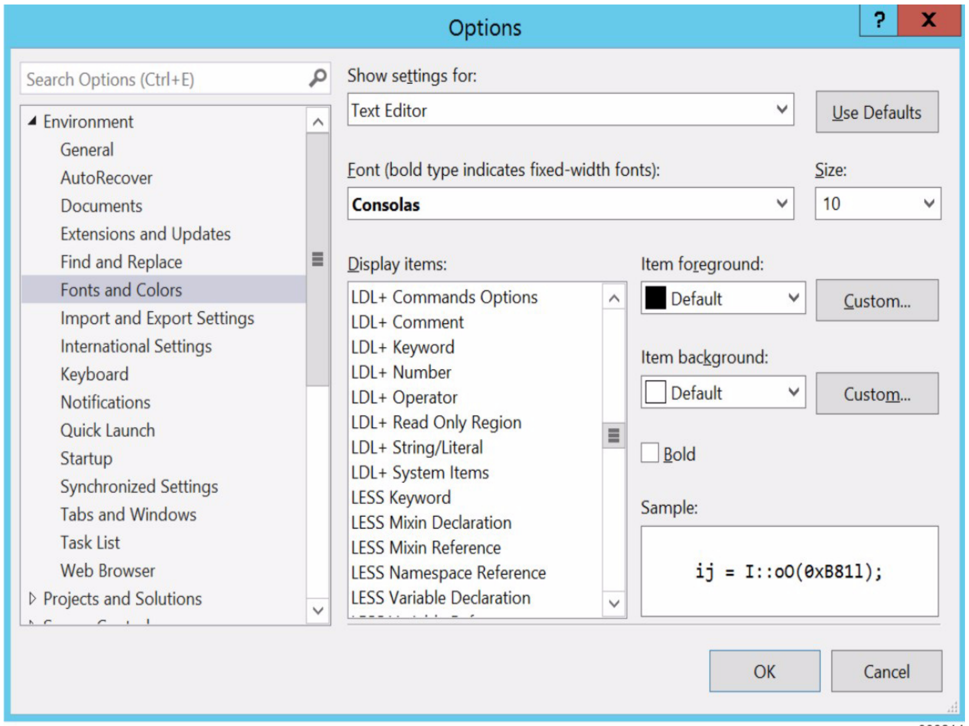
To set the font and color for display items, perform the following:

1. From the **Tools** menu, choose **Options**.

   The **Options** dialog box appears.

2. In the left pane, expand **Environment** and select **Fonts and Colors**.

3. From the **Show settings for** list, select **Text Editor**.



Show settings for: Text Editor

Font (bold type indicates fixed-width fonts): **Consolas**

Size: 10

Item foreground: ■ Default

Item background: □ Default

□ Bold

```
ij = I::oO(0xB81l);
```

008214

4. From the **Display items** list, select a logic element type. The following table lists some of the logic element types:

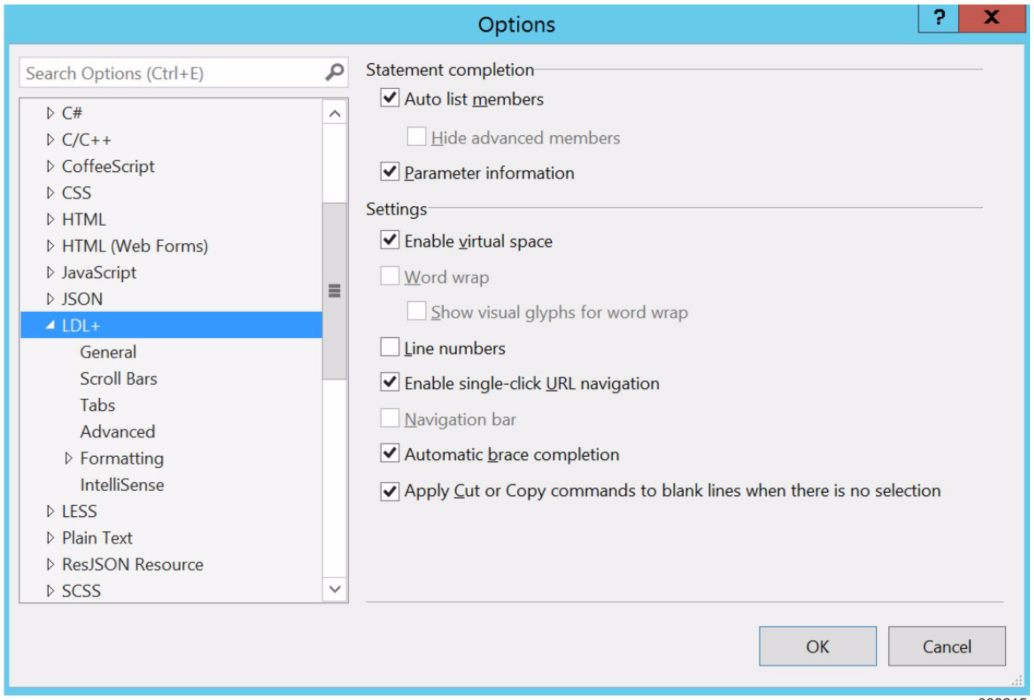
Logic Element	Description
LDL+ Commands	Logic commands. For example, Move, Lookup
LDL+ Commands Options	Options for logic commands. For example, Giving.
LDL+ Comment	Phrases in the text that are not part of the LDL+ logic. For example, logic preceded by colon (:).
LDL+ Keyword	LDL+ logic keywords. For example, If, End, Loop.
LDL+ Number	LDL+ number literals. For example, 123.45.
LDL+ Operator	LDL+ logic operators. For example, (), +, :=.
LDL+ Read Only Region	Read only region in the editor
LDL+ String/Literal	String literals. For example, "First Name"

Logic Element	Description
LDL+ System Items	Built-in segment, ispec, and event attributes. For example, MAINT, ISPEC, GLB.
Text	Names of AB Suite modeled elements. For example, variables (myCustomer) and methods(GetName()).

5. Select the Font, Size, Bold, and the Item foreground, and Item background colors for the selected display item.

Setting LDL+ Editor Options

The following diagram illustrates the options applicable to LDL+ logic:



Statement completion
- [✓] Auto list members
- [ ] Hide advanced members
- [✓] Parameter information

Settings
- [✓] Enable virtual space
- [ ] Word wrap
- [ ] Show visual glyphs for word wrap
- [ ] Line numbers
- [✓] Enable single-click URL navigation
- [ ] Navigation bar
- [✓] Automatic brace completion
- [✓] Apply Cut or Copy commands to blank lines when there is no selection

008215

Setting General Options

To set general options, perform the following:

1. From the **Tools** menu, choose **Options**.

   The **Options** dialog box appears.

2. In the Left pane, expand **Text Editor** and expand **LDL+**.

3. Select **General** to display options listed in the following table:

Display Options	Description
Statement Completion	
Auto List Members	When this option is selected, the logic editor displays a list of context-sensitive member items when a valid element is appended with a period (.).
Parameter information	When this option is selected, the logic editor displays information about the number, names, and types of parameters required by a method.
Settings	
Automatic brace completion	When this option is selected, the logic editor automatically inserts a closing bracket [], parenthesis (), or brace {}.

Note: *Refer to the* Microsoft Visual Studio Online Help *for more information on setting General editor options.*

Setting Advanced LDL+ Options

To set advanced options, perform the following:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box appears.
2. In the Left pane, expand **Text Editor** and expand **LDL+**.
3. Select **Advanced** to display options listed in the following table:

Display Options	Description
Highlighting	
Highlight references to symbol under cursor	When this option is selected, the logic editor highlights references to all instances of the element under the cursor.
Highlight related keywords under cursor	When this option is selected, the logic editor highlights the matching keyword to that under the cursor. For example, Positioning the cursor on a BeginCase will highlight EndCase.
Tool Tips	
Show tooltips on commands	When this option is selected, the logic editor displays information about the command the cursor is positioned over.
Navigation	
Open editor on goto definition.	When this option is selected, opens the logic editor of the target element on a goto definition operation.

Display Options	Description
Statement Completion	
Automatic quote completion	When this option is selected, the logic editor automatically inserts a closing quote.
Automatic block command completion	When this option is selected, the logic editor automatically inserts the terminating command when you start a block statement. For example, When you enter **If**, **foreach**, or **Determine** command, an **end** is automatically inserted.
Quick Action	
Quick Actions for ‘Item not found errors’	When this option is selected, the logic editor displays a light bulb icon when there are potential fixes available to resolve logic errors.
Name matching sensitivity	This option allows you to define the range of elements considered as potential fixes for errors in a misspelled element name. If you set the slider closer to **totally different**, range of elements suggested as a possible match will be high. Conversely, if u set the slider closer to **perfect match**, range of elements suggested will be less. This option is enabled only when **Quick Actions for ‘Item Not Found Errors’** option is selected.
Editor Compatibility	
Set logic editor options to provide an EAE like experience	When you click Set, the LDL+ logic editor will have default settings to provide an experience similar to EAE. A dialog box notifying the settings change appears. • Automatic brace completion • Automatic quote completion • Automatically formatting on enter • Show completion list after a character is typed • Member List Commit Aggressive Click **Yes** to apply settings change.

Setting Formatting Options

To set formatting options, perform the following:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box appears.

2. In the Left pane, expand **Text Editor** and expand **LDL+**.

3. Select **Formatting** to display options listed in the following table:

Display Options	Description
General	
Automatically format on enter	When this option is selected, the logic block is automatically formatted when you press enter. The logic is formatted according to the format defined in the **Command Style** options.
Automatically format on paste	When this option is selected, logic is automatically formatted when pasted into the logic editor. The logic is formatted according to the format defined in the **Command Style** option.
Command Style	
Command Format	Allows you to define the auto-format style for LDL+ commands. The available forms are: • Standard (Uppercase or Lower case). For example, LOOKUP, lookup • Mixed (Camel case) For example, LookUp • Abbreviated (Uppercase or Lower case) For example, LU, or lu
Command Case	Allows you to define the auto-format command case style to Lower case (lookup) or Upper case (LOOKUP).
Do When Command	
Use DoWhen instead of If	Replaces the If command with DoWhen command.
Blank Lines	
Remove Multiple Blank Lines	When this option is selected, multiple blank lines are removed when the logic is auto-formatted.

Setting IntelliSense Options

To set IntelliSense options, perform the following:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box appears.

2. In the left pane, expand **Text Editor** and expand **LDL+**.

3. Select **IntelliSense** to display options listed in the following table.

Display Options	Description
Completion Lists	
Show completion list after a character is typed	When this option is selected, the logic editor automatically suggests a list of commands, keywords and elements based on the characters entered.
Show keywords	When this option is selected, the auto list includes commands and keywords based on the characters you type in the Logic Editor.
Include Abbreviated Keywords	When this option is selected, the auto list includes abbreviated commands and keywords based on the characters you type in the logic editor. ***Note:*** *This option is enabled only if the **show keywords** checkbox is selected.*
Member list commit aggressive	When this option is selected, the logic editor commits the selection from the auto list when you press a commit character. For example, period(.), comma(,), bracket((), <space> in addition to the <tab> and <enter>.
IntelliSense	
Dynamic Validation	When this option is selected, the Logic Editor automatically validates logic as it is entered. Any invalid logic statement is underlined with a red squiggle and the associated error shown in the errors list.

Note: *Refer to the* Microsoft Visual Studio Online Help *for more information on Using IntelliSense.*

System Modeler Documentation Editor

To set the font and color for system modeler documentation editor items, perform the following:

1. From the **Tools** menu, choose **Options**.

   The **Options** dialog box appears.

2. In the left pane, expand **Environment** and select **Fonts and Colors**.
3. From the **Show Settings for** list, select **System Modeler Documentation Editor**.
4. From the Display items list, select **Documentation**.
5. Select the Font, Size, Bold, and the Item foreground, and Item background colors for the selected display item.

System Modeler Member Editor

To set the font and color for system modeler member editor display items, perform the following:

1. From the **Tools** menu, choose **Options**.

   The **Options** dialog box appears.
2. In the left pane, expand **Environment** and select **Fonts and Colors**.
3. From the **Show Settings for** list, select **System Modeler Member Editor** to display options listed in the following table:

Logic Element	Description
System Members	Built-In Attributes, Classes and Methods. For example, ACTMTH, Clear()
Inherited Members	Members inherited from user-defined super classes.
Members	Local Members of the owner. For example, Variables, Attributes, Profiles, Methods.

4. Select the Font, Size, Bold, and the Item foreground, and Item background colors for the selected display item.

Setting Validation Options

To set validation options, perform the following:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box appears.
2. In the left pane, expand **Environment** and expand **Projects and Solutions**.
3. Select from the Build and Run options as listed in the following table:

Display Option	Description
Save all changes	Saves and validates all modified documents.
Save changes to open documents only	Save and validates all modified documents.
Prompt to save changes to open documents	Validates all modified documents, and displays a list, to allow specification of which documents are to be saved.
Don’t save changes to open document	***Note:*** *This option is not supported by System Modeler.*

In multi-processor machines, logic validation can perform faster if multiple threads are used.

To set multi-thread validation options, perform the following:

1. From the **Tools** menu, select **Options**.
2. Navigate to **System Modeler** > **General category**.
3. Select the appropriate values for the Number of Threads for validation.
4. Click **OK**.

Validating Model Structure

The model validation feature enables you to capture validation errors of a model structure during the development phase and resolve them before building the model. Therefore, this feature provides real time error or warnings for the currently active configuration. The error representation process includes the following:

1. Model structure errors – Any changes in the model causes the error to be revalidated. If model structure errors exist, you can see the error in the Error List window during development phase. You are prompted to fix the errors before validating logic and starting the build. Refer to Model Structure Validation Rules for more information on different rules for specific platforms.

   Note: *The description of the warnings and errors appears in the dependency window.*

2. Language validation errors – You can use the Logic Status tab to validate logic for all methods within System Modeler through the Build menu. Refer to Click Show potential fixes link. for more information.

Following are the System Modeler functionalities available with this feature:

- Synchronous errors or warnings in the current session – The model structure errors are displayed instantly during development phase if you violate any validation rules. Refer to Model Structure Validation Rules for more information on different rules for specific platforms.
- Refresh and Multi-session – The model structure errors of a session are automatically displayed in another session after a refresh. For example, if you fix an error in a particular session, clicking Error from the Error list window for the same model in a different session displays a message box that the error no longer exists and is already fixed in another session.
- Auto-receive errors after import – The model structure errors are displayed automatically when you import a model through Model Importer.
- Model structure errors also available in Validation output – All model structure errors are also included in the Validation output window, which is also available during build.

System Modeler Policies

System Modeler Policies allows you to define various policy settings such as, Ask, Don't Ask, and Ignore. Depending on these settings, System Modeler prompts for a confirmation if the Owner property of an element is changed. And depending on these settings, when a new element is created, System Modeler prompts for a change of VersionFile property value.

To access System Modeler Policies page, perform the following:

1. From the **Tools** menu, select **Options**.
2. In the Options dialog box, navigate to **System Modeler**, and then click the **Policies** folder. This displays the Policies list with the following names:

Property	Function
Confirm move on Owner change	This option prompts you to confirm when an element is moved to a new owner. The default setting is **Ask**.
Set the VersionFile Property on creation	This option enables you to set the VersionFile property for newly created **Classes**, **Folders**, **Dictionaries**, or Diagrams. The default policy setting is **Don't Ask** and default value of **Additional Setting** is **All Classes owned by the Model or a Segment.**
Confirm property change impacting build	This option prompts you to confirm when some of the reserved properties of the elements are modified that requires an application rebuilt for a specific platform. These properties include: • Model or Segment properties • Segment configuration properties • Top level deployment folder properties • Top level deployment folder configuration properties For example, AllowedCore, DataSetBuffers reserved properties require an application rebuilt in an MCP environment. The default policy setting is **Ask**, which prompts a confirmation dialog box. You can select, 'No' to cancel the changes in the reserved properties. The **Don't Ask** policy setting does not prompt a confirmation dialog box but the policy is executed. The **Ignore** policy setting neither prompts a confirmation dialog nor executes the policy. This setting cancels the changes in the reserved properties of a Model or Segment.

3. From the Settings list, you can select either of the following for different policy settings:

Settings	Function
Ask	Prompts you whether to assign a value before executing the policy.
Don't Ask	Executes the policy.
Ignore	Does not execute the policy.

4. From the Additional Settings list, you can further select either of the following elements to allow the VersionFile property of those elements to be set on creation.
    - All Containers
    - Model or Segment Owned Container

This list is available only for the policy, **Set the VersionFile Property on creation**.

The policy settings are displayed in the **Policies** list.

Printing

System Modeler allows you to print the contents of a selected tab, or definition of selected element/s.

Note: *If a printer is not configured, all printing menu options are disabled.*

To print in System Modeler, perform the following:

1. Select a tab, or an element.
2. From the **File** menu, select Page Setup and customize the print output using the **Page Setup** dialog box.
3. From the **File** menu, select **Print Preview** to preview your print output.
4. From the **File** menu, select **Print**.

   Note: *The Print and Print Preview functions are also available from the toolbar.*

Restrictions

The following restrictions apply to the following tabs:

- **Members tab** – Printing a selected element does not print its definition, rather it prints the selected line's content. System Modeler allows you to select and print multiple lines.
- **Painter tab** – Prints either Painter's content, or selected controls.
- **Documentation tab** – Prints the entire contents of the tab.

Page Setup Dialog Box

Use the Page Setup Dialog Box to customize the page setup of your print output. When you have completed the fields described below, click **OK**.

Size – Select an appropriate paper size (options may vary between different printers).

Source – Select an appropriate paper source (options may vary between different printers).

Orientation – Select either a Portrait or Landscape orientation layout.

Margins – Enter the left, right, top, and bottom measurements for your page margins.

Header and Footer – Customize the header and footer text that appears at the top and bottom of the page. To print specific information as part of the header or footer, type the following characters as part of the text:

Characters	To include in the header or footer
&P	Product Name/Version
&n	Item Name
&e	Editor tab name
&t	Time in the format specified by the Regional Settings in the Control Panel
&T	Time in 24-hour format
&d	Date in short format (as specified by the Regional Settings in the Control Panel)
&D	Date in long format (as specified by the Regional Settings in the Control Panel)
&p	Current page number
&&	A single ampersand (&)

Buttons

Ok – Applies the preferences you have chosen.

Cancel – Cancels any changes made, and closes the dialog box.

Printer – Displays the Microsoft Windows Print dialog box.

Help – Displays the online help.

Model Entities

System Modeler allows you to create 4GL object-oriented applications. In addition to the inherent benefits of using standard object-oriented elements, System Modeler provides access to a host of stereotyped model entities that possess specific properties and roles within the Agile Business Suite processing cycles.

- Classes
- Methods
- Variables
- Profiles
- Teach Screens
- Dictionary
- Class Diagrams
- Groups
- Locations

Namespaces and Ownership

Namespaces uniquely identify a set of entities from other entities that may have the same name. Specifically, the distinct name of an entity (or qualifier) consists of its namespace and its local name (or identifier).

The namespace of an entity corresponds directly with the qualifier of its owning entity. As all member entities of a specific owning entity are constrained to have distinct identifiers, this ensures that no ambiguity can occur when referring to entities.

When referring to namespace relationships between entities, the following terms are used:

- This – refers to the current entity.
- Owner – refers to the owning entity of the current entity. The current entity is referred to as a ‘member’ of the owning entity.
- Component – refers to the owning segment of the current entity. This can be defined recursively as the successive owner of each owning entity, until the top-most level of the owner hierarchy is reached.
- Super – refers to the super (or base) class of the current class.

Kind

The Kind property of an entity determines the kinds of entities it can own. For example, a method cannot own another method.

The following kinds are available:

Owning Kind	Valid Member Kinds
Attribute	Class diagrams, folders and variables.
Class	Attributes, classes, class diagrams, externals, folders, methods, profiles, teach screens, and primitive classes.
Class diagram	Attributes, classes, class diagrams, externals, folders, methods, profiles, teach screens, and primitive classes as determined by the owner of the diagram.
External	Methods and attributes.
Folder	Attributes, classes, class diagrams, externals, folders, methods, profiles, teach screens, and primitive classes as determined by the owner of the folder.
Method	Classes, parameters, types, and variables.
Parameter	Attributes, classes, class diagrams, externals, folders, methods, profiles, teach screens, and primitive classes.
Profile	(None)

Owning Kind	Valid Member Kinds
Teach	(None)
Variable	Attributes, classes, class diagrams, externals, folders, methods, profiles, teach screens, and primitive classes

Visibility

The Visibility property of an entity determines its scope, which is its accessibility to other entities, in relation to inheritance and external to this namespace. For example, a method cannot refer to an attribute that is not within scope.

The following visibilities are available:

Visibility	Scope
Private	Accessible only to entities within its namespace.
Protected	Accessible to entities within its namespace, and entities inheriting from its namespace.
Public	Accessible to all entities.

To remove a column select it in the **Selected Columns** list and click **Remove**.

Classes

An object is a set of data combined with the logic to manipulate that data. A class describes an object. It is abstract in that it does not appear in the generated application, but it can be used to create objects which do. You should consult an object-oriented reference for details if you are unsure of its meaning. The following discussion is confined to aspects of classes that are specific to Agile Business Suite.

The following classes are available:

- Copy Events
- Copy Ispecs
- Enterprise Output Manager Reports
- Events
- Frames
- Insertables
- Ispecs
- Reports
- Segments

- SQL Scripts
- Messenger

Refer to Model Entities for more information on namespace and ownership, valid member kinds, and visibility of classes.

Stereotyping

The term ‘stereotype’ is a UML (Unified Modeling Language) concept that applies special meaning to a model element. In Agile Business Suite, classes can be stereotyped.

A class's stereotype determines and distinguishes the manner in which it is interpreted during the processing cycles. For example, all ispec-stereotyped classes (ispecs) have built-in methods such as Process and Construct that are automatically called at appropriate points of the ispec processing cycle.

The term ‘vanilla’ refers to classes that are not stereotyped. Vanilla classes do not possess any inherent behavior.

Inheritance

A class that inherits from another class has access to public and protected definitions in its superclass. Additionally, the inherited class forms a new type if the class definition is extended either by adding a member or by changing the semantic property of the class.

There are two types of properties relevant to a class namely, Descriptive and Semantic, which are explained as follows:

- **Descriptive Properties**

  These properties are either used for design or as a default. They do not affect the runtime behavior of the class, but are either descriptive such as Description property or influence the semantics of other entities such as MemberPersistence property.

- **Semantic Properties**

  These properties define the class behavior and may force the inclusion of framework members. These properties define the type and may be optional such as the AutomaticEntryCapable property. Some examples of semantic properties are Length, Primitive, PresentationType, and Stereotype. Once a semantic property has been specified it cannot be modified by an inheriting object (length).

Note: *Agile Business Suite supports single-inheritance. Multiple-inheritance is not supported.*

Inheritance is manifested in the Inherits property of a class.

Cautions

Inheritance cannot be used if:

1. It would create a cycle in the inheritance hierarchy because the superclass is the same as the subclass or is one of its subclasses.
2. It is an external class (IsExternal is True and Multiplicity is 0).
3. The class or object is a member of the proposed superclass (either directly or through inheritance).
4. The proposed superclass is an inner class and its outer class is contained by the class.
5. The class and its proposed superclass are both inner classes, and the outer class is not the same as, or a subclass of the proposed superclass's outer class.
6. The class is a member of a group and the proposed superclass is either not primitive or is a not group.
7. The class or the proposed superclass is a Group, Segment or Report class.
8. The class is an Insertable class.
9. The class has a different stereotype to the proposed superclass (except where the superclass has NoStereotype).

Additional restrictions may come from how the inheritance relationship is represented in the database. Where the superclass and subclasses are stored in the same table, the limit on the total number of attributes (4095 for MCP) is applied to the superclass and all its subclasses.

When referring to inheritance relationships between classes, the following terms are used:

- This – refers to the current class.
- Super – refers to the super (or base) class of the current class.

Refer to Class Properties for properties applicable to base Classes.

Copy Events

Copy events (copy-stereotyped event classes) are events that participate in the copy cycle. They behave similarly to copy ispecs, with the additional characteristics of events.

Only attributes that have graphical presentations can be copied. Copied attributes are those with their corresponding graphical object's **Is Copied** presentation property of set to true.

Copy events have a **built-in Edit method** as an inherent characteristic, in addition to those of events.

Refer to Copy Cycle for more information on the processing of copy events.

The default window is Members. Refer to Copy Event Properties for properties applicable to copy events.

Copy Ispecs

Copy ispecs (copy-stereotyped ispec classes) are ispecs that participate in the copy cycle.

Only attributes that have graphical presentations can be copied. Copied attributes are those with their corresponding graphical object's **Is Copied** presentation property of set to true.

Copy ispecs have inherent behavior related to:

- Initializing non-interface and non-persistent variables before the Construct and Main methods are invoked.
- Iteration through the Construct method (when invoked) for the number of times specified by the **Max Copies** property, incrementing the Glb.Copy segment attribute on each iteration. On each iteration, one set of the copied attributes are available to method logic.
- Iteration through automatic look-up, the Prepare method (when invoked), automatic validation, and the Edit method (when invoked) for the number of times specified by the **Max Copies** property, incrementing the Glb.Copy segment attribute on each iteration. On each iteration, one set of the copied attributes are available to method logic. If Glb.Error is not set to '*****', any automatic updates occur at the end of each iteration.

Copy ispecs have a **built-in Edit method** as an inherent characteristic, in addition to those of ispecs:

Refer to Copy Cycle for more information on the processing of copy ispecs.

Refer to Copy Ispec Properties for properties applicable to copy ispecs.

The default window is Members.

Enterprise Output Manager Reports

Enterprise Output Manager reports are specially formatted reports designed to be used with the Enterprise Output Manager print management application. The report is in a simple tag and value format, with the tag at the beginning of the line followed by the value. There are several tags used as commands that are interpreted by Enterprise Output Manager. The remainder are field names derived from the form file that is used to create the report in the Enterprise Output Manager wizard. The Output Report Generation Wizard is opened by default.

Enterprise Output Manager reports can be added in any situation where a standard report is valid. Refer to Wizard Output for more information on objects contained in a report once it is created using the Enterprise Output Manager wizard.

Refer to Using Enterprise Output Manager Reports for more information on creating a Enterprise Output Manager report

Events

Events (event-stereotyped classes) represent business transactions in your application, such as a payment. They affect the modification of data related to one of more ispecs.

Events have the following inherent characteristics, in addition to those of ispecs:

- Persistent attributes of an event automatically become the same as an equivalent attribute in either the event set structure or the segment. This is displayed in the Classifier property of the attribute

The default window is Members. Refer to Event Properties for properties applicable to Events.

The default window is Members.

External Classes

External classes provide an interface for AB Suite to call external programs and read external data sources.

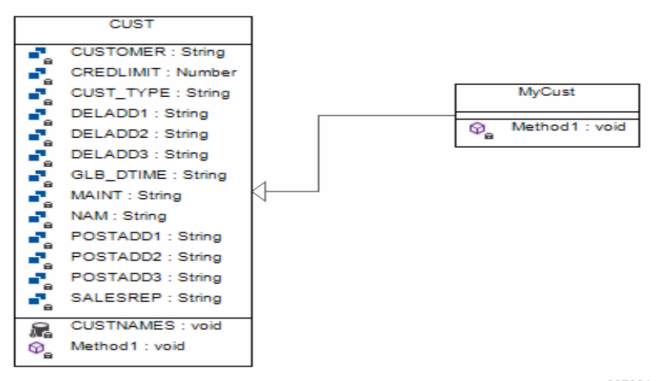
External programs provide specific functionality to applications that are generated and deployed with AB Suite. For example:

- A dynamic link library (.dll or MCP library) created to handle domain authentication for user names and passwords.
- An executable or MCP library used to perform file cleanup operations.
- A C# program used to call web services that provide data from external sources.

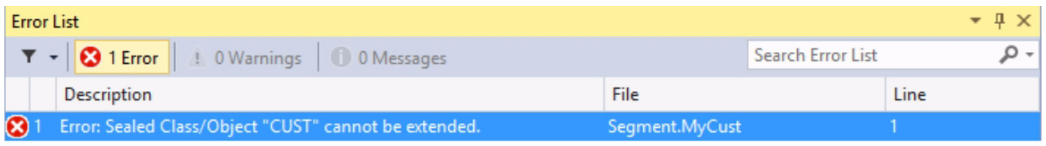
Note: *External programs must reside on the same machine as the AB Suite system. Refer to Calling External Components for more information on how to call external programs.*

External data sources are EAE OS 2200 database tables, EAE RDML schema files and OLTP view files. An external data source may not reside on the same machine as the AB Suite system. External classes with persistent members are used to read data from external data sources. Refer to Importing External Data Sources for more information on working with external EAE RDML schema files and OLTP view files.

External classes are defined when the **IsExternal** property is set to **True**. These classes can contain attributes, profiles, variables, methods, and parameters that provide an interface to the external resource. External classes do not have a stereotype and cannot be extended. Doing so generates a model validation error. For example, if MyCust with multiplicity = 1, inherits from the external class, Cust, and has a method Method1 added to it, a model validation error appears.



007021


Error List

1 Error | 0 Warnings | 0 Messages | Search Error List

	Description	File	Line
1	Error: Sealed Class/Object "CUST" cannot be extended.	Segment.MyCust	1

007022

External resources are components outside your application. For example, they may represent a .NET or COM component, a library, application, an EAE OS 2200 external database, an EAE RDML schema file, or an OLTP view file.

External classes support Latin 1 languages (Afrikaans, Basque, Catalan, Danish, Dutch, English, Faroese, Finnish, French, Galician, German, Icelandic, Indonesian, Italian, Malay, Norwegian, Portuguese, Spanish, Swahili, Swedish) and Japanese. Refer to the *Agile Business Suite Installation and Configuration Guide* for more information on adding the required libraries to support External persistent classes targeting Kanji data on OS 2200.

Methods

Methods of external classes can be used to access routines or programs such as:

- Methods of COM, COM+ or .NET components.
- Subroutines in libraries, such as .dll.
- Windows external programs, such as batch files (.bat) or other executable files that are not bound into the runtime application.

Refer to External Class Properties for properties applicable to External Classes.

For MCP-based Libraries, Agile Business Suite supports the following data types for external classes:

- Boolean
- Real

- Integer
- Ebcdic array

Multi dimension Ebcdic arrays are passed as single dimension ebcdic arrays whose size is item length multiplied by multiplicity (that is, item length *multiplicity). Where the array is defined as numeric its contents are held as packed decimal.

Calling External Components

External classes are used to connect AB Suite methods with functions contained in external components to call subroutines in dynamic link libraries, external batch files, or executable files.

An ExternalClassHost32 COM+ application allows AB Suite 64-bit runtime to call 32-bit external programs, such as Dlls, .NET assemblies, COM components, and EXEs. Refer to the *Agile Business Suite Installation and Configuration Guide* for more information on ExternalClassHost32.

An AB Suite system on a MCP platform uses the COMTI and J2EE Connector interfaces for programs existing in .NET or J2EE environments. This is achieved by processing external classes or public segment methods defined in the AB Suite system on a MCP platform.

You can map external classes to MCP Libraries and co-routines.

- MCP Libraries are programs that are compiled externally and that execute on the caller's process stack. These can be called by title or by function.

  MCP Libraries now support an extended range of library parameters (integer, real, Boolean, ebcdic array, file, and string for type procedures). Accessing libraries from AB Suite is no longer restricted to untyped, single ebcdic array exported procedures.

  Access to multiple Connection Libraries (CLs) residing within a single codefile is supported by including the interface name in the Library Name, for example, MyMultilib\Interface1
- Co-routines are C85/Algol/EAE/AB Suite modules that are separately compiled and that execute on their own stack. Calls to such modules are synchronous. Co-routines support the same simple method signature as EAE (glb.param).

Public Segment Methods support a true parameter interface and can be called from a variety of sources (MCP, CE, .NET, and J2EE).

The legacy interfaces of NOF, HUB, Business Integrator, OLTP, GLI, OFFLINE, RAS, and USER are also supported.

To call an external component from AB Suite, perform the following:

1. Create a class and set its **IsExternal** property to **True**.
2. Add a method for each function to be invoked from the external component. Parameters and return variables are limited to the AB Suite Primitive Types and must match those in the external component.

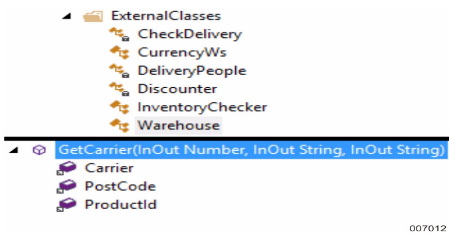
3. Right-click the external class and select **Properties** to set the configuration properties of the external class.
4. In the ExternalClass Property Pages window, set the **Component Type** property to the type of component you want the class to represent.

   Note: *You must specify the component type because AB Suite generates different external class wrappers for different component types.*
5. Set the External properties for the selected component type.

   Notes:

   - *Refer to External Configuration Properties of the Builder for more information on setting the connection information for external components.*
   - *Refer to General Configuration Configuration Properties of the Builder for more information on setting the connection information to call external MCP Libraries and co-routines.*
6. Invoke the external component.

   Refer to the following example:



```
returnVariable := Warehouse.GetCarrier(Carrier, PostCode, ProductId)
```

This returns an integer that is tested with the If command.

Notes:

For .NET components

- *A strong name key should be created for the assembly as part of the external component's project to make it visible to AB Suite's external class wrappers. This key (snKey.snk) is created with the snKey utility and is built with the external project.*
- *After building a .Net component assembly, you must deploy it for Runtime. In most cases, the runtime server is a different machine to the one used to develop the .Net assembly. So you must copy the built assembly to the runtime server and sometimes even register the built assembly. You can copy the built assembly manually, or use Visual Studio's built-in post-build events to automatically copy the built assembly. After you have copied the assembly to the runtime server, for AB Suite Runtime to locate the assembly, you can either add the path of the copied assembly to the PATH environment variable on the machine or register the assembly in the Global Assembly Cache (GAC) or Regasm.*
- *By default, during a call to an external 64-bit .NET assembly, a new instance is created and the same instance is reused for subsequent calls to the .NET assembly in the same transaction.*

However, you can set the CreateExternalClassInstanceAlways registry key that results in the following behavior; a new instance of the assembly is created when an external method is called and a new instance is created for every subsequent call.

The CreateExternalClassInstanceAlways is of type DWORD and you can set the key at the following locations.

- *The registry key for a particular system is stored under the following registry key:*

  ```
  \HKEY_LOCAL_MACHINE
    \SOFTWARE
      \Unisys
        \AB Suite
          \6.1
            \SM Runtime
             \<System Name>
  ```

  and

- *The registry key for machine wide settings is stored under the following registry key:*

  ```
  \HKEY_LOCAL_MACHINE
    \SOFTWARE
      \Unisys
        \AB Suite
          \6.1
            \SM Runtime
  ```

• *Registering the assembly in the GAC or Regasm can make the assembly easily accessible for other programs on the machine and requires simple utilities (such as GACUtil.exe/Regasm.exe, available in the .NET Framework 2.0 SDK), without needing to specifically configure your environment to locate the assembly.*

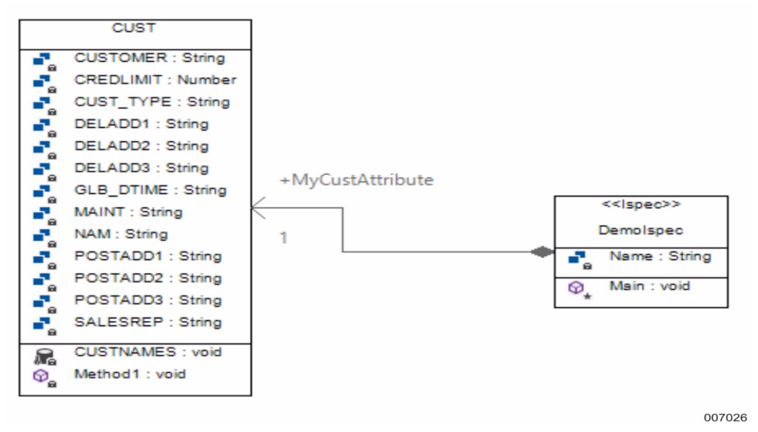
Reading External Data Sources

An external class with persistent members provides an interface to connect to an external data source and retrieve data. The schema of the external database or an OLTP view file can be imported into AB Suite.

Note: *Presently, external classes with persistent members are not supported on the MCP platform. Hence, you cannot access a DMSII database using external classes. However, you can access external MCP databases by using the existing MCP AccessExternal feature.*

Data is retrieved from the data source with a Determine or ForEach command over an instance (variable, parameter, attribute) of the external class.

The following example illustrates how an external class with persistent members can be instantiated. The Ispec 'DemoIspec' with Multiplicity = 1 has an attribute member 'MyCustAttribute' with Multiplicity = 1 that inherits from the external class Cust. The method Main() reads the contents of the external class by using a Determine command over MyCustAttribute.



External classes can be created manually or by using the External Class Wizard. Using the wizard requires a schema file or an OLTP view file that is imported to automatically create the external class and its members. Members typically consist of profiles and persistent attributes. Refer to Importing External Data Sources for more information on using the External Class Wizard.

Note: *For an imported external class with persistent members, it is recommended that you do not change the Profiles or Attributes because the changes are lost when the class is synchronized with the source RDML file or OLTP view file.*

To read an external database, perform the following:

1. Create a class and set the **IsExternal** property to **True**.
2. Add an attribute for every field in the record as defined in the external data source.Attributes are limited to the AB Suite Primitive types and must match those in the external data source.

   Note: *The sequence number is used to determine the order of the columns in an external database table.*
3. Add profiles to retrieve subsets of the records using the views or keys as defined in the external data source.
4. Right-click the external class and select **Properties** to set the configuration properties in the **ExternalClass Property Pages** window.

   Refer to the External Configuration Properties of the Builder for more information on setting the connection information for external data sources.
5. Use the Determine command variants to iterate over the external data source.

   Refer to the *Agile Business Suite Programming Reference Manual* for more information on using the Determine logic command variants to read external data.

Importing External Data Sources

You can create external classes with persistent members by importing the database schema file or the OLTP view file that defines the tables within the external data source. When the source file is imported, it is automatically copied to the project directory (Resources folder in the Solution Explorer view). As the synchronization process uses this file, it is recommended that you always update the source file in the project directory so that the changes are automatically detected. This prevents duplicate source filenames and is useful when you want to version control the solution files.

To import an external data source, perform the following:

1. Obtain access to the schema file of the external data source and save it on the AB Suite developer machine.
2. Right-click the element that owns the external class, and select **Add**, **Add New Item…**.
3. Select **External Class** from the Wizards pane in the **Add New Item** dialog box.
4. Click **Create**.

   The **Select Data Source** box appears.
5. On the **Selection** page, browse and select the external data source file.

   For example, an RDML$EXT.txt file or an OLTP (Online Transaction Processing) view file.
6. Select the tables to be imported from the external data source schema.

   You can select the **Select/Deselect All** check box to clear all the selected tables and select only those that must be imported.

   Note: *You can only select tables but not table members. All the members of the selected tables are imported except those data types that are not supported by AB Suite.*

   Click **Next**.
7. Confirm the selection and click **Finish** to return to the model.

   If you select the **Display Result** check box on the Confirmation page, the Results page shows the progress and displays a message indicating the status of the import process.
8. Click **Close**.

   The Output window displays the progress and the status during the creation of external classes. Errors appear in the Output window.

   This imports the structure of the external database. The **IsExternal** property of the classes is automatically set to **True** and the class members are persistent.

   The **SynchronizeState** property of the imported external class is also set to **InSync**.
9. Right-click the external class and select **Properties** to set the configuration properties in the **ExternalClass Property Pages** window.

10. Select either **Defined Locally** or **Location** in **Edit**, from the **Data Source Location** list.

    The Defined Locally option can be used to set the properties within the class configuration. Otherwise, you can select an existing location to specify the predefined configuration information.

    Refer to the **Data Source** properties in System Modeler Locations for more information on setting the properties for the **Defined Locally** option.

11. Use the Determine command variants to iterate over the external data source.

    Refer to the *Agile Business Suite Programming Reference Manual* for more information on using the Determine logic command variants to read external data.

Synchronizing External Classes

You can synchronize an external class with its resource, as specified by the **SynchronizeFile** property when the **SynchronizeState** property is set to **NeedsSync**. If you change the properties of an external class, such as SourceName, ViewName, ViewType, Length, Primitive, and Decimal, the **SynchronizeState** property is set to **NeedsSync**. The **SynchronizeState** property is also set to **NeedsSync** if you add or remove persistent members in an external class or change the member properties as mentioned earlier. In addition, modifying the resource file also changes the value of the **SynchronizeState** property to **NeedsSync**.

To synchronize an external class, perform the following:

1. Update the version of the Resource in the project directory.
2. Perform any one of the following:
    - Select **Synchronize** on the resource file in the Solution Explorer view.

      This synchronizes all the external classes that are associated with the selected resource file.
    - Select **Synchronize** on the external class.
    - Select **Synchronize** on the owner of the external class.

    This recursively synchronizes all the external classes within the namespace.

The value of the **SynchronizeState** class property changes to **InSync** after completion.

Web Services for MCP

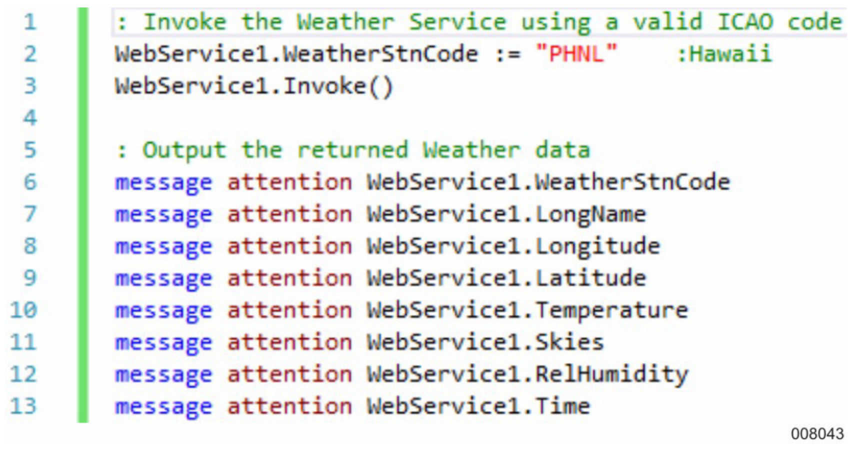
You can invoke a Web Service in your MCP application through the new Web Service Wizard. The wizard creates an AB Suite class that encapsulates the Web Service functionality ready for use by your Reports and Ispecs.

To create an MCP Web Service, perform the following:

1. Right-click the segment, and select **Add** > **Add New Item...**.
2. In the **Add New Item** dialog box, select **Web Service** > **Create**.

   The **Web Service Wizard** is displayed.
3. Enter a name for your Web Service, and click **Finish**.

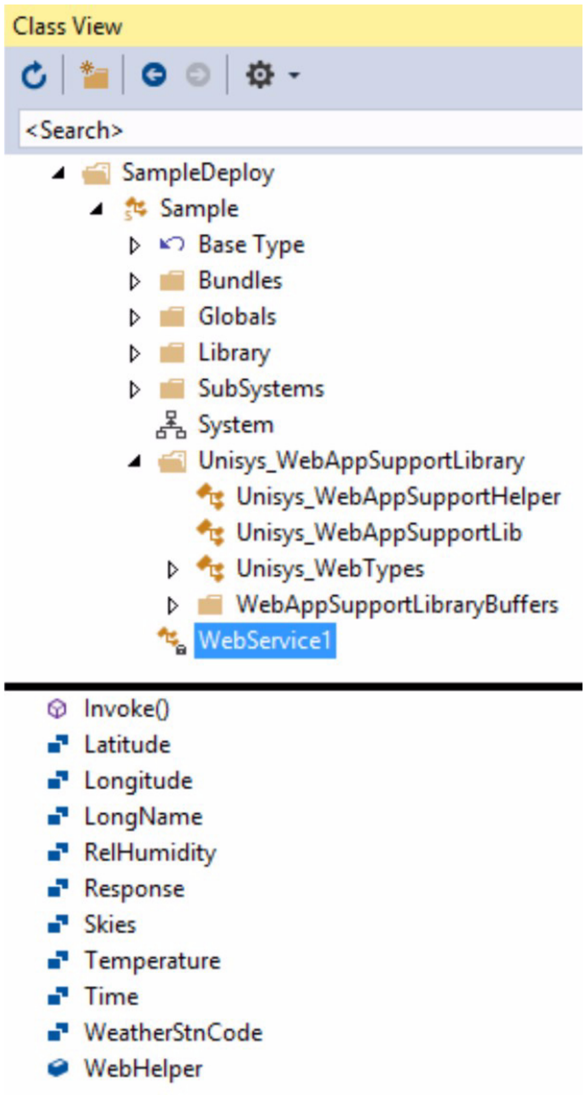
The created class contains pre-loaded logic and attributes to exercise the Web Service with the Unisys Weather Service. You can invoke the method from an Ispec or Report by setting the WeatherStnCode to any valid ICAO code to test it. For example, "PHNL" - to view the status of the weather in Hawaii.



```
: Invoke the Weather Service using a valid ICAO code
WebService1.WeatherStnCode := "PHNL"    :Hawaii
WebService1.Invoke()

: Output the returned Weather data
message attention WebService1.WeatherStnCode
message attention WebService1.LongName
message attention WebService1.Longitude
message attention WebService1.Latitude
message attention WebService1.Temperature
message attention WebService1.Skies
message attention WebService1.RelHumidity
message attention WebService1.Time
```

Note: *The added infrastructure to support the Web Service wizard includes the following classes.*

- ***Unisys_WebAppSupportLib***
  External class configured with all necessary entry points for the Web App Support Service.
- ***Unisys_WebAppSupportHelper***
  Class to assist with the implementation of the Web Service. Consolidating the common process for establishing and invoking a Web Service.
- ***WebAppSupportLibraryBuffers***
  Contains the specially configured buffers for invoking the Web App Support entry points.
- ***Unisys_WebAppTypes***
  All types associated to the Web App Support library.



008044

Pre-Requisites

The XML parser feature for Web App Support must be installed and configured on the host for MCP Web Services. Refer to WebAppSupport (WEBAPPSUPPORT Application Programming (http://www.support.unisys.com/aseries/docs/ClearPath-MCP-17.0/PDF/38265286-006.pdf)) for more information.

Frames

A Frame is a class that implements the functionality of a frame. It may have an interface and inherits a 'Main' method that is executed on a printframe LDL+ command. The Main method may be overridden. The Class functions as a report frame by providing a main method to override and restricting properties to those applicable to reports. When an attribute of this class is invoked, the main method is executed before printing the presentation. The default window is Members.

A Frame can be owned by any class, including another Frame and can be both, a class definition and an instance.

Insertables

Insertables (insertable-stereotype classes) consist of logic and/or a user interface that can be inserted (using the Insert logic command) in any number of ispecs, events, or reports. A dependency is created between the inserting class and the insertable. Insertables are migrated as attributes of the inserting class. The default window is Members.

Insertables without a user interface or any attributes can also be inserted in segment methods.

Insertables have the following inherent characteristic:

A built-in Main method, which can be overridden using the Overrides tab.

Note: *Insertables are not available for use in Messenger classes.*

For properties applicable to Insertable, see Insertable Properties.

Ispecs

Ispecs (ispec-stereotyped classes) represent an entity in the business world, such as a customer, product, or vendor.

Ispecs have inherent behavior related to:

- Control of the ispec processing cycle and error handling, which includes:
  - Initializing non-interface and non-persistent variables.
  - Performing appropriate validation of input variables, including automatic look-ups, value-checking logic, numeric validation, date validation, and ascertaining the presence of required variables.
  - Invoking the Prepare method.
  - Invoking the Main method.
- Assembling error and/or output messages for transmission to the application user.
- All ispecs have the following inherent members:

Built-in Construct, Prepare, and Main ispec methods.

Persistent ispecs with at least one attribute that has their **Is Key** property set to true also have an inherent Maint built-in attribute.

The default window is Members.

Ispec Usage

Ispecs can belong to one of the following categories:

- **Input**

  These ispecs have a user interface (**PresentationType** property set to a value other than None), but no persistent attributes (**Is Persistent** property set to false).

  Input ispecs do not perform an implicit automatic update of the database at the end of their Main method.

- **Output**

  These ispecs have no user interface (**PresentationType** property set to a value other than None), but they do have persistent attributes (**Is Persistent** property set to true).

  Output usage ispecs do not have any inherent behavior, as they are used solely to define database tables. However, they can have explicitly defined methods, which can be invoked independently of the Agile Business Suite processing cycles.

  Any attribute within the Segment class with IsPersistent=True, retains their value across ispec transactions within one session.

  Example with the SAMPLE application:

  ```
  Declare the MENU class with multiplicity=1 and Ispersistent=True
  In the “Construct Menu”
              Me attention           MENU.ACTION2
  In the “Prepare Menu”
              My ACTION2        MENU.ACTION2
  ```

  With this construct, when we execute the MENU ispec for the first time, the value for MENU.ACTION2 does not get displayed, but when the system displays the last value entered in the ACTION2 field this value “MENU.ACTION2” is stored as session attribute and each user can have a different value.

- **I/O (input-output)**

  These ispecs have both a user interface (**PresentationType** property set to a value other than None) and persistent attributes (**Is Persistent** property set to true).

  Ispec perform an automatic update of the database, provided Glb.Error has not been set to “*****”.

Depending upon usage type, ispecs may also have additional inherent characteristics. These characteristics vary accordingly with the addition and removal of persistent attributes and/or user interfaces.

Refer to Class Properties for properties applicable to Ispecs.

Persistence

Database Persistence

The typical usage of this property is to define columns in a database table. When you set IsPersistent on one or more attributes within an Ispec or Vanilla class to True, it would mean that a database table is created representing the class and that all IsPersistent attributes become columns within that database table. For example,

Ispec: Customer

Customer_Id IsPersistent=True

Name IsPersistent=True

Address IsPersistent=True

This example results in a database table called "Customer" with columns "Customer_Id", "Name" and "Address" being created. Note that the Customer element itself does NOT have IsPersistent=True.

Session Persistence

For attributes that are members of a Segment class, setting IsPersistent to True would mean that they are session persistent. This means that these values are retained for the duration of a user session. They arenot initialized during the ispec runtime cycle, but retains any values assigned to them. These can be used to pass data from one ispec transaction to another within a single user session.

This can apply to both primitive and non-primitive attributes. Typically, these attributes retain the values for the duration of a single user session. When the user signs off and later starts a new session these attributes is initialized at the start of the new session. However, if you set the "Preserve Session Data" Windows configuration property, then the value of these session persistent attributes carry over from one session to the next (by the same user).

The table below gives the session persistence support information about MCP and Windows® Runtime.

Session Attribute Type	MCP Runtime	Windows® Runtime
Primitive	Supported	Supported
<Group> Stereotyped Class	Supported	Supported
All Other Classes	Unsupported	Supported

Reports

Reports (report-stereotyped classes) present information from the database to the user, or are used to perform specialized batch processing of the database information. They also have a Main method. The default window is Members.

Refer to Report Properties for properties applicable to Reports.

Segments

Segments (segment-stereotyped classes) generally function as the top-level definition of your application. They are sometimes referred to as a 'component', and own all the model entities that the application consists of.

Segments have inherent behavior related to:

- Control of application user connection and disconnection.
- Control of database connection and disconnection.
- Persistence of session state for the Ispec cycle.
- Control of the segment cycle, which includes:
  - Unpacking of input messages to sufficiently identify their source and the ispec to be invoked
  - Invoking the ispec and its Process method with indication as to which part of the segment cycle the session is in
  - Assembling and delivering the output message
  - Enacting Recall and SwitchTo actions
  - Invoking public methods
- Transactional integrity

Segments have the following inherent members:

- Built-in segment attributes, such as Glb.Status, Glb.Error, Glb.User, and so on.
- Built-in StartUp, CloseDown, and ProcessMsq segment methods:
  - Startup is called on application start up
  - CloseDown is called on application close down
  - ProcessMsq is called to perform the segment cycle

A model can have multiple segment members.

Entities within a segment can be grouped using folders.

The default window is Members. Refer to Segment Properties for properties applicable to Segments.

SQL Scripts

SQL scripts (SQL script-stereotype classes) act as embedded containers for SQL statements. They can be used to optimize database queries and updates, by allowing access to native interface of the database. The performance gain that can potentially be achieved from the use of SQL scripts is dependent on the capabilities of the DBMS, the nature of the specific query or update, and the tuning of the physical database itself. The default window is Members.

Normal methods can also be in SQL. SQL Scripts have a processing cycle and are useful for iterating through the results of an SQL select.

Valid SQL commands are:

- INSERT
- DELETE
- UPDATE
- SELECT … INTO
- DECLARE … CURSOR FOR
- OPEN
- FETCH … INTO
- CLOSE

You should consult a SQL command reference for details of these commands. You should also consult your SQL Server documentation for details of their respective conformance with the ANSI SQL-92 standard. Invalid SQL commands in SQL scripts cause compilation errors.

SQL statements for SQL scripts are specified in the logic editor. Refer to the *Agile Business Suite Developer Online Help* for more information on using the logic editor.

- SQL scripts have the following inherent characteristics:
- SQL script processing.

Built-in construct, main, and destruct SQL script methods, which can be overridden using the Overrides Tab.

The built-in SQL script methods can only be invoked using the Determine Actual logic command. All other SQL script methods can be invoked directly.

Construct

The construct method (script) opens the cursor in preparation for iterating through the results of a SELECT statement.

To execute a set of SQL statements once only, they should be specified within the construct method. Additionally, a Break logic command should be invoked within the Determine Actual logic command loop. Refer to **Break** in SQL Script Variant for more information.

Main

The main method (script) is executed for each iteration of the Determine Actual logic command statement. It iterates through the retrieved records.

Destruct

The destruct method (script) releases any resources used by the SQL script.

Attributes

Attributes of the SQL script that are referenced by embedded SQL statements need to be identified as such. They should be prefixed with a colon (':') when referenced in a SQL statement. In the following example, MyA, MyB, MyC, and MyZ are attributes (instance variables) of the SQL script:

```
SELECT A, B, C INTO :MyA, :MyB, :MyC FROM TAB WHERE Z = :MyZ
```

Debugger Restrictions

SQL script method calls are ignored during debug sessions. Debugger disregards any invocations of SQL script methods it encounters, and continues execution at the logic statement following the method invocation.

Examples

- Selective Data Retrieval with Join Example
- Selective Update Example
- Database Issues

Refer to The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value. for properties applicable to SQL Scripts.

Multi-user Session Handling in SQL

AB Suite allows multiple users to access and update the AB Suite database. To reflect the changes made by another user, you need to manually refresh the domain. A domain here is the model element selected currently.

For example, when a user edits or deletes a model element, you have to manually refresh that model element to see the updates. If you refresh a particular domain, only the changes made within that domain is visible, whereas the changes made outside that domain remains invisible. You can increase the domain scope by selecting its owner.

Messenger

A Messenger is a class Stereotype that is used to define a class that acts as a message broker to process XML messages submitted as input to the AB Suite runtime system.

The Messenger class contains a built-in cycle for processing incoming messages.

In the Messenger processing cycle

1. An instance of the Messenger is created and the input message is parsed and validated.
2. The Messenger instance is populated from the message.
3. User-defined logic is executed.
4. A response message is created and returned to the client that sent the input message.

Methods

A method is a concept from object-oriented development. It is some logic that executes in the context of an object. You should consult an object-oriented reference for details if you are unsure of its meaning. The following discussion is confined to aspects of methods that are specific to Agile Business Suite.

Method Invocation

Methods can be called explicitly by name in logic, or implicitly as part of a processing cycle.

Built-in Methods

Agile Business Suite provides a number of built-in framework methods for certain model entities. Some these methods occupy specific roles within the Agile Business Suite processing cycles. Others are available as utility functions.

The following table lists a summary of some of the more significant built-in methods. Refer to Built-In Methods and Method Reference for a comprehensive list of built-in methods,.

Method	Owner	Description
StartUp	Segment	Called on segment start up.
CloseDown	Segment	Called on segment shut down.
ProcessMsg	Segment	Performs the segment cycle.
Construct	Ispec or event	Called before the class's user interface is displayed.
Prepare	Ispec or event	Called before the class's Main method is called.

Method	Owner	Description
Edit	Copy ispec	Called during the validation phase of the Copy processing cycle. The copy ispec's Main method is not called during this phase.
Main	Ispec, event, or report frame 0, migrated as Main	Called after the class's Prepare method (if it exists) is called.
Load	Class with persistent attributes	Called to load a class's persistent attributes.
Store	Class with persistent attributes	Called to store a class' persistent attributes.
Purge	Class with persistent attributes	Called to physically delete the most recently accessed class record from the database.
Update	Class with persistent attributes	Called to update the current record (last read into the instance).

Note: *You can override some of the above built-in methods. The overridable built-in methods can be identified by checking its IsFinal property. If this property is False you can override the method and if it is True you cannot. You can also check which methods are overridable by opening the Overrides tab for a class. Methods that can be overridden are shown in the Add New Override list. You cannot change the Visibility property of a built-in method or any method that overrides a built-in method.*

Refer to the *Agile Business Suite Developer Online Help* for more information on entering logic.

The default window is Logic. Refer to Method Properties for properties applicable to Methods.

Variables

A variable is memory location that stores data and is identified by name. An attribute is a variable that is a member of a class. A parameter is a variable that passes data in and/or out of a method. A variable can also be local to a method. The default window is Members.

Variables include the following:

- Attributes
- Parameters
- Reference
- Reference

Primitives

Variables can be of several types.

Number

Number-primitive variables can only contain decimal digits.

Decimal point position is implied and specified separately using the **Decimals** property.

Length is calculated from the sum of the number of digits in the integer part and the number of decimal places.

They are limited to a maximum length of 18 (maximum size of 2^18).

Signed Number

Signed number-primitive variables are the similar to number-primitive variables, except that they are signed.

String

String-primitive variables can contain any sequence of the following:

- Printable characters from the ASCII character set
- Embedded escape sequences representing non-printable characters in @xx@ notation, where “xx” is a hexadecimal representation of the binary character value.
- Embedded escape sequences representing non-printable characters in \x, where '\x' represents either a control-character equivalent such as '\n' for newline, or a hexadecimal equivalent of the binary representation.

  They are limited to a maximum length of 9999 characters.

Boolean

Boolean-primitive variables can be used in place of a logical expression in “If” (DoWhen) statements.

The following assignments are valid:

bool := “Y”

bool := “N”

bool := “Yes”

bool := “No”

bool := true

bool := false

But only true and false may be used in DoWhen/If statements.

Date

Date-primitive variables can only contain decimal digits.

They are comprised of two digits each for the month, day, year, and optionally the century. The **Date Format** property determines the order of these date units – IN(ternational), UK, and US are represented as (CC)YYMMDD, DDMM(CC)YY, and MMDD(CC)YY respectively.

The date format is only validated when the variable accepts input in a user interface, or when it is used as the input date of a DateConvert logic command.

Date-primitive variables are of a fixed length of 6 characters, or 8 characters if the date format includes the century.

Wide String

Wide string-primitive variables can contain any sequence of double-byte characters as a Unicode string.

They are limited to a maximum length of 2047 characters.

Refer to Variable Properties for properties applicable to Variable.

User Defined Classes

User defined classes comprises the structure of the items to be stored in a list.

Attributes

Attributes are variables that are members of a class.

Refer to Attribute Properties for properties applicable to Attributes.

Parameters

Parameters are variables used to pass data into and/or out of a method.

Refer to Parameter Properties for properties applicable to Parameters.

Reference

A Reference is an attribute whose value is contained either in part or whole within another attribute. The attribute being referenced must be within scope and not another reference. The Reference can only be accessed through a presentation (not from logic). A Reference attribute does however form a part of the owning class's structure.

The typical use of a Reference is to present a cell of an array. The association to another object is contained in the expression Constraint. Under this situation the expression represents a path to the referenced attribute.

The Inherits property is automatically set to the referenced attribute when the expression in the Constraint property is validated and a Use dependency exists. The expression is parsed on validation of the owner.

The value reference is initialized when the inherits property has been established. Only then is the Direction property available for placing the object in the presentation of the owner. To use a Reference in a presentation the owner must have its PresentationType set to a value other than None.

Refer to Reference Properties for properties applicable to References.

Profiles

Profiles define a set of criteria for selecting database records for a persistent class.

Specify constraints that limit the records selected using the Conditions tab.

Profiles belong to the class to which they apply. This allows the selection criteria to include private attributes without breaking encapsulation.

Note: *A profile on an event which is not itself acting as an EventSet is ignored during generation. That is, all profiles should be defined on the EventSet and not on individual events. Conditions can be added to profiles to restrict their operation to specific events.*

The default window is Conditions. See "Profile Properties" for more information on properties applicable to Profiles.

Teach Screens

Teach screens display help information in your applications. Teach screens can only be added to classes. Refer to the *Agile Business Suite Developer Online Help* for more information.

See "Teach Screen Properties" for properties applicable to Teach Screens.

Dictionary

A Dictionary is a form of Folder. It is however limited by the fact that it can only contain Objects. These are mainly considered Classes, a Dictionary can be described as a Class Dictionary. In the same way that Folders can be added to any Namespace, Dictionaries can also be added. The default window is Members.

Global dictionaries (defined directly under a model) can be used to share dictionary definitions across multiple applications. Local dictionaries (added to a segment) can be used to define data types for use only in that segment. Each dictionary may contain primitive and non-primitive classes.

By setting the EnforcePersistents segment property to True, you can specify that any persistent attribute added to that segment must inherit its type from an item in a dictionary. You can also specify, by setting EnforcePresentation to True, that any attribute appearing in a presentation must inherit from a dictionary item. These properties are also available on the model. Setting EnforcePersistents or EnforcePresentation at the model level applies the dictionary enforcement rule to all segments in that model.

Items can be added to multiple Dictionaries, providing that the IsUnique property of the Dictionary is set to False.

Folders

Folders act as a logical grouping of model entities.

A folder does not own its member entities. Entities contained in a folder continue to be owned by the folder's parent entity. This effectively means that an entity can be a member of more than one folder.

Adding a new entity to a folder is essentially the same as adding the entity to the folder's parent.

Entities that are members of the folder's parent can be added as existing entities to the folder.

Note: *Agile Business Suite folders are not the same as the virtual folders that can be added outside the model hierarchy using the New Folders button on the Class View toolbar.*

The default window is Members. Refer to Group Properties for properties applicable to Folders.

If you delete a folder and it contains other elements, you can optionally delete the folder's members at the same time. If you do not delete the folder's members, they are placed under the model when the folder is deleted.

The IsUnique model property of Folders ensures all members do not exist in other folders. This is a Boolean value where true applies the uniqueness constraint. A nested Folder assumes the setting of the owning Folder. This property does not apply to Diagrams.

For MCP, any folder with IsUnique property set to True must have a name that does not exceed 18 characters in length.

MCP runtime applies a build rule to cluster all non-persistent, generic or primitive attributes contained within the Segment Folder into a block. All persistent Segment attributes are made as a part of a GLB-WORK block, and all other Segment attributes are a part of a GLB-SDS block.

If the segment folder does not have the unique property set or the folder is not under the segment, it is not treated as a GSD block.

Class Diagrams

Class diagrams use standard UML (Unified Modeling Language) notation for class diagrams to represent the static relationships between model entities. You should consult a UML reference for details on UML notation. The following discussion is confined to how Agile Business Suite model entities are represented as class diagrams. The default window is Class Diagrams.

The following types of relationship can be expressed on a class diagram:

- **Aggregation**

  An aggregation is a relationship between a whole (the aggregating class) and its parts. It is indicated by a solid line, with a filled diamond at the aggregating class.

  An example aggregation is between a car (the aggregating class) and its parts, such as steering wheel, engine, tires, and so on. Each of these parts exists in their own right, but they are aggregated as a car.

- **Dependency**

  A dependency is a relationship where changes to the supplier class may cause changes in the client class. It is indicated by a dashed line, with an open arrowhead at the supplier class.

  An example dependency is between a encoder (the client class) and an encoding algorithm (the supplier entity). Changes to the encoding algorithm may affect the encoder.

- **Inheritance**

  An inheritance relationship is between a super class and its sub classes. It is indicated by a solid line, with a hollow arrowhead at the parent entity.

Class diagrams, like folders, do not own their member entities. Entities in a class diagram continue to be owned by the class diagram's parent entity. This effectively means that an entity can appear in more than one class diagram.

Adding a new entity to a class diagram is essentially the same as adding the entity to the class diagram's parent.

Entities that are members of the class diagram's owner can be added as existing entities to the class diagram.

Class diagrams are modified using the Class Diagram page of a diagram.

Refer to Class Diagram Properties for properties applicable to Class Diagrams.

Groups

Groups (group-stereotyped classes) are a way of representing structures in your application. They can contain only attributes as members. Member attributes can be primitives or instances of other groups. Member attributes of this class can be either all persistent or all non-persistent. Members can also be presented.

Members of a group are treated just as members of any other class. The exception to this case is with regards to persistency. Persistency of group members is group-defined and not member-defined; hence the group would be persistent or non-persistent as a whole.

The class can be treated as a string, allowing one group to be assigned to another. In addition to being a class, groups have special primitive string semantics. Assignment of a string to a group preserves the original string until it is subsequently modified. A group can act as a primitive. Here, the group behaves like a string with a string length equivalent to the sum total of the group members.

Groups feature no methods, and inheritance is not supported.

Refer to Group Properties for properties applicable to Groups.

Locations

A Location element is used to store platform-dependent location information, for example, file paths, database access, or Filegroup information. The Pack Type specifies a valid Data Source, External Pack Details, Filegroup, or Path for the files of a generated system on the target host computer.

A Location element can only be added at the Model level and cannot contain any members. A Location object is a named element and can be added, removed or updated as can any other element.

After adding a Location element to a model, you should configure the Location for a particular target platform.

To set the configuration properties for a Location, perform the following:

1. Click **Class View** on the View menu to open the Class View window.
2. Select a **Location** element from the members of the model.
3. Right-click the Location and select **Properties**.

   The **Location Property Pages** opens.
4. Select a configuration from the **Configuration** list.
5. Select a platform for deployment from the **Platform** list.
6. Specify the Pack Type (for Windows) or Pack Name (for MCP) properties depending on the platform that you select:

   Pack Type – Specifies a valid Data Source, External Pack Details, Filegroup, or Path. Select either one of the options from the Pack Type list.

   - Data Source – Specifies the connection information to connect to an external database. An appropriate error message is displayed for entries that are invalid for any of the data source configuration properties. The user needs to select either one of the options from the data source type list.
     - Data Source Type – Specifies the type of data source to connect from the various data source types such as, EAE OS 2200 database tables, <inherit from parent or default>, or OS 2200 Extract files.

Note: *EAE OS 2200 and <inherit from parent or default> options refers to the External Persistent Classes.*

The following table specifies the configurations for EAE OS 2200 database tables and <inherits from parents or default> options:

Configuration	Property
Authentication Method	Specifies the method used to authorize access to the external database table. • **ByUserIdPasswd** Requires a user id and password to access an OS 2200 host. • **ByAuxiliaryInfo** Requires a user-defined string that is passed to an onsite setup routine.
User Id	Specifies the registered user id that can access the host server. The user id can be • Alphanumeric character • Period (.) • Hyphen (-) • Of a length less than or equal to 12 The user id must not begin with an underscore or numeral.
Password	Specifies the password required for the authorized user id on the host server. The password can be: • Alphanumeric character • Of a length less than or equal to 18 A password must not contain a comma (,) and slash (/). The password is obfuscated.
External System	Specifies the name of the deployed EAE system.
External Host	Specifies the IP address or the name of the external host. For example, \\USTR-TIS2.UNISYS.COM ***Notes:*** *If "localhost" or the hostname is not provided, an error message appears as Invalid name.* *The external host name is case sensitive for OS 2200 data source type. Hence the name supplied must correspond to the name provided in ClearPathHosts.config file.*
Port Number	Specifies the TCIP port number to connect to the external HDBA host server. The port number must be in between 1 and 65,535 inclusive.
Auxiliary Information	Specifies a user-defined connection string if you have selected the **ByAuxiliaryInfo** option from the **Authentication Method** list.

The following table specifies the configurations for the OS 2200 Extract File Data Source types:

Configuration	Property
External Host	Specifies the name of the host as configured in the Clear Path Forward Application Integration Services (CPF AIS).
User Id	Specifies the registered user id that can access the host server. The user id can be • Alphanumeric character • Period (.) • Hyphen (-) • Of a length less than or equal to 12 The user id must not begin with an underscore or numeral.
Password	Specifies the password required for the authorized user id on the host server. The password can be: • Alphanumeric character • Of a length less than or equal to 18 A password must not contain a comma (,) and slash (/). The password is obfuscated.

- External Pack Details – Specifies the properties for physical host pack details to identify the extract file placement.

  The following table specifies the configurations for the External Pack Details pack type:

Configuration	Property
Access control record name(ACR)	Specifies the file access permissions and also identifies your access permission. Contact your OS 2200 System Administrator for more information. ***Note:*** *The length of this field cannot exceed 6 characters.*
Fixed Storage Pack	Specifies whether Fixed Mass Storage is used on OS 2200. ***Note:*** *If the property is set to 'false', you can customize the Pack Id's information. By default, this property is set to 'False'.* *If the property is set to 'true', Fixed Mass Storage is used.*

Configuration	Property
OS2200 Pack Settings	Specifies the disk equipment type and names for up to eight removable Packs. Click the OS 2200 Pack settings and select **Edit** to view the OS 2200 Pack Setting window. You can manually enter the disk equipment type and Pack Ids in the space provided. ***Note:*** *The length of the Pack Id should not exceed 6 characters.*
Host connection information	Specifies the connection information needed to connect to OS 2200. Click the Host connection information and select **Edit**, and then select the **Location** option. The Location Selection window appears. You can manually select the location and click **Ok**.

- Filegroup – Specifies the SQL Server database filegroup in the **Database Filegroup** property.
- Path – Specifies the path in the **Path Name** property where the pack is stored on the disk.
  - Pack Name – Specifies a unique name to define a storage area on the host for the system components.

7. Click **Apply**, and then click **OK** to save and close the Properties window.

The Properties window, if open, displays the properties of the selected location. Refer to Location Properties for properties applicable to a location.

General System Modeler Settings

To edit System Modeler’s general settings, perform the following:

1. From the **Tools** menu, select **Options**.
2. From the Explorer (located on the left-hand side of the Options dialog box), select System Modeler, then General.

The following properties can be configured:

Property	Function
Member Filter configuration file	Specifies the location of the Members Filter configuration. The default location is ‘%APPDATA%\Unisys\Agile Business Suite\6.1\System Modeler. ***Note:*** *The ‘Members Filter configuration file’ stores information about the Members pane filters.*

Property		Function
Logging	Style	Specifies the level of log details to be stored.
	Max File Size	Specifies the maximum size of the log file.
	File Location	Specifies the location of the log file.
Settings	Clear Navigator on exit	Specifies whether the content between sessions is maintained or not.
	Display Attribute name for frame attributes	Specifies whether the name of an attribute in a frame is displayed.
	Number of Threads for Validation	Specifies how many threads are used for validation.
	Max Update-Items In Multiple Selection	Specifies the max number of items that can be updated in a multiple selection scenario.

Summary

Agile Business Suite is a robust software that uses new technologies and leverages the functionalities of Visual Studio framework. Agile Business Suite is built on high productivity environment and offers:

- High level specification of the business model.
- Generate the complete application from the business model.
- One button deployment of the application.
- Lifecycle management.

Another key Agile Business Suite feature is its simple deployment to the targeted runtime platform. You do not need to select the target platform until application deployment. You can move the application from one platform to another instance of the supported platform with no changes to the application specification from which the application is generated. The one-button deployment handles the application framework and logic and the changes to the applications data model.

Section 3
Developing Applications

Developer is an Integrated Development Environment (IDE) for rapid application development. It operates as a plug-in to Visual Studio, running on a Windows workstation and enables designing, developing, debugging, and deploying applications. The development environment includes:

- System Modeler – for modeling information systems
- Debugger – for testing systems modeled in System Modeler
- Builder – for generating and deploying these systems

Creating System Modeler Projects

In System Modeler, a project is used to store the location and details of a model. When defining a project you can either create a new model or use an existing model.

Security

The model, and any of its elements can have security applied to it, to set security, the AccessControlled property at the Model level must be set to True.

To set the AccessControlled property of a Model, perform the following:

1. Select the Model in Class View.
2. Display the Properties window.
3. Select the AccessControlled property of the model and set the value to True.

Then each element of the model, or the entire model itself, can have security privileges applied to them.

When transferring the model database to another machine on a different domain always set the AccessControlled property to False. This ensures that the security is turned off at the Model. After transferring the database, you can set the required privileges for each user. To turn on the security, set the AccessControlled property to True.

If the security is turned off and the model database is transferred, then the users get insufficient privileges on the machine to which the database is transferred.

To set security for any element of a Model, perform the following:

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** menu, 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.

The Windows security dialog box enables you to add or remove users. You can also set the following privileges for each user or group:

- Full Control – Allows or denies the user to set the Security Information on an Element and all the privileges that applies to Read and Write permission.
- Read Logic – Allows or denies access to the Logic of a Method.
- Build – Allows or denies the ability to perform a build and deploy for a Model.
- Write – Allows or denies the ability to modify any attributes of an Element and all the privileges that applies to Read permission.

Allow and Deny Permissions

To explicitly allow or deny the permissions, select or clear the appropriate check box.

- Allow Permissions – If Allow is selected, then Permissions are allowed.
- Deny Permissions – If Deny is selected, then Permissions are denied.

Note: *The Deny entries take precedence over the Allow entries. So, if you are a member of both the groups, the Allow Permissions and Deny Permissions, then you are denied permission for the element.*

Explicit and Inherited Permissions

There are two types of permissions:

- Explicit Permissions – If you set the permissions on an element and select Allow or Deny, then it is Explicit Permission.
- Inherited Permissions – If you inherit the permissions from the parent of an element and do not select Allow or Deny, then it is Inherited Permission. If the check boxes under Permissions are shaded, the permissions are inherited from the parent. For example, an Element of a Model has the permissions inherited from a Model.

Note: *The Explicit Permissions take precedence over Inherited Permissions. This means, if you are a member of two groups, the Explicit Allow Permission and Inherited Permission which denied the same permission, then you are allowed that permission.*

Adding Projects

The following options present in the New Project dialog box of the System Modeler enables you to create a project:

- New Application – A new project for creating an AB Suite application or an AB Suite XML Framework application by using the built-in System Modeler Painter for user interface design.
- New Application For Client Framework – A new project for creating an AB Suite Client Framework application that allows you to design a user interface by using standard technologies, such as WPF, Silverlight, and ASP.Net.
- New Application From A File – A new project for restoring the backup file created using the AB Suite solution. Refer to Restoring the AB Suite Solutions for more information.
- Convert An Existing Model – A new project for creating an AB Suite Client Framework model or an AB Suite XML Framework model by converting an existing AB Suite model. See Converting the AB Suite Model for more information.
- Attach To An Existing Model – A new project that is associated with an existing AB Suite model or AB Suite Client Framework model.

Adding a Project to an Existing Model

To create a project by attaching to an existing AB Suite model or AB Suite Client Framework model, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.
2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Applications**.
3. In the Templates pane, select **Attach To An Existing Model**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to store the new project.
6. Click **OK**.

   The Attach To An Existing Model wizard appears.
7. From the **Server Name** list, select the SQL server.
8. From the **Database Name** list, select the database name.
9. Click **Next**.

   The project creation confirmation message appears in the wizard.
10. Click **Finish**.

A project file, with an .smproj extension, is created in the location specified. You can view the project through Solution Explorer, Class View, or Object Browser.

Adding a Project with an AB Suite Application

To create a project with an AB Suite application, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.

2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Applications**.
3. In the Templates pane, select **New Application**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to store the new project.
6. Click **OK**.

   The New Application Wizard appears.

7. From the **Server Name** list, select the SQL server. By default, this field displays the system name.
8. In the **Database Name** box, enter the database name. By default, this field displays the solution name.
9. In the **Model Name** box, enter the model name. By default, this field displays the solution name.
10. Select I**nclude XML Framework** check box, to create an AB Suite XML Framework model and enable the XML modelling extensions.
11. Click **Next** to specify the Configuration Settings. The configuration is set to Windows.

    Note: *If you are creating an XML Framework you cannot set the configuration to MCP as the XML functionality is not supported in an MCP model.*

12. Click **Next** to specify the Language Settings.
13. Select a primary language from the **Locale** list. By default, the primary language is set to 1033 - English (United States).
14. Click **Next** to specify the Presentation Settings. By default, the presentation type is set to Graphical.
15. Click **Next**.

    The project creation confirmation message appears in the wizard.

16. Click **Finish**.

A project file, with an .smproj extension, is created in the location specified. You can view the project through Solution Explorer, Class View, or Object Browser.

Note: *The features of an AB Suite model are fully enabled in an AB Suite XML Framework model.*

Adding a Project with an AB Suite Client Framework Application

To create a project with an AB Suite Client Framework application, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.
2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Applications**.
3. In the Templates pane, select **New Application For Client Framework**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to store the new project.
6. Click **OK**.

   The New Application Wizard appears.
7. From the **Server Name** list, select the SQL server. By default, this field displays the system name.
8. In the **Database Name** box, enter the database name. By default, this field displays the solution name.
9. In the **Model Name** box, enter the model name. By default, this field displays the solution name.

   Note: *The **Include XML Framework** check box is selected by default as the XML modelling extensions are fully enabled in a Client Framework model.*
10. Click **Next** to specify the Configuration Settings. The configuration is set to Windows.
11. Click **Next** to specify the Language Settings.
12. Select a primary language from the **Locale** list. By default, the primary language is set to 1033 - English (United States).
13. Click **Next**.

    The project creation confirmation message appears in the wizard.
14. Click **Finish**.

A Client Framework project file, with an .smproj extension, is created in the location specified. You can view the project through Solution Explorer, Class View, or Object Browser.

Note: *The XML modelling extensions are fully enabled in a Client Framework model.*

Adding System Modeler Items

You can add new items or existing items from several places within System Modeler, depending on the situation.

You can add elements from the

- Class View
- Members tab
- Subclass list of the class Properties tab
- Model Inheritance tab

You cannot add elements from the Object Browser.

Adding New Items

The process of adding a new element is the same for all System Modeler elements. The elements available to add differ according to the selected parent element.

To add a new element, perform either of the following:

1. Select a System Modeler element from any of the views described in Adding System Modeler Items.
2. Right-click and select **Add** from the cascading menu.
3. Select the element to add on the secondary, and any subsequent, cascading menu.

   The new element is added to your selected element with a default name.

Or

1. Select a System Modeler element from any of the views described in Adding System Modeler Items.
2. From the **File** menu, select **Add**.
3. Select **Add New Item**.

   The **Add New Item** dialog box is displayed.
4. Select the required element.
5. Enter a name for the new element.

   Note: *Element names must be unique amongst its siblings.*
6. If you want the document window for the new element to open immediately upon creation, select **Open the new item**.
7. Click **Create**.

   Note: *When you add a new element, the new element is selected. The element remains selected when it is updated.*

Add New Item Dialog Box

This dialog box allows you to add items to your System Modeler projects.

Templates

Displays the valid items that you can add to the selected element.

The templates are organized in three groups:

- Kinds
- Stereotypes
- Wizards

Name

Enter the name for the new item. If you do not specify a name, it is generated automatically.

Names can be between 1 and 64 characters in length and can contain a combination of alphanumeric characters and underscores. Names can contain non-ASCII characters.

A name must not begin with an underscore or numeral, and must not contain any white space, be that spaces or tabs.

Adding Existing Items

You can add existing elements to folders.

A folder is a convenient way of grouping together model elements; for example, all elements that are used by a particular business activity. A folder does not own the elements it contains. Ownership remains with the folder's parent element, meaning individual elements can be contained in more than one folder.

The **Add Existing Element** option enables you to select the elements you want to see in the folder. Elements can be added from either, Solution Explorer, Class View or the Members tab.

To add an existing item, perform either of the following:

1. Select a folder from any of the views described in Adding System Modeler Items.
2. Either right-click and select **Add Existing Item**, or on the **File** menu, select **Add Existing Item**.

   The Add Existing Item dialog box is displayed, which contains the elements available to be added to the selected model or folder. If no elements are available the dialog box is empty.
3. Either scroll or use the **Name** field to locate an element.

4. Select the required element.
5. Click **Open**.

Add Existing Item Dialog Box

You can use this dialog box to add existing items to a folder. Following is the description of the fields.

Members Filter

Allows user to filter members in the add dialog box.

From

The From field:

- Displays the namespace to which the folder belongs.
- Displays the items contained in the namespace that you can add to the folder.
- Displays the items either in a simple List or with Details. In both the modes, the items are sorted alphabetically. You can change the sorting in the Details mode.

Use filters to display only the required items.

To change the sort or set a filter, perform the following:

1. Right-click any column heading.
2. Select the action from the context menu.

Name

You can enter the name of the item to add to the folder. While entering the first letters of a name, the cursor scrolls to the item that most closely matches in the display field.

Notes: *Following rules are applied to all the items including Folders and Dictionaries except Diagrams:*

- *When you add an existing item such as report or ispec within a folder to another folder, duplicate instances are created in second folder. The changes made in any of the instances are reflected in other instance.*
- *When you add an existing item such as a report or ispec which is not within a folder to another folder moves the items to the folder. The items that are not within a folder are termed as Orphaned Items.*

Synchronize Selection in Class View

In windows where elements, like Members list, Search result List, Diagrams, and so on are displayed, you can synchronize a current element with the Class View.

For synchronizing a selected element, perform the following:

1. Right-click the selected element.
2. Select **Synchronize Class View** option.

This option is also available in the Quick Navigator window.

Adding IGraphicalPresentation for Client Framework Applications

The IGraphicalPresentation is an element that defines the attributes exposed by the AB Suite Access Layer for creating client applications by using the technology of your choice. It is automatically added when you add an Ispec, an Event, an Insertable, a CopyIspec, or a CopyEvent.

Note: *The IGraphicalPresentation is not added to a class automatically. Therefore, if you want to add the IGraphicalPresentation to a class, you must do it manually.*

To add an IGraphicalPresentation, perform the following:

1. Right-click the model, point to **Add**, and then select **Folder** from the context menu.

   A folder with the default name Folder1 appears under the model.
2. Rename the folder from Folder1 to an appropriate name.
3. Right-click the folder, point to **Add**, and then select **Segment** from the context menu.

   A segment with the default name Segment1 appears under the folder.
4. Rename the segment from Segment1 to an appropriate name.
5. Right-click the segment, point to **Add**, and then select any of the following elements from the context menu:

   - Ispec
   - Event
   - Insertable
   - CopyIspec
   - CopyEvent

   The element that you select from the context menu appears under the folder and theIGraphicalPresentation node appears automatically under the selected element.
6. Rename the element as appropriate.

You can also add the IGraphicalPresentation to a class by performing the following:

1. Perform step 1 through step 4.
2. Right-click the segment, point to **Add**, and then select **Class** from the context menu.
3. Rename the class from Class1 to an appropriate name.
4. Right-click the ispec, point to **Add**, and then select **Presentation** from the context menu.

   An IGraphicalPresentation node appears under the class.

You can now add attributes to the IGraphicalPresentation node by performing the following:

1. Right-click the **IGraphicalPresentation** node, point to **Add**, and then select **Attribute** from the context menu.

   An attribute with the default name Attribute1 appears in the Members pane.
2. Rename the attribute from Attribute1 to an appropriate name and set its properties as required.

   Note: *The attribute created under the IGraphicalPresentation node also appears in the Members pane for an ispec or a class definition. Different properties of an attribute can be set when it is selected in the context of the IGraphicalPresentation node, and when it is selected in the context of an ispec. The properties of an attribute under the IGraphicalPresentation node deal with the data being exchanged between the user interface and the runtime system. The properties of the same attribute that appears under the ispec deal with modelling and persistence of the attribute.*

After you add attributes to the IGraphicalPresentation node, you can generate the Client Framework projects and design the user interface by using the development tools available for the chosen client technology.

If you want to add an Ispec, an Event, an Insertable, a CopyIspec, or a CopyEvent without an IGraphical Presentation node, perform the following:

1. Right-click the model, point to **Add**, and then select **Folder** from the context menu.

   A folder with the default name Folder1 appears under the model.
2. Rename the folder from Folder1 to an appropriate name.
3. Right-click the folder, point to **Add**, and then select **Segment** from the context menu.

   A segment with the default name Segment1 appears under the folder.
4. Rename the segment from Segment1 to an appropriate name.

5. Right-click the segment, point to **Add**, and then select **Class** from the context menu.

   A class with the default name Class1 appears under the folder.

6. Open the Properties window of the class, and then select any of the following elements, which you want to add to the segment, from the Stereotype list.

   - Ispec
   - Event
   - Insertable
   - CopyIspec
   - CopyEvent

This adds the elements of your choice to the segment without the IGraphicalPresentation node.

Adding Attributes, Variables, or Parameters

In an XML Framework model or Client Framework model you can add attributes, variables, or parameters using a dialog box.

To add attributes, perform the following:

1. In the Class View window, right-click an element for which you want to add an attribute and then select **Attribute**.

   Or

   Right-click the element, point to **Add**, and then select **Add New Item…**.

   The **Add New Item** dialog box appears.

2. Select Attribute.

   The **Add New Attribute** dialog box appears.

   You can define a simple attribute, array attribute, or list attribute using this dialog box.

   To define a simple attribute

   a. In the **Template** box, optionally enter or browse to the class or type that this attribute derives from. If left blank, the attribute will be created with Primitive = String and Length = 1.

   b. In the **Name** box, enter an appropriate name for the attribute, or retain the default name.

   c. Click **Create**.

To define an array attribute, perform the following:

a. In the **Template** box, optionally enter or browse to the class or type that this attribute derives from. If left blank, the attribute will be created with Primitive = String and Length = 1.

b. In the **Name** box, enter an appropriate name for the attribute, or retain the default name.

c. In the **Multiplicity** box, enter a valid Multiplicity value. For example, '5', '2,2' (for a two dimensional array). By default, Multiplicity field is set to 1.

d. Click **Create**.

To define a list attribute

a. In the **Template** box, enter or browse to the class or type that this attribute derives from.

b. In the **Name** box, enter an appropriate name for the attribute, or retain the default name.

c. Select the **List** check box.

   Notes:
   - *Template is required for a List definition.*
   - *If you select the List check box the Multiplicity field is disabled because it is not valid to have an array of lists.*

d. Click **Create**.

Note: *The Kind property of the List attribute displays List Attribute.*

You can add more attributes by selecting the **Keep Open** check box.

You can also define a variable or a parameter by using a dialog box similar to the Add New Attribute dialog box. When you right-click a method, point to Add, and select Variable or Parameter from the context menu the Add New Variable or Add New Parameter dialog box appears. The fields in this dialog box are similar to the Add New Attribute dialog box, except the title of the dialog box.

Grouping Elements

It is possible to group attributes, parameters, and variables within your project.

Groups are displayed in the Members pane as an expandable list. Elements in groups cannot be sorted, but can be rearranged by dragging and dropping, or by changing the sequence numbers.

To add an element to a group, perform the following:

1. Select the element in the Members window.
2. Right-click the element and select **Add New Item to <selected item name>**.

Setting Properties

For each element in your project, the Properties window displays properties which influences how the element functions.

Properties are displayed for elements selected in the tool windows (Solution Explorer, Class View, Object browser) and the editors in the document windows.

For the Painter and UML Designer tabs, the object name field lists all available elements from the active tab. For other tabs, however, the object name field lists only the selected element.

The Properties window displays the properties of single or multiple selected elements. If multiple elements are selected, only the properties common to all selected elements are displayed.

The Properties window displays different types of editing fields, depending on the needs of a particular property. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes such as the Element Picker.

Properties shown in gray are read-only.

When a value is changed from the default it is displayed in bold. To revert a property to the default value, right-click and select **Reset**.

Refer to the *Agile Business Suite Developer Online Help* for more information on setting property values for graphical objects.

All Properties

The table below lists all properties for all elements. The following links describe the properties applicable to each element type:

- All Properties
- NOF Format Property
- Attribute Properties
- Class Properties
- Class Diagram Properties
- Dictionary Properties
- Copy Event Properties
- Copy Ispec Properties
- Event Properties
- External Class Properties
- Folder Properties
- Group Properties

- Frame Properties
- Insertable Properties
- Ispec Properties
- Location Properties
- Method Properties
- Messenger Properties
- Model Properties
- Parameter Properties
- Profile Properties
- Reference Properties
- Report Properties
- Segment Properties
- SQL Script Properties
- Serialization Properties
- Teach Screen Properties
- Variable Properties

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
AccessControlled	Allows an administrator to specify if the model, and all its elements are under access control. When set to True, all elements can individually have security applied to them.
ActmthNotEntered	Specifies whether end-users are allowed to alter the accounting month. When this property is set to True, the accounting month cannot be changed. The value of the accounting month is stored in the ACTMTH System attribute.
AllowPurge	Specifies whether users of your generated system are allowed to physically delete records using the PUR command in the Maint field of the standard Ispec screen. When the record is displayed, an entry of PUR in the Maint field causes a physical deletion to take place. If this value is False, users are not allowed to enter PUR in the Maint field. This is the default.
AlphaClearWhenCharacter	Specifies the DEF.WHEN.CLEAR property for alphanumeric attributes.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.

Property	Function
AutomaticEntryCapable	Specifies whether the element is able to accept Automatic Entries.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
AutoRecallCapable	Specifies whether you want to enable all data values for an Ispec record to be recalled to the screen, based on a specified Key value.
BaseYear	Specifies the year upon which the DateConvert command bases relative day numbers.
Caption	Language dependent description of an element to appear in messages.
CenturyStartYear	Specifies a secondary base year to define which century a date belongs to. This is independent of the value of the Base Year.
Conditions	Specifies logic that selects a subset of records from a file associated with the selected profile. This property is reflected in, and can be changed from, the Conditions tab of the selected profile.
Constraint	An expression that is an attribute qualified path to the value being represented by a Reference attribute.
ConvertToUpperCase	Specifies that all input is converted into upper case characters.
Copies	Specifies the number of copies that can be painted onto a form in Painter for an attribute that is a member of a Copy Ispec.
Created	Read-only. The date on which the selected element was created.
CurrencySign	Specifies the character to be output with EDIT $ Attributes in the Report to be modified or added to the selected Element. By default, the value is $.
DateConvertSetsGLB.CENTURY	Specifies whether Glb.Century is reset by a complex DateConvert command.
DateFormat	Specifies the format of the date throughout the segment and its element. Options are UK, International, or US.
DecimalCharacter	Specifies the character to be used as a decimal place.
Decimals	Specifies the number of decimal places required. The Decimals property is only displayed if the Primitive property is set to number or signed number.
DecimalsKeyed	Specifies the default for how decimal points are used in numeric attributes. This property's value is inherited.

Property	Function
DefaultCursorField	Specifies the Default cursor field position, otherwise cursor positioning may be random. You can set this property by using any one of the following ways: • Use the Element Picker to select the attribute in the DefaultCursorField property field. • Enter the attribute name in the DefaultCursorField property field manually. ***Notes:*** • *This property is valid only if the Direction property is set to In or InOut.* • *The attribute being set in the field must be of primitive type or a reference.*
DefaultDevice	Specifies the device to which the report is print or display.
Fixed Type	Specifies the default settings for background color, foreground color and font for Fixed Presentation types in Painter at Model or Segment levels. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
Graphical Type	Specifies the default settings for background color, foreground color, font, ScrollBars, ShowActmth, ShowHeader, and TransmitToCursor for Graphical Presentation types in Painter at Model or Segment levels. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
Print Type	Specifies the default settings for BlankWhenZero, background color, FloatingSign, foreground color and font for Print Presentation types in Painter at Model or Segment levels. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information. ***Note:*** *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties using AB Suite Runtime Administration Tool.* *Refer to the* Agile Business Suite Runtime for Windows® Operating System Administration Guide *for more information.*
Direction	For an attribute, painted on a form, a direction of Out indicates that the control is read-only.
DuplicatesAllowed	Specifies whether the profile is permitted to contain duplicate key values.

Property	Function
EnforcePersistent	Specifies that when this property is set to true, all objects within this namespace that are persistent, or are made persistent, must inherit their definition from an item in a Dictionary. Applies to Model and Segment elements only.
EnforcePresentation	Specifies that when this property is set to true, all objects within this namespace that have their direction set to a value other than None, must inherit their definition from an item in a Dictionary. Applies to Model and Segment elements only.
EventSet	Defines an AutoPersist dependency between two Events. This dependency causes all the persistent attributes of this event to persist in the EventSet. The value of the EventSet property must be the name of another event.
FullSuppression	Display a zero numeric value as spaces.
HasMaint	Specifies whether the External Ispec contains the MAINT field. ***Note:*** *This property becomes visible only if the IsExternal property of an ispec is set to True.*
IfPresent	Specifies whether to look up the object if the value for the key is defined. If set to True, the profile becomes conditional and returns a view instead of an index. ***Note:*** *This property becomes visible only when a key is selected.*
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an "instance of" when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
Integrity	Indicates if Dataset locking is enabled automatically, and at what level. This ensures processing integrity and full-synchronized recovery.

Property	Function
InquiryOnly	Specifies whether to prevent Report logic from updating a database. The following LDL logic commands should not be used: FLAG, STORE, and PURGE. When you set Inquiry Only to True, all existing logic will need to be validated again.
IsAscending	Keys are sorted in ascending order.
IsConstant	Specifies that an object cannot have its value changed in logic.
IsExternal	Specifies whether the class is acting as a proxy for an external component, class or library. Even when this property is set to True, the elementstill contains the following Ispec system Attributes: Actmth, Glb.Source, Input_Date, Ispec, and TranNo. ***Note:*** *These attributes are displayed in Inherits dialog box.*
IsFinal	Specifies that the method cannot be overridden in a sub class.
IsInnerClass	Instances of an inner class are linked to an instance of its owner and has have access to all its members of its containing class. Only one class in an inheritance hierarchy can have IsInnerClass set to True. When a complex class is inherited, this property is derived by the child class.
IsKey	Specifies whether the attribute acts as a key in the database table containing its owning class.
IsPersistent	Specifies whether the element is persistent.
IsRequired	Specifies whether the primitive attribute requires a value.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to False. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsSynchronous	Specifies whether the method can execute synchronously.
IsUnique	Specifies whether Items can be added to multiple Dictionaries, if this property is true items can only be added to that Dictionary.
Kind	Read-only. Identifies the kind of element selected.
Language	Specifies the programming language to be used for the logic.
Length	Specifies a numeric value for the length of the attribute.
Leftfill Numerics	Specifies that the numeric screen items are filled from the left, not the right.

Property	Function
LineLength	Specifies the maximum length of the Report print line. This property is read-only for frames. ***Notes:*** • *For reports that are generated in an MCP non-ROC system, you should consider the following rules:* • *If DefaultDevice is LP or RP, LineLength is restricted to 80-132.* • *If DefaultDevice is DP, LineLength is restricted to 255.* • *If DefaultDevice is VD, LineLength is restricted to 80.* • *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties using AB Suite Runtime Administration Tool.* *Refer to the* Agile Business Suite Runtime for Windows® Operating System Administration Guide *for more information.*
LineSpacing	Specifies the spacing between the lines.
Locale	Specifies the Locale Id of the given language.
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
MultiByteClearWhen Character	Specifies the clear-when character to be used with MultiByte strings. ***Note:*** *This property becomes visible only if Internationalization Support is set to Multibyte String Support.*
MultiByteStringValidation	Specifies the type of validation to be applied to MultiByte strings. ***Note:*** *This property becomes visible only if Internationalization Support is set to Multibyte String Support. When Internationalization Support is set to some other value, this property must be set to spaces.*
Modified	Read-only. The date the element was last modified.
Namespace	Specifies a namespace for qualifying element used in XML.
Multiplicity	Number of instances.

Property	Function
NationalString	Specifies the type of internationalization support for Single Byte and MultiByte Coded Character Sets. The runtime subsystem allows users to make use of different collating sequences on the same single-byte Character Code Set. The national strings in a segment could use one collating sequence when a system is generated from one configure set and a different collating sequence in a system generated from another configure set. The following options are available for this property: • None – Indicates that there is no international support. • SingleByte string support – Provides the ability to generate systems without support of KANJI characters. • MultiByte string support – Indicates internationalization support of KANJI characters. • Unicode – Indicates that there is no international support. Any MCP generatefails if Unicode is selected.
NOFLength	Specifies the size of an object in a fixed presentation.
NOFOrder	Redefining the NOF Order.
NOF Format	CR/DR or +/- ***Note:*** *NOF Format is also used to define the NOF Format Presentation/Painter properties.*
NumericClearWhenCharacter	Specifies the DEF.WHEN.CLEAR property for numeric attributes. This value can be set to blank or None.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Pitch	Specifies the default pitch setting (characters per line) for printed Report output. The following pitches are available: 220, 160, 132, 99, 66, and 49. ***Note:*** *This property is preset and disabled if the Default Device for your Report is Enterprise Output Manager (EOM) generated Report.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to Presentation Type formats for more information on options. For Copy Ispecs the default value is Graphical and is read-only. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.

Property	Function
Primitive	Specifies the kind of data that the element can contain. The following values are available for this property: Boolean, String, Signed Number, Date, Class, National string, and MultiByte string. ***Note:*** *A National String behaves the same as a String until the National Support is enabled on the Segment.*
PrintIfPresent	Specifies whether a line is to be printed if the attribute value is not equal to spaces or zeros.
Project File Path	Specifies the full path name of the project on the disk.
ProductionSystem	Specifies whether the Database auditing of the System's application data must occur when the System has been generated.
RefreshPresentation	Specifies whether the framework Construct method should always be called at the end of the segment cycle.
ReportParameter	Specifies the report attribute that revives the parameter used to invoke the report.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved. An element can be reserved in two ways: Explicitly by the user or implicitly by the model when any of the documentation windows are edited.
RSNCapable	Specifies that when this property is set to true, includes a built-in persistent attribute called Identifier from the framework that acts as the auto incrementing number. The RSN Capable property can be set when the class has persistent members. The RSNCapable must be set on the base class of an inheritance hierarchy. All subclasses derive the property value from the base class.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Semantics	Specifies the semantics to be used by the element. You can select either System Modeler specific or conventional semantics.
SeparatorCharacter	Specifies the character to be used to delimit groups of three digits.
Sequence	Specifies the sequence number of the selected element. Changing the sequence number of one element affects the order of the other elements in the list.
Server Name	Specifies the server that hosts the model database.
StandardHeading	Specifies whether to print the standard heading on each page of the Report.

Property	Function
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
SuppressZeros	Suppresses the leading zeros for numeric data types in Ispec screen layouts. This option does not apply to data types in Reports and Usage Inquiry Attributes. The SuppressZeros property results in non-significant zeros to be replaced by spaces on output and leading spaces changed to zeros on input. If a separator character is also specified, the leading separators are also changed to spaces. A zero value appears as 0.
SynchronizedField	Specifies whether a control in a presentation has its height and width adjusted when the length of the contents are changed. Fixed Width indicates that only controls with fixed width fonts are synchronized. This property is only available at the Segment level.
Template	Derives the class or type for an attribute. It allows to instantiate an instance that cannot be extended. This property can be set to any type (Primitive or List), Class, or Serialization. ***Note:*** *In an AB Suite model (that is, a model that does not have the model extensions enabled) all attributes are created as Inherits-based. Template-based attributes are not available in such a model.*
Value	Specifies the initial value for the Attribute.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
VideoCapable	Specifies whether to direct the output of a Report to a video device. ***Note:*** *In Windows® runtime, this property is ignored as all reports are automatically Video Capable.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

NOF Format Property

The NOF Format object property is used to define the NOF format as a signed number attribute for Model properties and Painter properties. The following table describes the values, and identifies the primitive type to which they are applicable.

Value	Description	Primitive Type
CR	Signed (CR or DR) numeric, default CR	Signed Number
-	Signed (CR or DR) numeric, default -	Signed Number
+	Signed (CR or DR) numeric, default +	Signed Number
DR	Defaults to DR	Signed Number

Note: *For signed attribute having direction as None, NOF Format property is not displayed in property window.*

Attribute Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
ConvertToUpperCase	Specifies that all inputs are converted into upper case characters.
Created	Read-only. The date on which the selected element is created.
Decimals	Specifies the number of decimal places required. The Decimals property is only displayed if the Primitive property is set to number or signed number.
DecimalsKeyed	Specifies the default for how decimal points are used in numeric attributes. This value is inherited.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
Direction	Specifies how the parameter passes data; in, out, or both in and out. For an Attribute, painted on a form, a direction of Out indicates that the control is read-only.

Property	Function
Element Name	Specifies the name of the element in the XML message. This need not be same as the attribute name.
EnableMaskDefinition	Specifies that when this property is set to true, the attribute data is masked in the log file at runtime. By default, the value is False. To support data masking in the MCP database, this property is enabled for attributes when the • Direction property is set to None. • IsPersistent property is set to True. • Primitive property is set to either Number or String. Data masking of persistent attributes in the database is only available on the MCP platform. ***Notes:*** • *MCP has an optional LogLibSecurity host utility at runtime that lets you enter the attributes to be masked along with the starting offset, mask length, and mask character. Refer to the* Agile Business Suite Runtime for ClearPath MCP Administration Guide *for more information on the LogLibSecurity utility.* • *EnableMaskDefinition is used in conjunction with the Obfuscate Level MCP segment configuration property to mask data in the MCP database. Data masking occurs when EnableMaskDefinition is set to True and when the* – *Obfuscate Level = 1 or* – *Obfuscate Level = 2 or* – *Obfuscate Level = 3 and Extended Edition is set to True.*
ExcludeIfEmpty	When formatting an XML message if ExcludeIfEmpty is set to True and the attribute value is blank or zero, then the element is not included in the output XML message.
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an “instance of” when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
Integrity	Indicates if Dataset locking is enabled automatically, and at what level. This ensures processing integrity and full synchronized recovery.

Property	Function
Is Key	Specifies whether the attribute acts as a key in the database table containing its owning class.
IsPersistent	Specifies whether the element is persistent. If IsPersistent is set to True, an attribute that is a member of an Ispec or Vanilla class is database persistent, that is, saved as a column in the database.
IsRequired	Specifies whether the primitive attribute requires a value. For an attribute in a Serialization (serializable interface), in an XML Framework application, if IsRequired is set to True it indicates that an input XML message must contain the element and it must contain a value.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element is resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
Lock Sign	Check this check box to specify that a signed field is to be locked at run time. The numeric value preceding the signed field may be entered or modified, but the sign itself may not.
MaskDefinition	Specifies the mask definition that can be applied to the attribute data at runtime when the EnableMaskDefintion property is true. This property is optional and if left blank, the default mask character, '*' is applied to mask the entire field value. You can specify a string of characters as the mask definition to mask a field value at runtime. Only non-Unicode and printable characters can be specified in the mask. A period (.) indicates that the underlying position in the field is not masked. For example, the mask definition "….####....&&&&" has two separate masks at different positions in the field. Hence "ABCDEFGHIJKLMNOP" would be masked as "ABCD####IJKL&&&&" This property is enabled for attributes that have the Direction property set to an option other than None and the Primitive property set to either Number or String. ***Notes:*** • *MCP has an optional LogLibSecurity host utility at runtime that lets you enter the attributes to be masked along with the starting offset, mask length, and mask character. Refer to the* Agile Business Suite Runtime for ClearPath MCP Administration Guide *for more information on the LogLibSecurity utility.* • *This property is ignored for the masking of data in the MCP database. DMSII uses its own obfuscation algorithms.*
Modified	Read-only. The date the element was last modified.

Property	Function
Multiplicity	Number of instances. ***Caution:*** *In MCP Runtime, Enterprise Database Server (DMSII) does not allow the number of dimensions to change on an existing persistent attribute.*
NOFOrder	Redefining the NOF Order.
NOFType	CR/DR or +/-
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Primitive	Specifies the kind of data that the element can contain. The following values are available for this property: Boolean, String, Signed Number, Date, Class, National string, and MultiByte string. ***Note:*** *The National String option is made visible only if Internationalization Support is set to National String Support.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Semantics	Specifies the semantics to be used by the element. You can select either System Modeler specific or conventional semantics.
Sequence	Specifies the sequence number of the selected element. Changing the sequence number of one element affects the order of the other elements in the list. For an attribute in a Serialization (serializable interface), in an XML Framework application, Sequence specifies the order in which the element is written into an output XML message. The order of elements in an input XML message is not significant.
StoreIfPresent	A property that can be applied to persistent, non-key attributes to improve database performance when handling large data types with low expected numbers. The database space is optimized by maintaining a database structure for only those attributes that do not have a space or zero as their value. This feature is supported only in the MCP runtime environment.
SuppressZeros	Suppresses the leading zeros. ***Note:*** *This property is applicable only when:* • *Attribute's primitive type is set to a Number or Signed Number.* • *Attribute's owner has a Presentation and its either Fixed, Graphical, or Graphical and Fixed.* • *Attribute's Direction is either In, Out, or InOut.*

Property	Function
Template	Derives the class or type for an attribute. It allows to instantiate an instance that cannot be extended. This property can be set to any type (Primitive or List), class, or Serialization. ***Note:*** *In an AB Suite model (that is, a model that does not have the model extensions enabled) all attributes are created as Inherits-based. Template-based attributes are not available in such a model.*
Value	Specifies the initial value for the element. ***Note:*** *This value is validated automatically after the text is entered and errors are shown in the Error List.*
VersionFile	Specifies the name of the file which contains details of the element in the Source Control Bank.
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.
XsdType	Specifies the XML type when generating an XML Schema Definition (XSD) from a Messenger class or a Serializable class definition.

Initial Value

The Value property is used to specify the initial value for an attribute.

Different initial value can be specified for each language translation associated with the attribute. Each value can be edited either from the Translations tab or by selecting the language from the Session Language drop-down list and editing the value property of an attribute.

Initial value must confirm to the following rules:

- Length of the value must not be greater than the length of attribute.
- Double-quote (“) and backslash (\)must be ‘escaped’ by doubling it or preceding it with a backslash.
- Initial value must be compatible with attribute data type.
- Numeric:
  - Number data type cannot have sign (+/-) in the initial value.
  - Number of decimals in the initial value must be less than or equal to the decimals specified in the decimal property of an attribute.
  - If the decimal property is not set, initial value of an attribute cannot have decimal character.

The initial values associated with an attribute for the different language translations can be validated by using validate option in the attribute level.

In the Painter, the Value property of a field specifies its default value that gets loaded into the field when the host sends a blank value.

Note: *The value of this property is displayed in the Painter and Debugger screens. This helps to visualize the look-and-feel of the form with representative data shown in the associated fields. So, the Value property is applicable only for Winforms and not for CE clients.*

Examples of Initial Value

Example 1: When the attribute is of data type Number

Valid:

123
1.23 (Decimal property set to 2)

Invalid:

+123 (Number must be unsigned)
1.2 (Decimal property set to 0)
1.234 (Decimal property set to 2)
1+2 (Arithmetic expression in the initial value)
1a2 (Alpha numeric in the initial value)

Example 2: When the attribute is of data type SignedNumber

Valid:

+123
-1.23 (Decimal property set to 2)

Invalid:

1+23 (Arithmetic expression in the initial value)
1a2 (Alpha numeric in the initial value)

Example 3: When the attribute is of data type Boolean

Valid:

True
False
Yes
No
Y
N

Invalid:

123 (Incompatible)
Abc (Incompatible)
1a2 (Incompatible)

Example 4: When the attribute is of data type String

Valid:

123
\\abc
\"1+2
""12a

Invalid:

\123 (Backslash must be escaped)
"abc" (Double-quote must be escaped)
1"a2 (Double-quote must be escaped)

Example 5: When the attribute is of data type Date

Valid:

12122012
01021999
17011986

Invalid:

11Jan2016 (Incompatible)
Jan112009 (Incompatible)

Class Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.

Property	Function
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an "instance of" when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
Integrity	Indicates if Dataset locking is enabled automatically, and at what level. This ensures processing integrity and fully synchronized recovery.
IsConstant	Specifies that an object cannot have its value changed in logic.
IsExternal	Specifies whether the class is acting as a proxy for an external component, class or library.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.

Property	Function
Multiplicity	Number of instances. ***Note:*** *MCP-based systems support a maximum of 255 instances of any persistent class. There is no specific limit for non-persistent classes.*
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. By default for Copy Ispecs, the value is True and is read-only. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Class Diagram Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.

Property	Function
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.

Dictionary Properties

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsUnique	Specifies whether Items can be added to multiple Dictionaries, if this property is true items can only be added to that Dictionary.

Property	Function
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings,.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved. An element can be reserved in two ways: Explicitly by the user or implicitly by the model when any of the documentation windows are edited.

Copy Event Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Copies	Specifies the number of copies that can be painted onto a form in Painter for an attribute that is a member of a Copy Ispec.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces.

Property	Function
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an "instance of" when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* Refer to IsInnerClass property for more information.
Integrity	Indicates if Dataset locking is enabled automatically, and at what level. This ensures processing integrity and a fully synchronized recovery.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. By default for Copy Events, the value is Graphical and is read-only.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.

Property	Function
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Copy Ispec Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticEntryCapable	Specifies if the element is able to accept Automatic Entries.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Copies	Specifies the number of copies that can be painted onto a form in Painter for an attribute that is a member of a Copy Ispec.

Property	Function
Copy From Line	Specifies the line number to be the starting point in COPY.FROM Ispecs. Use this property in conjunction with the Copy To Line property and the Max Copies field to specify the group of lines to be repeated in a COPY.FROM Ispec.
Copy To Line	Specifies the line number to be the final line used in COPY.FROM Ispecs. Use this property in conjunction with the Copy From Line property and the Max Copies field to specify the group of lines to be repeated in a COPY.FROM Ispec.
Created	Read-only. The date on which the selected element was created.
DefaultCursorField	Specifies the Default cursor field position, otherwise cursor positioning may be random. You can set this property by using any one of the following ways: • Use the Element Picker to select the attribute in the DefaultCursorField property field. • Enter the attribute name in the DefaultCursorField property field manually. ***Notes:*** • *This property is valid only if the Direction property is set to In or InOut.* • *The attribute being set in the field must be of primitive type or a reference.*
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces.
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an “instance of” when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* Refer to IsInnerClass property for more information.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsSynchronous	Specifies whether the method can execute synchronously.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. By default for Copy Ispecs, the value is Graphical and is read-only. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policies settings.*

Property	Function
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Event Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticEntryCapable	Specifies if the element is able to accept Automatic Entries.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
EventSet	Defines an AutoPersist dependency between two Events. This dependency causes all the persistent attributes of this event to persist in the EventSet. The value of the EventSet property must be the name of another event.
FullSuppression	Display a zero numeric value as spaces.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsSynchronous	Specifies whether the method can execute synchronously.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. Select a value other than None to add the Painter tab to the document window. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
RefreshPresentation	Specifies whether the framework Construct method must always be called at the end of the segment cycle.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*

Property	Function
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

External Class Properties

The following table lists all the properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties become visible only when a dependent property is set to a specific value.

The Synchronize properties are relevant only to external classes with persistent members. These properties indicate the external database schema file that is imported and whether the external class should be synchronized with the imported schema file.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER. Reflected in the Properties tab.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsExternal	Specifies whether the class is external and acts as a proxy for an external component, class, library, or data source.
Kind	Read-only. Identifies the kind of element selected.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the settings of System Modeler Policies. Refer to System Modeler Policies for more information.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, System Modeler prompts for a change in the value of the VersionFile property based on the System Modeler Policies. Refer to System Modeler Policies for more information.*
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Created	Read-only. The date on which the selected element was created.

Property	**Function**
Modified	Read-only. The date the element was last modified.
MemberPersistence	Specifies the default persistence of the members of an external class. ***Note:*** *This property is False for an imported OLTP view file.*
InterfaceType	Indicates additional framework methods required to represent the interface. Set to None for external classes that are manually created. This property is not available for an imported EAE RDML schema file. The **OltpView** option is set for an imported OLTP view file. ***Note:*** *The* ***ViewName*** *and* ***ViewType*** *properties are available if you have selected the OltpView option.*
SourceName	Set by the External Class Wizard to the external resource name. For manually created external classes, the SourceName is automatically set to the class name. You can also set it to a name other than the class name so that it does not change if the class name is changed. ***Notes:*** • *The SourceName is not case-sensitive.* • *This property is available for external classes that are used for external data sources.* • *If the SourceName does not match the name on the host, Debugger reports a schema mismatch error.*
ViewName	Should match the View name of the server view files. ***Note:*** *This property is available only if you have selected the OltpView option in the InterfaceType for an external class.*
ViewType	Should be set to X_COMMON as AB Suite currently supports only this view type. ***Note:*** *This property is available only if you have selected the OltpView option in the InterfaceType for an external class.*
SynchronizeFile	Set by the External Class Wizard to the resource used to create and synchronize the external class and is read-only. ***Note:*** *This property is available only for external classes with persistent members.*

Property	Function
SynchronizeState	Indicates the status of the resource if it is present in the SynchronizeFile. It is read-only. • **InSync** Indicates that the resource has not changed since the class was last created or synchronized. • **NeedsSync** Indicates that the resource is present but has changed since the class was last created. This value is changed to InSync after an external class is synchronized or it is changed to NoFile if the resource file is removed. • **NoFile** Indicates that the resource does not exist. This value is changed to NeedsSync after the resource file is specified or it is changed to InSync if the resource file is removed and then added after an external class is imported. ***Note:*** *This property is available only for external classes with persistent members.*
Length	Read-only. Specifies the sum of all the lengths of its members.
Multiplicity	Specifies the number of instances. Each instance represents a unique connection to the external resource.
MemberVisibility	Specifies the default visibility of the members of an external class. By default, it is set to Private. Is set to Public for an external class that contains an imported database schema file.
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace that inherits from it. Public – Visible outside its namespace. Is set to Public for an external class that contains an imported database schema file.

Folder Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER. Reflected in the Properties tab.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsUnique	Enforces the Uniqueness of all its member elements within the model.
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.

Group Properties

The table below lists all properties for Group. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element.
Kind	Read-only. Identifies the kind of element selected. The elements can be Class, Attribute, Keyword or Variable.
Length	Read-only. Specifies the sum of all the lengths of its members.
MemberVisibility	Specifies the default visibility for new member elements. On Groups, it is always set to Public.
Modified	Read-only. The date the element was last modified.

Property	Function
Multiplicity	Number of instances. ***Note:*** *MCP-based systems support a maximum of 255 instances of any persistent class. There is no specific limit for non-persistent classes.*
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options.
Primitive	Read-only. Specifies the kind of data that the element can contain. On Groups, it is always set to String.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system. Changing this value makes the class no longer act like a group.
Type	Read-only. The Type property displays the name of the class that defines this object. In groups, the type always refers to this class as they can never derive their definition.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its owner. Protected – Visible within its owner and any class inheriting from the owner. Public – Visible within and outside its owner and any class inheriting from the owner.

Frame Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an “instance of” when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* Refer to IsInnerClass property for more information.
IsConstant	An object that is constant cannot have its value changed in logic.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.
Iskey	Does this Attribute act as a key in the database table containing its class?
IsPersistent	Is this object persistent?
IsRequired	Specifies whether the primitive attribute requires a value.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsSynchronous	Specifies whether the method can execute synchronously.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
LineLength	Maximum length of the Report print line. This property is read-only for Frames. ***Notes:*** • *For reports that are generated in an MCP non-ROC system, you should consider the following rules:* • *If DefaultDevice is LP or RP, LineLength is restricted to 80-132.* • *If DefaultDevice is DP, LineLength is restricted to 255.* • *If DefaultDevice is VD, LineLength is restricted to 80.* • *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties by using AB Suite Runtime Administration Tool.* *Refer to the* Agile Business Suite Runtime for Windows® Operating System Administration Guide *for more information.*
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. Select a value other than None to add the Painter tab to the document window. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.

Property	Function
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Sequence	Sequence number in the ordered list.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Insertable Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Created	Read-only. The date on which the selected element was created.
Decimals	Specifies the number of decimal places required. The Decimals property is only displayed if the Primitive property is set to number or signed number.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces.

Property	Function
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsSynchronous	Specifies whether the method can execute synchronously.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. Select a value other than None to add the Painter tab to the document window. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*

Property	Function
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Ispec Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
AllowPurge	Specifies whether users of your generated system are allowed to physically delete records using the PUR command in the Maint field of the standard Ispec screen. When the record is displayed, an entry of PUR in the Maint field causes a physical deletion to take place. If this value is False, users are not allowed to enter PUR in the Maint field. This is the default.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticEntryCapable	Specifies whether the element is able to accept Automatic Entries.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
AutoRecallCapable	Specifies whether you want to enable all data values for an Ispec record to be recalled to the screen, based on a specified Key value.
Created	Read-only. The date on which the selected element was created.

Property	Function
DefaultCursorField	Specifies the Default cursor field position, otherwise cursor positioning may be random. You can set this property by using any one of the following ways: • Use the Element Picker to select the attribute in the DefaultCursorField property field. • Enter the attribute name in the DefaultCursorField property field manually. ***Notes:*** • *This property is valid only if the Direction property is set to In or InOut.* • *The attribute being set in the field must be of primitive type or a reference.*
DefaultProfile	Defines the keys or profile to be used for the class.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces
HasMaint	Specifies whether the External Ispec contains the MAINT field. ***Note:*** *This property becomes visible only if the IsExternal property of an ispec is set to True.*
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an “instance of” when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* Refer to IsInnerClass property for more information.
IsExternal	Specifies whether the class is acting as a proxy for an external component, class or library. Even when this property is set to True, the element still contains the following Ispec system Attributes: Actmth, Glb.Source, Input_Date, Ispec, and TranNo.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsPersistent	If IsPersistent is set to True, an attribute that is a member of an Ispec or Vanilla class is "database" persistent, that is, saved as a column in the database.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberPersistence	Sets the default to Persistent property for the members.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
PresentationType	Specifies if the selected element has a user interface, and what format such an interface takes. Refer to PresentationType formats for more information on options. Select a value other than None to add the Painter tab to the document window. Refer to the *Agile Business Suite Developer Online Help* for more information on using the Painter tab.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*

Property	Function
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Location Properties

Note: *This element is applicable for MCP and Windows® based systems.*

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element.
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	Read-only. The namespace to which the element belongs.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.

Method Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Caption	Language dependent description of an element to appear in messages.
Created	Read-only. The date on which the selected element was created.

Property	Function
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsFinal	Specifies that the method cannot be overridden in a sub class.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Messenger Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element. SUPER, COMPONENT, THIS and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.

Property	Function
Created	Read-only. The date on which the selected element is created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsInnerClass	Instances of an inner class are linked to an instance of its owner and have access to all its members of its containing class. Only one class in an inheritance hierarchy can have IsInnerClass set to True. When a complex class is inherited, this property is derived by the child class.
Kind	Read-only. Identifies the kind of element selected.
Length	Read-only. Specifies a numeric value for the length of the attribute.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Primitive	Read-only. This property is set to Class.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Stereotype	Read-only. This property is set to Messenger in a Messenger class.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Model Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
AccessControlled	Allows an administrator to specify if the model, and all its elements are under access control. When set to True, all elements can individually have security applied to them.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
Fixed Type	Specifies the default settings for background color, foreground color and font for Fixed Presentation types in Painter at the model level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
Graphical Type	Specifies the default settings for background color, foreground color, font, ScrollBars, ShowActmth, ShowHeader, and TransmitToCursor for Graphical Presentation types in Painter at the model level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
Print Type	Specifies the default settings for background color, foreground color and font for Print Presentation types in Painter at the model level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information. ***Note:*** *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties using Agile Business Suite Runtime Administration Tool. Refer to the* Agile Business Suite Runtime for Operating System Administration Guide *for more information.*
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER. A model can be renamed ***Note:*** *The model name and the database name are not necessarily same.*
ProjectFilePath	Specifies the full path name of the project on the disk.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.

Property	Function
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Server Name	Specifies the server that hosts the model database.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policiesfor more information on System Modeler Policies settings.*

Parameter Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticTab	Cursor automatically tabs to next field on a presentation screen at runtime. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
Caption	Language dependent description of an element to appear in messages.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces.

Property	Function
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an "instance of" when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Primitive	Specifies the kind of data that the element can contain. The following values are available for this property: Boolean, String, Signed Number, Date, Class, National string, and MultiByte string.
PrintIfPresent	Specifies whether a line is to be printed if the attribute value is not equal to spaces or zeros.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.

Property	Function
Sequence	Specifies the sequence number of the selected element. Changing the sequence number of one element affects the order of the other elements in the list.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
SuppressZeros	Suppresses the leading zeros.
Value	Specifies the initial value for the element.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*

Profile Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Caption	Language dependent description of an element to appear in messages.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
DuplicatesAllowed	Specifies whether the profile is permitted to contain duplicate Key values.
IfPresent	Specifies whether to look up the object if the value for the key is defined. If set to True, the profile becomes conditional and returns a view instead of an index. ***Note:*** *This property becomes visible only when a key is selected.*
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*

Property	Function
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

To create a profile for more than one event, perform the following:

1. Copy the key attributes for the profile from the relevant Event into the Event acting as the EventSet for the group and create a profile under the EventSet using these keys.
2. Create a profile conditions that includes each of the relevant Event classes in the profile. For example,

   ispec = EVNT1 and

   ispec = EVNT2

Reference Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.

Property	Function
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Constraint	An expression that is an attribute qualified path to the value being represented by a Reference attribute. ***Note:*** *This value is validated automatically after the text is entered and errors are shown in the Error List.*
Decimals	Specifies the number of decimal places required. The Decimals property is only displayed if the Primitive property is set to number or signed number.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
Direction	Specifies how the parameter passes data; in, out, or both in and out. For an Attribute, painted on a form, a direction of Out indicates that the control is read-only. The Direction property is only visible after the element has been validated and the Inherits property set.
Inherits	Read-only. Specifies a class from which the selected class inherits. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *The Inherits property is automatically set to the referenced attribute when the expression in the Constraint property is validated and a Use dependency exists. The expression is parsed on validation of the owner.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
NOFLength	Specifies the size of an object in a fixed presentation.
NOFOrder	Redefining the NOF Order.
Owner	Read-only. The namespace to which the element belongs.
Primitive	Specifies the kind of data that the element can contain. The following values are available for this property: Boolean, String, Signed Number, Date, Class, National string, and MultiByte string. ***Note:*** *The National String option is made visible only if Internationalization Support is set to National String Support.*

Property	Function
SuppressZeros	Suppresses the leading zeros. ***Note:*** *This property is applicable only when:* • *A Reference has a valid Constraint and references to a Number or Signed Number.* • *Reference's Owner has a Presentation and its either Fixed, Graphical, or Graphical and Fixed.* • *The Direction of Reference is either In, Out, or InOut.*

Report Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
Author	Read-only. The identifier of the person who created the selected element.
Created	Read-only. The date on which the selected element was created.
CurrencySign	Specifies the character to be output with EDIT $ Attributes in the Report to be modified or added to the selected Element. By default, the value is $.
DecimalCharacter	Specifies the character to be used as a decimal place.
DecimalsKeyed	Specifies the default for how decimal points are used in numeric attributes. This value is inherited.
DefaultDevice	Specifies the device to which the report prints or displays.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
FullSuppression	Display a zero numeric value as spaces.
InquiryOnly	Specifies whether to prevent Report logic from updating a database. The following LDL+ logic commands should not be used: FLAG, STORE, and PURGE. When you set Inquiry Only to True, all the existing logics needs to be validated again.
IsInnerClass	An inner class has access to all the members of its containing class. When a complex class is inherited, this property is derived by the child class.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsPersistent	If IsPersistent is set to True, an attribute that is a member of a Reportis "critical point" persistent, that is, saved at CriticalPoint commands.
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
LineSpacing	Specifies the spacing between the lines.
LineLength	Specifies the maximum length of the Report print line. ***Notes:*** • *For reports that are generated in an MCP non-ROC system, you should consider the following rules:* • *If DefaultDevice is LP or RP, LineLength is restricted to 80-132.* • *If DefaultDevice is DP, LineLength is restricted to 255.* • *If DefaultDevice is VD, LineLength is restricted to 80.* • *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties using AB Suite Runtime Administration Tool.* *Refer to the* Agile Business Suite Runtime for Windows® Operating System Administration Guide *for more information.*
MemberPersistence	Do members default to Persistent?
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Pitch	Specifies the default pitch setting (characters per line) for printed Report output. The following pitches are available: 220, 160, 132, 99, 66, and 49. ***Note:*** *This property is preset and disabled if the Default Device for your Report is Enterprise Output Manager generated Report.*

Property	Function
Print Type	Specifies the default settings for background color, foreground color and font for Print Presentation types in Painter at Model or Segment levels. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information. ***Note:*** *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties using AB Suite Runtime Administration Tool.* *Refer to the* Agile Business Suite Runtime for Windows Operating System Administration Guide *for more information on system modeler policy settings.*
ReportParameter	Specifies the report attribute that revives the parameter used to invoke the report.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
SeparatorCharacter	Specifies the character to be used to delimit groups of three digits.
Stereotype	Specifies the stereotype which identifies how the class operates and is interpreted in the system.
StandardHeading	Specifies whether to print the standard heading on each page of the Report.
VersionFile	Specifies the name of the file which contains details of the element in the Source Control Bank.
VideoCapable	Specifies whether to direct the output of a Report to a video device. ***Note:*** *In Windows®runtime this property is ignored as all reports are automatically Video Capable.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Segment Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Alias	An automatically generated unique name for the selected element. You can change this name, but if it clashes with an existing alias, you are prompted for another name or to cancel the name change.
AlphaClearWhenCharacter	Specifies the DEF.WHEN.CLEAR property for alphanumeric attributes.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
AutomaticTab	Cursor automatically tabs to next field. ***Note:*** *This option is applicable only to the Presentation and Winform Client.*
BaseYear	Specifies the year upon which the DateConvert command bases relative day numbers.
CenturyStartYear	Specifies a secondary base year to define which century a date belongs to. This is independent of the value of the Base Year.
ConvertToUpperCase	Specifies that all input are converted into upper case characters.
Created	Read-only. The date on which the selected element was created.
DateConvertSetsGLB.CENTURY	Specifies whether Glb.Century is reset by a complex DateConvert command.
DateFormat	Specifies the format of the date throughout the segment and its element. Options are UK, International, or US.
DecimalCharacter	Specifies the character to be used as a decimal place.
DefaultFixedSettings	Specifies the default settings for background color, foreground color and font for Fixed Presentation types in Painter at the segment level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.

Property	Function
DefaultGraphicalSettings	Specifies the default settings for background color, foreground color, font, ScrollBars, ShowActmth, ShowHeader, and TransmitToCursor for Graphical Presentation types in Painter at the segment level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
DefaultPrintSettings	Specifies the default settings for background color, foreground color and font for Print Presentation types in Painter at the segment level. These settings can be changed for each child element individually in Painter. Expand each sub property to select your chosen setting. Refer to PresentationType Property under Defining User Interfaces for more information.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
EnforcePersistent	Specifies that when this property is set to true, all objects within this namespace that are persistent, or are made persistent, must inherit their definition from an item in a Dictionary. Applies to Model and Segment elements only.
EnforcePresentation	Specifies that when this property is set to true, all objects within this namespace that have their direction set to a value other then None, must inherit their definition from an item in a Dictionary. Applies to Model and Segment elements only.
FullSuppression	Display a zero numeric value as spaces.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element that refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
IsPersistent	If IsPersistent is set to True, only Primitive or Group that is a member of the Segment is “session” persistent, that is, saved as session data.
Kind	Read-only. Identifies the kind of element selected.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.

Property	Function
MultiByte ClearWhen Character	Specifies the clear-when character to be used with MultiByte strings. ***Note:*** *This property becomes visible only if Internationalization Support is set to Multibyte String Support.*
MultiByteStringValidation	Specifies the type of validation to be applied to MultiByte strings. ***Notes:*** • *This property becomes visible only if Internationalization Support is set to Multibyte String Support. When Internationalization Support is set to some other value, this property must be set to spaces.* • *To enable Debugger to emulate the MCP Kanji runtime behavior, set this property to SOK/EOK. The SOK/ EOK characters in a string primitive is represented by a dummy SOK/EOK character (0xA0) for the following:* • *Extract file field* • *Screen field* • *Watch Window item*
NationalString	Specifies the type of internationalization support for Single Byte and MultiByte Coded Character Sets. The runtime subsystem allows users to make use of different collating sequences on the same single-byte Character Code Set. The national strings in a segment could use one collating sequence when a system is generated from one configure set and a different collating sequence in a system generated from another configure set. The following options are available for this property: • None – Indicates that there is no international support. • SingleByte string support – No support for Unicode or Multibyte character sets. • MultiByte string support – Support for the Kanji Shift-JIS character set on Windows. Attributes defined with a Primitive setting of "NationalString" are implemented and stored in the database as Kanji (only) text coded in the Shift-JIS multibyte character set. Kanji text can also be stored within attributes defined with a Primitive setting of “String”. Note that there is currently no support for any multibyte character sets other than the Kanji Shift-JIS character set. • Unicode – Support for Unicode text. Attributes defined with a Primitive setting of NationalString are implemented and stored in the database as Unicode strings. Any MCP generate fails if Unicode is selected.
NumericClearWhenCharacter	Specifies the DEF.WHEN.CLEAR property for numeric attributes. This value can be set to blank or None.

Property	Function
Owner	The namespace to which the element belongs. ***Note:** When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved. An element can be reserved in two ways: Explicitly by the user or Implicitly by the model when any of the documentation windows are edited.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
SeparatorCharacter	Specifies the character to be used to delimit groups of three digits.
Stereotype	Specifies the stereotype which identifies how the class operates and is interpreted in the system.
SuppressZeros	Suppresses the leading zeros.
SynchronizedField	Specifies whether a control in a presentation has its height and width adjusted when the length of the contents are changed. Fixed Width indicates that only controls with fixed width fonts are synchronized.
VersionFile	Specifies the name of the file which contains details of the element in the Source Control Bank.

Serialization Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element. SUPER, COMPONENT, THIS and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Created	Read-only. The date on which the selected element is created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab..
Element Name	Specifies the name of the element in the XML message.

Property	Function
Inherits	Read-only. Specifies the interface from which the selected interface inherits. This property is set to the internal built-in interface Framework.ISerializable.
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Namespace	Specifies a namespace for qualifying element used in XML.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace

SQL Script Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.

Property	Function
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Teach Screen Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Created	Read-only. The date on which the selected element was created.

Property	Function
Description	Specifies a short description of the selected element. Reflected on the Properties tab.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Modified	Read-only. The date the element was last modified.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Visibility	Specifies the level of visibility for the element. Private – Only visible within its namespace. Protected – Visible within its namespace and any namespace inheriting from it. Public – Visible outside its namespace.

Variable Properties

The table below lists all properties for the particular element. Not all properties may be visible when an element is added, or an existing element is selected. Some properties only become visible when a dependent property is set to a specific value.

Property	Function
(Name)	Specifies the logical name of the selected element. The following are reserved words and cannot be used to name an element: SUPER, COMPONENT, THIS, and OWNER.
Author	Read-only. The identifier of the person who created the selected element. Reflected on the Properties tab.
Caption	Language dependent description of an element to appear in messages.
Created	Read-only. The date on which the selected element was created.
Description	Specifies a short description of the selected element. Reflected on the Properties tab.

Property	Function
Inherits	Specifies a class from which the selected class inherits. By leaving the field empty or deleting an existing entry you are specifying that the selected class has no inheritance. Inherits can also mean an "instance of" when the deriving attribute has a multiplicity >0 and does not extend the superclass. This property is reflected in, and can be changed from, the Properties tab of the selected class in the Superclass field. ***Note:*** *Inheritance from a persistent class is not supported. This restriction is not enforced by System Modeler and does not produce a validation error. However, it results in a compilation error when generated.* For the primitive attributes the following properties are inherited: 1. Primitive 2. Length 3. Decimals (for Number/Signed Number)
IsConstant	An object that is constant cannot have its value changed in logic.
IsResolved	Specifies whether the element exists in the model or whether it has been created during the import of an element, which refers to it. ***Note:*** *This property becomes visible only when an element is unresolved, that is when it is set to false. This property can only be changed from False to True. Once an element has been resolved it cannot be made unresolved again.*
Kind	Read-only. Identifies the kind of element selected.
Length	Specifies a numeric value for the length of the attribute.
MemberVisibility	Specifies the default visibility for new member elements. This can be overridden for individual elements.
Modified	Read-only. The date the element was last modified.
Multiplicity	Number of instances.
Owner	The namespace to which the element belongs. ***Note:*** *When namespace is changed, system modeler prompts for a confirmation based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*
Primitive	Specifies the kind of data that the element can contain. The following values are available for this property: Boolean, String, Signed Number, Date, Class, National string, and MultiByte string.
ReservedBy	Read-only. The identifier of the user that currently has the selected element reserved.
Security	Specifies the security level for an element for this user. The AccessControlled property of the model must be set to True to provide this option.
Stereotype	Specifies the stereotype that identifies how the class operates and is interpreted in the system.

Property	Function
Value	Specifies the initial value for the element.
VersionFile	Specifies the name of the file that contains details of the element in the Source Control Bank. ***Note:*** *When you create a new element, the System Modeler prompts for a change of VersionFile Property value based on the System Modeler Policies settings. Refer to System Modeler Policies for more information on System Modeler Policy settings.*

Element Picker

The Element Picker is used to select an element as a value for a property.

The Element Picker for the Inherits property consists of a dual pane window and a search capability to quickly retrieve a valid element. The dual pane is similar to Windows Explorer with the Navigation pane on the left and the Details pane on the right.

- Navigation pane – Displays the project node and relevant dictionaries in the left pane.
- Details pane – Displays a list of elements that can be inherited or that match the search criteria.

The Navigation pane displays the project node and only those dictionaries that contain elements that can be inherited by the selected element in the class view or the Members pane.

The Details pane displays the Name, Primitive, Length, and Stereotype property values for each element in the list. If the Inherits property contains a value, the Element Picker opens with that element highlighted. Otherwise, the project node is highlighted in the left pane and the first element in the list is highlighted in the right pane. The OK button is enabled if you select a valid element to inherit from in the left pane. Alternately, you can double-click a valid element to inherit it.

The Search feature in the Element Picker brings up a list of elements that match the specified search criteria. You can specify one or more letters of an element's name for a general search, or you can prefix the element kind before the colon in the search string. The Search feature supports elements such as class, attribute, parameter, profile, and reference. The scope of the search is always the root model.

For example,

- att:XXX, attribute:XXX

  You can use one or more letters of an element's name with the element kind as a prefix.
- Cla:, class:

  You can use a full string or a substring of an element kind as the search string.

- Eta:XXX

  If the element kind is not supported, the entire search string including the colon is considered in the search.

Model Structure Validation Rules

The validation process of a model structure includes a specified set of predefined rules for validating the model structure. There are two rules implemented for each of Windows Runtime, Debugger mode and 30 rules implemented for the MCP platform. Following list describes the rules and the specific platforms where they are applicable. You can refer to the list to fix any model structure error.

Rules	Mode/Platform	Category	Description
IDS_EXCEEDED_MAXPROFILE_LENGTH	This rule is applicable only for MCP platform.	This rule is applicable for Profiles and Configurations.	If the Name or Alternate Name property of a Profile in a Class exceeds nine characters, an error appears. To validate the model structure, you must enter maximum nine characters for either the Name or Alternate Name property of a Profile in a Class.
IDS_DB_PROF_NO_KEYS	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Profiles.	If a model does not have a key attribute for the profile, an error message appears. To validate the model structure, • You must add a key for the profile in Windows® and MCP platform • For Debug mode, you must add a key for the profile and set the debugger configuration properties to online system.

Rules	Mode/Platform	Category	Description
IDS_INVALID_AUTOPERSIST_DEPENDENCY	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Objects.	Suppose a model has an event, Event1 with Event Set of another event, Event2 and if Event1 has a persistent attribute that is not present under Event2 for establishing AutoPersist dependency, an error message appears. To validate the model structure, • You must add a persistent attribute for Event2 in Windows® and MCP platform. • For Debug mode, you must add a persistent attribute for Event2 and start debugging by setting the debugger configuration properties to online system.
IDS_NO_ISCOPIED_ATTR	This rule is applicable only for MCP platform.	This rule is applicable for Object category.	If Copied Ispecs do not have at least one attribute with IsCopied property set, an error message appears. To validate the model structure, you must define an attribute with the IsCopied property set to True or False.
IDS_DASDL_ATTR_NAME_TOO_LONG	This rule is applicable only for MCP platform.	This rule is applicable for Profile and Configuration categories.	If Name or Alternate Name property of a persistent attribute exceeds 10 characters, an error message appears. To validate the model structure, you must enter maximum 10 characters for either the Name or Alternate Name property of a persistent attribute.
IDS_INVALID_SEPARATOR_CHARACTER	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If DECIMAL.KEYED option is set to NO for a signed attribute with the SeparatorCharacter property defined, an error message appears. To validate the model structure, you must not define the SeparatorCharacter property for a signed attribute.
IDS_PERSIST_EXTRACT_FILE	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If an extract file has a persistent member, an error message appears. To validate the model structure, an extract file must not have a persistent member.

Rules	Mode/Platform	Category	Description
IDS_NUMERIC_LENGTH_EXCEEDED	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If length of a numeric attribute exceeds 19, an error message appears. To validate the model structure, you must specify a numeric value for the length of the attribute that does not exceed 19.
IDS_EXTRACT_NAME_RESERVED	This rule is applicable only for MCP platform.	This rule is applicable for Profiles and Configurations.	If an extract file has reserved words, such as MOVE for the Name property, an error message appears. To validate the model, you must change the Alternate Name property of an extract file.
IDS_AG_PERS_CLASS	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If a model has an aggregated persistent class, an error message appears. To validate the model structure, you must remove any aggregated persistent instances.
IDS_INVALID_AUTOLOOKUP	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Objects.	If a persistent class is performing autolookup on non-persistent class, an error message appears. To validate the model structure, you must ensure that invalid autolookup do not exist in a model.
IDS_PROFILE_IN_EVENT	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Objects.	If a leaf event has a profile, then an error message appears. To validate the model structure, you must ensure that • Events which has event set defined do not have profile members. • Events which don't have event set defined can have profiles.
IDS_CAPTION_LENGTH_EXCEEDED	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	A restriction is imposed on the upper pane of the Translations tab that defines all the captions for a default primary language. If length of the Caption property of a Class exceeds 16 characters, a warning message appears. To validate the model structure, you must ensure that length of caption does not exceed 16 characters.

Rules	Mode/Platform	Category	Description
IDS_NO_CLASS_PARAM	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If you add a class parameter for a public segment method, an error message appears. To validate the model structure, you must either change the visibility property of the segment to private, protected or remove the class parameter for the public segment method.
IDS_MIXED_GROUP_DIRECTION_FOUND	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If you have defined the direction of a Group and the direction of the members of the group doesn't match with the direction of the Group, an error message is displayed. To validate the model structure, you must ensure that the direction of the members of the group is same as the direction of the Group.
IDS_SAME_POF_DUPLICATE_FAMILY	This rule is applicable only for MCP platform.	This rule is applicable for Segment Configurations.	If you specify same pack names for POF Family and Duplicate POF Family, an error message appears. You can validate the model structure by changing either of the pack names.
IDS_REORDB_DB_NOT_DEFINED	This rule is applicable only for MCP platform.	This rule is applicable for Segment Configurations.	If you do not specify the configuration property, REORGDB Title of a segment, an error message appears. To validate the model structure, you must specify the name of the database copy in the segment configuration property, REORGDB DB Title.
IDS_UNDEFINED_NATIONAL_SUPPORT	This rule is applicable only for MCP platform.	This rule is applicable for Segment Configurations.	If the National Language is not specified, an error message appears. To validate the model structure, specify the National Language for the segment.
IDS_MAX_TRANSLATIONS	This rule is applicable only for MCP platform.	This rule is applicable for Folder Configurations.	If you specify more than 15 as the list of languages for Translations property of Build Target Filter, a warning appears. Hence, you should ensure to specify a maximum of 15 because only first 15 languages are included for translation.

Rules	Mode/Platform	Category	Description
IDS_RTU_SOURCE_CONFIG_NOT_SET	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If the RTU build is set to True for a deployment folder, and the RTU Source Set property is blank, an error message is displayed. To validate the model structure, ensure to set the RTU Source Set property.
IDS_RTU_Filename_Missing	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If you do not specify the RTU File name property when building RTU, an error message appears. To validate the model structure, you must enter the RTU File Name property.
IDS_RTU_SOURCE_DEFAULT_PACK_NOT _SET	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If you do not set the Default Pack property of the RTU Source set, an error message appears while building RTU. To validate the model structure, you must set the Default Pack for RTU Source Configuration.
IDS_RTU_CONFIG_DEFAULT_PACK_NOT_SET	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If the RTU build is set to True for a deployment folder, and the Default pack is not set, an error message is displayed. To validate the model structure, ensure to set the Default Pack property.
IDS_RTU_RDB_CONFIG_NOT_RDB	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If you do not set the builder configuration property, RDB Configure set to RDB type, an error message appears when building RTU. You must set the property, RDB Configure set as RDB to validate the model structure.
IDS_RTU_RDB_DEFAULT_PACK_NOT_SET	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If the RTU build is set to True, and the default pack for the RDB configure is not set, an error message is displayed. To validate the model structure, you must set the default pack for RDB Configure set to BASE configuration.

Rules	Mode/Platform	Category	Description
IDS_RTU_RDB_PORTTIMEOUTS_IDENTICAL	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	At the same time, if Primary and Secondary Port Timeouts for RDB Configure set are set to 0, an error message is displayed. To validate the model structure, ensure the Primary and Secondary Port Timeouts are not set to 0 at the same time.
IDS_RTU_ADDITIONAL_CONFIG_NOT_CFG	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If you do not set the property, Additional RTU Configure sets to CONFIGURE; an error message appears when building RTU. To validate the model structure, you must set the Type of the Additional Configuration set as CONFIGURE.
IDS_RTU_ADDITIONAL_DEFAULT_PACK_NOT_SET	This rule is applicable only for MCP platform.	This rule is applicable for RTU Configurations.	If the Additional Configuration for the BASE configuration is not set, an error is displayed. To validate the model structure, ensure to set the Default pack for the Additional Configuration Set.
IDS_FRAME_CLASS_FOUND	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If you define a Frame class in a Report without setting the Multiplicity property of Frame to 1, a warning message appears. To validate the model structure, you must set the Multiplicity property of Frame to 1.
IDS_WIDE_STRING	This rule is applicable only for MCP platform.	This rule is applicable for Objects.	If you do not enable the National Support on a segment attribute, a National String behaves as a String. So, if you define National Support as None or Unicode, National Support is not enabled and a warning message appears. To validate the model structure, you must define National Support as SingleByte string support or MultiByte string support.
IDS_INVALID_ELEMENT_NAME	This rule is applicable only for MCP platform.	This rule is applicable for class, insertable, attribute, group, diagram, dictionary, method, and folder.	If there are invalid characters detected, an error message appears. To validate the model structure, you must use the valid characters. This rule checks the Alternate Name, Alias, and Name property in order.

Rules	Mode/Platform	Category	Description
IDS_NO_VALID_PRESENTATION	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Class.	If the presentation type of a class is not contained by the presentation type of the class it inherits from, an error message appears. To validate the model structure, you must set the presentation type of the class be contained by the class it inherits from.
IDS_NOT_MATCH_PRESENTATIONTYPE	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Attribute.	If an attribute is presented but its owner's presentation type is not contained by the presentation type of the class it inherits from, an error message appears. To validate the model structure, you must set the presentation type of its owner be contained by the presentation type of the class it inherits from.
IDS_GROUP_ATTRIBUTES_OVERLAP	This rule is applicable only for MCP platform.	This rule is applicable for Attribute.	If the owner of an attribute and the group it inherits from have the same presentation type, and the group has its own group attributes that have different directions with the attribute, an error message appears. To validate the model structure, you must set all the attribute’s directions to be the same.
IDS_CANNOT_EXTEND_SEALED_CLASS	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Class.	If a class inherits from an external class, then it cannot have any member and its multiplicity cannot be zero. An error message appears. To validate the model structure, you must either remove the inheritance or ensure that the class does not have any member and its multiplicity is greater than zero.
IDS_GROUP_LENGTH_INVALID	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Group.	If a group has a length zero, an error message appears. To validate the model structure, you must add some members to the group and ensure that the length of the group is greater than zero.

Rules	Mode/Platform	Category	Description
IDS_SAME_INSERTABLE_NUM_EXCEED	This rule is applicable for MCP, Windows® platform and Debugger mode.	This rule is applicable for Objects.	If an insertable is instantiated more than once in a class, an error message appears. To validate the model structure, you must remove the other instances of the insertable and ensure that there is only one instance of the insertable in the class.

Element Reservation

In System Modeler a session is always a multi user session. To protect the integrity of the Model database there are a number of choices for ensuring that elements of the Model, or the whole model itself, can be reserved (locked) for exclusive use. Reservation prevents more than one user having access to, and changing the content of the Model at the same time.

Logic Editor is initially opened in view mode. View mode does not request a write lock for the method from the model. A lock is then requested when the logic is modified, preventing other users from obtaining write access to the same method. When the method is locked, the **Reserved By** property indicates the user holding the lock. Other users are able to obtain read-only access to the method in its form prior to the change. The method is unlocked when the logic is saved, or Logic Editor is closed.

In all types of element reservation, if a reservation fails because the element is reserved by another user, you are notified. When a reservation is successful, the Reserved By property in the Property window for the reserved element displays the name of the user who has reserved it.

This level of protection occurs irrespective of whether the additional security of Source Control is available in the project. When Source Control is available it is implied that the Model database is read-only. This means that elements can't be modified until they are checked out. System Modeler automatically checks out elements from Source Control prior to making it available for editing.

Note: *All the functions of element reservation and source control rely on the user having been granted security access to the Model and the element(s) in the Model.*

Refer to the following for more information on element reservation:

- Validating Logic
- Automatic Reservation
- Reservations Tab

Manual Reservation

In addition to the automatic reservation of elements in the Model you can manually reserve one or more individual elements. You may want to do this to make the elements available to you for editing at a future time. However, you should remember that doing so would prevent other users who may have immediate need to edit the elements, so this option should be used with discretion.

You can reserve selected elements only, or recursively. A recursive reservation locks all the child members of the selected element and the element itself.

To manually reserve an element, and optionally its children, perform the following:

1. If it is not already open, open the Class View window by clicking the View, **Class View** menu item.
2. Sort the hierarchical list into the most appropriate form to locate the element you wish to use.
3. If necessary, click the plus (+) symbol next to a node to list all the elements within the node.
4. Select the **File** menu.
5. Select **Reserve Element**.
6. Select one of the menu items listed below:
   - **Reserve** – to reserve the selected element or elements. If one or more elements are reserved to another user the operation fails, and you are notified. Other users can view the element but not change it.
   - **Unreserve** – to unreserve (clear the reservation) the selected element or elements. You can only unreserve elements that are reserved to you. The ability to unreserve another users' reservation is subject to Access Control.
   - **Reserve Recursively** – to recursively reserve the selected element, or elements, and each individual child member. If one or more child members of selected elements are reserved by other users the whole operation fails, and you are notified. Other users can view the elements but not change them.

     Note: *Reserving elements recursively does not display all the elements in the reserve list. The Reservation list is not automatically updated like other lists, that is, members list. This is done for performance reasons as maintaining this list automatically causes considerable degradation. Therefore, this behavior is as designed in AB Suite.*
   - **Unreserve Recursively** – to recursively un-reserve (clear the reservation) the selected element, or elements, and each individual child member of the selected element. You can only unreserve elements that are reserved to you. The ability to unreserve another users reservation is subject to Access Control.
   - **Use Model Exclusively** – to lock the database so that no other user can access it, even to read or browse the records. This is useful if you want to perform major maintenance on the model database, such as an upgrade.

When elements have been manually reserved, you can view the list of all reserved elements from the Reservations tab of the editor window.

Automatic Reservation

To ensure integrity of the model and to prevent two persons from modifying the same element at the same time, an element is automatically reserved (locked) by System Modeler to that user, when it is about to be modified. System Modeler determines that an element is about to be modified if the element is opened in its editor and a key press is made, or the mouse moved, when any part of the content of the editor is selected.

Automatic reservation occurs regardless of whether Source Control is available or not. If Source Control is available the element is also be checked out to this user if it has not already checked out by another user, or separately by the same user.

The point at which a reservation is removed depends on the type of edit being done, as follows:

- When the changes are saved.
- When the editor is closed for a Documentation or Logic editor, whether or not you have selected the save option.
- In the case of renaming or changing an attribute, when the change has been made persistent by saving it in the model.

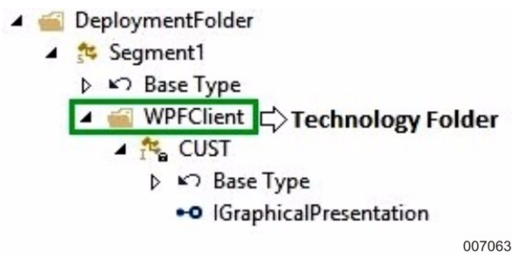
With automatic reservation, the granularity for reservation is at the element level. For example, in the case of Logic, the method is locked, and in the case of any modification made in the editor, the editor element gets locked. Also, refer to Manual Reservation.

Generating Client Framework Projects

System Modeler integrates the generation of Client Framework projects by using the concept of Technology folders and Graphical Presentation interfaces for ispecs and classes. You can create a Technology folder for a required client technology to generate the Client Framework projects and design the user interface by using the development tools such as WPF Designer, Microsoft Blend, and others for the chosen technology. One or more Technology folders can be created under a segment in the Class View window. Each Technology folder can specify either the same or different target technology type and include the set of ispecs and classes that are required for your client application.

The following example illustrates the usage of Technology folders:

The DeploymentFolder contains Segment1, and Segment1 contains a Technology folder named WPFClient. The Technology folder contains an ispec named CUST and an IGraphicalPresentation node that is exposed as a DataModel for client application development.



The DeploymentFolder is configured to deploy general application components, such as runtime system, database, and reports. The Technology folder is configured to generate Client Framework projects for user interface development.

Note: *The Technology folder output is normally generated in the background as changes are made to the AB Suite application model definitions. Typically, a build operation performed on a deployment folder does not result in any output from the Technology folder, because the output would have been generated as changes were made to an application.*

To create a Technology folder and generate Client Framework projects, perform the following:

1. Add a **Folder** to a segment, and then drag or add the required ispecs or classes into the folder. These ispecs or classes are exposed through the Access Layer API, if they contain an IGraphicalPresentation node. Refer to Adding IGraphicalPresentation for Client Framework Applications for more information on adding an IGraphicalPresentation node.

   A folder with the default name Folder1 appears under the model.

2. Rename the folder from Folder1 to an appropriate Technology folder name. For example, if you want to create a WPF application you can rename the folder to WPFClient.

3. Right-click the Technology folder and select **Properties** from the context menu.

   The **<TechnologyFolderName> Property Pages** dialog box appears.

4. Select or enter the **Client Technology** that you want to use from the list.

   You can select or enter any of the following client technologies in the Client Technology field:

   - **WPF (Windows Presentation Foundation)** – It is a user interface framework that creates rich, interactive client applications. The WPF development platform supports a broad set of application development features such as application model, resources, controls, graphics, layout, data binding, documents, and security. You can choose this option to generate projects for DataModels, DataViewModels, and Views. The Views can be designed using the

WPF Designer (or Blend) and deployed for use by the AB Suite WPF Client application, which is a WPF Container solution provided by Unisys. Alternatively, you can develop your own WPF Container application and still make use of the infrastructure provided by this technology option.

- **.Net Framework DataModels** – This option generates the corresponding DataModels project for the ispecs or classes specified in the System Modeler Technology folder. The compiled DataModels assembly can then be referenced by any .Net application by using the full .Net Framework; for example, ASP.Net Model View Controller (MVC), ASP.Net Web Forms, Web Services, and others. You can choose this option, if you want to develop a custom .Net client application by using the generated DataModels and the Access Layer API assemblies.
- **.Net Portable DataModels** – This option generates the corresponding Portable DataModels project for the ispecs or classes specified in the System Modeler Technology folder. The compiled Portable DataModels assembly can then be referenced by any .Net application that requires the .Net Portable subset; for example, Silverlight, Windows Store Apps, and others. You can choose this option, if you want to develop a custom .Net client application by using the generated Portable DataModels and the Remote Access Layer API assemblies. Note that the Remote Access Layer interface communicates through the AB Suite WCF (Windows Communication Foundation) Gateway service.
- **.Net Portable DataModels and ViewModels** – This option generates the corresponding Portable DataModels project and ViewModels project for the ispecs or classes specified in the System Modeler Technology folder. You can choose this option if you want to develop a custom .Net client application that employs the Model-View-ViewModel (MVVM) pattern, such as SilverLight and Windows Store Apps.
- **WCF Library (Windows Communication Foundation)** – This option generates a WCF Library project that can be used to deploy a WCF Service that exposes the generated DataModel definitions. A WCF Client can then call this service with a data contract based on the DataModel for an ispec.

Note: *For all the options selected above, the standard .NET DataModels is always generated, because they are required by the Access Layer Connector module (which is referenced by the Client application either directly or through the WCF Gateway).*

5. Set the **Generate Client Framework Projects** property to **True**.
6. Set the **Include ACTMTH** property to **True** or **False** as required.
7. Set the **Include Ispec Header Items** property to **True** or **False** as required.
8. Click **OK**.

Note: *The **Client Technology** field in the **<Technology Folder Name> Property Pages** dialog box is disabled after generating the Client Framework projects for a specific client technology.*

The relevant Client Framework projects are generated automatically.

For example, if the client technology is WPF/XAML, the following additional folders are generated in the Solution Explorer window.

- A Models folder comprising C# projects named, <Application Name>.<Folder Name>.DataModels and <Application Name>.< Folder Name>.DataViewModels.
- A C# project named <Application Name>.<Folder Name>.Views. This contains empty XAML Views and Data Sources that can be in the WPF Designer (or Blend), to create a user interface for an ispec.

  Note: *For a class, the XAML file is generated under the Classes folder. For an ispec, an event, a CopyIspec, and a CopyEvent, the XAML file is generated under the Stereotyped folder.*
- DataModels and DataViewModels are generated based on the Graphical Presentation interface definitions.
- DataViewModels are generated so they can be used by the WPF Client to exchange data with the WPF View in a typical MVVM pattern, which is usually employed when developing a WPF/XAML application. The Data Sources assist in designing Views in the WPF Designer. Refer to Designing User Interfaces for Client Framework Applications for more information on designing Views in the WPF Designer.

Note: *If you turn off the Auto Generate Client Framework Projects option you must generate the Client Framework projects manually by selecting* ***Generate Client Framework Projects*** *from the* ***Build*** *menu.*

Synchronizing AB Suite Client Framework Applications with System Modeler

The default synchronization mode for Client Framework projects is to automatically update the associated project files in the background when changes are made to the AB Suite Client Framework model definitions.

Note: *During synchronization, the Client Framework regenerates the Data Models for a Technology folder. For the WPF technology, it also regenerates the DataViewModels and the Data Sources. It does not update any designed Views in the existing client View projects. Any change made to the ispec definition must be manually applied to the View by using the WPF Designer.*

To verify the synchronization of Client Framework with System Modeler changes, you can perform either of the following:

- Add a new ispec or class.

  The new ispec or class is automatically generated into the Data Models, DataViewModels, and Client Views projects.
- Update an ispec by adding, renaming, or deleting one or more existing attributes.

  The ispec or class is automatically synchronized by creating new DataModel, DataViewModel, and Data Source definitions for the ispec as required, in the projects that reference the ispec or class.

- Delete an ispec or class.

  The ispec or class is automatically synchronized by deleting the ispec or class from all the DataModel and DataViewModel projects that reference the ispec or class. The Data Source definitions for this ispec are also removed, but any designed View for the ispec in the Views project is retained. The user can delete the View manually, if it is not essential.

You can control the automatic synchronization and generation of Client Framework projects by using the **Auto Generate Client Framework Projects** option.

To turn off the automatic synchronization of Client Framework projects, perform the following:

1. From the **Tools** menu, select **Options**.

   The **Options** dialog box appears.
2. Expand **SystemModeler** from the left pane, and then select **Client Framework**.
3. Clear the **Auto Generate Client Framework Projects** check box in the right pane.

If you want to propagate updates to Client Framework projects after turning off the **Auto Generate Client Framework Projects**, you must generate the Client Framework projects manually by selecting **Generate Client Framework Projects...** from the **Build** menu. This also generates any outstanding changes, as determined by Change Analysis, for a Technology folder.

Using the Client Framework Classes

The AB Suite System Modeler offers a new Framework object, Glb.ClientManager, to improve list management and dynamic attribute specification that can be integrated with the Client Framework.

The System Modeler also offers List attributes to provide dynamic list behavior. Lists can be instantiated and used in logic to hold list items, such as user-defined classes and primitive classes.

To add a List attribute, perform the following:

1. Create a definition of the items that you want to add to the list.

   To create a user-defined class, perform the following:

   a. In the Class View window, add a class to a segment or a Technology folder.

   b. Add member attributes to the class to form a structure.

   c. Set the **Visibility** property of the member attributes to **Public**.

To create a primitive class or Primitive type, perform the following:

a. In the Class View window, add a class or a Primitive to a segment or a Technology folder.

b. In the Properties window of the class, set the following properties to complete the definition:

   – Primitive – Select a primitive type, such as a String, National String, Date, Number, or Signed Number.

   – Length – Enter the maximum length (number of characters or digits) the attribute can hold.

   – Decimal – Enter the number of decimal places. This property only applies to numeric data.

2. Create an instance of the list by performing the following:

   a. Add an attribute in the ispec where you want to use the list.

   b. In the Add New Attribute dialog box set the **Template** property to the user-defined class or Primitive that you defined in Step 1.

   c. Select the **List** check box.

3. Add a method to the ispec, and then open the logic editor.

4. Use the **Add()** method in logic to add items to the list object.

   Refer to the *Agile Business Suite Programming Reference Manual* for more information on using the Add() method in logic.

The Glb.ClientManager framework class provides the ability to send a list to the client form by using the SendDynamic method. The list is exposed to the form in a DataModel as a collection, which can be easily bound to list-based controls.

Performing Backup of AB Suite Solutions

You can back up an AB Suite solution, including any selection of projects, and then restore it using the AB Suite Application from Backup option in the New Project dialog box.

To back up an AB Suite solution, perform the following:

1. Open the AB Suite solution.

2. On the **File** menu, click **Backup...**.

   The Backup Wizard appears.

3. Select the projects that you want to back up in the Backup Wizard.

   Note: *The AB Suite model projects are mandatory for every backup and are therefore selected by default.*

4. Browse to and select the location where you want to save the file, by clicking ... located next to the **Please select destination** box.

The project is saved as a .bck file.

Note: *By default, the .bck file is saved in the location where the solution file is saved.*

5. Click **Next**.

   The Confirmation page appears.
6. Click **Finish**.
7. Click **Close**.

You can now restore the AB Suite solution into an AB Suite environment. Refer to Restoring the AB Suite Solutions for more information on restoring the AB Suite solution.

Restoring the AB Suite Solutions

You can restore an AB Suite backup (.bck) file or a .model file into a new System Modeler solution and get the solution structure based on the projects that was exported previously.

To restore the AB Suite solution, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.
2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Applications**.
3. In the Templates pane, select **New Application From A File**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to restore the new projects.
6. Click **OK**.

   The New Application From A File wizard appears.
7. Browse to and select the file that you want to restore, by clicking ... located next to the **Please select file to import** box. You can select a .model file or a .bck file.

   Note: *If you select a .bck file, the restored projects appear in the Application from Export or Backup File Wizard.*
8. Click **Next**.
9. From the **Server Name** list, select the SQL server.
10. From the **Database Name** list, select the database name. By default, this field displays the model name.
11. In the **Model Name** box, enter the model name. By default, this field displays the model name.

    Note: *If there is more than one AB Suite model while restoring, you must enter the Server Name, Database Name, and Model Name for each model.*

12. Click **Next**.

    The project creation confirmation message appears in the wizard.

13. Click **Finish**.
14. Click **Close**.

On completion, the restored solution appears in the current window.

Converting the AB Suite Model

You can convert an existing AB Suite model to an AB Suite Client Framework model or an AB Suite XML Framework model by using the Converter.

Converting the AB Suite model to the AB Suite Client Framework model creates the following:

- A new Client Framework model database.
- A new Client Framework project.
- A clone of the original model database.

  The cloned model database is used to

  – Extract all the presentation information from the AB Suite model and create matching WPF or ASP.NET MVC client projects.

  – Extract all the application model structures and populate the new Client Framework model database.

To convert an existing AB Suite model to an AB Suite Client Framework model or an AB Suite XML Framework model, perform the following:

1. Create a new AB Suite project.

   Refer to Adding a Project with an AB Suite Application for more information on creating an AB Suite project.

2. Import an existing AB Suite model that you want to convert to an AB Suite Client Framework model by using Model Importer. Refer to Exporting and Importing Model Elements for more information on importing an AB Suite model.
3. Save the changes, and then close the AB Suite project.
4. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.

5. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Applications**.
6. In the Templates pane, select **Convert An Existing Model**.

   Note: *Select .NET Framework 4.6.1 or higher from the Target Framework list. You can find the Target Framework list on top of the Templates pane.*

7. In the **Name** box, enter the project name.

8. In the **Location** box, enter the path or browse to the location where you want to store the new project.
9. Click **OK**.

   The New Application Wizard appears.
10. From the **Server Name** list, select the SQL server.
11. From the **Database Name** list, select the database name of the AB Suite model that you want to convert to an AB Suite Client Framework model.

    The model name appears in the **Model Name** box.
12. Select **Include Client Framework** check box to convert the existing AB Suite model to an AB Suite Client Framework model.

    Note: *Converting the existing AB Suite model to an AB Suite Client Framework model enables the XML modelling extensions as the* ***Include XML Framework*** *check box is selected by default.*

    If you want to convert the existing AB Suite model to a XML Framework model only you must not select the **Include Client Framework** check box.
13. Click **Next** to create a new database for the Client Framework model or the AB Suite XML Framework model.

    Note: *An error appears if you try to convert an AB Suite application with an MCP configuration to a Client Framework model or an XML Framework model as these models do not support MCP.*
14. From the **Server Name** list, select the SQL server.
15. In the **Database Name** box, enter the database name. By default, this field displays the database name of the AB Suite model.
16. If you have selected the **Include Client Framework** check box, click **Next** to specify the technology to which you want to convert the AB Suite model. If you have selected the **Include XML Framework** check box only skip to step 20.
17. Click **Next** to specify the technology to which you want to convert the AB Suite model.
18. From the **Technology** list, select a technology of your choice. You can select more than one technology if you want to convert the AB Suite model to more than one technology in the Client Framework model.

    Note: *The WPF and Metadata technology is selected by default. The Metadata technology must be selected to create an ASP.NET MVC or Web API projects. The generated metadata is used in conjunction with the AB Suite Client Framework Scaffolder.*
19. In the **Folder Name** field, enter a folder name for the specified technology.
20. From the **Configuration Name** list, select a configuration for the technology folder.

21. Click **Next**.

    The project creation confirmation page appears.

22. Click **Finish**.

Note: *When the AB Suite model is converted to an AB Suite Client Framework model, a log file appears displaying the conversion details.*

The Visual Studio development environment starts and displays the project files for the newly created AB Suite Client Framework model.

You can check if the AB Suite model is converted to the AB Suite Client Framework model by verifying the following:

In the Class View window of the Client Framework project, verify if

- The Client Packages folder is created under the segment and all other elements appear within the Technology folder.
- The following properties are set for the specified Technology folder:
  - The **Client Technology** property is set to the technology specified in the New Application Wizard; for example, WPF.
  - The **Generate Client Framework Projects** property is set to **True**.

  Note: Include ACTMTH *and* ***Include Ispec Header Items*** *properties can be set to* ***True*** *depending on the settings in the original AB Suite model.*
- All elements in the AB Suite model with the PresentationType set to Graphical or Graphical & Fixed have the IGraphicalPresentation node under the elements (class or ispec) in the converted AB Suite Client Framework model.
- All members defined for an element in the AB Suite model appear under the IGraphicalPresentation node, if the **Direction** property of the members in the AB Suite model is set to a value other than **None**.

In the Solution Explorer window of the Client Framework project, verify if

- The converted elements (class or ispec) with an IGraphicalPresentation node have the corresponding project files, such as DataModels, DataViewModels, and Views generated for the WPF Client Technology folder.
- The converted screens appear similar to the forms in the AB Suite model.

  You can verify this by opening each of the converted screens from the Classes or Stereotyped folder present under the Views folder in the Solution Explorer window. For a class, the XAML file is generated under the Classes folder. For an ispec, an event, a CopyIspec, and a CopyEvent, the XAML file is generated under the Stereotyped folder.

  Notes:

  - *The converted screens generated for the Client Framework displays a warning when you access the designer as the UserControl for the screens does not include a Resource Dictionary reference to the Generic.xaml file. This reference is not intentionally added because of xaml restriction. To view* *the designer without the warning, you must add the <ResourceDictionary Source="/Themes/Generic.xaml" /> tag within the <ResourceDictionary.MergedDictionaries> tags as shown in the following code snippet:*

```
<UserControl.Resources>
   <ResourceDictionary>
         <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/Themes/Generic.xaml" />
          </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
 </UserControl.Resources>
```

- *At design time, the converted screens may not display the background colors applied to the forms in the AB Suite model. However, you can view the background colors in the converted screens at runtime.*

- The converted CopyFrom screens look different from the CopyFrom forms in the original AB Suite application.

  The CopyFrom screens generated for the WPF or the XAML technology in the converted Client Framework model may not appear exactly as they appear in the forms present in the AB Suite model. This is because a CopyFrom area is converted to a DataGrid after conversion. The copied fields are represented as columns in the DataGrid. Therefore, you generally have to modify the generated DataGrid to suit the requirements of your user interface. For example, you can delete any Copy.From columns as the DataGrid column header provides that information.

You can now run the converted Client Framework application by performing the following:

1. Configure the properties of the model and the deployment folder.

   Refer to Debugger Configuration Properties and Creating and Configuring a Deployment Folder for more information on configuring the properties of the model and the deployment folder.

2. Build the WPF Client technology projects (DataModels, DataViewModels, and Views projects) from the Solution Explorer window if you want to update the individual project files.

3. Right-click the deployment folder, and then select **Build** from the context menu that appears.

   The <TechnologyFolderName>_Config.rtxml file is generated in the location where you have saved the project:

   For example, C:\Users\<UserName>\Documents\Visual Studio 2015\Projects\<TechnologyFolderName>\Access Layer API Deploy\WpfClient_Config.rtxml

4. Double-click the <TechnologyFolderName>_Config.rtxml file, if you have associated the "rtxml" extension with the WPF Client executable. Otherwise, set up a shortcut for the WPF Client with the configuration file as a command line option. Refer to Using the WPF Client Container for Windows Platform for more information on executing the WPF Client application.

   The WPF Client Container displays the user interface by using the generated XAML Views that is converted from the forms in the original AB Suite model.

Perform a few transactions in the WPF Client Container to test if the user interfaces in the converted Client Framework model work similar to the forms in AB Suite model at runtime.

If changes are made to the AB Suite model, you can use the Synchronize Project… option in the File menu to update the AB Suite Client Framework model with the changes made to the AB Suite model.

Refer to Developing AB Suite Applications in Mixed Mode for more information on updating the converted model with the changes made to the AB Suite model.

Developing AB Suite Applications in Mixed Mode

Modifying or enhancing the AB Suite Client Framework application that is converted from the AB Suite application may take some time. Therefore, to facilitate a phased approach to carry out this conversion, you can connect a Client Framework application (for example, WPF Client) to a runtime system deployed from either an AB Suite model or a Client Framework model.

This allows you to not only continue development in the AB Suite model and the existing Client Tools interfaces, but also develop Client Framework applications that connect to the runtime system deployed from the AB Suite model. You can used the mixed mode development until all existing AB Suite forms are converted to Client Framework user interfaces.

This approach involves keeping the Client Framework model in sync with the AB Suite model by performing the following operations:

1. Convert an existing AB Suite model to an AB Suite Client Framework model. Refer to Converting the AB Suite Model for more information.
2. Build the converted Client Framework application, by selecting **Build** from the **Build** menu. Alternatively, you can right-click the model and select **Build**.
3. Run the Client Framework applications against the runtime system deployed from the original AB Suite model. Refer to Using the WPF Client Container for Windows Platform for more information on running the Client Framework application.

If any changes are made to the AB Suite model after converting it to the AB Suite Client Framework model, you can update the changes in the converted model.

Updating the Converted Model

To update the changes made to the AB Suite model after converting it to the AB Suite Client Framework model, perform the following:

1. In the Solution Explorer window, select the project that you want to update.
2. From the **File** menu, select **Synchronize Project…**.

   The Synchronize Application Wizard appears, displaying the confirmation message about the updates being made to the converted Client Framework model.

3. Click **Finish**.

A log file appears displaying the details about the updates made to the Client Framework model.

Note: *If you modify the forms in the AB Suite model, you must manually modify the screens in the converted Client Framework model. However, the log file provides information about the forms that are modified in the AB Suite model.*

If changes are made to the AB Suite model, you can use the **Synchronize Project…** option in the **File** menu to update the AB Suite Client Framework model with the changes made to the AB Suite model. Refer to Converting the AB Suite Model for more information on updating the converted model with the changes made to the AB Suite model.

In mixed mode development, the Client Framework model is never deployed. It is only used to develop client applications by using the Client Framework interfaces. Therefore, you cannot take advantage of certain new Client Framework features in System Modeler (for example, List classes). The new features can only be used when the Client Framework model is deployed. This can be done when all the existing user interfaces are converted to technologies that use the Client Framework.

You can now run the AB Suite application and the converted Client Framework application side by side in two different instances of Visual Studio to check if the user interfaces in the Client Framework application work similar to the forms in the AB Suite application.

Processing XML Messages

AB Suite allows you to receive and process XML messages in AB Suite Windows Runtime. XML message processing allows an AB Suite runtime system to act as a message broker for XML messages. In XML message processing

- A client program submits XML messages through an API.
- Each message is parsed and validated.
- Runtime objects are populated from the message and user-defined method logic is executed.
- A response that indicates the status of the completed message processing is returned to the client program. This response can optionally include an output XML message which can then be returned to the system that was the original source of the input XML message.
- The processing of each message is handled as a single transaction.

Receiving and processing data typically includes the following:

- Defining the format of messages through a special type of interface definition known as “Serialization” or “Serializable Interface”.
- Defining classes that can implement the interfaces. These classes are referred to as Messenger classes and Serializable classes. These classes contain the implementation definition for processing the XML messages.

Serialization

A serialization is a construct that can be used to define the format of an XML message. This follows the object oriented concept of interfaces in which they form a contract between a client and the implementing class and define the format of data that is passed in and out of the class.

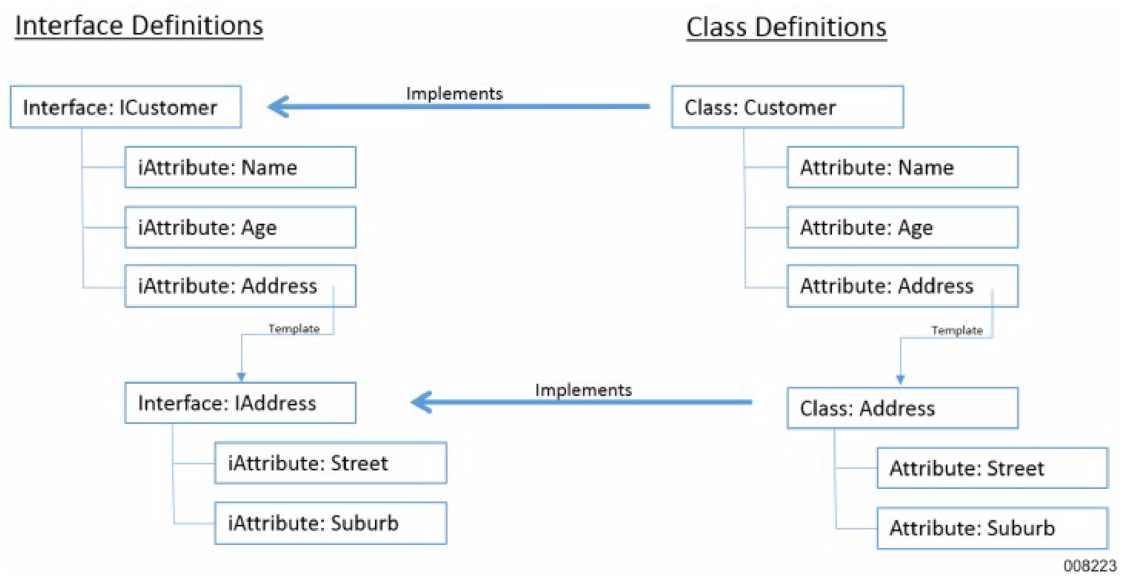
Serialization is an interface construct that is used to allow message structures, such as XML to be modelled in Agile Business Suite. The term Serialization refers to the fact that the data can be serialized to and from message formats like XML. You can use Serialization to define any structured message format as it is not tied to any specific message format.

Note: *In AB Suite 6.1 Serialization to and from XML messages is supported, and other message formats, such as JSON may be supported in the future.*

The content and structure of an XML message can be defined through one or more Serializations. The Serialization comprises attributes that define individual primitive elements and other Interfaces that define complex sub-elements.

The AB Suite model can contain a definition of the message structure as a Serialization, and can also contain a corresponding class that implements the Serialization. At runtime, the XML message is de-serialized according to the Serialization definition and the XML element data is populated into the attributes of an instance of the corresponding class. This AB Suite instance can then be processed internally through certain business rules implemented as LDL+ logic.

The following figure illustrates how the message can be modelled, with the layout of the message modelled through the Serialization interface definitions. It also illustrates how the behavior and processing can be implemented in the corresponding class definitions, which are said to 'implement' the interfaces. For example, the classes can contain LDL+ logic within their methods to store the XML data in database tables.



There are two types of classes that can be used to implement Serialization and process message data

- Messenger class
- Serializable class

A Messenger class is a stereotyped class with Stereotype="Messenger". It includes built-in behaviors allowing it to process incoming messages.

A Serializable class is an ordinary class (not Stereotype) that is used to implement the handling of all or part of a message by implementing a Serialization.

Messenger Class

A Messenger class is used to process input data that is submitted to the system in the form of a message. Such messages are submitted through a programmatic API that processes the data through a processing cycle. Note that currently the API supports processing of messages that are in XML format.

The Messenger class contains a built-in cycle for processing input messages. This involves reading and parsing the input message, populating an instance of the Messenger class with the data from the message, and then executing user-defined business rules written as LDL+ logic in built-in methods called Receive() and Respond(). The Messenger class cycle can optionally return a response message.

In addition to the built-in methods you can create user-defined methods. However, there is one important difference with the user-defined methods in a Messenger class compared to all the methods in other types of classes, related to the way in which the return type is defined. In standard methods, throughout the rest of the AB Suite model, the return type of a method is defined by creating a "Return Variable" which is simply a variable with the same name as the method. Methods within a Messenger class do not have a "Return Variable". Instead, you define the return type directly through the ReturnType property of the method. This is actually defining a "type" rather than an instance. Therefore, you can return any instance of the specified type from the method logic.

Data Processing in a Messenger Class

Data processing in the Messenger class primarily involves defining the appropriate definitions and logic to receive XML input data and process it - typically to store it in a database or retrieve data from the database, and to respond with output XML data. For example, an input XML document may have customer details, such as name and address that can be processed through a class named "Customer". Based on the defined definitions and logic, the customer details can be stored in a database table named CUST. The necessary logic can be defined to assign attributes from the Messenger class instance to an instance of CUST.

At runtime, a client or a protocol adapter calls the Process() method to submit input messages to the system. This method accepts two string arguments: a string containing the XML message and a string with the name of the receiving class; for example "Customer".

The XML cycle then passes the received message to an instance of the Messenger class; for example, "Customer". This instance is then populated with the data from the input message. The processing cycle then executes the defined logic to process the input data; for example to store data or retrieve data from the database.

For a Messenger class to receive an XML input message and process data through an XML cycle, it must be defined with the following:

Stereotype = "Messenger"

Defining an XML Structure

You can define a simple XML structure or a complex XML structure by adding a Messenger class and a corresponding Serialization. You can create an XML structure either manually or by importing an XML Schema Definition (XSD) file.

Defining a Simple XML Structure Manually

To define a simple XML structure you must add a Messenger class in the model, define Serialization with the necessary attributes and implement the interface into the Messenger Class for processing the XML data.

To manually define a simple XML structure, perform the following:

1. Right-click the segment, point to **Add**, and then select **Messenger** from the context menu.

   The Messenger class with the default name “Messenger1” appears under the segment.

2. Rename the Messenger class from “Messenger1” to an appropriate name.
3. Right-click the Messenger class, point to **Add**, and then select **Serialization**.

   The Serialization with the default name “I<Messenger name>” appears under the segment.

4. Rename the Serialization from “IMessenger1” to an appropriate name.

   You can now define the layout of the XML message by adding attributes to the Serialization.

5. Right-click the Serialization, point to **Add**, and then select **Attribute** from the context menu.

   The **Add New Attribute** dialog box appears.

6. In the **Template** box, optionally enter a type or class that this attribute will derive from.
7. In the **Name** box, enter an appropriate name.
8. Click **Create**.

   You can add more attributes to the Serialization by setting the **Keep Open** checkbox and repeating step 6 through step 8.

9. In the Properties pane of the attributes change the property of each attribute as required.

You must now add the implements relationship to indicate that the Messenger class will implement the Serialization.

To create the Implements relationship, perform the following:

1. Right-click the Messenger class, select Quick Actions…, and then select **Implement an Interface…**.

   The **Select Interface for <Messenger Name>** dialog box appears.

2. Expand the list and select the Serialization for which you want to create the Implements relationship.
3. Click **OK**.

At this stage the definition is still incomplete because the class does not yet have an implementation of all of the attributes in the interface. Therefore, the Errors pane displays a list of errors indicating that the Messenger class does not implement the interface members.

To resolve these errors you must Implement the interface members by performing the following:

- Right-click the Messenger class, select **Quick Actions...**, and then select **Implement Interface Members...**. This automatically adds attributes to the Messenger class that correspond to each of the attributes within the interface.
- Alternatively, you can add attributes to the class manually with corresponding names and types.

Note: *It is important to understand that it is the interface definitions and not the class definitions that define the messages and their layout.*

You can preview the message layout for any class that implements a Serialization. The Message Layout Preview window shows a representation of the corresponding message layout.

To preview the message layout, perform the following:

- Right-click a Messenger class or other class that implements the Serialization, and select **Preview Message Layout**.

  The Message Layout Preview window appears displaying a representation of the layout of the corresponding messenger attributes.

Defining a Complex XML Structure Manually

To define a complex XML structure manually, perform the following:

1. Add a Messenger class to a segment and add a Serialization. See "Defining a Simple XML Structure Manually" for more information on adding a Messenger class and Serialization.
2. Add another Serialization anywhere within the model structure by right-clicking on the element and selecting **Serialization**.

   For example, if you want to represent the following XML structure, you must add one Serialization to the Messenger class to represent the complete <Customer> message (for example, ICustomer) and another Serialization (for example, IAddress) anywhere within the model structure to represent the <Address> element:

   ```
   <?xml version=ö1.0ö?>
   <Customer>
       <Name>John Smith</Name>
       <Age>54</Age>
       <Address>
           <Street>115 Main St</Street>
           <Suburb>Mornington</Suburb>
       </Address>
   </Customer>
   ```
3. Add attributes to the Serialization to define the layout of the XML message.

   For example, you must add attributes, such as Name, Age, and Address to the ICustomer interface and attributes, such as Street and Suburb to the IAddress interface.

See “Defining a Simple XML Structure Manually” for details on adding attributes to the interface.

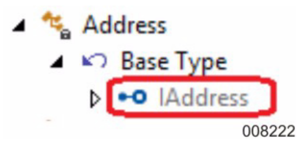
4. Define the Address attribute under the ICustomer interface as an instance of the IAddress interface by performing the following:

   a. Open the Properties of the Address attribute in ICustomer interface.

   b. Set the Template property to IAddress.

   The ICustomer interface represents the complete XML message format.

   After adding the Implements relationship, the IAddress interface is shown in the Class View window under the Base Type node as shown in the following image. In this case the Address class implements the ‘Address’ interface. See “Defining a Simple XML Structure Manually” for details on adding an implements relationship.



Generating an XML Schema Definition (XSD) File

You can generate an XSD file from any Messenger or Serializable class definition.

To generate an XSD file, perform the following:

1. Select a Messenger class or a Serializable class.
2. From the **File** menu, select **Export As Xsd...**.

   The Xsd Export Wizard appears.
3. In the **File Name** box, enter the location of the XSD file name or browse to and select the location where you want to export the class definition.
4. Click **Finish**.

   An XSD representing the selected class is written to that file.

This process also creates an automatic mapping between the AB Suite primitive types and the corresponding standard XML types in an XSD.

For example, any attribute with the Primitive property set to String is mapped to xsd:string or any attribute with the Primitive property set to Number is mapped to an appropriate type depending on the length and decimals (for example, xsd:unsignedLong or xsd:decimal).

You can override this automatic mapping and specify a particular XML type. For example, you may use a String in AB Suite to hold an element that is of a type not fully supported by a corresponding AB Suite primitive type, such as xsd:dateTime. Hence, even if the attribute in AB Suite that holds this data is a String, you may want the exported XSD definition to represent it with the correct XML type (that is, xsd:dateTime).

To override the automatic mapping, perform the following:

- Select the attribute within the Serialization (Serializable interface definition) and set the XsdType property of the attribute to an appropriate XML type.

To set the XsdType property, perform the following:

1. In the Properties window of the attribute, click the Element Picker **...** corresponding to the XsdType property.

   The XML Type Definition window appears.

2. Select an appropriate XML type and click **OK**. Alternately, you can double-click the XML type.

This property does not influence any of the internal processing of the element when processing the XML messages. It is only used in the Export As Xsd operation.

Importing XSD Files

You can create a class in your model that defines an XML document using the XSD Import Wizard. The input file for this process must be an XSD file that defines the schema for an XML document. After completion of this process your model will contain a class or classes that define the structure of the XML document.

To create a class in your model that defines an XML document using the XSD Import Wizard, perform the following:

1. In the Class View, right-click the segment, point to **Add**, and then select **Add New Item...**.

   The **Add New Item** dialog box appears.

2. From the templates available in the Add New Item dialog box, select **XSD Import**, and then click **Create**.

   The Xsd Import Wizard appears.

3. Enter the location of the XSD file or browse to the location of the XSD file that you want import by clicking ... .

4. Click **Load**.

   The Xsd Structure section displays the element name in the selected XSD file.

5. Click **Next**.

   The Confirmation page appears in the wizard.

6. Click **Finish** to complete importing the XML definition.

   A new folder, MessageDefinitions, is created at the folder level under the segment. The XML definition is imported to a folder, Messages, under the MessageDefinitions folder.

Defining Lists in an XML Framework Model

Lists can be defined within Serializations and Serializable classes to represent repeating elements in an XML message format.

A List can be defined in either of the following ways:

- As a special kind of attribute or variable within classes and interfaces. That is, you can define a List attribute by selecting the List check box in the Add New Attribute dialog box when adding a new attribute or variable.
- As a List type. This is a re-usable list definition that can be used in ordinary attributes to define a list through the Template property. You can do this by selecting List in the Add New Item dialog box.

To understand the usage of List we can use the following example. This is a simple XML message with a repeating element:

```
<?xml version="1.0"?>
<People>
  <Surname>Smith</ Surname >
  < Surname >Jones</ Surname >
  < Surname >Harris</ Surname >
</People>
```

To define a List Attribute to represent the repeating <Name> element, perform the following:

a. Right-click the Serialization that represents the “People” message format, point to **Add**, and then select **Attribute** from the context menu.

   The **Add New Attribute** dialog box appears.

b. In the **Template** box, enter or browse to the type that this attribute will derive from. For example, the Primitive type for Surname. This can be any Primitive type, Class, or Serialization.

c. In the **Name** box, enter an appropriate name; for example, Surname.

d. Select the **List** check box.

e. Click **Create**.

A corresponding List attribute has to be defined within the class that implements the above Serialization.

Implementing Multiple Interfaces

You can also implement multiple interfaces for a Serializable class or a Messenger class when you want to have a single class implementation and have more than one variation of an XML message returned from it.

To implement multiple interfaces for a Serializable class or a Messenger class, perform the following:

1. In the Class View window, add a Serializable class or a Messenger class; for example, CustomerInfo.
2. Add Serializations (Serializable interfaces) to define the XML messages; for example, ICustomerInfo and ICustInfoSummary.
3. Implement the interfaces (once for each interface) for the Serializable class or the Messenger class.

   Note: *You can also implement interfaces present anywhere within the model.*
4. When writing logic to return an XML message from a Respond() method you can specify the interface you want to use to serialize the class to XML by using the AsA command option in the Return logic command.

   For example:

   ```
   If FullDetails
       Return aCustInfo AsA ICustomerInfo
   End

   Return aCustInfo AsA ICustInfoSummary
   ```

If you have a Messenger class that implements multiple interfaces you have to specify the interface to be used for de-serializing an input XML message.

To specify the interface to be used for de-serializing an input XML message, perform the following:

1. In the Class View windowLimits on LINC+ Logic commands and System Attributes are described, expand the Messenger class after creating the Implements relationship.
2. Select the Serialization (serializable interface) that is to be used to de-serialize input XML messages, and then invoke the Properties window.
3. Set the **Use for Cycle** property under the **Misc** group to **True**.

   The Use for Cycle property also specifies the interface to be used to serialize a method parameter that is being passed in an external method call.

Sending XML to External Systems

You can not only process an XML input message that originates outside the AB Suite system and respond with an XML reply message but also process the XML messages in the reverse direction. That is, you can initiate the sending of an XML message to an external system from within the AB Suite system and receive an XML message in reply. You can do this by calling an external library passing an XML message as a parameter and receiving an XML reply message as the return value from that call.

To achieve this, you must add methods in an external library that takes a string parameter expected to contain an XML message. In the AB Suite model you can define the parameter to be a Serializable class. In your LDL logic you can populate an instance of this class and pass it as an argument in the external method call. The runtime infrastructure automatically serializes the argument into an XML message and the external library receives it as a string.

Similarly, you can add a method in an external class that returns a string value containing an XML message as the return value of the method (or as the value of an InOut or Out parameter). The ReturnType of the method in the AB Suite model can be defined to be a Serializable class. In this case the XML message in the return value is automatically de-serialized to populate an instance of this class.

Refer to *HowTo Process XML in AB Suite 6.1* in the Documentation Libraries page on the Product Support site for an example on calls made to an external library that send and receive XML messages.

Introducing the Runtime Cycle

Transaction Processing

Transaction processing is implemented via the Agile Business Suite segment cycle. This functionality occurs since all segment methods implicitly process transactions, and the segment cycle processes ispecs as transactions.

The transaction processing cycle also determines the context in which logic commands operate. When a logic command is executed, its operating context is resolved to the stereotype of the initial class activated by the segment cycle (for ispecs and events) or the called report (for reports).

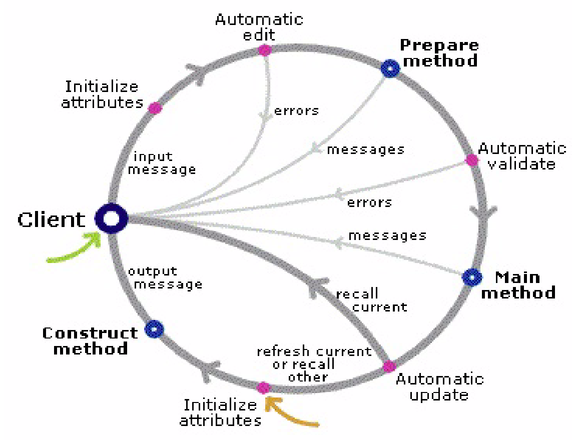
Other processing occurs within the context of the segment cycle (including the ispec cycle), such as copy cycle processing (transaction processing of copy ispecs and events), SQL script processing (of SQL scripts), and automatic entry processing.

Segment Cycle

The segment cycle is the processing cycle that occurs when the application executes an ispec or event transaction. It is controlled by the segment and defines the order in which built-in methods are called.

Note: *In the following text of this topic, the term ispec should be read to denote both ispecs and events.*

The following diagram is of the basic segment cycle:



Requesting an Ispec

An ispec can be requested as a result of one the following:

- A request from the Select Ispec dialog box.
- A Recall logic command invoked by another ispec.
- A Recall logic command invoked by the same ispec.
- An Abort logic command.
- A Roc logic command.
- A request/incoming message from an external caller via the segment's public (COM) interface.
- An automatic refresh of the ispec.

When an ispec is requested, the following process steps occur before the ispec is ready to accept input (for ispecs with a user interface, this corresponds with its display to the application client). The orange arrow in the diagram above indicates the starting point:

1. Segment and ispec attributes without defined initial values are initialized to their corresponding values from the input message.
2. The Construct method is called unless either the ispec is being requested due to an automatic refresh and a Message logic command has not been invoked, or the ispec has not been requested as a result of a Recall logic command invoked by the same ispec.

   The Construct method can be used for reasons such as the pre-filling of user interface fields, or security checking.

Transmitting an Ispec Update

An ispec update initiates the following process steps (the green arrow in the diagram above indicates the starting point):

1. Segment and ispec attributes without defined initial values are initialized to their corresponding values from the input message if they are in the presentation, or to the appropriate “empty” value (0, “”, or false), depending on the attribute type.
2. Automatic edit occurs – attributes with decimals are validated.

   Any errors are returned to the application client.
3. The Prepare method is called.

   The Prepare method can be used for reasons such as generating a customer number, performing any necessary validation of user input data, performing logic actions based on the user input, or recalling another ispec without processing the current ispec.

   Any Message or Recall logic commands invoked halts processing at the end of the prepare method.
4. Automatic validation occurs – keys, dates, and required fields are validated; the database records corresponding to the specified keys are retrieved if they have an automatic lookup dependency.

   Any errors are returned to the application client.
5. The Main method is called.

   The Main method can be used for reasons such as checking stock-on-hand, or checking a customer credit limit for a sale.

   Any errors are returned to the application client.
6. Automatic update occurs – for a persistent ispec, the database record is updated (or written).
7. At this point, one of the following process steps occurs:

   - If a Recall logic command was invoked on the same ispec, the Construct method call is skipped, and the segment cycle repeats from step 1 above.
   - If a Recall logic command was invoked on a different ispec, the specified ispec is requested. Refer to Requesting an Ispec for more information.
   - If the ispec's **Refresh Screen** property is set to true, and no Recall logic command was invoked, the current ispec is requested. Refer to Requesting an Ispec for more information.
   - If the ispec's **Refresh Screen** property is set to false, and no Recall logic command was invoked, the **Select Ispec** dialog box is displayed. Refer to Requesting an Ispec for more information.

Transmitting an Ispec Inquiry

An ispec inquiry occurs when an ispec is transmitted with its Maint built-in presentation attribute is set to FIR, LAS, NEX, BAC, or REC. It initiates the following process steps:

1. Segment and ispec attributes without defined initial values are initialized to their corresponding values from the input message.
2. Automatic edit of keys occurs – numeric fields are validated, separators and decimal points are removed.

   Any errors are returned to the application client.
3. The database record corresponding to the specified keys is retrieved.
4. The retrieved record is made available (for ispecs with a user interface, this corresponds with its display to the application client)

Glb.Error

The behavior of the segment cycle is affected by the Glb.Error built-in segment attribute in the following manner:

- Automatic processing (edit, validation, or update) does not occur if Glb.Error is set to "*****". If Glb.Error is set to "*****" upon completion of the Main method call, the automatic update of the database update does not take place.
- Glb.Error is initialized to spaces prior to both the Prepare and Construct method calls. Consequently, automatic processing in the Construct method occurs even if Glb.Error has previously been set to "*****" in the Prepare or Main methods.
- A Message or Recall logic command invoked in the Prepare or Main methods sets Glb.Error to "*****" and cause the Construct method call to be skipped. However, explicitly setting Glb.Error to "*****" does not cause the Construct method call to be skipped if the ispec's **Refresh Screen** property is set to true.

  If Glb.Error is set to Glb.Spaces following the invocation of a Message or Recall logic command, database updates, including the automatic update following the Main method call, occurs, and construct automatic processing is performed if the ispec's **Refresh Screen** property is set to true.
- If database updates in the Prepare or Main methods are completed successfully, the message "SUCCESSFUL ENTRY" is displayed on the status line. However, if database updates are disabled because Glb.Error is set to "*****", the message "THIS ENTRY DID NOT UPDATE" is displayed instead, regardless of any successful database update that might subsequently occur in the ensuing Construct method. This situation results because the status line message is output after the Main method call, at which point the application does not know whether the Construct method attempts to (successfully or otherwise) update the database.

Ispec Cycle

The ispec cycle is a subset of the segment cycle and consists of the processing of an input message by a single ispec. It is controlled by each individual ispec.

The ispec cycle can also be called independently of the segment cycle, such as with external automatic entry processing.

Copy Cycle

The copy cycle is the processing cycle that occurs when the application executes an copy ispec or copy event transaction. It is controlled by the segment and defines the order in which built-in methods are called.

Note: *In the following text of this topic, the term "ispec" should be read to denote both ispecs and events.*

The copy cycle is similar to the segment cycle, except that it consists of three phases – the validate, update, and refresh phases; and each phase is performed multiple times for each copy ispec.

Requesting a Copy Ispec

When a copy ispec is requested, the following process occurs before it is ready to accept input (this corresponds with its display to the application client). The orange arrows in the diagram above indicates the request phase:

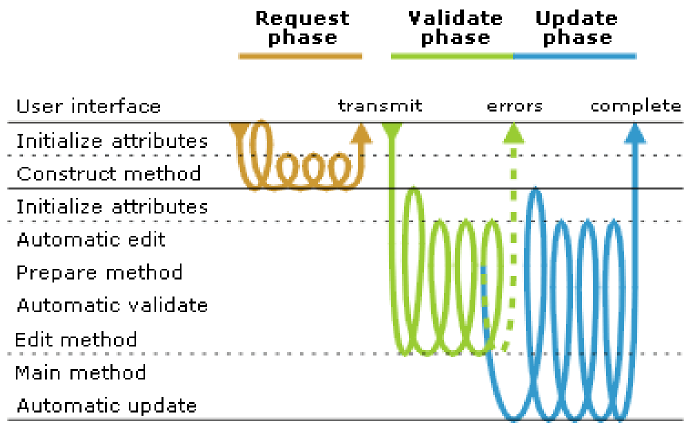
1. Segment and copy ispec attributes without defined initial values are initialized to their corresponding values from the input message. This occurs for the first copy of the copy ispec only.
2. The construct method is called once for each copy of the copy ispec.
3. The graphical presentation is displayed.

A copy ispec can be requested similarly to an ispec. Refer to Requesting an Ispec in Segment Cycle for more information.

Transmitting a Copy Ispec

Upon transmission of a copy ispec, two passes through the logic of the copy ispec are performed - the first pass is the validate phase, and the second pass is the update phase.

The update phase is only performed if there are no errors in the validate phase.



Validate Phase

The validate phase consists of the following process steps (indicated in the diagram above by the green arrows):

1. Segment and copy ispec attributes without defined initial values are initialized to their corresponding values from the input message. This occurs for the first copy of the copy ispec only.
2. The following loop is performed once for each copy of the copy ispec. Any errors that occur in one copy never affect the processing of other copies:
   - Non-copied attributes (those without graphical presentations, or with their corresponding graphical object's **Is Copied** presentation property set to false) are set to their corresponding values from the input message.
   - Automatic edit occurs. If an error occurs, the remainder of the loop is not performed for the current copy.
   - The prepare method is called. If an error occurs, the remainder of the loop is not performed for the current copy.
   - Automatic validation occurs. If an error occurs, the remainder of the loop is not performed for the current copy.
   - The edit method is called. If an error occurs, the remainder of the loop is not performed for the current copy.
3. At this point, one of the following process steps occurs:
   - If any errors were detected in any copy, they are returned to the application client. All error messages are stored until all copies have been processed.
   - If a Recall logic command was invoked in any copy, the specified ispec is requested. Only the last invoked Recall logic command is executed (after all copies have been processed).
   - If no errors occurred, no Recall logic commands were invoked, and the Glb.Error built-in segment attribute is set to Glb.Spaces, processing proceeds to the update phase.

Update Phase

The update phase effectively repeats the validate phase, except that it also calls the main method and performs automatic update for each copy. It consists of the following process steps (indicated in the diagram above by the blue arrows):

1. Segment and copy ispec attributes without defined initial values are initialized to their corresponding values from the input message. This occurs for the first copy of the copy ispec only.
2. The following loop is performed once for each copy of the copy ispec.
    - Non-copied attributes (those without graphical presentations, or with their corresponding graphical object's **Is Copied** presentation property of set to false) are set to their corresponding values from the input message.
    - Automatic edit occurs again
    - The prepare method is called again.
    - Automatic validation occurs again
    - The edit method is called again
    - The main method is called.
    - If the Glb.Error built-in segment attribute is set to Glb.Spaces, automatic update occurs.
3. At this point, one of the following process steps occurs:
    - An ispec is requested.
    - The current copy ispec is refreshed.
    - The Select Ispec dialog box is displayed.

An EndExit logic command terminates processing for the current method only (prepare, edit, or main) for the current copy. It has no effect on the processing of subsequent copies.

Messenger Cycle

The Messenger cycle is the XML processing cycle that consists of the processing of an input message within a single database transaction. The input in a messenger cycle comes from an XML message and the output includes a response XML message. The Messenger cycle also includes framework methods that are used to include user-defined logic. In a Messenger cycle steps, such as initialization, validation, parsing message, and serialization are automatic.

Defining the Processing Logic

A special type of serializable class is used to define the processing logic to be executed for an input XML message. This class is called a Messenger class and is defined with Stereotype=Messenger. The Messenger class defines the format of an XML message and the processing logic to handle the message when it is input to the system. The Messenger class is an entry point for processing an input XML message. An XML message is passed to an instance of the corresponding Messenger class for processing.

In a Messenger cycle a Messenger class is defined to implement the Serialization that defines the structure and layout of the input XML message. Then, the class implements an automatic processing cycle to receive an XML message and execute user-defined processing logic.

Basic Steps in a Messenger Class

The basic steps in a messenger class are as follows:

1. Parsing the input message and populating an instance of the Messenger class with the data from the message.
2. Validating the input data from the message.
3. Executing user-defined logic.
4. Returning a response XML message (optional).

Including User-Defined Logic

Built-in framework methods are available in which user-written logic can be included to perform any user-defined business rules or processing including reading from and writing to the database.

The framework methods are

- Receive() – Intended for any logic to process the input message. This method can include validating input data, storing data in the database, reading data from the database, and so on.
- Respond() – Intended for any logic to formulate a response message. This method typically includes logic to read data from the database, populate a serializable class instance (that is, a class that implements the Serialization), and then return that instance. A returned serializable instance is automatically serialized into the corresponding XML message and included in the response.

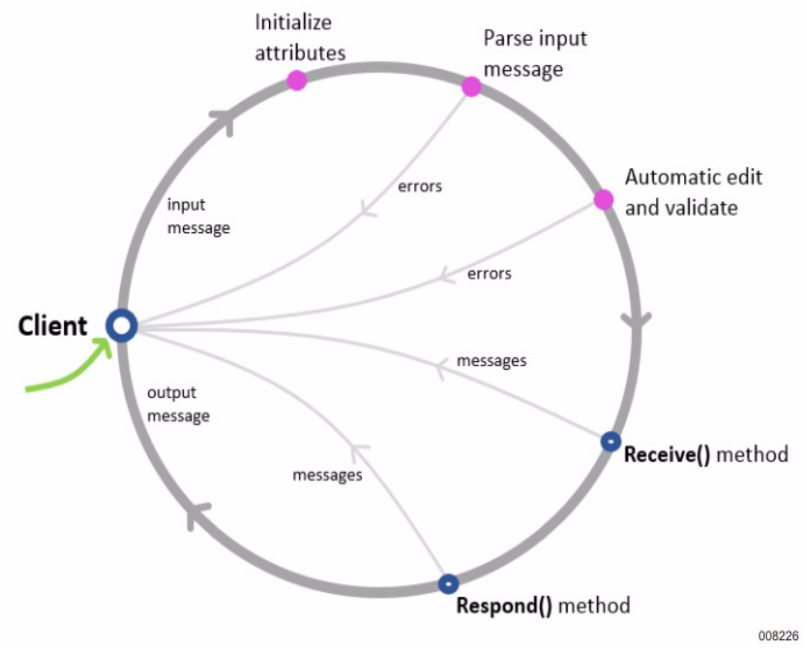
  The Respond() method is unique, that is, it is defined with a ReturnType of 'String'. The String return value contains the XML message stream. To return the XML message you must use the Return command to return a serializable class instance that is cast to the Serialization through the 'AsA' command option. The returned instance is implicitly converted to a string value by serializing the instance according to the interface definition.

  The syntax of the Return command when used in this context is

  ```
  Return <instance> AsA <interface>
  ```

  Where, <*instance*> is the serializable class instance and <*interface*> is the Serialization that is implemented by the class.

The following image shows the Messenger cycle:

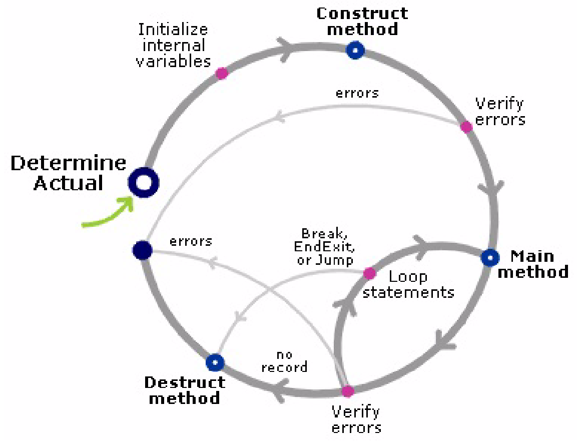


The Messenger Cycle is similar to the Ispec cycle, except that the input comes from an XML message instead of a client form and the output can include a response XML message.

SQL Script Processing

SQL script processing occurs when the application executes the SQL script variant of the Determine Actual logic command. It is controlled by each SQL script and defines the order in which the built-in SQL script methods are called.

The following diagram is of the basic SQL script process:



This process occurs upon invocation of a Determine Actual logic command specifying the SQL script:

1. Any internal variables, such as error result codes, are initialized.
2. The SQL statements in the SQL script's construct method are executed.
3. The SQLCODE return code is verified, and sets the JumpTo built-in segment attribute.

   If errors are detected, rollback occurs and the Determine Actual logic command loop is exited.

   If no record is retrieved, the Glb.Status built-in segment attribute is set to "*****" and logic control passes to the end of the Determine Actual logic command loop to execute the SQL script's destruct method (Refer to step 7).
4. The SQL statements in the SQL script's main method are executed.
5. The SQLCODE return code is verified, and sets the Glb.MainSQLCode built-in segment attribute.

   If errors are detected, rollback occurs and the Determine Actual logic command loop is exited.

   If no record is retrieved and it is also the first iteration through the Determine Actual logic command loop, the Glb.Status built-in segment attribute is set to "*****" and logic control passes to the end of the loop to execute the SQL script's destruct method (Refer to step 7), otherwise the Glb.Status built-in segment attribute is set to spaces.
6. Logic command statements within the Determine Actual logic command loop are executed.

   If the Determine Actual logic command loop is exited using a Break or EndExit logic command, control passes to the end of the Determine Actual logic command loop to execute the SQL script's destruct method. If exited using a Break All logic command variant, control passes to the end of the outermost loop. If exited using an EndExit logic command, control passes to the end of the current method.

   If the Determine Actual logic command loop is exited using a JumpTo logic command, control passes to the end of the Determine Actual logic command loop to execute the SQL script's destruct method before transferring control to the Label destination of the JumpTo logic command.

   If the Determine Actual logic command loop is not exited using a Break or JumpTo logic command, execution returns to step 4 above.
7. The SQL statements in the SQL script's destruct method are executed.

   The SQLCODE return code from the destruct method sets the Glb.PostSQLCode built-in segment attribute.

Examples

- Selective Data Retrieval with Join Example
- Selective Update Example
- Database Issues

Refer to Determine Actual and SQL Script Variant for more infomration on SQL script usage.

Database Issues

Database Integrity

SQL scripts operate within the context of the existing Agile Business Suite transaction structure. A commit would destroy the atomicity of the transaction and potentially leave the database in an indeterminate state. For ispecs, this means that any command which would cause an implicit or explicit commit of database changes is specifically not allowed, and results in a validation error when the component is built. For reports, commands that force an implicit or explicit commit can be used; as such a capability is already available with the Sleep logic command.

Resilience and Recovery

Use of SQL scripts exposes the Agile Business Suite database to uncontrolled updates. It is possible to perform updates that do not take into consideration the internal semantics that Agile Business Suite imposes on a database table. Due care should be taken to ensure that the database is not corrupted or internal semantics invalidated.

OLEDB

Upon completion of the construct built-in method, the Glb.PreSQLCode built-in segment attribute is set to the Database return SQLCode value.

Upon completion of the main built-in method, the Glb.MainSQLCode built-in segment attribute is set to the Database return SQLCode value.

Upon completion of the destruct built-in method, the Glb.PostSQLCode built-in segment attribute is set to the Database return SQLCode value.

Upon completion of any non-built-in SQL script method, Glb.MainSQLCode is set to the Database return SQLCode value.

If there are multiple SQL statements in any method, only the last value of SQLCode is set.

The following example checks SQLCODE values:

```
Determine Actual MySQL
DoWhen Glb.PreSQLCode <> 0
Message "Construct method caused a warning"
: We don't want to process this for each iteration, so clear the condition
Glb.PreSQLCode := 0
End
```

```
DoWhen Glb.MainSQLCode <> 0
Message "Main method caused a warning"
: Clear the condition for the next iteration
Glb.MainSQLCode := 0
End
: Logic
End     : End of Determine Actual MySQL
: Check for any warnings from the destruct method
DoWhen Glb.PostSQLCode <> 0
Message "Destruct method caused a warning"
: Clear the condition
Glb.PostSQLCode := 0
End
```

Selective Data Retrieval with Join Example

This example uses the SQL script “LargePolicies” to iterate over data retrieved concurrently from two database tables by means of a join operation.

Construct

The construct method consists of the following SQL statements:

```
DECLARE Cursor1 CURSOR FOR
SELECTPolicy.Acct Cust.Name Cust.Address Policy.Value
FROMPolicy, Cust
WHEREPolicy.Acct = Cust.Acct AND Policy.Value > :MaxValue ;
OPEN Cursor1 ;
```

This declares a cursor “Cursor1” over the set of data represented by the SELECT and WHERE clauses. It then opens the cursor. As an example, this would retrieve the name and address of every customer with a policy value greater than $1,000,000.

Main

The main method consists of the following SQL statement:

```
FETCH Cursor1 INTO :PolicyAcct :CustName :CustAddress :PolicyValue ;
```

This fetches the next record into the instance variables defined by the DECLARE … CURSOR FOR statement in the construct method.

Destruct

The destruct method consists of the following SQL statement:

```
CLOSE Cursor1 ;
```

This closes the cursor Cursor1.

Attributes

The following attributes should be defined for the SQL script, for use as instance variables:

- PolicyAcct – string-primitive, length 15.
- CustName – string-primitive, length 30.

- CustAddress – string-primitive, length 50.
- PolicyValue – number-primitive with 2 decimal places, length 12.
- MaxValue – number primitive, length 10.

Invocation

The SQL script is invoked from a report as follows:

```
MaxValue := 1000000
Determine Actual LargePolicies
PolicyTotal := PolicyTotal + PolicyValue
F10_Acct := PolicyAccount
F10_Name := CustName
F10_Address := CustAddress
Frame10.Print()
End
```

Upon invocation, processing occurs as described in SQL Script Processing.

Selective Update Example

This example uses the SQL script “PremiumUpdate” to update all records from a single table that meet the selection criteria.

Construct

The construct method consists of the following SQL statements:

```
UPDATE Policy SET Policy.Premium = Policy.Premium * :PremiumIncr
WHERE Policy.StartDate > 19851103 ;
```

This is a singleton SQL statement that increases the policy premium by 11.5% (specified by PremiumIncr) for all policies started after November 3, 1985.

Main

The main method is not defined or empty.

Destruct

The destruct method is not defined or empty.

Attributes

The following attribute should be defined for the SQL script, for use as an instance variable:

- PremiumIncr – number-primitive with 3 decimal places, length 4.

Invocation

The SQL script should only be executed once. Consequently, an unconditional Break logic command should be invoked from within the Determine Actual logic command loop:

```
PremiumIncr := 1.115
Determine Actual PremiumUpdate
: Logic
Break
End
```

Upon invocation, processing occurs as described in SQL Script Processing .

Automatic Entries

Automatic entries are used to pass ispec information between deployed applications. This information passing can occur between applications on the same or on different hosts. Transactions that initiate inter-application communication are fully recovered in the event of a system failure (except where the two-phase commit process abandons a transaction).

Internal Automatic Entries

Internal automatic entries occur between applications using the same database (or within the same application).

The automatic entry sends information from the originating ispec or report to the database via the database record buffer of the target ispec.

External Automatic Entries

External automatic entries occur between applications using different databases. The target application must have an ispec identical to that specified by the originating application, sharing the same attributes, and in the same order.

External automatic entries can be used to interface to an external system, using the NOF or USER interface methods. Refer to the *Agile Business Suite Runtime for Windows® Operating System Administration Guide* for your host type for more information.

Automatic Entry Processing

The target ispec must be an automatic entry ispec (its **Auto Entry Capable** property is set to true).

Note: *An external-only ispec (its* ***External Only*** *property is set to true) is inherently an automatic entry ispec.*

The data object used to pass information to the target application is specified as an instance of the target ispec as a non-persistent ispec attribute of the originating segment.

To send an automatic entry, perform the following:

In logic:

1. Call the target ispec's Initialize built-in method to clear it.
2. Set the members of the target ispec to the required values.

   For members of the target ispec that are not given a value by an explicit assignment statement:

   - For new records, any members omitted are set to zero (for number-primitive variables) or a space character.
   - For existing records being modified, any members omitted retain their existing values. However, if the member's **Clear When** property is set to zero (for number-primitive variables) or a space character, the value is set to zero or a space character, as appropriate.
3. Specify the target application by setting one or more of the following built-in segment attributes:

   - Glb.Destination to the destination database; it is set by default to Glb.Self.
   - Glb.DestHost to the destination host; it is set by default to Glb.SelfHost.
   - Glb.DestEnv to the destination environment; it is set by default to Glb.SelfEnv.

   To specify an internal automatic entry, set the Glb.Destination built-in segment attribute to Glb.Spaces or Glb.Self and the Glb.DestHost built-in segment attribute to Glb.Spaces or Glb.SelfHost.
4. Call the target ispec's Store built-in method to send the automatic entry. Refer to Store for more information.

   If the target ispec has at least one attribute set as a key, the Maint built-in attribute value should be passed as a parameter to Store.

Internal Automatic Entry Processing

For internal automatic entries, no transaction processing of the target ispec occurs. The database is updated immediately with the information.

External Automatic Entry Processing

For external automatic entries:

1. After the automatic entry is sent, the originating application is suspended while waiting for a response from the automatic entry.
2. The target application receives the automatic entry in the input buffer and processes the information as normal client input. All phases of the segment cycle , except for the Construct method, are performed for the target ispec.

   The Glb.Origin, Glb.OriginHost, and Glb.OriginEnv built-in segment attributes can be used to identify the source of the transaction.

Whilst the target ispec is processed, a recall of a different ispec (using the Recall logic command) is ignored. A recall of the same ispec causes a reply to be returned to the originating application. Otherwise, a simple acknowledgment is returned, indicating that processing is complete. If the Glb.Error built-in segment attribute is set to "*****", only an error acknowledgement is returned upon completion.

3. The originating application receives the response and the reply can be determined from the values of the members of the targeted ispec.

   The Glb.Status and Glb.HubStatus built-in segment attributes can be used to identify the result of processing the message. Refer to Glb.Status and Glb.HubStatus Settings for more information on the values returned to Glb.Status and Glb.HubStatus.

   The Auto Write&Clear command only clears the automatic entry buffer if no data is returned to the buffer, in which case it clears the buffer regardless of the Glb.Error built-in segment attribute value. However, if data is returned, the Auto Write&Clear logic command behaves identically to the Auto Write command.

Failure Behavior

An automatic entry is rejected if:

- The Maint built-in attribute value is passed to the Store method as ADD, and the record already exists in the target database.
- The Maint built-in attribute value is passed to the Store method as CHG or DEL, and the record does not exist in the target database.

Refer to Glb.Status and Glb.HubStatus Settings for more information on failure behaviour.

Data Truncation

If a mixed string and wide string value is input to a wide string-primitive variable, the value is moved character-by-character, and truncated to the length of the variable.

If a mixed string and wide string value is input to a string-primitive variable, the value is moved byte-by-byte, and truncated to the length of the variable. This truncation may occur after the first byte of a double-byte character, resulting in an invalid character. However, truncation does not occur when moving these values into a wide string array variable.

Automatic Entry Restrictions

The following restrictions also apply to the use of automatic entries:

- They are not valid in Edit methods of copy ispecs.
- Signed number-primitive variables are corrupted if passed in an external automatic entry to an application deployed to a platform with a different character set, that is, between EBCDIC (applications deployed to MCP) and ASCII (applications deployed to UNIX, *Windows®* , or OS 2200).

- Only the values ADD, CHG, and DEL can be passed as parameters to the Store method. The Glb.Status built-in segment attribute is set to "*****" when the automatic entry is sent if any other value, such as PUR, is passed.

  A signed number-primitive value cannot be transferred between different host platforms. To transfer a signed value between applications deployed to different host platforms, specify a separate variable for the sign, and transfer the absolute value.
- A string value cannot be transferred to an unsigned number-primitive variable with decimals (or vice versa).
- A string value cannot be transferred to a signed number-primitive variable (or vice versa).
- A number value cannot be transferred to a wide string-primitive variable (or vice versa).
- Number array variables are implemented as COBOL COMPUTATIONAL fields. Moving a number array to anything other than an identically defined number array may result in data that is invalid for use in either computation or display.

Numeric Arrays

Numeric arrays are stored in COMP data format, which is different from how string arrays are stored (as string values).

To calculate the required size for the receiving attribute to store an incoming numeric array:

1. Allow one byte for every two digits.
2. Add half a byte for each array element, rounded up to the nearest byte.

   For example, an array with ten elements, each three digits in length, would be calculated as:

   ```
   ((10 * 3) / 2 + (0.5 * 10)) := 20
   ```

Graphical Interface Workbench

If a Graphical Interface Workbench interface is used to view the output from copy ispecs that contain external automatic entries, the automatic entry buffer that holds the information from the external automatic entry is also used for the video output items for the copy ispec for the Graphical Interface Workbench interface. This can result in corrupted data on the output screen.

To avoid this, when coding copy ispecs to use automatic entries for Graphical Interface Workbench-type interfaces:

1. Code any Initialize method calls before setting up any display fields on the screen. The BEGIN.EDIT logic command can be used to help do this. Refer to Segment Cycle for Copy.From Ispecs in the *Enterprise Application Host Builder Guide* and BEGIN.EDIT and End.EDIT later in this section for more information.

2. Store the fields in the automatic entry buffer immediately into SDs or GSDs, by using the Move.ARRAY; command. This prevents the fields from being initialized on each copy cycle, being overwritten by other external Automatic Entries, or being overwritten by setting up display fields on the screen.
3. Reference the SD or GSD, rather than using the Move.ARRAY; command, in the Ispec user logic.

Automatic Entry Example

This example in the logic of a report calculates each customer's bank balance and makes internal automatic entries updating the interest amount. It also makes an external automatic entry to the database of a different application if the customer is unemployed.

```
LookUp Every Cust                  : Iterate through every customer
 Determine Total Deposits (Cust.Customer) Amount : Determine the total amount
                         :  of the customer's deposits
  Glb.Total := Glb.Total * Cust.Interest     : Calculate the interest
  SD_Interest := "Interest"            : Give a description
  Deposit.Initialize()
  Deposit.SetMaint("CHG")
  Deposit.Customer := Cust.Customer
  Deposit.Amount := Glb.Total
  Deposit.Narration := SD_Interest
  Deposit.Store()                  : Store in Deposit database table
  Deposit.Initialize()              : Clear the Deposit buffer values

  : Logic

  Journal.Initialize()
    Journal.SetMaint("ADD")
    Journal.Ledger := SD_Interest
    Journal.Amount := Glb.Total
    Journal.Amount.Contra()
  Journal.Store()                  : Store in Journal database table
  Journal.Initialize()              : Clear the Journal buffer values
  DoWhen Cust.EType = "N"             : For unemployed customers
    Glb.Destination := "EmployDB"        : The interest amount is sent to
    Deposit.Initialize()           :  the Employ application
    Deposit.SetMaint("ADD")
    Deposit.External(true)           :  (running against EmplyDB)
    Deposit.Customer := Cust.Customer
    Deposit.Amount := Glb.Total
    Deposit.Narration := SD_Interest
    Deposit.Store()
    Deposit.Initialize()
    DoWhen Glb.Status = "NOEXT"         : Check external automatic entry
      Message Error "External application not identified"
    EndExit
    Glb.Destination := Glb.Self         : Reset Glb.Destination
  End
End
```

Runtime Limits

This section summarizes the limits that are currently in effect with Agile Business Suite.

- Ispec Class Limits
- Ispec Class and Report Ordering
- Reports Class Limits

- Profiles Limits
- Ispec Attribute Limits
- Segment Attribute Limits
- Methods Limits
- Database Limits

Limits on LINC+ Logic commands and System Attributes are described in the *LINC+ Programming Reference Manual* and *Logic Online Help*.

Conventions

Limits are summarized in tables. Additional information may be included after each table.

The phrase No limit means that there is no direct limit, although system software and size considerations (and the associated increase in edit and generate times) may have an impact.

The phrase System means that the limit is imposed by system software, such as the COBOL compiler.

Ispec Class Limits

The following table contains a summary of the limits for an Ispec Class for each target host. It is intended as a guide only and should be read in conjunction with the information in the subsections following the table:

Limit	Windows®	MCP
Number of Ispec Classes	No Limit	1500 (1000 for Output or I/O)
Number of Attributes in Output Structure	1024	3020
Length of Attributes in Output Structure	29 characters (For Numeric) 2^{31}-1 characters (For Text)	24570 characters
Byte length of Input/ Output Ispec classes	4000 bytes (65000 for Big Buffer Ispecs)	26,450 characters
Number of Logic Lines in an Ispec class	2097151 Number of lines in a Class	999999
Number of Nested Database Commands	32	No limit

Limit	Windows®	MCP
References to other Ispec classes	65535 (within a method) 2097151 (within a class)	6009
References to Methods		2000 (includes references to other Ispec classes)
Painted Items in an Ispec class	9990	320
Painted Items in a Copy.From area	9990	100
Maximum size of External Automatic Entry (HUB) transaction	9999 characters	2000 bytes

Number of Attributes in the Output Structure

In Windows-based systems, there is a limit of 950.

In MCP-based Systems, the maximum number of Usage Input-Output and Usage Output items is 3,020 (and is limited by the accumulated size).

The number of fields reserved for the different types of generated fields are listed in the following table:

Generate	Subtract
GLB_DTIME (Ispec)	1
GLB_DTIME, MAINT (Std component)	3
ACTMTH, GLB_DTIME, GLB-REPORT, INPUT-DATE, ISPEC, XTRANNO	6
Descending key fields	1 for each

Notes:

- *Component arrays often result in a long Attribute.*
- *In SQL Server, this limitation of allowing only a single field within a table to exceed 4000 characters is not present. However, this limit is enforced at the validation phase of Generate for all database types to ensure that specifications developed in Agile Business Suite are consistent for all deployed database types.*

Size of Ispec Class or Profile Record in MCP-based Systems

Usage Input-Output and Usage Output Attributes are limited to an accumulated size of 4,065 words (24,570 characters), which is the DMS II record size limit.

Size of Ispec Class or Profile Record in Windows-based Systems

An Attribute Item of type long may not be used as a Key or Profile Key.

For SQL Server, the size of an index must be smaller than 900 bytes.

Byte Length Limit for Windows-based Systems

For Input/Output Ispecs that are to be generated to a Windows® operating system, there is a 4,000 byte data limit. The combined maximum size of user defined fields is 3,893 bytes. The remaining 107 bytes are made up of the following system-defined fields:

- ACTMTH
- LB.SOURCE
- INPUT-DATE
- MAINT
- TRANNO
- 1 byte for the @-sign field
- 80 bytes for the status line

This limit is due to the internal buffer RDATA-AREA having a maximum length of 4,000 bytes. RDATA-AREA stores the Input/Output buffer for the Ispec. Anything over the 4,000 byte limit causes memory to be overwritten.

The limit is checked during the Model Validation phase of the Builder generate process. If the maximum length is exceeded an error is given and the generate is stopped.

Note: *This limitation does not apply to Big Buffer Ispecs classes. For Input/Output Big Buffer Ispec classes, the maximum byte length is 65000 bytes for a Windows® oerating system.*

Database Commands

In Windows-based systems, there is no limit to the number of Database commands.

In MCP-based systems, there is a limit of 999 references to other Ispec classes (all Events count as one reference) from an Ispec class.

Painted Items in an Ispec Class

For *Windows®* and MCP-based systems the limit to the number of painted items in an Ispec Class, or a Copy.From area is as follows:

Limit of 320. Maximum of 100 items in the Copy.From area of the screen layout for a Copy.From Ispec classes.

Note: *This limitation does not apply to Big Buffer Ispec classes. The maximum number of painted items in a Big Buffer Ispec class is 1000.*

Internal Database Storage Order

In MCP-based systems, Ispec Attributes are stored in the database in the following order: alphanumeric items (in alphabetical order), numeric items (in alphabetical order), and a filler item if defined.

In Windows-based systems, Attributes used as Keys are defined first (in alphabetical order), followed by all other Attributes (in alphabetical order).

Ispec Class and Report Ordering

MCP is an EBCDIC based platform, the Windows environment is ASCII based. Builder converts ASCII to EBCDIC during the generate file transfer process.

Because of this difference, Ispec and Report Classes appear differently in lists in Developer and MCP Runtime, when their names contain numeric characters. MCP Runtime displays the name containing the numeric after those with no numeric characters. Developer displays the name containing the numeric before those with no numeric characters. For example, MCP displays Ispec classes in the following order, ABCD, ABC2, while Developer displays them in the opposite order, ABC2, ABCD.

You must also be aware that the sort order is based on the collating sequence of the native character set for the host platform on which you are running the application. This affects the order in which items are stored and retrieved from the database.

You should make appropriate allowances for this if your application is to run on both Windows® and MCP operating systems.

Reports Class Limits

The following table contains a summary of the limits for Reports for each target host:

Limit	Windows®	MCP
Reports in a System	No limit	No limit
Logic Lines for a Frame class	999,999	999,999
Nested Database Commands	No limit	No limit
References to Ispec classes (all Events count as 1 reference) from a Report class	999	600
Segment Method logics in a single Frame class		2000 (includes references to ispec classes)
Segment Methods in a report		2000 (includes references to ispec classes)
Methods in a single Frame class	900	900
Methods in a Report	900	900
Display items in one Frame class	204	204
Display and Attributes in one Frame class	450	450
Global Setup Attributes in a Report class	64K bytes	999x1 Mb bytes
Setup Attributes in a Frame class	64K bytes	1 Mb
Size of generated Report code	System	System

Limit	Windows®	MCP
Files Used in a Report class	System	System
Extract files in a Report class	26	600 (only first 26 CP recoverable)
Physical size of Extract files created within a Report class	No Limit	
Size of Extract file (Extracted Frame, 80-byte records)	No Limit	Expected number field to 99,999,999,999
Size of Extract file (Ispec class)	No Limit	Expected number field to 99,999,999,999
Attached Reports running at one time for a System		500
Standalone Reports running at one time for a System	System	System
Sorting	No Limit	Sort memory 999K words, Sort disk 999,999K words
Size of RIP file card equates		5K bytes

Number of Logic Lines for a Report

999,999 lines for each Frame class, and 999,999 lines for Main logic.

Setup Attributes in a Report Frame in MCP-based Systems

A separate limit of 64K bytes applies to Attributes with initial values and Attributes without initial values.

Database Commands

There is no limit for Database Commands in the Windows® operating system.

In MCP-based systems, there is a limit of 999 references to other Ispec classes (all Events count as one reference) from an Ispec class.

Physical Size of Extract Files Created Within a Report on MCP-based Systems

If the Extract is a Frame, a limit of 99,999,999,999 records applies (set on the Extract File Options screen).

If not an Extract Frame, the file size is taken from the value of the Expected Number field for the Ispec class (limit of 99,999,999,999 records set on the Extract File Options screen).

Profiles Limits

The following table contains a summary of the limits for Profiles for each target host.

Limit	Windows®	MCP
Number of Profiles	99	No Limit
Number of Keys in a Profile	16	20
Length of Profile Keys	240 bytes	No Limit
Number of Lines in a Profile Definition	99	99
Alphanumeric Key Length	78	78
Usage Output Alphanumeric Key	4095	4095
Unsigned Numeric Key	12	12
Signed Numeric Key	11	11

Number of Profiles

The number of Profiles over any Ispec class is limited to 99 for all hosts.

Length of Profile Keys

On MCP-based systems, there is no limit, although DASDL may impose an upper limit.

On Windows-based systems, the sum of the lengths of all Profile Keys is limited to half the database block size. An Attribute of type long may not be used as a Key. For SQL Server, the size of an index must be smaller than 900 bytes.

Number of Lines in a Profile Definition

For Windows® and MCP-based systems the limit on the number of lines in a Profile definition is as follows:

There is a limit of 99 lines when defining your Profile (that is, lines for COMP.NAME; or EVENT.NAME; commands, KEY; commands, DO.WHEN; conditions and for MCP-based Systems, DATA; commands). You should use this limit at your discretion as your system may experience memory problems.

Ispec Attribute Limits

The following table contains a summary of the limits for Setup Attributes for the different target hosts.

Limits	Windows®	MCP
Number of Ispec Attributes	2097151	No limit
Size of an alphanumeric Ispec Attribute	$2^{31} - 1$ characters	9,999

Limits	Windows®	MCP
Size of an Ispec Attribute with alphanumeric literal	2^{31} - 1 characters	160
Size of a numeric Ispec Attribute	29	18
Size of an Ispec Attribute with numeric literal	29	18
Total size of an Ispec Attribute array (characters)	Size of public Array variable of any datatype	1MB
Size of all Ispec Attributes with initial values	29 characters (for numeric) 2^{31} - 1 characters (for text)	No limit
Size of all Ispec Attributes without initial values	29 characters (for numeric) 2^{31} - 1 characters (for text)	See Text
Ispec Attributes in an Ispec class	2097151	No Limit
Ispec Attributes in Report	2097151	No Limit
Ispec Attributes in a Method	2097151	See Text

Size of Ispec Attributes on MCP-based Systems

Ispec Attributes without initial values have a maximum limit of 99 Ispec classes with greater than 64K byes of Ispec Attributes or the equivalent size in one Ispec class.

Ispec Attributes in a Method

If you intend to access a Method using the Component Enabler interface, you can only use 5000 Ispec Attributes to define the parameters of the Method. Also, if a parameter is longer than 5000 bytes, only the first 5000 bytes are transferred.

These limits only apply if you use the Component Enabler interface. They are not enforced by Developer.

Segment Attribute Limits

The following table contains a summary of the limits for Global Setup Attributes.

Limit	Windows®	MCP
Number of Folders	999	999
Number of Segment Attributes	2097151	No Limit
Size of an alphanumeric Segment Attribute	2^{31} - 1	9,999
Size of a Segment Attribute with alphanumeric literal	2^{31} - 1	160

Limit	Windows®	MCP
Size of a numeric Segment Attribute	29	18
Size of a Segment Attribute with numeric literal	29	18
Total size of a Segment Attribute array (characters)	65,000	1MB
Size of all Segment Attributes with initial values	29 characters (for numeric) 2^{31}-1 characters. (for text)	No Limit
Size of all Segment Attributes without initial values	29 characters (for numeric) 2^{31}-1 characters. (for text)	No Limit
Segment Attributes in an Ispec class	2097151	No Limit
Segment Attributes in Report	2097151	No Limit

Size of Segment Attributes on MCP-based Systems

Total size of all Segment Attributes without initial values has a maximum of 6,336K bytes.

Methods Limits

The following table contains a summary of the limits for Methods for each target host.

Limit	Windows®	MCP
Number of Methods	2097151	No limit
Logic Lines in a Method	2097151	9,999
Methods in an Ispec class	2097151	No limit
Nested Methods	2097151	No limit
Macro Methods in Module Master	No Limit	No limit
Macro Methods inserts in Module	No Limit	3,000
Size of an alphanumeric Attribute in a Method	2^{31}-1	4,095

Methods in an Ispec Class

MCP-based Systems

There is a limit of 9,000 on the total number of inserts of a Method into another Method.

Windows-based Systems

For *Methods*, there is a limit of 500 on the number of inserts of a Global Logic into an Ispec class (including nested inserts).

Nested Methods

The upper limit for nested Methods is the limit on the number of Methods in an Ispec class.

Length

By default, the value is 1.

The following table contains limits for Attributes.

Type of Attribute	Maximum Length
Alphanumeric	2^{31}-1
Usage Inquiry, Alphanumeric	2^{31}-1
Usage Output, Alphanumeric	2^{31}-1
Ispec class, Numeric, with NO.DECIMAL.KEYED	29
Ispec class, Numeric, with DECIMAL.KEYED	29
Date	By default, exactly 6 (can be specified as 8 to include the century)

Database Limits

Limits of Persistent Agile Business Suite Constructs

The Database Reorganization module of AB Suite translates the persistent constructs of the model into database structures. The naming conventions for the database structures and the corresponding limits is listed in the table below:

Model Construct	DB Construct	DB Naming Convention	Length_Limit
Database Schema (Segment property)	Database Schema	Database Schema	8 characters
Persistent Ispec	Table	SegmentName_IspecName	30 characters
Persistent Attribute	Column	ColumnName	30 characters
Profile	Index	SegmentName_ProfileName	30 characters
Conditional Profile	Materialised View	SegmentName_MaterialisedView Name	30 characters
SQL Scripts	Stored Procedures	Stored Proc Name=CONSTRUCT, PREPARE, MAIN	30 characters

Defining User Interfaces

The Agile Business Suite Developer is used to design and construct user interfaces, for you to interact with deployed applications. It allows you to add simple graphical objects, such as lines and images, and graphical objects, such as edit boxes and radio buttons, which are bound to attributes as their data source.

When a graphical object is bound to an attribute, it can read and display the data in the attribute and modify the contents if applicable.

Note: *In Teach Screens, graphical objects cannot have associated attributes.*

You can use the following two modes to design the forms:

- **Graphical Mode** to design GUI screens for application ispecs and methods. These screens can display the greatest range of visual objects, such as check boxes, list boxes, images, and so on.
- **Fixed Mode** (Green Screen Painter) to design character screens for different runtime platforms. These screens use fixed type fonts and cannot display complex graphics.

You can toggle between the two modes in AB Suite applications by selecting the required mode from the list above the form. You can design the screens by using the toolboxes available for each mode.

Note: *In AB Suite Client Framework applications, you cannot toggle between the Graphical and Fixed modes. When a* ***Presentation Interface*** *is added to a class or an ispec, the* ***PresentationType*** *property of the class or the ispec is set to Graphical and is read-only.*

User interfaces for Client Framework applications are designed using the standard tools for the chosen technology (for example, WPF Designer for WPF Client applications). A Client Framework model does not allow the System Modeler Painter to be used for designing the user interface for Client Framework applications. Refer to Designing User Interfaces for Client Framework Applications for more information.

Any named element can have a form if its **PresentationType** property is set to a value other than None, with the exception of a model or folder.

The first step in creating an interface for AB Suite applications is to add a form to the element. You can then add graphical objects to the form so that a user can interact with the underlying elements of the application.

Differences between Graphical and Fixed Screens

The Fixed mode screens (green screens) are similar to the Graphical screens; however different types of items are handled differently in the Fixed mode and Graphics mode. The main differences are summarized in the following table.

Property	Graphics Screen	Fixed Screen
Fonts	Supports multiple fonts and point sizes.	Supports fixed type fonts only (the fonts used are not proportional to the width of the characters). Any font change applies to all items in a form and is not specific to an item in the form.
Controls	Supports the use of all graphical controls in the System Modeler (SM) Graphical Painter Toolbox.	Supports use of only the following controls in the SM Fixed Painter Toolbox: Pointer, Labels, TextField, and PasswordField.
Overlapping of fields	Supports overlap of characters or fields without restriction.	Prevents overlapping of characters or fields. The character or field that is positioned on top of another character or field is truncated. In addition, if the available space where a field is placed is less than the defined length of the attribute then the field is truncated.
Grids	Displays the default grid size.	Displays a different grid size. The size of the grid depends on the fixed font type and size.
Foreground and background colors	Supports custom colors.	Limits the use of colors to the following: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta.

PresentationType Property

A presentation format is selected by specifying the PresentationType property. The PresentationType property comprises the following options:

- Graphical
- Fixed
- Graphical and Fixed
- Print
- None

Graphical

The Graphical format allows the addition of graphical objects listed in the toolbox to a form for creating AB Suite applications. This format is most suitable for GUI screens that can display the greatest range of visual objects, such as check boxes, list boxes, and images. For Client Framework applications, the PresentationType property is set to Graphical and is read-only. Furthermore, you cannot use System Modeler Painter for developing user interfaces for Client Framework applications. The development of user interface for Client Framework applications is achieved by using external tools for the chosen technology.

Fixed

The Fixed format is most suitable for character-based screens that are unable to display complex graphics. This format limits the graphical objects to Labels, TextFields, and PasswordFields. Note that there is a limited range of available colors and fonts in a Fixed presentation.

Graphical and Fixed

The Graphical and Fixed format allows you to create either a graphical or fixed form, or both. It allows you to have different versions of the same form, which can be accessed from either a graphical or character-based screen.

Print

In the Print format, the addition of graphical objects to the form is limited to only Labels. This format is ideal for printing report outputs.

None

Generally, this is the default value for the PresentationType property and is selected for elements that do not require a form. The exception to this is CopyEvents and CopyIspecs, which have a default value of Graphical but can be changed to Fixed and Graphical & Fixed.

Refer to Common Properties for more information on setting the Font, Foreground color, and Background color for each of these presentation types of AB Suite applications.

Adding a Form

You can add a form to an element by selecting the element from the Class View.

To add a form to an element from the Class View, perform the following:

1. Open the Class View window and select the element to which you want to add a form.

   Note: *If you wish to select an element lower in the hierarchy of the element node that is displayed in the Class View, press + (plus key) on the numeric keypad to expand the element and to display its members.*

2. Open the Properties window of the selected element, and set the **PresentationType** property to any value other than None.

   This adds the **Painter** tab to the document window. Refer to PresentationType Property for a description of the available presentation types.

The **Painter** tab is available at the bottom of the document window. Click this tab to open the newly created form on which you can add graphical objects. Before you add objects, you may want to display the form grid to have your objects snap to the grid for more accurate positioning. You may also want to set the properties of the form, which can be done before or after you add the objects.

Setting the Properties of a Form

The visual characteristics that you specify for a form are inherited by all graphical objects that you add to the form, when they are appropriate for an object type. You can specify different visual attributes for a particular object or group of objects after the objects have been added to the form. If you wish to return the changed visual attribute of an object to its original inherited value, select the object and the property that you wish to reset, click Reset from the context menu.

To set the properties of a form in the Properties window, perform the following:

1. Click anywhere on the form to select the form.
2. On the **View** menu, right-click **Properties** window to open the Properties window.
3. Specify the properties as listed in the following table.

Property	Function
Name	Specifies the name of the element that contains the form. This property is read-only.
GridSize	Specifies the size in pixels of the horizontal and vertical guidelines to help you align objects on a form. The grid size may differ between Graphical and Fixed screens. The grid size depends on the fixed type font and font size for a Fixed screen.
ShowGrid	Specifies whether the form displays grid lines. By default, the grid is turned on.
SnapToGrid	Specifies whether the objects on the form get locked to the nearest grid lines. By default, the value is True.
FormLayout	Specifies the display format of form objects. By default, the Grid option is selected. The following two options are available: • Grid – Specifies absolute positioning behavior of form objects. • Flow – Specifies a logical flow layout of form objects in a top-left, bottom-right arrangement.

Property	Function
AutoTabbing	Specifies if the cursor automatically moves to the next field on the form after data is entered in the current field. Options include True and False. ***Note:*** *This option is applicable only for the Presentation and Winforms clients.*
BackgroundColor	Specifies a background color for the form. Select a color from one of the tabs in the drop-down list. The colors for a Fixed screen are limited to the following: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta.
BackgroundImage	Specifies an image to be displayed as the form background.
Font	Specifies font attributes that are applied directly to the text appearing on the form. The fonts for a Fixed screen are limited to fixed type fonts only. Any font change applies to all items in a form.
ForegroundColor	Specifies a foreground color for the form. Select a color from one of the tabs in the drop-down list. The colors for a Fixed screen are limited to the following: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta.
ScrollBars	Specifies if a form displays scroll bars when the content of a form exceeds a screen display size. The following options are available: • No • Horizontal • Vertical • Both
ShowActmth	Specifies whether the accounting month field appears. Options are True (default) and False.
ShowHeader	Specifies whether the header fields of a form appear. Options are True (default) and False.
ShowToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, you have to hover the mouse on the control and press the F1 key. This tooltip is only visible at Runtime.
Size	Specifies the size of the form representing the height and width in pixels. By default, the value is 80 by 24 pixels. You can adjust the size, if needed.
TransmitToCursor	Specifies that the form data up to the position of the cursor (in the order of tabbed fields) is transmitted to the host. By default, the value is False, and it sends all the form data to the host at transmit time.

Using the Form Grid

The Form Grid provides horizontal and vertical guidelines designed to help you align objects on your form, or a Panel that was added from the Toolbox.

ShowGrid – Specifies whether form display the sizing grid. By default, the grid is turned On. The grid size may differ between Graphical and Fixed screens. For Fixed screens, the grid size depends on the fixed type font and font size.

SnapToGrid – Specifies whether forms objects snap to the grid. When this feature is turned on, objects are automatically aligned to the grid when resized or moved. The resizing and movement of objects on the form are constrained to the grid increment when this feature is turned on.

Having **SnapToGrid** turned on makes it easier to align objects precisely, but limits the freedom with which you can place objects. By default, SnapToGrid is turned On.

To display the grid, select the **ShowGrid** property in the Properties window of the selected form.

The default Grid Settings can be set in **Windows Forms Designer** from the **Tools**, **Options** dialog box.

Inherited Form Objects

When you click the Painter tab to create a new form, you may find that graphical objects enclosed in a GridPanel already exist on the form.

This can occur in the following cases:

- **Inherited Interface** – If the class for which you are creating an interface inherits from another class that contains an interface, the form objects of the parent class are added to the current form as immovable objects and cannot be modified. Instead, if an attribute inherits from a class that contains an interface, the form objects of the parent class are added to the current form as a movable GridPanel when the attribute is dragged onto the form.
- **Attribute Group** – If the class for which you are creating an interface contains an Attribute Group. In previously released versions of the product, these were referred to as Keywords. The existing GridPanel can be moved within the form, but its contents cannot be modified or removed.

Adding Graphical Objects

There are several ways in which you can add graphical objects to a form in the Painter tab.

- Drag an object from the Toolbox onto a form.
- Change the Direction property of an attribute in the Class View.

- Copy an existing object on a form and paste it as an additional object on the same form, or on a different form.
- Drag an attribute from the Class View window on to a form to create an object on the form.

Note: *A quick way to add a label to a form is to position the cursor on the form where you want the label to be positioned, and start typing. A label object is automatically added to the form and your text is placed in the label as you type.*

Each object that you add to a form possesses certain properties. The properties that are available vary depending on the type of object. You can modify the properties of the object to suit the requirements of your form. You may also change the type of some Graphical Objects. For instance, after you have added a Text Field to a form or GridPanel, you may decide that it would be more useable as a Text Area.

Note: *The properties of form objects are transient until a form is saved. It is only after a form is saved that a form and its contents are committed to the project database.*

SM Graphical Painter Toolbox

The SM Graphical Painter Toolbox contains all the graphical objects that can be added to a form. All graphical objects added to a form from the Toolbox possess one or more identifier properties that are used internally by the software. For example, the properties Id and Name. These properties are read-only and cannot be changed. They however appear in the Properties window for your reference.

Two tool boxes are available for different modes in the Painter tab. The SM Graphical Painter Toolbox contains all the elements listed in the following table; the SM Fixed Painter Toolbox contains only the Pointer, Label, TextField, and PasswordField controls.

The table below describes the objects that are generally available in the Toolbox. Click an object on a form to view the available properties in the Properties window.

Note: *To access the tools that are applicable for designing a form in the Painter tab, click the SM Graphical Painter tab in the Toolbox.*

Object	Description
Pointer	By default, this tool is selected when the Toolbox opens. It cannot be deleted. The Pointer enables you to drag objects onto the Design view surface, resize them, and reposition them on the form.
Button	Inserts a Button object. To change the text appearing on the button, edit the Text property. By default, Id=Button1 for the first inserted Button, Id=Button2 for the second, and so on. Multiple buttons on a form or in a GridPanel exhibit particular functionality. Refer to Panel Function for more information.

Object	Description
CheckBox	Inserts a Checkbox object. By default, Id=CheckBox1 for the first inserted checkbox, Id=CheckBox2 for the second, and so on. Multiple checkboxes on a form or in a GridPanel exhibit particular functionality. Refer to Panel Function for more information.
ComboBox	Inserts a single line ComboBox object with Size =27 (Width), 22 (Height). You can change the width by editing the Size property or by dragging the bject. By default, Id=ComboBox1 for the first inserted Combobox, Id=ComboBox2 for the second, and so on.
Label	Inserts a text label object containing text.
ListBox	Inserts a multiline List box object with Size = 27, 38. To display a longer list, edit the size property or drag the object. By default, Id=ListBox1 for the first inserted List box, Id=ListBox2 for the second, and so on. ***Note:*** *The maximum aggregate size of a "list box entry" is 999 characters. This includes delimiters, values sent to the host, and the value appearing in the list box.*
Line	Inserts Line. By default, Id=Line1 for the first inserted Line, Id=Line2 for the second, and so on.
Panel	Inserts a GridPanel object in which other graphical objects can be inserted and are identified as a group of objects sharing common attributes. The border of the panel can be formatted with a Border Style. By default, Id=GridPanel1for the first inserted GridPanel, Id="GridPanel2" for the second, and so on. ***Note:*** *Objects in a GridPanel belong to the same parent.*
Image	Inserts an Image. Edit the ImageSource property to browse to the required image file. By default, Id=Image1 for the first inserted Image, Id="Image2" for the second, and so on.
RadioButton	Inserts a Radio Button object. By default, Id=RadioButton1 for the first inserted Radio Button, Id=RadioButton2 for the second, and so on. Multiple Radio Buttons on a form or in a GridPanel exhibit particular functionality. Refer to Panel Function for more information.
TextField	Inserts a Text Field object. To change the default text, edit the Value property. By default, Id=TextField1 for the first inserted Text field, Id=TextField2 for the second, and so on. The TextField has delimiters (< >) to show the start and end of characters in Fixed mode screens.
SubmitButton	Inserts a Submit button, which transmits the form and the form data to the application at runtime. By default, Id=SubmitButton1 for the first inserted Submit button, Id=SubmitButton2 for the second, and so on.
PasswordField	Inserts a Password field object. By default, Id=PasswordField1 for the first inserted Password Field, Id=PasswordField2 for the second, and so on. The PasswordField has delimiters (^< >) to show the start and end of characters in Fixed mode screens.

Object	Description
TextArea	Inserts a Text Area object to enable a user to enter multiple text lines. You can change the default text by editing the Value property. By default, Id=TextArea1 for the first inserted Text Area, Id=TextArea2 for the second, and so on.
Rectangle	Adds a Rectangle to a page, for example, to visually group other objects on a form. After you add a rectangle, you can modify its properties to change its appearance.

Dragging Objects from the Toolbox

To drag an object from the Toolbox and drop it onto a form, perform the following:

1. Open the Class View window from the **View** menu.
2. Select the element that you want to open in the **Painter**.

   Note: *Sort the hierarchical list into the most appropriate form to locate the attribute you wish to use. If necessary, expand a node to list all the items within the node.*
3. Open the form by clicking the **Painter** tab available at the bottom of the document window.

   Note: *Ensure that the PresentationType property is set to a value other than None.*
4. In the Toolbox, click the desired object and drag it to your form.
5. Drop the object onto the form by releasing the mouse button.

The object is added to the form at the specified location in its default size. You can then modify the properties of the object.

If the object is added to the form by dragging from the Toolbox, and is required to be bound to an attribute (a dynamic object) to be functional, an attribute is created for the object when the object is dropped onto the form. The Attribute property appears in the object's Properties window. Objects for which an attribute is optional and have been added to the form by dragging from the Toolbox are treated as static objects and cannot have an attribute bound to them.

Changing an Attributes Direction Property

You can add a dynamic object to a form by changing the Direction property of its associated attribute. When the direction is changed, a default Text box is placed on the form.

To add an attribute to a form by changing its Direction property, perform the following:

1. Click **Class View** on the **View** menu to open the Class View window.
2. Open an existing form, or create a new one.

3. Select the attribute you wish to add to the form as an object.

   Note: *Sort the hierarchical list accordingly to locate the attribute you wish to use. If necessary, expand a node to list all the items within the node.*

4. Open the Properties window from the **View** menu or press **F4**.
5. Set the Direction property to a value other than None.

   A default text box is placed on the form.

To change the text box to another object type, refer to the instructions in Changing the Object Type.

Dragging an Attribute from the Class View

You can add a dynamic presentation to a form by dragging an existing attribute from the Class View window and dropping it onto a form or GridPanel.

Several conditions apply to performing this operation as described below.

When the attribute is a member of the current class

- If an attribute's graphical object is already on the form, then dropping the attribute is rejected unless you press and hold the control key prior to dragging the attribute from the Class View. In this case, a copy of the attribute is created and added to the current class. Any existing object presentation of the item is read from the Model. If no graphical object exists for the attribute, a default Text Field object is created.

When an attribute is not a member of the current class:

- If an attribute's graphical object is already on the form then dropping the attribute is rejected unless you press and hold the control key prior to dragging the attribute from the Class view. This causes a copy of the item to be created and added to the current class. Any existing object presentation of the item is read from the Model. If no graphical object exists for the attribute, a default Text Field object is created.
- Press and hold the shift key prior to dragging an attribute from the Class View. The attribute is moved from the original class and added to the current class. Any existing object presentation of the item is read from the Model. If no graphical object exists for the attribute, a default Text Field object is created.

When pasting an attribute, a copy of the attribute is created and added to the current class. Any existing object presentation of the item is read from the Model. If no graphical object exists for the attribute, a default Text Field object is created

To drag an attribute from the Class View window and drop it onto a form, perform the following:

1. Open the form by clicking the **Painter** tab available at the bottom of the document window.
2. If it is not already open, open the Class View window by clicking the **View**, **Class View** menu item.

3. Sort the hierarchical list into the most appropriate form to locate the attribute you wish to use.
4. If necessary, click the plus (+) symbol next to a node to list all the items within the node.
5. Select the attribute you wish to assign to a graphical object, and drag it onto the form.
6. Drop the attribute onto the form, following the conditions described earlier.
7. If you want to assign another type of object to the attribute, select the existing object and click the arrow adjacent to the Object Type property in the Property window.

   This displays a list of other object types, which can replace the existing object while still maintaining the binding to its attribute.

   Note: *The list only contains dynamic objects required, or allowed, to be bound to an attribute: Button, CheckBox, Image, Label, ListBox, PasswordField, RadioButton, TextArea, and TextField.*

Some special types of attributes require additional handling when dragged from the Class View window. Refer to Adding an Array and Adding a CopyFrom for more information on these attributes.

Adding a Reference

A Reference is a form of an Attribute and is added to a form in the same way as any other Attribute. They allow for storage of multiple values and to process information in a tabular form, accessing individual items in the array by using a Reference to present the cell in an Array.

To add a Reference to a form, perform the following:

1. Set the **PresentationType** property of the owner of the Reference element to a value other than None.
2. Enter a logical condition in the **Constraint** field that must remain true for the Reference element.
3. Set the **Direction** property of the Reference to a value other than None, or drag the Reference from the Class View onto the form of the owner. Follow the instructions for Dragging an Attribute from the Class View on how to add a Reference element.

Refer to Reference for more information.

Adding a CopyFrom

Attributes that are members of the stereotype class <<Copy>> (described as a Copylspec) possess the capability of being painted on a form multiple times, each instance being a unique representation of the attribute. This capability is known as a Copy.From and is determined by the ‘Is Copied’ property of the graphical object.

To add a CopyFrom to the form, open the Copylspec in the Painter. Refer to Dragging an Attribute from the Class View or Dragging Objects from the Toolbox.

If an attribute is dragged onto the form, a default Text Field is created. Select the attribute and set the IsCopied property to 'True'. This creates additional copies of the text field corresponding to the Copies property of the CopyIspec and a sequence number is appears in the Sequence property of the graphical object in the Properties window. The sequence number determines the sequence in which the objects are displayed at runtime and is adjusted whenever an object is removed from the form. Default TextField objects are created if there is no existing object associated with the attribute.

Therefore, if the number of the Copies property of the CopyIspec is 5 and the Iscopied property is set to True, 5 object copies are painted. These text field copies behave as individual controls, except that they are attached to the same attribute and have the CopySequence property.

If you drag an object from the Toolbox onto the form, instead of dragging an attribute from the Class View, the object appears on the form. Set the IsCopied property to "True" again to create the additional copies of the object.

Once the object copies have been painted on the form, you can change the number in the Copies property of the CopyIspec to a different value. If you do so, you are prompted to "Save and Refresh" the form to display an equivalent number of copies, or to "Ignore" and restore the original value in the Copies property.

Once you have added the number of CopyFrom objects to a form, they can be manipulated individually in the same way as any other object, and can be grouped using a GridPanel .

Note: *If you attempt to add an attribute, that does not have the CopyFrom capability, more than once to a form, it does not succeed. Refer to Copy in the Classes in the* Agile Business Suite Developer Online Help *for more information on the <<Copy>> class.*

Presentation of Insertable Classes

The attributes of an Insertable class are aggregated independently to form the presentation. These attributes have corresponding graphical objects that appear in the presentation. For CopyIspecs, attributes with fixed and graphical presentations are copied. The IsCopied property of each of the graphical objects must be set to True for the object to be copied. If the attribute is copied, it is copied for all presentations (format and language).

Fixed screen presentation format has a Copy Region Panel that can be used to copy controls based on the IsCopied property. The copies are kept in sync through the copy panel. Once the first presentation is modified, the subsequent copies are updated to reflect the change.

Graphical screens do not have such a copy panel. For graphical screens, copies can be moved around separately, independent of the other copies.

Manipulating Objects

Once you have added the object to the form, you can change its size and position by selecting a handle on the border of the object and dragging it to the required position. You can do so by using the Snap to Grid feature for maximum precision, or turning Snap to Grid off for greater freedom of movement.

You can also change the size or position of the object by selecting the object and specifying the Size or Location property in the Properties window.

Changing the Position of an Object Incrementally

You can change the position of an object, or multiple objects, incrementally by “nudging” the selected object with the Up, Down, Left, or Right arrow keys. Select one or more object as described below, and press the appropriate arrow key.

The distance that the object moves depends on the setting of the Snap to Grid option.

- With Snap to Grid on, the object moves one grid increment.
- With Snap to Grid off, the object moves one pixel.

Placing Objects on Fixed Screens

By default, overlapping a character or a field on top of another character or field in a Fixed screen is prevented. In addition, if the available space for the placement of a field on a form is less than the defined length of the attribute, then the field is truncated.

Making an Object Invisible

You may wish to make one or more objects, on a form invisible to the user at runtime. To do this, change the Visibility property of the object from True to False.

Selecting Multiple Objects

You can also select multiple objects that you want to delete, cut or copy, and paste onto another area of the form or onto another form. You can also move multiple objects, in one operation, to a different position on the form.

To select multiple objects, perform the following:

1. Identify the desired objects for selection.
2. Position the cursor on the form at a location outside the total area of the objects.
3. Click and hold down the left mouse button and drag the cursor around the objects. A “bounding” box is drawn on the screen to identify the area within which all the objects are selected.
4. Release the mouse button which in turn selects all the objects within the area.

Note: *Selecting multiple objects in this way groups them only temporarily and should not be confused with creating a group of objects with the GridPanel control.*

You may also select multiple objects by holding down the Shift key and clicking each object that you wish to select.

Changing the Object Type

If you want to change an existing object to another object type, select the object and click the arrow adjacent to the Object Type property in the Property window. This displays a list of other object types, which can replace the existing one. If the existing object is not capable of being assigned to an attribute, the object does not provide an Object Type property and cannot be changed.

Note: *If the existing object is of a type that is required (dynamic), or allowed to be assigned to an attribute to be functional, the list only contains dynamic objects capable of being assigned to an attribute and of the allowed type. These dynamic objects include Button, CheckBox, Image, Label, ListBox, PasswordField, RadioButton, TextArea, and TextField.*

The following objects are static and cannot be bound to an attribute, and their object type cannot be changed using the Object Type property. If you wish to change such an object from one type to another, you must delete the existing object and add another object of the preferred type:

- GridPanel
- Line
- Rectangle
- Submit

Removing an Object

Select the object to be removed from a form, and on the **Edit** menu or the context menu, click **Delete**.

Note: *When an object is removed from a form and has been bound to an attribute, the attribute is not deleted from the Model unless the following steps are taken.*

To remove an object and delete the attribute, perform the following:

1. If it is not already open, open the Class View window by clicking the **View**, **Class View** menu item.
2. Sort the hierarchical list into the most appropriate form to locate the attribute you wish to keep.
3. If necessary, click the plus (+) symbol next to a node to list all the items within the node.

4. Select the attribute you wish to delete.
   - In the Property window select the **Direction** property.
5. Set the property to None.

This removes the object from the form in the Painter window and deletes the attribute.

Sizing an Object Dynamically

Certain graphical objects that contain text, or are attached to an attribute, can have their height and width resized dynamically to fit the text or attribute to which they are attached.

The graphical objects that can be resized are listed in the following table with the applicable resize function.

Control Type	Attribute	Text	Both
Label			YES
Button		YES	
SubmitButton		YES	
PasswordField			YES
RadioButton		YES	
CheckBox		YES	
TextField	YES		
TextArea	YES		

"Both" means that the control can be resized to either the length of the attribute or the length of the text whichever is longest.

The resize calculation considers text alignment for the controls from the "TextAlign" property. Resizing does not occur when the "SynchronizeField" property is set to "Yes" on the component.

To set the relative size of an object, perform the following:

1. Open a form.
2. Select the graphical object that you wish to resize.
3. On the **Format** menu, click **Make Same Size**.
4. On the sub-menu, click either **As Text** or **As Attribute**.

   The menu entries are only available if the object has either Text or an Attribute attached.

When a group of objects are selected, and none of the selected objects have text or an attribute attached, then none of the menu entries are available. If the group has at least one object with text and at least one object with an attribute attached, both menu entries are available.

If an object has both text and an attribute attached, the object is resized to either the length of the attribute or the width of the text whichever is longer.

As Text – The width and height of the object are resized to the pixel width and height of the current text based on the font for this object.

To make an object with a PresentationType property of Fixed or Print the same size as text, the object width is resized to the character width of its text based on the font for this control.

As Attribute – The width and height of the object are resized to the pixel width and height of the **total length** of the attribute based on the font for this object.

To make an object with a PresentationType property of Fixed or Print the same size as the attribute, the object width is resized to the character width of its **total length** based on the font for this control.

The **total length** is the sum of attribute length and extra characters applicable for value of the PrintFormat property, if the attribute is a SignedNumber. The table below identifies the number of extra characters required for each PrintFormat value and PresentationType.

Graphical	**Extra Characters**	**Fixed**	**Extra Characters**	**Print**	**Extra Characters**
CR	2	CR	2	Blank	1–2
DR	2	DR	2	CR	2
+	1	+	1	DR	2
_	1	_	1	+	1
				_	1
				$	2–3
				*	1–2

Graphical Object Properties

To set the properties of a graphical object by using the Properties window, perform the following:

1. If the item that you want to modify is not selected, click the down arrow button next to the object name box to display the list of objects and select it, or click the object on the form.

2. In the Properties window, if the properties are displayed alphabetically, select the property that you want to modify. If the properties are displayed within groups and the property you want to set is not visible, click the plus (+) symbol next to the appropriate group.

   For example, click the Misc group and select tabindex to set the index for the tab order of the object.

3. Specify a value for the property.

Depending on the property, you may be required to type a value, select a value from a list, or set the value in a custom editor. If you wish to restore a changed value to its default value, select the object, right-click the appropriate property, and then click **Reset** on the context menu.

Setting a Single Property for Multiple Objects

You can specify the same property value for a group of objects. This technique can be useful, for example, if you add several button objects to a form or document and want each button to have the same value for the height and width properties.

Note: *If you select multiple objects of different types, the Properties window displays only the properties that are common to all the selected objects.*

To set the value of a property for multiple objects, perform the following:

1. Select the first object in the group of objects that you want to modify.
2. Hold down the Ctrl or Shift key while selecting other objects that you want to modify.
3. In the Properties window, specify the value for the property.

   The value is set for each selected object.

You may also select multiple objects by creating a “bounding” box around objects as described in Manipulating Objects under “To select multiple objects”.

- All Object Properties
- Common Properties
- Pointer Object Properties
- Button Object Properties
- CheckBox Object Properties
- ComboBox Object Properties
- Label Object Properties
- ListBox Object Properties
- Line Properties
- GridPanel Properties
- Image Object Properties
- Radio Button Object Properties

- TextField Object Properties
- Submit Button Properties
- PasswordField Object Properties
- TextArea Object Properties
- Rectangle Object Properties

Common Properties

The properties listed in the following table are common to all, or multiple graphical objects. The read-only Design properties reflect the properties of the attribute attached to the form object. These Design properties can be modified by changing the attribute properties.

Property	Function
Direction	Read-only design property. Specifies the value of the Direction property of the attribute associated with the form object.
Id	Read-only design property. By default, it is automatically generated. Identifies the object type and number.
Length	Read-only design property. Specifies the Length of the attribute associated with the form object.
Name	Read-only design property. Specifies the Name of the attribute associated with the form object. You can change this property by renaming the attribute. Its value is set to Static Control for unassociated form objects.
Primitive	Read-only design property. Specifies the Primitive type of the attribute associated with the form object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. All objects initially inherit the background color of their parent. Note that you can specify a custom color for graphical screens and specify one of the following colors for character screens: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an Image to display in the background of the object.
BorderStyle	Specifies a style to control the border of the container that surrounds the object. Options are None, FixedSingle, or Fixed3D.
ControlType	Specifies the type of object the selection should be. To change the ControlType, choose from the list of available types.
DefaultValue	Specifies the default text thatdisplays in the drop-down box entry field when the host does not send any data.

Property	Function
DropDownStyle	Specifies the appearance and functionality of the ComboBox. Select from the values Simple, DropDown, and DropDownList. Note that if you change the height by using the mouse or entering values in the Height property, the physical height of the control appears only for a Simple style, though at runtime the height setting is maintained for all styles.
Font	Specifies font attributes that are applied directly to the text in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color for graphical screens and specify one of the following colors for character screens: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
ReadOnly	Specifies whether the content of the object is read-only.
ScrollBars	Specifies whether the Form should have scroll bars or not. Setting this property only have an impact on systems generated for CE. There is no impact at design time. Options are No, Vertical, Horizontal, and Both (default).
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
Text	Specifies the text to be displayed in the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
TextPosition	Specifies the position of the label relative to the object. The choices are Left and Right (default).
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winform. To view the tooltip in Presentation Client, the user has to hover on the control and press the F1 key. This tooltip is visible only at runtime.
ValidateEntry	When set to True, specifies that the value entered by the user at runtime is one of the values in the drop-down list.
Value	Specifies the value to be displayed in or on the object. This value is returned to the application when a transaction occurs at runtime.
Visible	Specifies if the object is visible to the user at runtime, options are True (default) and False.

Property	Function
Wrap	Specifies whether the browser automatically performs word-wrap on the text of the object.

To see properties specific to any particular object, which also include common properties for each object, click the object in the list below.

- Pointer Object Properties
- Button Object Properties
- CheckBox Object Properties
- ComboBox Object Properties
- Label Object Properties
- ListBox Object Properties
- Line Properties
- GridPanel Properties
- Image Object Properties
- Radio Button Object Properties
- TextField Object Properties
- Submit Button Properties
- PasswordField Object Properties
- TextArea Object Properties
- Rectangle Object Properties

Pointer Object Properties

There are no properties associated with the Pointer object.

Button Object Properties

Add a Button to a form to enable the user to provide a value that is sent back to the application when the button is clicked at runtime. Button objects have special characteristics when added to a GridPanel or a form.

Note: *The read-only Design properties are displayed for Button objects associated with attributes. These properties provide a better understanding of the object. You can change the property values in the Properties window after selecting the associated attribute in the Class View.*

Property	Function
Direction	Read-only. Specifies the value of the Direction property of the attribute associated with the Button object.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=Button1 for the first inserted Button, Id=Button2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the Button object.
Name	Read-only. Specifies the name of the attribute associated with the Button object.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the Button object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an image to appear in the background of the object. The image always appears as a tiled image.
ControlType	Specifies Button as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Selected	Specifies whether the button should be selected by default. You can select only one button on a form. Therefore, when set to True, any other button in the same group is reset to False.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object. Button groups are defined as a set of button controls within a GridPanel. Set the TabIndex property on the GridPanel to define the position of the button group as a whole within the overall form tab order. The TabIndex values for the individual buttons within the GridPanel can be set to any arbitrary values to define the order of tabbing from button to button within the button group.
Text	Specifies the text to be displayed in the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.

Property	Function
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Value	Specifies a value to be returned to the server when the form is submitted at run time. This is to provide keyword/value pairs for button groups. Refer to Panel Function for more information on this property.

CheckBox Object Properties

Add checkboxes to your form when you want a user to select one or more items, or none at all. Checkboxes have special characteristics when added to a GridPanel or a form.

When you add a checkbox to a form, the checkbox has a label attached to it automatically. You can change its Text property and specify a label.

Property	Function
Direction	Read-only. Specifies the value of the Direction property of the attribute associated with the CheckBox object.
Id	Read-only. By default, it is automatically generated. Id=CheckBox1 for the first inserted CheckBox, Id=CheckBox2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the CheckBox object.
Name	Read-only. Specifies the name of the attribute associated with the CheckBox object.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the Checkbox object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an image to appear in the background of the object. The image always appears as a tiled image.
Checked	Specifies the initial state of the check box. Can also be specified for Radio Buttons.
CheckedValue	Specifies the string value to be returned to the server for a single checkbox when the checkbox is checked and the form is submitted at run time. For multiple checkboxes in a group, this property does not exist. Refer to GridPanel Function for more information on this property.
ControlType	Specifies CheckBox as the type of object added to the form. To change the ControlType, choose from the list of available types.

Property	Function
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
Text	Specifies the text to be displayed in the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
TextPosition	Specifies the position of the label relative to the object. The choices are Left and Right (default).
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Unchecked Value	Specifies the string value to be returned to the server for a single checkbox when the checkbox is unchecked and the form is submitted at run time. For multiple checkboxes in a group, this property does not exist. Refer to Panel Function for more information on this property.

ComboBox Object Properties

Property	Function
Id	Read-only. By default, it is automatically generated. Id=ComboBox1 or the first inserted ComboBox, Id=ComboBox2 for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ControlType	Specifies ComboBox as the type of object added to the form. To change the ControlType, choose from the list of available types.

Property	Function
DefaultValue	Specifies the default text that appears in the drop-down box entry field when the host does not send any data.
DropDownStyle	Specifies the appearance and functionality of the ComboBox. Select from the values Simple, DropDown, and DropDownList. Note that if you change the height by using the mouse or entering values in the Height property, the physical height of the control appears for a Simple style, though the height setting is maintained at runtime for all styles.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ListContents	Opens the List Contents Editor to specify settings for the List Box. Can also be specified for a Combo Box.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
ScrollBars	Specifies whether the ComboBox should have scroll bars or not. Options are No (default) and Horizontal.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
ValidateEntry	When set to True, specifies that the value entered by the user at runtime is one of the values in the drop-down list.

Label Object Properties

Add a Label to your form to identify the name of a control or group of controls. A Label can optionally be bound to an attribute.

Note: *When a Label is added to a form that has the identity of a Report Frame, the Label properties are different. Refer to Label Object Properties in Report Frames for information on these properties.*

Labels are of two types: Static and Dynamic.

Static Labels are directly typed on the form in the Painter tab or created with a drag and drop operation from the Toolbox (these labels are not associate with any attribute).

Dynamic Labels are labels whose ControlType is changed to Label from another control that was associated with an attribute. If the control is associated with an Attribute, then deleting the attribute results in deleting the related presentation on the Painter.

When a Label is added to a form that has the identity of a Fixed screen, the Label properties are different. Refer to Label Object Properties in Fixed Screens for more information on these properties.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=Label1 for the first inserted Label, Id="Label2" for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BorderStyle	Specifies options to control the border of the container that surrounds the object. The following options are available: • None – The object appears without a border. • FixedSingle – The object has a single line border. • Fixed3D – The object has a three-dimensional border. • UnderBar – The object has a border only for the lower line.
ControlType	Specifies Label as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object. If the label contains a "shortcut key" definition (a character preceded by the "&" character), then you should set the TabIndex value to the same value as the TabIndex value of the field to which you want focus to go when the shortcut key is activated.
Text	Specifies the text to be displayed in the object in the form.

Property	Function
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Wrap	Specifies whether the browser automatically performs word-wrap on the text of the object.

ListBox Object Properties

When you want user to choose options from a list, add a List box to your form. Only one choice can be made from the list at runtime.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=ListBox1 for the first inserted ListBox, Id=ListBox2 for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BorderStyle	Specifies a style to control the border of the container that surrounds the object. Options available for List Box control are None and Fixed3D. ***Note:*** *The appearance of a ListBox is always Fixed3D, regardless of whether the None or Fixed3D options are selected.* Client generators such as Component Enabler and Winforms interpret the BorderStyle property set here in the Property window and generate the Graphical User Interface accordingly.
ControlType	Specifies ListBox as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking “Reset” on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ListContents	Opens the List Contents Editor to specify settings for the List Box. Can also be specified for a Combo Box.

Property	Function
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
ScrollBars	Specifies whether the ListBox should have scroll bars or not. Options are No (default) and Horizontal.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.

Line Properties

You can add a Line to a page, for example to separate text or areas of a form. After you add a line, you can modify its properties to change its appearance.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=Line1 for the first inserted Line, Id="Line2" for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
ControlType	Read-only. Specifies Line as the type of object added to the form.
Direction	Specifies if the line drawn is in the vertical or horizontal direction.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Length	Specifies the length of the line in pixels.
LineStyle	Specifies the style of the line and can be selected from the list.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
TabIndex	Specifies the index that defines the tab order for the object.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.

GridPanel Properties

The GridPanel acts as a container for other graphical objects and exhibits the following characteristics:

- A GridPanel can be dropped on a form from the Toolbox using the Panel tool.
- A GridPanel can be pasted on a form from the clipboard.
- A GridPanel can contain one or more objects including other Panels.
- Objects can be moved into a GridPanel.
- Objects can be moved out of a GridPanel.
- Objects can be dropped into a GridPanel from the Toolbox.
- Objects can be pasted into a GridPanel from the clipboard.
- If a panel is deleted, all the objects that it contains are also deleted.

When a GridPanel contains other objects, all objects within the GridPanel are treated as a single entity. The objects can be moved with the GridPanel while maintaining their relative positions to the border of the GridPanel. The objects can inherit the presentation style applied to the GridPanel by using the properties selected for the GridPanel in the Properties window, unless the objects have their own individual presentation style applied to them. Note that the form itself is in fact a “parent” GridPanel.

A GridPanel shares the same qualities as a form, in that any object placed in the GridPanel inherits the visual attributes of the GridPanel. You can specify different visual attributes for any particular object or group of objects after they have been added to the GridPanel. If you wish to return the changed visual attribute of an object to its original inherited value, select the object and the property that you wish to reset, and then click Reset on the context menu.

Note: *There is a unique property Value, assigned to Buttons and Radio Buttons to accommodate their functionality when in a GridPanel or a form. There is also a Checked Value property specific to checkboxes. Refer to GridPanel Function for more information on how to use these objects in a GridPanel.*

Property	Function
Id	Read-only. By default, it is automatically generated. Id=GridPanel1 for the first inserted GridPanel, Id=GridPanel2 for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an image to appear in the background of the object. The image always appears as a tiled image.
BorderStyle	Specifies the width or type of border to draw around the GridPanel.

Property	Function
ControlType	Read-only. Specifies GridPanel as the type of object added to the form.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object. The TabIndex value of a GridPanel object defines the position of the GridPanel as a whole within the overall form tab order, relative to the other controls on the form outside the GridPanel. The TabIndex value of controls within the GridPanel defines the tab order within that GridPanel only. As you tab through a form, the tab order moves in the GridPanel according to the TabIndex value of the GridPanel object. Focus initially goes to the control within the GridPanel with the lowest TabIndex value, and the tab order then moves through all controls within the GridPanel in the TabIndex sequence. The tab order then moves outside the GridPanel to the control on the form with the next highest TabIndex value to the GridPanel object.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.

Image Object Properties

You can add images with the following file formats: GIF (standard and animated), JPEG (standard and progressive), BMP (Windows and OS/2), TIFF, TGA, RAS, EPS, PCX, PNG, and WMF. The formats that are generally used for forms are GIF, JPEG, and PNG.

An image can optionally be bound to an attribute.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=Image1 for the first inserted Image, Id=Image2 for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BorderStyle	Specifies the width of the border to draw around the object.
ControlType	Specifies Image as the type of object added to the form. To change the ControlType, choose from the list of available types.

Property	Function
ImageSource	Specifies the file name of the image used.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Sizing	ResizeImage and ResizeBox are the two options available. By default, the value is ResizeImage. When an image is loaded with the ResizeImage option set, then the loaded image fits the size of the box set in the Painter. On toggling to ResizeBox, the box stretches to fit the original image size. When an image is loaded with the ResizeBox option set, then the box stretches to fit the original image size. Resizing of the image is not allowed in this mode.

Radio Button Object Properties

When you want a user to select one option in a group, add radio buttons to your form. Only one radio button in a group can be selected at a time. Radio buttons enable the user to provide a value that is returned to the application when a transaction occurs at runtime. Radio buttons have special characteristics when added to a GridPanel or form.

When you add radio button to a form, it has a label attached to it automatically. The label can have its text properties specified for that particular combined object.

Property	Function
Direction	Read-only. Specifies the value of the Direction property of the attribute associated with the RadioButton object.
Id	Read-only. By default, it is automatically generated. Id= RadioButton1 for the first inserted RadioButton, Id=RadioButton2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the RadioButton object.
Name	Read-only. Specifies the name of the attribute associated with the RadioButton object.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the RadioButton object.

Property	Function
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an image to appear in the background of the object. The image always appears as a tiled image.
Checked	Specifies the initial state of the radio button. When set to True, any other radio button in the same group is set to False.
ControlType	Specifies RadioButton as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
Text	Specifies the text to be displayed on the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
TextPosition	Specifies the position of the label relative to the object. The choices are Left and Right (default).
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Value	Specifies the value to be displayed on the object. This value is returned to the application when a transaction occurs at runtime.

TextField Object Properties

When you want to accept one line of information from a user, add a one-line Text Field to your form.

Property	Function
Direction	Read-only. Specifies the value of the Direction property of the attribute associated with the TextField object.
Id	Read-only. By default, it is automatically generated. Id=TextField1 for the first inserted TextField, Id=TextField2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the TextField object.
Name	Read-only. Specifies the name of the attribute associated with the TextField.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the TextField object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BorderStyle	Specifies an attributes to control the border of the container that surrounds the object. Options are None, FixedSingle, or Fixed3D or UnderBar. When the property is set to None, the selected control appears without a border. When the property is set to FixedSingle, the selected control has a single line border. When the property is set to Fixed3D, the border of the selected control has a three-dimensional appearance. When the property is set to UnderBar, only the bottom line appears for the selected control. The other three border lines do not appear.
ControlType	Specifies TextField as the type of object added to the form. To change the ControlType, choose from the list of available types
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking “Reset” on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
ReadOnly	Specifies whether the content of the object is read-only.

Property	Function
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies a number value that indicates the tab order for the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Value	Specifies the value to be displayed in the field. This value is returned to the application when a transaction occurs at runtime.
Visible	Specifies if the object is visible to the user at runtime, options are True (default) and False.

Submit Button Properties

Add a Submit button as a static control to cause the form to be transmitted to the application at runtime.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=SubmitButton1 for the first inserted SubmitButton, Id=SubmitButton2 for the second, and so on.
Name	Read-only. The value of this property is set to Static control.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BackgroundImage	Specifies an image to appear in the background of the object. The image always appears as a tiled image.
ControlType	Read-only. Specifies SubmitButton as the type of object added to the form.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.

Property	Function
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
Text	Specifies the text to be displayed in the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, centered, or justified.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.

PasswordField Object Properties

When you want a user to enter a password on the form, add a password field to the form. A password field is just a one-line text box. When a user types in this field, most Web browsers display the password as asterisks to protect confidentiality.

Property	Function
Direction	Read-only. Specifies the value of the Direction property of the attribute associated with the PasswordField object.
Id	Read-only. By default, it is automatically generated. Id=PasswordField1 for the first inserted PasswordField, Id=PasswordField2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the PasswordField object.
Name	Read-only. Specifies the name of the attribute associated with the PasswordField object.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the PasswordField object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.

Property	Function
BorderStyle	Specifies an attributes to control the border of the container that surrounds the object. Options are None, FixedSingle, or Fixed3D or UnderBar. • When the property is set to None, the selected control appears without a border. • When the property is set to FixedSingle, the selected control has a single line border. • When the property is set to Fixed3D, the border of the selected control has a three-dimensional appearance. • When the property is set to UnderBar, only the bottom line appears for the selected control. The other three border lines do not appear.
ControlType	Specifies PasswordField as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking "Reset" on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking "Reset" on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Read Only	Specifies whether the content of the object is read-only.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Value	Specifies the value to be displayed in the field. This is the value, which is returned to the application when a transaction occurs at runtime.

TextArea Object Properties

Add a text area to your form when you want to accept multiple lines of text from a user.

Property	Function
Id	Read-only. By default, it is automatically generated. Id=TextArea1 for the first inserted TextArea, Id=TextArea2 for the second, and so on.
Length	Read-only. Specifies the Length of the attribute associated with the TextArea object.
Name	Read-only. Specifies the name of the attribute associated with the TextArea.
Primitive	Read-only. Specifies the value of the Primitive property of the attribute associated with the TextArea object.
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BorderStyle	Specifies an attributes to control the border of the container that surrounds the object. Options are None, FixedSingle, or Fixed3D or UnderBar. • When the property is set to None, the selected control appears without a border. • When the property is set to FixedSingle, the selected control has a single line border. • When the property is set to Fixed3D, the border of the selected control has a three-dimensional appearance. • When the property is set to UnderBar, only the bottom line appears for the selected control. The other three border lines do not appear.
ControlType	Specifies TextArea as the type of object added to the form. To change the ControlType, choose from the list of available types.
Font	Specifies font attributes that are applied directly to the text appearing in the object. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking “Reset” on the context menu.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking “Reset” on the context menu.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
ReadOnly	Specifies whether the content of the object is read-only.
Size	Specifies the size of the object representing the height and width in pixels.
TabIndex	Specifies the index that defines the tab order for the object.
TextAlign	Specifies whether the text in the object is left-aligned, right-aligned, center-aligned, or justified.

Property	Function
ToolTip	Specifies an optional tooltip to appear when the user at runtime hovers the mouse over a control in Winforms. To view the tooltip in the Presentation Client, the user has to hover over the control and press the F1 key. This tooltip is visible only at runtime.
Value	Specifies the value to be displayed in the field. This value is returned to the application when a transaction occurs at runtime.

Rectangle Object Properties

You can add a Rectangle to a page, for example to visually group other objects on a form. After you add a rectangle, you can modify its properties to change its appearance.

Property	Function
BackgroundColor	Specifies a background color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color. If you change this value, you can restore the default value specified for the parent object by right-clicking the property and clicking **Reset** on the context menu.
BorderStyle	Specifies the style of the border of the object. Choose from the list of available types to change the border style.
ControlType	Read-only. Specifies Rectangle as the type of object added to the form.
ForegroundColor	Specifies a foreground color for the object. Select a color from one of the tabs in the drop-down list. Note that you can specify a custom color.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Size	Specifies the size of the Rectangle object representing the height and width in pixels.

List Contents Editor

Use the List Contents Editor dialog box to enter and edit a list of options displayed in a List box or Combo box on a form.

To display this dialog box, select a ListBox or ComboBox object on a form and click the ListContents property from the Property window for the selected object.

The dialog box allows you to create a list of selectable options, which are displayed at design time and runtime, and possible values that the selection returns.

Runtime Created Contents

List name – This field specifies the name of the list that is created at runtime by using the SENDLIST logic commands. The format of the name of the list object must be one of the following:

ispec.attribute (LBX; in previous versions of Developer)

.attribute (LBX; in previous versions of Developer)

file name (specifies the file name that contains the list items).

Object name from host – This field specifies the object name that is sent to the form by the host application program at runtime.

Item displayed in list – This field specifies which column from the data file is displayed in the list.

Value sent to host – This field specifies which column in the data file is sent to the host when you select the associated list item at runtime

Start line – This field specifies the number of the first displayable line in the file. This field applies only to List box controls.

Fixed Contents

Text – Specifies the literal string displayed for an option within this box, and allows you to edit existing option texts.

Value – Specifies the value returned when an option is selected, and allows you to edit existing option values.

Insert – Inserts a new blank line in the list box after the line currently highlighted.

List – Displays the options that this list includes currently. To modify an existing option, select the options in the list, and then edit the values in the Text and Value fields.

Up and Down Arrows – Changes the order in which Options appear in the List box or Combo box.

Delete – Removes an Option from the List box or Combo box.

To add properties in the dialog, perform the following:

1. Enter the literal string that you want to display in the **Text** field.
2. Enter a value in the **Value** field.
3. Click **Insert** to add to the list.

Note: *The data for static list boxes is stored in a data file named <RuntimeDatabaseName>_<DeployedApplicationName>_listbox.dat in the ..\NGEN\DATA\Private folder, which is created at the time of installation of Developer. This file is created whenever you run a report or execute logic by using the SendListStatic command. You may wish to backup this file periodically.*

Color Picker

Use the Color Picker to select the color that you want to apply to the Background or Foreground. You can click any one of the three tabs to select a color.

Custom Colors

There are 48 preset basic colors that cannot be modified.

To select one of these basic colors, perform the following:

1. Click the square of the basic color that you want.
2. Click **OK** to close the dialog box.

You can fill 16 custom color squares with colors of your choice. You can fill an empty square or replace the color in a square with a new color.

To create a custom color, perform the following:

1. To create a custom color, right-click one of the available custom color squares. The Define Color dialog box appears.

   This dialog box modifies the red, green, blue, hue, saturation and luminance values of the current color.
2. To adjust these values, click inside the color spectrum box or drag the crosshair pointer to the desired position.

   The thin color ribbon box modifies the red, green, blue, and luminance values of the current color.

   To adjust these values, click inside the color ribbon box or drag the pointer on the right of the box up or down. The Color/Solid display box changes as you move the pointer.

   The controls affect the contents of the Color/Solid display box, which displays the currently selected color.
3. Click **Add Color** to add the custom color to the previously selected custom color square.
4. Click **OK** to close the **Color Picker** dialog box.

To change a custom color, right-click or double-click the custom color square in the lower section of the dialog box and repeat steps 1 through 4 for creating a custom color.

Web Colors

The Web tab displays the standard 216 safe colors. These safe colors are composed of combinations of the values 00, 33, 66, 99, CC, and FF for red, green, and blue, and they should be correctly displayed by any browser.

To select one of these colors, perform the following:

1. Click the box of the color you want.
2. Click **OK** to close the dialog box.

System Colors

The System tab displays the system colors that are set in the Windows Display Control Panel.

To select one of these colors, perform the following:

1. Click the box of the color you want.
2. Click **OK** to close the dialog box.

Graphical Objects and Attributes

Unless the graphical object is a static object and has a specific and unique purpose such as a Line, it requires binding to an existing attribute as a data source to be functional.

The following objects are static objects and **cannot** be bound to an attribute:

- GridPanel
- Line
- Rectangle
- Submit Button

The following objects are optionally dynamic and **may** be bound to an attribute, but are not required to be:

- Label
- Image
- List Box

These optional dynamic objects when bound to an attribute cause the attribute to be of the type Usage Inquiry. This enables an attribute value to be displayed on the screen, but not receive input from the screen or write to the database. Usage Inquiry is only used to display data values on the screen.

Usage Inquiry Attributes can be receiving fields only. They may not be used as source fields for arithmetic or data manipulation in Logic.

An example where an image could be bound to a Usage Inquiry attribute is where you may want to change the image displayed on a form depending on a previous input action of a form, say a warning sign appears if an error occurred in an input transaction.

The relationship between dynamic graphical objects and an attribute exists to enable the user to interact with the attribute so that records in the underlying database can be read, created, modified, and deleted by the user (where applicable). The interaction is carried out by the user manipulating the objects on the form with the actions permitted by a given object, for instance a Button can be clicked and an item in a List box can be selected.

Special consideration needs to be made for the GridPanel functionality when populated with Radio Button, check box, or Button objects.

Group Control Function

The graphical objects that are grouped together to represent one attribute are called Group Controls. The graphical objects button, check box, and radio button are called Group Controls. For example, check boxes can be used as the multiple choice answers for one question.

Group Controls do not leave their container and cause attribute disassociation. For example, if a Button is dragged from the Toolbox onto the Graphical Painter Form, then the parent of the Button is always the Form. If the Button is dragged into a GridPanel, then the parent of the Button is always the GridPanel.

Moving an Object in Graphical Painter

Moving an object in the Graphical Painter Form does not change the container of an object. It only changes the location of the object. For example, if you move an object into the region of a GridPanel, the GridPanel does not become the container of the object. Instead the GridPanel and object overlap each other.

If you want to change the container of controls, such as label, text field, list box, you must press the Ctrl key while moving them.

Group Controls within the same container associate with the same attribute. To change the container for Group Controls, you must:

1. Select all the Group Controls that associate with the same attribute. Since Group Controls never leave their container that associate to the same attribute, changing container for a few Group Controls results in attribute dissociation, which is not permitted.
2. Press and hold the Ctrl key when selecting multiple Group Controls.

   Note: *You should move the Group Controls to a container that does not have the same kind of Group Controls.*

For example, moving Group Control buttons from the GridPanel:

1. Add four buttons, Button1, Button 2, Button 3, and Button 4, to a GridPanel in the Painter.

   Note: *All the buttons associate with attribute 1.*
2. Try to change the container of Button1, Button 2, and Button 3 by selecting and moving them.

The change container operation is not allowed.

To change the container of the buttons, you should select all the four buttons and move them.

For example, to move a Text Field to the Grid Panel:

1. Add a Text Field and a Panel to the Painter.
2. Select the Text Field and press and hold the left mouse button.
3. Press and hold the Ctrl key and then move the selected Text Field into the Panel.

   The boundary of the GridPanel gets highlighted and the GridPanel now becomes the container of the Text Field.

Since the Ctrl key is also used for multi-selection, the steps to move the controls with the Ctrl key is as follows:

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.

Panel Function

The GridPanel performs a secondary function in the Painter. In addition to acting as a container for objects, it has a special relationship with Buttons, Check Boxes, and Radio Buttons when they are added to a GridPanel.

Each object type shares the same characteristics in relation to its default Id and Attribute Name properties as follows.

Adding a Button Object to a GridPanel

When a button is initially added to a GridPanel, its default Id is Button1. It also creates a default attribute with the name Attribute 1. When a second button is added to the same GridPanel, it receives a default Id Button2 but retains its binding with the attribute created by the addition of the first button, which is Attribute1. This continues for all buttons that are added to the same GridPanel.

Adding a Radio Button Object to a GridPanel

Adding a radio button to a GridPanel follows the same rules as adding a button object, with the exception that the default Id for each radio button is RadioButton followed by the number of the button. Each radio button is bound to the same attribute as the first radio button that is added.

Adding a Checkbox Object to a GridPanel

Adding a checkbox to a GridPanel follows the same rules as adding a button object, with the exception that the default Id for each checkbox is CheckBox followed by the number of the checkbox. Each checkbox is bound to the same attribute as the first checkbox that is added.

When moving a button or a checkbox from one group to another, the button is re-assigned to the attribute of the new group.

Value Property of a Button or Radio Button

Though each button (or radio button) within a group is bound to the same attribute, each object maintains a different Value property. This allows the user to return a different value to the same attribute when the transaction is completed. One example of the use of this is where a GridPanel contains several buttons, each of which can return a different value to the MAINT field of an attribute when clicked.

Checked and Unchecked Value Properties of a Checkbox

The value of a checkbox differs from other objects. Instead of the Value property, a checkbox has the Checked Value property that specifies a string value to be sent to the application when the object is checked and the form is transmitted at runtime. There is also an Unchecked Value property that specifies the value to be sent to the application when the control is unchecked. These properties are only available for a single checkbox; when there are multiple checkboxes, these properties do not exist.

Multiple checkbox objects within the GridPanel are considered to be a group and are assigned to one attribute in the same way as buttons and radio buttons. At runtime, a representation of the state of the checkboxes in the group is sent to the application. For example, a representation of 1010 means the first and third checkboxes are selected and the second and fourth checkboxes are clear.

GridPanel Behavior

The behavior of the GridPanel varies with the movement of objects in it. For example:

- In a GridPanel, you can change the location of objects by moving them. This changes the location of objects only in the GridPanel without disturbing the location of the GridPanel. To do this, select the objects in the GridPanel and move them. The GridPanel does not move but the objects within it move. However, it may impact the size of the GridPanel.

  Note: *To select multiple objects, press and hold the Ctrl key.*

- In a GridPanel, if you move some of the objects out of the boundary, the GridPanel area expands to include all the objects. If you move some of the objects within the boundary, the location of the objects changes within the boundary without expanding the GridPanel area. The location of the GridPanel does not change.
- In a GridPanel, when you try to move the objects out of the boundary, the boundary of the GridPanel is highlighted.

Tool Tips at Design Time

At design time, information about an attribute that is bound to a graphical object can be displayed in a Tool Tip when the mouse hovers over an object. The tool tip can display the following information:

- Attribute name
- Edit type
- Length
- Caption

Note: *If an object, which requires an attribute, has been added to the form by dragging it from the Toolbox, but the form has not yet been saved, the tooltip displays the text "Attribute not assigned." To display the tool tip for such an object, you must first save the form to create an attribute for the object.*

Graphical objects that cannot be bound to an attribute display the text "Static Control" in the tooltip. Objects for which an attribute is optional and that have been added to the form by dragging from the Toolbox are treated as static controls and they display the same tooltip.

Synchronizing Form Objects

The **Synchronize Class View** option selects an element in the Class View that is associated with a form or a form object. You can select any graphical object on a form and instantly determine its corresponding data attribute in the Class View hierarchy.

To synchronize a form object with its data attribute in the Class View, right-click the object, and select **Synchronize Class View**.

This highlights the data attribute that is associated with the graphical object on the form. If the Properties window is open, you can also edit the properties of the associated attribute that is selected in the Class View. For form objects that are not associated with any data attribute, the class that contains the form is selected. The **Synchronize Class View** option on a form, or a multiple selection of objects, selects the class in the Class View that contains the form.

The Properties window also displays certain Design properties of form objects associated with primitive data types. The properties that are displayed depend on the type of form object and are read-only, such as Length, Primitive, Direction, Decimals, and Constraint. This provides a quick view of some of the properties of the primitive data type associated with the selected form object. The changes that are made to the Design properties of data attributes associated with form objects, such as Button, CheckBox, Radiobutton, TextField, PasswordField, or TextArea, are automatically reflected in the Design properties of the form object.

You can also synchronize objects that are inherited from a form of a parent class. For example, if an attribute with Direction other than None inherits from a class that contains a form, the form objects of the parent class are added to the current form as a movable GridPanel.

Note: *If an attribute in an Ispec or Class with a Fixed Presentation inherits from an Insertable class with a Fixed Presentation, the form objects of the parent class are added to the current form as an immovable GridPanel. In such instances, you cannot synchronize the inserted form objects.*

Creating Fixed Screens

A Fixed mode screen (green screen painter) is designed by using the Fixed mode Painter for different runtime platforms. These platforms require character-based interfaces; therefore, you need to paint fixed width (use a font that is not proportional to the width of the characters) objects to display text and fields.

You can use only the following graphical objects to paint on a character screen: Pointer, Labels, Textfields, and Password field. The Label and Textfield object properties for a character screen differs from those for the graphical screen.

You can also generate presentations for the Insertable Stereotypes, Aggregated Presentations, and CopyIspec or CopyEvent Stereotypes on a green screen. For CopyIspec/CopyEvent Stereotypes, the Character Mode Painter defines the number of copies based on the IsCopied property.

All overlapping of fields or characters on a Fixed screen are prevented. In addition, if the available space for placing a field is less than the defined length of the attribute, then that field is truncated.

When control is initially created, the size is equal to the corresponding attribute's length. When you increase the control's size, the corresponding attribute's length is adjusted accordingly. When the control's size is reduced, the corresponding attribute's length is not adjusted.

You can set the properties of the data entry fields on a character screen to CR, DR, +, and -.

To create a Character mode screen, perform the following:

1. Follow the instructions in Adding New Items.
2. Select an element and provide a name for it.
3. Once the element has been added, select it from Solution Explorer or Class view.
4. Click the **Painter** tab to open the form.
5. Select **Fixed** from the drop-down list above the form.
6. Add the required objects to design the interface.

Label Object Properties in Fixed Screens

Add a Label as a static label object used for entering static text to the Character screen or as a dynamic label control. To display the value of an attribute in a Character screen at runtime, the following properties can be selected for each label object.

Property	Function
BackgroundColor	Specifies a background color for the object. Select a color from the drop-down list. Note that you can specify only one of the following colors for character screens: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta.
Big	Specifies whether the selected object is printed in large characters; that are in pitch 66, about 50 percent taller than normal. Not displayed at design time, but only at runtime. ***Note:*** *The changes you apply to this property do not affect the display of a report.*
Blink	Specifies whether to display the selected object in blinking video.
Box	Specifies whether to surround the selected item with a box.
Bright	Specifies whether the selected object is highlighted. ***Note:*** *If two controls overlap on setting this property, then an overlap error is reported. Property remains False.*
ControlType	Specifies the type of object selected. This property is read-only from within the Painter document window and is always a Label.
Foreground Color	Specifies a foreground color for the object. Select a color from the drop-down list. Note that you can specify a custom color.
Id	Read-only. Generated by the software.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Reset	Specifies that all current highlighting and color attributes are reset after the selected object.
Reverse	Specifies whether the selected object is displayed with reverse video highlighting.
Underline	Specifies whether the selected object is underlined. Not displayed at design time, but only at runtime.
Underscore	Specifies whether the selected object is displayed with a horizontal delimiter beneath. Not displayed at design time, but only at runtime.
Upperscore	Specifies whether the selected object is displayed with a horizontal delimiter above. Not displayed at design time, but only at runtime.

Text Field Object Properties in Fixed Screens

When you want to accept one line of information from a user, add a one-line Text Field to your form. To display the value of an attribute in a Character screen at runtime, the following properties can be selected for each Text Field object.

Property	Function
BackgroundColor	Specifies a background color for the object. Select a color from the drop-down list. Note that you can specify only one of the following colors for character screens: Black, White, Red, Green, Blue, Yellow, Cyan, and Magenta.
Big	Specifies whether the selected object is printed in large characters; that are in pitch 66, about 50 percent taller than normal. Not displayed at design time, only at runtime. ***Note:*** *The changes you apply to this property do not affect the display of a report.*
Blink	Specifies whether to display the selected object in blinking video.
Box	Specifies whether to surround the selected item with a box.
Bright	Specifies whether the selected object is highlighted. ***Note:*** *If two controls overlap on setting this property, then an overlap error is reported. Property remains False.*
ControlType	Specifies the type of object selected. This property is read-only from within the Painter document window and is always a TextField.
Foreground Color	Specifies a foreground color for the object. Select a color from the drop-down list. Note that you can specify a custom color.
Id	Read-only. Generated by the software.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
Name	Specifies an attribute to which the object is bound. This property is read-only from within the Painter document window, though it can be renamed from any other window that displays existing attributes.
Reset	Specifies that all current highlighting and color attributes are reset after the selected object.
Reverse	Specifies whether the selected object is displayed with reverse video highlighting.
TabIndex	Specifies the index that defines the tab order for the object.
Underline	Specifies whether the selected object is underlined. Not displayed at design time, but only at runtime.
Underscore	Specifies whether the selected object is displayed with a horizontal delimiter beneath. Not displayed at design time, but only at runtime.
Upperscore	Specifies whether the selected object is displayed with a horizontal delimiter above. Not displayed at design time, but only at runtime.

Creating Teach Screens

A Teach Screen in a system displays help information, which you create for your user, about an element. Many teach screens can be associated with a single element.

Use the Painter to enter the text for a teach screen. The teach screen painter allows you to define teach screens and translate them into any available language. When the Painter is used to create teach screens, it provides a subset of the functionality provided by the screen painter.

All controls from the Tool Box can be dropped onto a teach screen, and all painter operations can be performed on the objects.

Only the following exceptions exist:

- No object can be bound to an attribute.
- Attributes cannot be dragged and dropped onto a teach screen.
- Attributes cannot be created when a teach screen is saved.

To add a teach screen, perform the following:

1. Follow the instructions in Adding New Items.
2. Select a Teach Screen class and provide a name for it.
3. Once the teach screen has been added, select it from Solution Explorer or Class view.
4. Click the **Painter** tab to open the Painter window and begin adding objects.

Designing User Interfaces for Client Framework Applications

Before creating an interface for AB Suite Client Framework applications, you must add an IGraphicalPresentation interface node to an ispec or a class. This ispec or class is exposed as a set of corresponding DataModels that can then be used by various popular client development tools to create desktop, web, mobile, and service based applications. When the IGraphicalPresentation interface node is added to the ispec or the class, the PresentationType property of the ispec or class is set to Graphical and is read-only. You can add attributes or drag attributes from an ispec definition to the ispec or class with the IGraphicalPresentation interface node. The DataViewModel definitions and an empty XAML View appears for each ispec or class, which has an IGraphicalPresentation interface node, when you generate the necessary Access Layer projects from the Build menu. For WPF/XAML client type, the Data Sources project is also generated. You can now design the XAML View for the ispec or class that has the IGraphicalPresentation interface node.

Note: *The Data Sources are created in the <ApplicationName.TechnologyFolderName>.Views project.*

You can now design the user interface by using the Microsoft WPF Designer in Visual Studio. To design the user interface open the Solution Explorer window, and then double-click the <IspecName or ClassName>.Xaml file in the Stereotyped or Classes folder.

Note: *You can design the user interface by using Microsoft WPF Designer in Visual Studio if you have set the Client Technology option to WPF.*

This opens the View in the WPF Designer window along with the Common WPF Controls Toolbox and the All WPF Controls Toolbox.

You can now add attributes to the View from the corresponding Data Source or add controls from the Toolbox to design Views for a specified client technology. To access the Data Sources window, on the Views menu, point to **Other Windows**, and then click **Data Sources**.

Note: *Always ensure that the View matches the corresponding Data Source from where you add the attributes.*

Refer to Client Framework - Working with the WPF/XAML Designer Using Data Sources in the Documentation Libraries page on the Product Support site for more information on designing the Views for a WPF application.

Manipulating Graphical User Controls

The following points highlight the usage of WPF Designer interface and manipulation of various graphical user controls:

- The default view of WPF Designer interface contains a user control, a top-level grid control, and a canvas control. The WPF Designer shows a split view with the design view at the top and the XAML window at the bottom. The XAML window can be collapsed if required. The canvas control is the container for all other controls placed on the view. To increase the size of the design surface, you must adjust the size of the user control. You can select the user control from the Document Outline tab, and then drag the handles on the sides or the corner of the control to the desired size. Alternatively, you can set the Height and Width properties of the user control in the XAML. You can search any control by using the Document Outline tab. The Document Outline tab displays a hierarchy of all the user controls for a selected <IspecName or ClassName>.Xaml.
- The WPF graphical user controls that are added to the View from a Data Source appears in a composite grid control containing a label and a user control, which is automatically bound to an attribute in ViewModel. If you do not need the label automatically created, you can drag the user control outside the grid and then delete the grid containing the label. You can change the position of a control, or multiple controls, incrementally by pushing the selected object with the Up, Down, Left, or Right arrow keys. Select one or more controls and press the appropriate arrow key.
- To associate a control with a grid, select the control and drag it onto the grid. A blue highlighted text, “Press Alt to place inside [Grid]” appears on the interface. Now, you can press the Alt key and the control is associated with the new grid.
- You can make one or more graphical controls invisible on a view at runtime. To do this, set the **Visibility** property of a user control to Hidden or Collapsed.

- The Properties Editor in the WPF Designer can be used to configure the properties ofa user control. In some instances, you might need to access the advanced options fora specific property, such as DataBinding. These advanced properties can be identifiedby a small square toward the right of the property. You can click this square to select the required option from the context menu.
- You can have some classes that are aggregated into an ispec. This enables the creation of separate ViewModels and Views for these classes when the Access Layer is built. So, you can design these classes separately and reuse them by inserting as user controls in the Views for other ispecs.
- An image control can be optionally dynamic and a Convertor is required to resolve the location and type of the image control for dynamic controls.
- The user controls that are grouped together to represent one attribute are called Group User Controls. The graphical user controls, such as button, checkbox, and radio button are called Group User Controls. All buttons in a group user control can be associated with an attribute. For example, check boxes can be used for selecting answers to multiple choice questions.
- Combo box and list box controls can be bound to a ListModel that is generated from a List attribute in System Modeler. The ListModel is created when the Access Layer is generated and is available as a ViewModel in the ViewModels project. You can then bind the ItemsSource property of the combo box or list box to the ListModel. At runtime, the list sent from the AB Suite system is displayed in the combo box or list box.
- All Copy.From fields are generated as a CopyFromItems collection that can be bound to a DataGrid control type, so you do not need to place each copied field separately on the View.

Creating Reports

The Painter is not only used for creating screen interfaces and Teach Screens, but is also used for creating reports and report frames. A report is part of a system that is generated and used to produce output or to carry out specialized batch processing of a Database. It consists of report frames and report methods, and a number of options that define the output of the report.

To add a report, perform the following:

1. Follow the instructions in Adding New Items.
2. Select a Report class and provide a name for it.

To add a report frame, perform the following:

1. Follow the instructions in Adding New Items.
2. Select a Frame class and provide a name for it.
3. Once the frame has been added select it from Solution Explorer or Class view.
4. Set the **Multiplicity** property to 1.

5. Set the **PresentationType** property to Print to create a **Painter** tab on the document window.
6. Click the **Painter** tab to open the Painter window and begin adding labels.

Report Options

Global options for the text font size, type, and foreground color can be set for all reports from the **Tools**, **Options** menu. Refer to Document Windows in System Modeler for more information on how to do this. The Form Grid setting is based on the font size either selected from the Tools, Option setting or overridden at the Frame level from the Report Frame Properties window.

Report Frame Properties

To set the properties of a Report Frame using the Properties window, perform the following:

1. On the Class view, select the Report Frame.
2. On the **View** menu, click **Properties** to open the Properties window.
3. Select the property that you want to specify and select it from, or type it in, the edit box.

The following properties are available for the Report Frame.

Property	Function
Foreground Color	Specifies a foreground color for the object. Select a color from the drop-down list.
Line Length	Specifies the line length for a report, which determines the Report page size. By default, it is 132 characters in width with a minimum of 80 and a maximum of 260. The height is set at 100 characters and cannot be change. ***Note:*** *The changes you apply to this property affect the display of a report. To get a desired output in a printed report, you can configure the print properties by using AB Suite Runtime Administration Tool.* Refer to the *Agile Business Suite Runtime for Windows®Operating System Administration Guide* for more informatioin.
Show Grid	Specifies whether forms display the sizing grid. By default, the grid is turned on.
Snap to Grid	Specifies whether forms snap objects to the grid.

Add Labels to a Report Frame

Only a limited amount of painter functionality is available for report frames, as it simply provides the functionality to use a character-based font (fixed fonts). The only object which can be added to a report frame is a Label, whose behaviors depend on the methods used to add the label to the form.

To drag a label from the Toolbox and drop it onto a form, perform the following:

1. Click the **Painter** tab available at the bottom of the document window.

   The Form appears.
2. If it is not already open, click the Class View window.

   The Class View window appears.
3. Click the Class View menu item.
4. Sort the hierarchical list into the most appropriate form to locate the attribute you wish to use.
5. If necessary, click the plus (+) symbol next to the appropriate node.
6. Select the attribute that you wish to assign to a label, and drag it onto the form.
7. Drop the attribute onto the form to create a default dynamic Label object.

When adding a Label in this method, the Label is added as a static label object, and is used for entering static text into the report.

Note: *You can specify a qualified object name as the value to make this object a dynamic Label object.*

To drag an attribute from the Class View window and drop it onto a form, perform the following:

1. Open the form by clicking the **Painter** tab available at the bottom of the document window.
2. If it is not already open, open the Class View window by clicking the **View**, **Class View** menu item.
3. Sort the hierarchical list into the most appropriate form to locate the attribute you wish to use.
4. If necessary, click the plus (+) symbol next to the appropriate node.
5. Select the attribute you wish to assign to a label, and drag it onto the form.
6. Drop the attribute onto the form by releasing the mouse button to create a default dynamic Label object.

When dropping an attribute from the Class view onto a report frame, a dynamic Label object is added to the report. A dynamic label control is used to specify where to insert the value of an attribute in the report at runtime.

Note: *This differs from dragging an attribute onto a form in the screen painter, where it adds a Text Box.*

Label Properties in Report Frames

Add a Label as a static label object that is used for entering static text to the report, or as a dynamic label control. To display the value of an attribute in the report at runtime, the following properties can be selected for each label object.

Property	Function
Big	Specifies whether the selected object is printed in large characters; that is in pitch 66, about 50 percent taller than normal. Not displayed at design time, but only at runtime. ***Note:*** *The changes you apply to this property do not affect the display of a report.*
BlankWhenZero	Specifies whether spaces are to be printed if the value of the Frame Attribute is zero. Options are True and False.
Bright	Specifies whether the selected object is printed with highlighting. ***Note:*** *If two controls overlap on setting this property, then an overlap error is reported. Property remains False.*
Control Codes	Specifies output control codes into your model. Not displayed at design time, but only at runtime. Click the ellipses button to open the **Control Codes** dialog box.
ControlType	Specifies the type of object selected. This property is read-only from within the Painter document window and is always a Label.
Foreground Color	Specifies a foreground color for the object. Select a color from the drop-down list.
Id	Read-only. Generated by the software.
IsFloatingSign	Specifies whether the sign is to be printed immediately before the first significant digit. Options are True and False.
Location	Specifies the point in pixels that represents the upper-left corner of the object relative to the upper-left corner of its container as an X and Y location.
(Name)	Specifies the type of object. This property is read-only from within the Painter document window and is either Static Label or Dynamic Label.
Pitch	Specifies the number of characters to be printed per line for the selected object. ***Note:*** *The changes you apply to this property do not affect the display of a report.*
PrintFormat	Specifies how to present numbers and zeros in reports. Values are Blank, CR, -, +, $, *, and None. Refer to the PrintFormat Property for more information on these values.
Reset	Specifies that all current highlighting and color attributes are terminated after the selected object.
Reverse	Specifies whether the selected object is printed with reverse video highlighting.

Property	Function
Underline	Specifies whether the selected object is underlined. Not displayed at design time, but only at runtime.
Underscore	Specifies whether the selected object appears and print with a horizontal delimiter beneath. Not displayed at design time, but only at runtime.
Upperscore	Specifies whether the selected object is displayed and printed with a horizontal delimiter above. Not displayed at design time, but only at runtime.
Value	Specifies the value to be displayed in the object. Note that you can specify a qualified object name to make this object a dynamic Label object.

PrintFormat Property

The PrintFormat property is only available for a Label that is bound to an attribute with a Primitive Type of Number or Signed Number. The following table describes the values and identifies the Primitive Type to which they are applicable:

Value	Description	Primitive
Blank	Print nothing if value is zero	Signed Number
CR	Signed (CR or DR) numeric, default CR	Signed Number
-	Signed (CR or DR) numeric, default -	Signed Number
+	Signed (CR or DR) numeric, default +	Signed Number
$	Signed numeric, print nothing if the value is zero, print floating $ if non zero	Signed Number
*	Signed numeric, print asterisks if the value is zero	Signed Number
None	No formatting is applied	Number
DR	Defaults to DR	Signed Number
Z	Suppress leading zeros, print 0 if the value is zero	Number
X	Print nothing if the value is zero	Number

Control Codes Dialog Box

Use the Control Codes dialog box to add Output Control codes into your model. The control codes are used to associate user-defined highlighting characteristics for report data or display items that are directed to terminal printers, video devices, or ROC remote printers.

Note: *Control Codes are ignored if the Default Device for your Report is an Enterprise Output Manager generated Report.*

To enter or change control codes, perform the following:

1. Enter the required code in the **Control Code** edit box.
2. Click **Insert** to add the control code to the list.
3. To add additional control codes repeat steps 1 and 2. Each additional control code is added to the end of the list.
4. To change an existing control code, select the code from the list to display it in the edit box.
5. Enter the required changes.
6. Click **Change**.

To delete a control code, select the code in the list and click **Delete**.

Add Logic to Report Frames

Logic is added to report frames from the Agile Business Suite Developer Logic Editor. You can enter the logic by selecting and opening the method for a report frame in a Logic Editor Window from the Class View.

Using the Windows Communication Foundation (WCF) Gateway

The AB Suite Client Framework WCF Gateway acts as a single access point for different client technologies. It allows remote clients, such as .NET, Silverlight, Windows 8 (WinRT) Store Apps, and others, that do not support direct COM/DCOM connectivity to communicate with the runtime system by using the Client Framework access layer infrastructure. The WCF gateway exposes a WCF service that allows client applications to communicate with the AB Suite runtime system over the network by using Windows Communication Foundation.

The WCF Gateway is installed as a service with the AB Suite Runtime software. It can also be executed within a self-hosted application named Client Framework WCF Gateway Host, which is supplied with the AB Suite Developer software. This allows developers to use the WCF Gateway service on a machine that does not have the AB Suite Runtime software installed. It exposes the same service on a development machine so that you can develop and test your client applications.

The WCF Gateway exposes an endpoint named AccessServiceEndPoint. This endpoint implements the IGateway interface that offers a full-duplex asynchronous pattern allowing you to run colon commands, report, and receive unsolicited messages from the runtime system. In conjunction with the Remote Access Layer API, you can use this WCF service to implement rich client applications by using technologies that support a full-duplex WCF binding. For example, .NET Framework applications, Windows Store apps, Silverlight, and others.

To interface with the Gateway service you must use the Client Framework Remote connector assemblies that expose the same Access Layer API as the direct connect assemblies. You can install the Remote connector assemblies by using the Client Framework Remote Connect NuGet package.

FileStoreGateway Service

The Client Framework also provides a WCF Service named FileStoreGateway. It exposes an endpoint named AccessServiceFileStoreEndPoint that implements the IFileStoreService interface. This allows the remote client application to download files from a file repository on the server where the FileStoreGateway Service is running. You can also upload files if you have the required permissions.

The FileStoreGateway Service exposes a more basic interface than the Gateway Service and does not require a duplex channel for its operation. The FileStoreGateway can be installed as a service with the AB Suite Runtime software or can be hosted by the WCF Gateway Host application on a development machine for testing purpose.

Running the WCF Gateway

To host the WCF Gateway on a machine with AB Suite Developer installed, perform the following:

- Point to the bottom-left corner of the screen to enable the Start icon, click **Start** > **Apps** > **Agile Business Suite 6.1** > **Development Environment** > **Client Framework WCF Gateway Host**.

  This starts the WCF Gateway Service Host application and displays the following services in the Hosted Service pane:

  - Gateway – for connecting to the Runtime system
  - FileStoreGateway – for downloading and uploading files

  Refer to the *Agile Business Suite Installation and Configuration Guide* for more information on the Client Framework WCF Gateway Host application.

When the gateways start, the Gateway Server Console and Service Host window displays the machine details and the endpoint addresses in the Trace Output pane.

To connect a client with this gateway, perform the following:

1. Create a client application that you want to connect to the gateway.

   Note: *You can create a console application, WPF application, Windows 8 apps, Windows Service application, or others as required.*

   As an example, we can use a WPF application to reference the Portable DataModels by using the Remote Access Layer API and connect to the WCF Gateway with the Remote.DotNET connection.

2. Add the following references to your project:

   - Your portable DataModels assembly generated from the AB Suite solution for the specified technology folder (for example, Sample.Remote.DataModels.Portable.dll)
   - The Gateway Service Interface assembly (Unisys.ABSuite.AccessLayer.GatewayServiceInterface.dll) that defines the Service Contract with the WCF Gateway.
   - The Remote Core assembly (Unisys.ABSuite.AccessLayer.Connector.Remote.Core.dll) that implements the Access Layer API for remote connections through the WCF Gateway.
   - The Connector assembly for establishing connections to the WCF Gateway from a Windows application, such as WPF (Unisys.ABSuite.AccessLayer.Connector.Remote.DotNet.dll).

3. In your Connection processing, create a Remote connection instance by using the ABSuite.AccessLayer.Remote.DotNet.Connection class, by passing an IspecFactory instance for your Portable DataModels.

```
ABSuite.AccessLayer.Connector.Remote.Core.Connection RemoteConnection = new
            ABSuite.AccessLayer.Connector.Remote.DotNet.Connection(new
                Sample.Remote.DataModels.Portable.Core.IspecFactory());
```

   Note: *In this example, we are using a WPF application that uses the full .NET Framework. Hence, we can employ the Connection class in the Unisys.ABSuite.AccessLayer.Connector.Remote.DotNet assembly. If you were developing a Windows 8 Store application, then you must use the Connector class in the Unisys.ABSuite.AccessLayer.Connector.Remote.Windows assembly.*

4. Set up a ConnectionDetails object.

```
ConnectionDetails cDetails = new ConnectionDetails()
{
  // The name or IP Address of the host machine the
  // WCF Gateway will connect to:
        Host = "localhost",
  // Is it an anonymous connection?
       IsAnonymous = true,
  // The name of the deployed runtime system (usually the
  // segment name)
      System = "Sample",
  // The name or IP Address of the machine where the WCF
  // Gateway is running
      GateWayAddress="localhost",
 // The location where the .NET DataModels can be found for
 // the WCF Gateway to use.
 // This must be a folder that is accessible by the WCF
 // Gateway process on the machine where the WCF Gateway is
 // running.
     DownLoadURI = @"C:\Gateway Applications\Sample\Access Layer API Deploy",
 // This is a list of assemblies that the WCF Gateway needs to
 // process the transactions for a particular application. In
 // this case, the .NET DataModels are specified
DownLoadFiles = new string[] { "Sample.Remote.DataModels.dll" },
// Do we want to override any existing connections for the
// same user?
  ForceLogin = true
};
```

Once the ConnectionDetails is established, you can pass this to the Connect() method of your Remote Connector instance

```
TransmissionObject trObj = null;
trObj = RemoteConnection.Connect(cDetails, null);

if (trObj != null && (trObj.State == TransmissionReturnCode.Ok ||
trObj.State == TransmissionReturnCode.OkWithSwitch))
{
    CurrentIspecName = trObj.ObjectClassName;
    CurrentTO = trObj;
    SessionConnected = true;
}
else
SessionConnected = false;
```

5. The Remote Connector establishes a full-duplex TCP/IP channel to the WCF Gateway and makes requests to the Gateway by using the Access Layer API Remote interface that exposes the same interface as the Direct Connect approach. The difference here is that the messages are being passed through an intermediary gateway by using a WCF endpoint.

Custom Gateway

You can create your own custom WCF gateway by using the WCF Service Library option in the <Technology Folder Name> Property Pages window. This builds a basic WCF Service that can transmit requests and receive responses in a synchronous manner. It does not expose the full Access Layer API. Therefore, it does not implement callbacks, nor does it require a full-duplex asynchronous session.

The WCF Service Library technology option generates two projects

- A WCF Service library project that defines a WCF service. The WCF service can be exposed through various hosting options, such as an IIS Web application, a Windows Service, or a Windows Self-Hosting Service.
- A WCF Service Gateway project that allow you to host the Service Library.

Note: *If you do not want to use this Service Gateway, then you can choose to host the Service Library in the Service Gateway supplied with the AB Suite software.*

Due to its relatively simple interface, this WCF Service is ideal for use by remote clients where callbacks and duplex channels cannot be used (for example, mobile devices, Java, Web Service client, and others). It does not allow you to execute colon commands and other asynchronous operations, such as running reports or receiving unsolicited messages. However, with these client types, such operations are typically not required.

To create a custom service library, perform the following:

1. Create an AB Suite Client Framework application.
2. Right-click the Technology folder, and then select **Properties**.

   The <Technology Folder Name>Property Pages window appears.
3. From the **Client Technology** list, select **WCF Library (Windows Communication Foundation)**.

4. Click **OK**.

   This creates two projects in the solution

   - The WCF Service library project named <SegmentName>.<FolderName>Library (for example, Sample.MyWCFServiceLibrary).

     This project creates the WCF Service Library that can be hosted.

   - A default WCF Hosting project named <SegmentName>.<Foldername>Gateway (for example, Sample.MyWCFServiceGateway).

     This project creates hosting application that hosts your service library. It can be run as an application or can be installed as a Windows Service.

   Refer to the *Agile Business Suite Installation and Configuration Guide* for more information on installing the WCF Hosting application as a Windows Service.

5. In the Solution Explorer window, right-click the **<Project Name>.<FolderName>Library** project, and then select **Start New Instance**.

   The WCF Service Host application starts and displays the generated service being hosted (for example, Sample.MyWCFServiceLibrary.SampleMyWCFService). The Microsoft WCF Test Client window also appears, which allows you to discover the hosted service and perform operations on it for testing purposes.

To connect a client with this gateway, perform the following:

1. Create an application in Visual Studio.

   Note: *You can create a console application, WPF application, Windows Service application, Windows Store application, or others as required.*

2. Right-click the project, point to **Add**, and then click **Service Reference** from the context menu.

   The **Add Services References** dialog box appears.

3. In the **Address** box, enter the endpoint address. You can either discover the endpoint address or copy it from the left pane of the WCF Test Client window

4. Click **Go**.

   The methods available as part of the endpoint are listed in the Operations box.

5. In the **Namespace** box, enter a name; for example, SampleRef.

6. Click **OK** to close the Add Services References dialog box.

   The name you entered in the Namespace box appears under the Service Reference folder in Solution Explorer window.

7. Double-click the Service Reference to open the Service definition in the Object Browser window

8. Add code in your application class to create and use the service. For example, the following code is executed when a button is pressed in a WPF Client application:

```
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        // Create an instance of the Service
SampleMyWCFClient svc = new SampleMyWCFClient("WSHttpBinding_ISampleWcfService");
```

or

```
SampleMyWCFClient svc = new SampleMyWCFClient("BasicHttpBinding_ISampleWcfService");
            MessageData messageData;
         // Perform the Connect operation on the service
            messageData = svc.Connect();

         // The fireup ispec is a type of MENUmodel
            MENUModel menu = (MENUModel) messageData.Data;
            menu.ACTION2 = "PROD"; // Navigate to the PROD ispec

         // Transmit the MENUModel
            messageData = svc.Transmit(menu);
            PRODModel prod = (PRODModel)messageData.Data;
            prod._UserMAINT = "FIR"; // Set the UserMaint field

         // Tranmit the PRODModel
            messageData = svc.Transmit(prod);
            prod = (PRODModel)messageData.Data;

          // Extract the Name information for the first record
             string name = prod.NAM;
          // Check the Status information
             string status = messageData.Status;
       }
```

In your logic, you can call the operations that the service provides.

In the example above

- The service is created.
- (WSHttpBinding_ISampleWcfService) and (BasicHttpBinding_ISampleWcfService) are two endpoints for service reference. The interface reference, ISampleWcfService is I<SegmentName><webservice foldername>.
- The Connect() operation is performed that returns the fireup ispec.
- The MENUModel is populated and sent using the Transmit() operation.
- The PROD ispec is returned and the PRODModel is populated.
- The PRODModel is sent using the Transmit() operation and the data is extracted from the information returned.
- The Status message is also extracted from the returned MessageData.

Debugging Applications

You can debug your application logic interpretively using Debugger in Agile Business Suite System Modeler. Debugger performs JIT (Just-In-Time) compilation of the components being debugged (and other required components), and thus does not require a full generate of the application. JIT compilation is only for Winforms and WPF Containers. The reason a full generate is not required is because most things are never generated – they are interpreted. The only things that are generated are interfaces to mimic the Windows® runtime interface for the system so that the Windows infrastructure can provide services to the Debugger. Debugger also allows the components being debugged to be run against a test database, rather than your production database. In this way, Debugger helps you to rapidly develop robust applications that can be tested in a controlled manner.

Note: *You cannot debug parts of an application that contain Unresolved elements. Unresolved elements must be resolved before debugging.*

To use Debugger, you must first specify a configuration to run, and then set the configuration to Debug Mode.

Specify a configuration using the Configuration Manager. Refer to Configuration Manager Dialog Box in the *Microsoft Visual Studio Online Help* for more information on using the Configuration Manager.

To set a configuration to Debug Mode, perform the following:

1. Right-click the project in Class View and select **Properties**.
2. Set the Debug Mode configuration property to the type of debug session you want to run:
    - Online System

      An Online System debug session runs a segment through a Windows Forms interface for an AB Suite application. It also runs a client application through a Designer interface for an AB Suite Client Framework application.
    - Report Call

      A report debug session runs a report.
    - Method Call

      A method call debug session runs a public method of a segment that would be exposed through the generated interface.
    - External Application

      An external application debug session runs methods on classes that would be exposed by the Windows generated system, as they are called from an external application. This can be used for scripting a test run.

      When the Debug Mode configuration property is set to Off, none of the Debugger settings are available.

      Note: *The External Application option is not available when you debug an AB Suite Client Framework application, as the Online System option runs the external client applications supported by Client Framework.*
    - Messenger

      A Messenger debug session runs a segment through a Messenger cycle interface for an AB Suite application. The Messenger Client program starts and allows you to submit XML messages for processing by Messenger classes.
3. Specify other configuration properties as required.

Note: *The Debug Mode configuration properties are stored in the Visual Studio Solution files and not the database. Hence, these properties cannot be exported.*

Accessing an Existing Runtime Database to Debug an AB Suite Application

Existing runtime database can be used for AB Suite debug session in the following ways:

- Local
- Remote Server (Distributed environment)
- Backup / Restored Runtime Data base

To access locally deployed runtime database, perform the following:

1. In the Administration Tool on the Developer workstation, create a Database Server Registration to LOCAL HOST.
2. Right-click on the Database Server Registration, and then select **All Tasks** > **Attach** > **Existing Database**.
3. Select the Runtime Development Database, and then click **OK**.
4. Setup the Debug Configuration to specify the Database Server Registration and Database Name under Test Database. Set **Reorganise Database** to NO.
5. Perform a build for the Debug Configuration:

   It adds the Debugger System name under the Database Name in the Administration Tool.

To access remotely deployed runtime database, perform the following:

1. In the Administration Tool on the Development Runtime Server, right-click the Dev AB Suite Runtime System, and select **Change DB Password**. Set the DB Password, and then select **Apply this password change to the schema**.
2. In the Administration Tool on the Developer workstation, create a Database Server Registration to the AB Suite Development Runtime Server.
3. Right-click the Database Server Registration, and then select **All Tasks** > A**ttach Existing Database.**
4. Select the Runtime Development Database, and then click **OK**.
5. Setup the Debug Configuration to specify the Database Server Registration and Database Name under Test Database. Set **Reorganise Database** to NO.
6. Perform a build for the Debug Configuration.

   It adds the Debugger System name under the Database Name in the Administration Tool.
7. In the Administration Tool, right-click the Debugger System Name, and select**Change DB Password**. Set the DB Password to the same value set in Step 1. Deselect **Apply this password change to the schema**, and then click **OK**.

   You can run the Debugger session, and it connects to the shared Development Runtime Database.

To use backup or restored runtime database for debug session, perform the following:

1. In case, the backup or restored runtime database using another name is too huge it is required to set **Reorganise Database Flag** to YES.
2. Set the same **Database Schema Name** as the one initially used while deploying the runtime database for each involved user or configuration by using the debugger for restored runtime database.

Debugger Configuration Properties

The following tables list configuration properties that affect debugging.

- Client Properties
- Misc Properties
- Start Options Properties
- Test Database Properties
- Test Report Output

Refer to Running a Debug Session for more information on using these properties.

Client Properties

Property	Values	Function
Application To Start	Application path	Specifies the external application to run. This property is only available if Debug Mode is set to Online System.
Command Line Arguments	String literal	Specifies the command line arguments to run the specified application. This property is only available if Debug Mode is set to Online System.
Working Directory	Directory path	Specifies the working directory to run the specified application. This property is only available if Debug Mode is set to Online System.

Property	Values	Function
Client Package Folder	Path to Client Application folder	Specifies the Client Technology folder created for the AB Suite Client Framework application. This property is only available if Debug Mode is set to Online System. ***Note:*** *If you enter an invalid value, in the Client Package Folder field, an alert message "Property value is not valid" appears. When you click Details in the alert message box the following message appears:* *"The name <Invalid Value Entered> could not be resolved as a valid Client Package Folder for <Segment Name>. Please enter the full qualified name or use the element picker dialog window for selecting a valid item".*

Note: *The Client Properties are available only when you debug an AB Suite Client Framework application.*

Misc Properties

Property	Values	Function
Debug Mode	{Off, Online System, Report, Method Call, External Application, Messenger}	Specifies whether to debug this configuration, and if so, the type of debug session to run.
Value of Glb.Machine	Host	Specifies the type of host on which an application is running. If you specify As Per Host the host is set to A for MCP or the host is be set to N for Windows (NT). If you specify P for Debugger the host is set to P.

Start Options Properties

Property	Values	Function
Application To Start	Application path	Specifies the executable file to run. This property is only available if Debug Mode is set to External Application. ***Note:*** *This property is not available in Start Options properties, when you debug an AB Suite Client Framework application.*

Property	Values	Function
Segment To Debug	Qualifier	Specifies the segment to run if Debug Mode is set to Online System. This property is only available if Debug Mode is set to Online System. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Command Line Arguments	String literal	Specifies the command line arguments to run the specified application with. This property is only available if Debug Mode is set to External Application. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Deployment Folder	Qualifier	Specifies the Builder configuration settings folder. This property is not available if Debug Mode is set to Off.
File Name	Qualifier	Specifies the file containing the data to be sent to the Messenger when starting the debug session. This property is only available if Debug Mode is set to “Messenger”.
Language	Qualifier	Specifies the defined language (one of a set of defined languages in a multiple language environment) to run. This property is not available if Debug Mode is set to Off.
Messenger To Debug	Qualifier	Specifies the Messenger to run if Debug Mode is set to Messenger. If this property is left blank you can submit XML messages to any Messenger class. This property is only available if Debug Mode is set to “Messenger”.
Method To Debug	Qualifier	Specifies the public method to run. This property is only available if Debug Mode is set to Method Call.
Parameter Values	Comma-delimited list of parameters	Specifies the parameter values to pass to the specified method. This property is only available if Debug Mode is set to Method Call.

Property	Values	Function
Report To Debug	Qualifier	Specifies the report to run if Debug Mode is set to Report. This property is only available if Debug Mode is set to Report.
Report Parameter	String literal	Specifies the parameter to pass to a report. This property is only available if Debug Mode is set to Report.
Working Directory	Directory path	Specifies the working directory to run the specified application with. This property is only available if Debug Mode is set to External Application. ***Note:*** *This property is not available in Start Options properties, when you debug an AB Suite Client Framework application.*
Client Language Mode	Language Mode	Specifies the type of client language to use. Specify Locale if you want to simulate MCP host behavior using SQL Database. Specify MCP Slot for MCP host application. ***Note:*** *This property is only available if Debug Mode is set to External Application for MCP Configuration, and is not available when you debug an AB Suite Client Framework application.*

Test Database Properties

Property	Values	Function
Alternate Name	String literal	Set a unique value for the Alternate Name segment property, which have not been used in any other debug configuration. This property value overrides the segment alternate name property.
Database Name	String literal	Specifies the name of the database to run the debug session against. Refer to the *Administration Tool Online Help* for more information on running against a database. This property is not available if Debug Mode is set to Off.
Database Server Registration	String literal	Specifies the database server registration of the server hosting the database to run the debug session against. Refer to the *Administration Tool Online Help* for more information on running againts a database. This property is not available if Debug Mode is set to Off.

Property	Values	Function
Enable Host Database Access	Boolean	Specifies whether to access the host database or not. By default, the value is False. This property is only available for MCP configuration. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Enable User Maintained Tables	Boolean	Specifies whether to enable User Maintained Tables for the debugger database or not, if the structure is configured as User Maintained. By default, the value is False. This property is only available for Windows configuration.
Host Name	String literal	Specifies MCP host details. This property is only available for MCP Configuration. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Password	String literal	Specifies MCP user code password. This property is only available for MCP configuration. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Port Number	Numeral	Specifies HDBA port number. By default, the value is 1871. This property is only available for MCP configuration. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*
Allow Recovery from Failed Reorganization	{Yes, No}	Specifies whether a backup of the database must occur before the reorganization. By default, this property is set to 'Yes'. If this property is set to 'Yes' the database is restored on failure of the database reorganization. ***Note:*** *It is recommended to set this property to 'Yes' as a failure may lead to the non-usability of the database. This property is not available if Debug Mode is set to 'Off'.*
Reorganize Database	{Yes,No}	Specifies whether the database is reorganized when starting the debug session. If this property is set to No, and the model does not match the schema of the physical database, logical reorganization occurs. This property is not available if Debug Mode is set to Off.

Property	Values	Function
User Name	String literal	Specifies MCP user name. This property is only available for MCP configuration. ***Note:*** *This property is not available when you debug an AB Suite Client Framework application.*

Refer to Test Database Issues for more information on Debugger interaction with databases.

Test Report Output

Property	Values	Function
Output Location	(Output Window, Use ROC)	Specifies the location of the report output. By default, the value is the Output window. However, users who have installed the Windows® Runtime and the ROC system for Windows can also set the output location as Use ROC. The Use ROC option is not available for MCP configurations, and the report is always sent to the Output Window.

Running a Debug Session

To initiate a debug session for an AB Suite application, perform the following:

1. Select the desired configuration.
2. Select **Start Debugging** from the **Debug** menu or **Start** from the toolbar.

To initiate a debug session for an AB Suite Client Framework application, perform the following:

1. Select the desired configuration.
2. Select **Start Debugging** from the **Debug** menu or **Start** from the toolbar.

   Note: *Ensure that you set the AB Suite model project as the Startup project before starting the Debug session. To set the AB Suite model project as the Startup project, you can right-click the AB Suite model project in the Solution Explorer window, and then select **Set as Startup Project**.*

   Alternatively, you can right-click the AB Suite model project in the Solution Explorer window, point to **Debug**, and then select **Start New Instance**.

If the **Reorganize Database** property of the configuration is set to Yes, a warning message is displayed, allowing you to continue the debug session with or without reorganizing the database as required. If the Reorganize Database property is set to No, the debug session is started without reorganizing the database. Refer to Logical Reorganization for more information on debug session behavior when the database is not reorganized.

According to the **Debug Mode** property of the configuration:

- If Debug Mode is set to Online System, the initial user interface is invoked in the language specified by the **Language** property. Refer to Multiple-Language Environments for more information on Debugger language use.

  The initial user interface is specified by the **Class To Debug** property of the configuration. If it is set to a segment, it is the user interface of the ispec specified by the **Fireup** property of that segment. If the Fireup property of the segment is not specified, a message is displayed, allowing you to select the initial ispec user interface.
- If Debug Mode is set to Report, the report specified in the **Report To Debug** property is run with a parameter specified in the **Report Parameter** property. The report is invoked in the language specified by the **Language** property.
- If Debug Mode is set to Method Call, the qualified public segment method specified by the **Method To Debug** property is called, and passed the parameters specified by the **Parameter Values** property of the configuration.
- If Debug Mode is set to External Application, the application specified by the **Application To Start** property is invoked using the working directory specified by the **Working Directory** property, with the arguments specified by the **Command Line Arguments** property of the configuration.
- If Debug Mode is set to “Messenger” the system is started up for use through the Messenger Client. XML messages can be opened and submitted to any Messenger class through the Messenger Client interface. The **Language** property affects the translated values of attributes within logic execution. The **Messenger To Debug** property defines a Messenger class that is pre-loaded into the Messenger Class field when the Messenger Client first comes up. The **Messenger To Debug** property can be left blank. This allows you to debug any Messenger class

To run a debug session, perform the following:

Once started, a debug session can be controlled using the commands listed in the following table, which are available from the Debug menu or toolbar.

Command	Description
Break All	Halts logic execution before interpreting the next logic statement.
Stop Debugging	Terminates the debug session.
Continue	Resumes logic execution.
Step Into	Interprets the next logic statement, or enters a method call in the statement, before stopping logic execution.
Step Over	Interprets the next logic statement, including logic in any methods called by that logic statement, before stopping logic execution.

Command	Description
Step Out	Interprets logic until the current method returns, before stopping logic execution.
Run To Cursor	Interprets logic until the logic statement on the selected line in the logic editor is reached, before stopping logic execution.

Debugger Windows

The following Microsoft Visual Studio Windows display information about the current debug session:

Call Stack

The Call Stack window displays methods currently on Debugger's call stack.

Breakpoints

The Breakpoints window displays all defined breakpoints for the solution.

Autos

The Autos window displays all variables referenced in the current method logic, including the executing object ('this'). It also displays the owning component (the segment) of the executing object. This includes Glb, providing access to the Glb.Status and Glb.Error built-in segment attributes.

To edit the value of a listed variable, type the required value directly in the **Value** field of the variable.

Locals

The Locals window displays local variables and parameters of the current method, including the executing object ('this').

To edit the value of a listed variable, type the required value directly in the **Value** field of the variable.

This

The **This** window displays the executing object ('this'). It also displays the owning class (‘owner’) of the executing object if there is one, and of the owning component (the segment) of the executing object. Note in all these sections, ‘component’ is a keyword like ‘this’ and ‘owner’

To edit the value of a listed variable, type the required value directly in the **Value** field of the variable.

Watch1, Watch2, Watch3, Watch4

The four Watch windows display user-specified variables. You can use the Watch windows to monitor or manipulate your choice of variables.

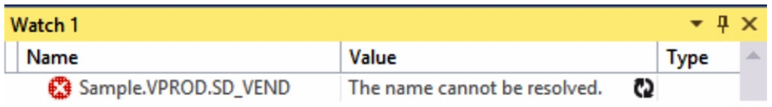
For example,

- To add a variable to a Watch window, type its name directly in the Watch window. You can also drag-and-drop or copy-and-paste a variable into the Watch window.

  Note: *When you drag-and-drop or copy-and-paste a variable, you might notice that the fully qualified name of the variable appears instead of only the attribute in the Watch window. The AB Suite Debugger cannot read the attribute from the fully qualified variable. Therefore, you need to manually edit the variable name to display only the attribute and its value.*

  For example,

  – After a fully qualified variable is copied to Watch window, the value of variable is not displayed as shown below.

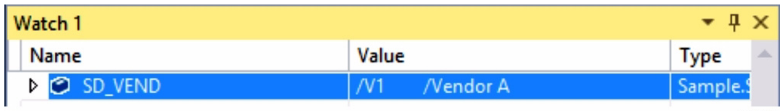


Watch 1

Name	Value	Type
Sample.VPROD.SD_VEND	The name cannot be resolved.	

007032

  – After editing the name of the variable to display only the attribute, the value is displayed as shown below.



Watch 1

Name	Value	Type
SD_VEND	/V1 /Vendor A	Sample.S

007033

- To remove a variable from a Watch window, select the variable in the Watch window, right-click, and select **Delete Watch**.
- To edit the value of a listed variable, type the required value directly in the **Value** field of the variable.

Search Debugging User Interface Reference in the *Microsoft Visual Studio Online Help* for more information on Debugger windows.

Breakpoints

Breakpoints halt logic execution at specified points, allowing observation and analysis of logic execution and variable and parameter values.

The model can be edited when logic execution is stopped. Refer to Edit a Model for more information.

Debugger supports Function and File breakpoints. Function breakpoints halt logic execution at a specified location within a specified method. File breakpoints halt logic at a specified location within a file or logic editor window. Effectively, both these types of breakpoints act the same since each method has its own editor.

Refer to Working with Breakpoints for more information on breakpoint use.

Working with Breakpoints

To create a breakpoint, perform the following:

1. Select **New Breakpoint** from the **Debug** menu or **Breakpoints** window toolbar to display the New Breakpoint dialog box.
2. To create a Function breakpoint, in the **Function** field on the Function tab, enter the name of the breakpoint in the following format:
   ```
   <model name>.(<class name>.)*<method name>
   ```
   To create a File breakpoint, in the File field on the File tab, enter the name of the breakpoint in the following format:
   ```
   <server>\<model name>.(<class name>.)*<method name>
   ```
   Search New Breakpoint Dialog Box in the *Microsoft Visual Studio Online Help* for more information on this dialog box.

   Alternatively, to create a File breakpoint on the current line in the logic editor, click in the left margin or right-click and select **Insert Breakpoint**.

To modify a breakpoint, perform the following:

1. Select the desired breakpoint in the **Breakpoints** window.
2. Select **Properties** from the **Breakpoints** window toolbar, or right-click and select **Properties**.

To remove a breakpoint, perform the following:

1. Select the desired breakpoint in the **Breakpoints** window.
2. Select **Delete** from the **Breakpoints** window toolbar, or right-click and select **Delete**.

   Alternatively, click the desired breakpoint glyph in the logic editor or right-click and select **Remove Breakpoint**.

To remove all breakpoints, select **Clear All Breakpoints** from the **Debug** menu or **Breakpoints** window toolbar.

To disable or enable a breakpoint, perform the following:

Clear or select (as required) the check box corresponding to the desired breakpoint in the **Breakpoints** window.

Alternatively, right-click in the logic editor and select **Disable Breakpoint** or **Enable Breakpoint** (as required).

To disable or enable all breakpoints, select **Disable All Breakpoints** or **Enable All Breakpoints** (as required) from the **Debug** menu or **Breakpoints** window toolbar.

Saving breakpoints

Breakpoints are automatically stored and loaded on a per user basis, and are persistent between Visual Studio sessions.

Restrictions

The following restrictions apply to the use of breakpoints:

- Address and Data breakpoints are not supported by Debugger.
- The Hit Count breakpoint property is not supported by Debugger.
- Place Function or File breakpoints at each reference to the required variable for Reference Breakpoints.
- Condition breakpoints in Agile Business Suite are not supported.

Search Breakpoints in the *Microsoft Visual Studio Online Help* for more information on the use of breakpoints, including a description of breakpoint glyphs.

Edit a Model

The model can be edited during a debug session, when logic execution is halted.

The debug session responds differently, depending on the nature of the change to the model:

Logic Changes

Logic changes to the model are not saved upon resumption of logic execution, they must be saved manually. Logic is validated before a method is called. If the logic changes are not successfully validated, you can choose to correct the error (if the method is already on the call stack or called from within another method being debugged), skip the method call, or terminate the debug session.

User Interface Changes

For user interfaces that have already been displayed during the debug session, user interface changes to the model are not applied immediately upon resumption of logic execution, or even when the changed user interface is next displayed (recalled). User interface changes are displayed only after the debug session is restarted.

For user interfaces that have not yet been displayed during the debug session, user interface changes to the model are displayed during the debug session. However, adding or removing a field leads to a mismatch between the user interface and the debug session. Only display changes such as color and position is applied.

Database Changes

Logical reorganization occurs if database changes to the model result in a model that does not match the schema of the physical database.

Multiple-Language Environments

In a multiple-language environment, debug sessions can be run in one of the set of defined languages. The Language property specifies the language in which to invoke the debug session. However, Visual Studio continues to run in the language defined by the Windows User Locale setting.

For all of the following examples, assuming that the Windows User Locale is set to English, the Visual Studio debug interface runs in English:

- If the Debug Mode is set to Online System, and the Language property is set to French, the application interface is invoked in French.
- If the Debug Mode is set to Report, and the Language property is set to Russian, the report is invoked in Russian.
- If the Debug Mode is set to Method Call, and the Language property is set to German, the method for the class is invoked in German.
- If the Debug Mode is set to External Application and the language property is set to Italian, the external application is invoked in the language it is configured for, but methods called on the classes are invoked in Italian.
- If the Debug Mode is set to “Messenger” and the language property is set to “Spanish”, the application runs with the Language set to Spanish. This affects the translated values of attributes in logic

For further details on the Debug Mode and Language properties, see Running a Debug Session and Configuration Properties.

Switching between Applications

If the SwitchTo logic command is invoked during a debug session, the solution for the target segment is launched in a new Visual Studio instance, and a new debug session is initiated. Switching back uses the original Visual Studio instance.

Note: *The registered value of the **Debug Mode** property of the target segment(its value when the segment was most recently built) must be any value other than Off, otherwise the new debug session isnot initiated. The language and database settings at this build time isalso used by Debugger.*

The target segment must have been built with a debug build. If the segments are in different models, the user hasto create their own configuration file for Winforms and run Winforms via External Application to be able to switch (this is automatically handled if the segments are in the same model). The build of the second segment needs to be manual and done with the correct deployment folder if the segments are in different deployment folders. But this is automatic if they are in the same folder.

Run Multiple Concurrent Debugging Sessions

Concurrent debugging implies that multiple users can concurrently debug the same AB Suite application, running on a Windows Terminal Server machine.

To perform concurrent debugging, perform the following:

1. Connect to the Windows Terminal Server machine as **User 1**.
2. Open an AB Suite application.
3. Create a new debug configuration for **User 1** in the model.
4. Set the following model properties:
    - Debug Mode
    - Class To Debug
    - Deployment Folder
    - Database Server Registration
    - Database Name
5. Set the following deployment folder properties:
    - Deployable
    - Deploy Application Components
6. Set the following segment properties:
    - Com Prog Id
    - Alternate Name
    - Database Schema Name

    Notes:

    - *Ensure that you set unique values for the **Com Prog Id** and **Alternate Name** segment properties, which have not been used in any other debug configuration.*
    - *If different data is to be entered in the same database, then a unique value should be set for the **Database Schema Name** segment property, which has not been used in any other debug configuration.*
7. Start the debugging process.
8. Connect to the Windows Terminal Server machine as **User 2** and perform step 2 through step 7.

    This enables multiple users to concurrently debug the same AB Suite application.

Testing Dynamic Attributes with Component Enabler

Component Enabler (CE) clients such as Presentation Client, ASP.NET, and VB.NET support the Dynamic Presentation Attribute feature.

To view the Dynamic Presentation Attributes in Debugger, you can use a CE client with Debugger via RATL. You can create a new configuration or use the configuration that you used for deployment. If you have already deployed the system on the Windows® platform and plan to run debugger on the same machine, then you need to provide different values for COM ProgID, database schema name, and Alternative name in Segment level property to avoid a compilation error.

Note: *Developer and CE must be installed on the same workstation. Alternate Name field in Configuration property dialog box must be blank or same as segment name.*

To use Debugger to Test Dynamic Attributes with CE, perform the following:

1. Ensure that the Debug Mode configuration property is set to Off.
2. Configure the properties for the Bundle for any of the CE clients.
3. Generate the Bundle.
4. Setup the Virtual Directory, and other setting for the bundle.

   Refer to the *Agile Business Suite Component Enabler User Guide* for more information.
5. Open the Model configuration property dialog box of the model and set the following configuration properties:
   - Set the Debug Mode configuration property to External Application.
   - Set the Application to Start configuration property value to RatlSocket.exe. For example, C:\Program Files\Unisys\AB Suite 6.1\Bin\ratlsocket.exe.
   - Set the Working Directory configuration property value to point to the Bin directory. For example, C:\Program Files\Unisys\AB Suite 6.1\Bin.
   - Set the Database name and Database Server Registration name configuration properties.
6. Open the Deployment folder configuration property dialog box and set the Deployable and Deploy Application Components (and Deploy reports, if required) configuration properties to true.
7. Open the Segment configuration property dialog box and ensure that all the properties are set in the Component Enabler User Interface section.

   Refer to the *Agile Business Suite Component Enabler User Guide* for more information on setting the properties for Component Enabler User Interface.
8. Create a View using Administration Tool and:
   - Specify the View Name.
   - Specify the System Name.
   - Specify the Server Name as localhost.

- Specify the Protocol (RATL over TCP/IP or MSMQ).
- Click RATL Login Settings and specify the required login details.

9. Stop the AB Suite 6.1 Protocol Adapter - RATL service(s) from the Services management console (if running).

   You can access the Services management console from **Control Panel** > **Administrative Tools** > **Services** or you can type Services.msc in the Run command on Start menu.

10. Start the Debugger.

    Debugger starts the RATL application and a message box Click to exit ratlsocket6.1 is displayed.

11. Start the CE client.
12. Select the required debug option to continue. For example, F5, F10, F11, and so on.

    After the debug session is established, the Dynamic Presentation Attributes are displayed in the Debugger.

13. After testing the application, to stop debugging either select Stop Debugging from the Debug menu or click the **Click to exit ratlsocket6.1**.

Using a Test Database on a Host Machine

From Debugger, you can connect to a database deployed on a host machine. This allows you to test a fully functioning application with a complete set of data. You can still use the local test database for unit testing and audit and recovery purposes.

Caution

This facility is for testing purposes only. Do not use it with your deployed production databases. Applying untested code to a production database could destroy the integrity of the database, risk loss of data, and risk database corruption.

Preparing Your Application in Developer

Before you can connect to a test database on a host machine you must configure and generate your system.

In order for Debugger to access the deployed system database there are several configuration properties options you must set. Refer to the *Agile Business Suite Developer User Guide* for more information on how to set configuration properties options and generating systems.

Preparing to Connect to a Test Database on a Host Machine

Refer to the *Agile Business Suite Runtime for Windows® Operating System Administration Guide* for your respective platform.

Test Database Issues

Refer to Test Database Properties for more information on configuration properties related to the Debugger test database.

Test Database Migration

Test databases used should be migrated before Debugger is run against them. Refer to the *EAE to Agile Business Suite Migration Guide* for more information on Test Database Migration.

Logical Reorganization

Logical reorganization of the database occurs during a debug session that refers to a model that does not match the schema of the physical database.

During logical reorganization:

- Database-only members are ignored during the debug session.
- Model-only members are accessible during the debug session, but they are not read from or saved to the database.
- Array members that do not match are accessible during the debug session, but they are not read from or saved to the database. They are only session-persistent and are created with a default value, corresponding to their Initial Value property.
- Non-primitive members (that is, composite members that are stored as a single column in the database) that do not match are not read from or saved to the database. They are only session-persistent and are created with a default value, corresponding to their Initial Value property.
- Members with a primitive property that does not match the database are accessible during the debug session, but they are not read from or saved to the database. They are only session-persistent and are created with a default value, corresponding to their Initial Value property.
- Members with a length property and/or decimals property that do not match the database are accessible during the debug session, and are read from and written to the database, but they are truncated to shorter of the model-defined and database schema-defined lengths and/or decimals.

  If the model-defined length is shorter than the database schema-defined length or decimals, the value written to the database is padded with spaces or zeros, as appropriate.

Database Security

To run debug sessions against a database, it is necessary to grant access to the Windows® user running the debug session (the Debugger user).

Using the Administration Tool, the Debugger user should be made a member of the Secure Users role, as described in the *Administration Tool Online Help*.

Note: *Administrators of the machine are automatically given access by the installation process.*

Restrictions

The execution of application logic by Debugger is subject to the following restrictions:

Determine Actual

The SQL script variant of the Determine Actual logic command is ignored. Debugger disregards any Determine Actual logic statements that specify SQL scripts that it encounters, and continues execution at the logic statement following the End or EndExit logic statement terminating the Determine Actual loop. Other Determine Actual variants (the database and extract file variants) are not ignored and are executed normally.

SQL Scripts

SQL script methods are ignored. Debugger disregards any invocations of SQL script methods it encounters, and continues execution at the logic statement following the method invocation.

Performance and Resource Usage

The first time a Winform is shown in the Debugger it performs JIT (Just-In-Time) compilation of the components being debugged (and other required components). Also, Debugger interprets the model such that it can pick up changes and validation is done during a debug session. These actions dictate the performance. Consequently, the performance of the segment during a debug session is not an indicator of the performance of the deployed application. Runtime performance should generally be much better, as it is executing compiled native code.

Memory requirements during a debug session vary, as memory is dynamically allocated as objects are accessed. Allocated memory is not released until the segment caller no longer requires the owning object. Therefore although initial memory requirements are relatively small, the memory requirements may grow to become quite significant, especially for debug sessions accessing many parts of a large segment.

Debugger Administration

When a debug build occurs, wrapper COM DLLs are optionally built for the Segments and/or Reports being built. The choice of whether to build these are determined by the setting of Deploy Application Components and Deploy Reports in the Build Target Filter categories of the Folder Configuration Properties, being True or False.

When built, these COM DLLs are registered in the Windows registry with the same ProgId as for the same objects in the Generated Windows® Runtime if the same configuration is used to build them. To unregister and remove them, the Debugger Administration utility can be used.

The Debugger Administration utility is accessed from the Debugger Administration sub menu item of the Development Environment program menu.

When the Debugger Administration dialog box displays, it contains a list of all the <<Segment>> and <<Report>> Class interfaces which are registered with the Debugger. The Name column of the list contains the name of the <<Segment>> Class in the Model, or the name of the <<Report>> Class qualified from its owning <<Segment>> Class in the Model. The COM ProgId column of the list contains the ProgId which was used to register the Class' interface.

To unregister any of the interfaces in the list, select them and click **Remove**. You can refresh the list at any time by clicking **Refresh**.

Debugging through EBCDIC Tool

A Debugger Extended Binary Coded Decimal Interchange Code (EBCDIC) tool allows you to interact with EBCDIC data. You can access the Debugger EBCDIC tool menu by selecting the EBCDIC tool under Agile Business Suite Developer. The options of the Debugger EBCDIC tool menu are also available on the Debugger EBCDIC Toolbar.

To open the EBCDIC debugger tool, perform the following:

1. Point to the bottom-left corner of the screen to enable the Start icon, click **Start** > **Apps** > **Agile Business Suite 6.1** > **Development Environment** > **EBCDIC Tool**.
2. Click **EBCDIC** Tool.

   The Debugger EBCDIC Tool window appears

Alternatively, you can perform the following:

1. Point to the bottom-left corner of the screen to enable the Start icon, click **Start**, type and select **EBCDIC** Tool on the desktop.

   Note: *If the EBCDIC Tool is not present on the desktop, pin the application to the desktop.*
2. Click **EBCDIC Tool**.

   The Debugger EBCDIC Tool window appears.

Use the Debugger EBCDIC Tool

The Debugger EBCDIC tool enables you to

- Access databases in EBCDIC format
- Convert text files
- Change Languages
- Access various windows by using the Windows menu option

Access Databases in EBCDIC Format

Data sets are encoded in EBCDIC format for all AB Suite applications built on the MCP platform. The EBCDIC tool enables you to

- View EBCDIC data sets in ASCII format.
- Add data sets directly to a database.
- Update/modify data in a database.
- Delete data sets directly from a database.
- Query data sets present in a database.

Viewing EBCDIC Data Sets in ASCII Format

To view data sets from the Debugger EBCDIC tool menu, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.
2. Select **Open Table** on the **Database** menu option of the EBCDIC tool menu.

   The **Open Table** dialog box appears.
3. Select a server from the **Server Name** list of the **Open Table** dialog box.

   Note: *By default, the server is local.*
4. Select a database from the **Database** list of the **Open Table** dialog box.
5. Select a table from the **Table** list of the **Open Table** dialog box.
6. Click **Open**.

   The selected table appears in ASCII format.

Adding Data

To add data sets directly to a database, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.
2. Select **Open Table** on the **Database** menu option of the EBCDIC tool menu.

   The **Open Table** dialog box appears.

3. Select a server from the **Server Name** list of the **Open Table** dialog box.

   Note: *By default, the server is local.*

4. Select a database from the **Database** list of the **Open Table** dialog box.
5. Select a table from the **Table** list of the **Open Table** dialog box.
6. Click **Open**.

   The selected table appears.

7. Select **Add** on the **Database** menu option of the EBCDIC tool menu, or click ✚ on the Debugger EBCDIC Toolbar.

   A new row is added to the open table.

8. Enter data directly into the table.
9. Select **Save** on the **Database** menu option of the EBCDIC tool menu or click 💾 on the Debugger EBCDIC Toolbar.

   The newly added data is saved in the database.

Updating/Modifying Data

To update/modify data sets in the database, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.

2. Select **Open Table** on the **Database** menu option of the EBCDIC tool menu.

   The **Open Table** dialog box appears.

3. Select a server from the **Server Name** list of the **Open Table** dialog box.

   Note: *By default, the server is local.*

4. Select a database from the **Database** list of the **Open Table** dialog box.
5. Select a table from the **Table** list of the **Open Table** dialog box.
6. Click **Open**.

   The selected table appears.

7. Select the row that you wish to update/modify in the open table.
8. Update/modify the necessary data.
9. Select **Save** on the **Database** menu option of the EBCDIC tool menu or click 💾 on the Debugger EBCDIC Toolbar.

   The updated data is saved in the database.

Deleting Data

To delete data sets directly from a database, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.
2. Select **Open Table** on the **Database** menu option of the EBCDIC tool menu.

   The **Open Table** dialog box appears.
3. Select a server from the **Server Name** list of the **Open Table** dialog box.

   Note: *By default, the server is local.*
4. Select a database from the **Database** list of the **Open Table** dialog box.
5. Select a table from the **Table** list of the **Open Table** dialog box.
6. Click **Open**.

   The selected table appears.
7. Select one or more rows that you wish to delete in the open table.
8. Select **Delete** on the **Database** menu option of the EBCDIC tool menu, or click ✕ on the Debugger EBCDIC Toolbar.

   The selected rows are deleted.
9. Select **Save** on the **Database** menu option of the EBCDIC tool menu, or click 💾 on the Debugger EBCDIC Toolbar.

   The changes are saved in the database.

Querying the Database

To query data sets present in a database, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.
2. Select **New Query** on the **Database** menu option of the EBCDIC tool menu.

   The **Connect to Database** dialog box appears.
3. Select a server from the **Server Name** list of the **Connect to Database** dialog box.
4. Select a database from the **Database** list of the **Connect to Database** dialog box.
5. Click **Connect**.

   A database connection is established.
6. Enter the query in the Query window.
7. Select **Execute SQL** on the Database menu option of the EBCDIC tool menu, or click ! on the Debugger EBCDIC Toolbar.

   The result set of the query appears on the Query window.

Converting Data Files

The EBCDIC tool enables you to

- Create a new text file.
- Update/Modify an existing text file.
- Convert text files from EBCDIC to ASCII/Unicode and vice-versa.

Note: *If you create a text file by using the EBCDIC tool, by default the file is saved in the EBCDIC format. To view the contents of the file in ASCII, you must open the relevant file using the EBCDIC tool.*

Creating a New Text File

To create a new text file, perform the following:

1. Open the Debugger EDCDIC Tool.

   The Debugger EBCDIC tool menu appears.
2. Select **New File** on the **Extract File** menu option of the EBCDIC tool menu.

   A new window appears.
3. Enter text in the editor.
4. Select **Save** as on the **Extract File** menu option of the EBCDIC tool menu, or click on the Debugger EBCDIC Toolbar.
5. Enter a file name in the **File Name** list of the **Save File as** dialog box.

   The new file is saved in the default location.

Note: *To save a file in a different location, browse to the desired location and save the file.*

Updating/Modifying an Existing Text File

To update/modify an existing text file, perform the following:

1. Open the Debugger EBCDIC tool.

   The Debugger EBCDIC tool menu appears.
2. Select **Open** on the **Extract File** menu option of the EBCDIC tool menu.
3. The **Open EBCDIC Extract File** dialog box appears.
4. Browse to the location and select the file.
5. Click **Open**.

   The content of the selected file appears on the editor.
6. Modify the content and select **Save** on the **Extract File** menu option of theEBCDIC tool menu, or click on the Debugger EBCDIC Toolbar.

   The updated content is saved.

Converting Text Files

Text files can be converted from one format to another by using the EBCDIC Debugger tool or the Command line interface.

Note: *The following is the list of input and the output file formats:*

- *EBCDIC V24-K*
- *UNICODE*
- *ASCII*
- *EBCDIC V24+K (Japanese)*

Using the EBCDIC Debugger Tool

To convert files by using the EBCDIC Debugger Tool, perform the following:

1. Open the Debugger EBCDIC Tool.

   The Debugger EBCDIC Tool menu appears.
2. Select **Convert File** on the **Extract File** menu option of the EBCDIC tool menu.

   The **Convert Extract File** dialog box appears.
3. Click **Browse** to select the **Input Extract File**.
4. Select the Input file format.
5. Click **Browse** to specify a location and file name where the Output Extract file is available.
6. Select the Output file format.
7. Click **Convert** to convert the file.

   The converted file is saved in the Output Extract file location
8. Click **Close**.

   The converted text file is saved.

Using the Command Line Interface

To convert files by using the Command line Interface, perform the following:

1. Point to the bottom-left corner of the screen to enable the Start icon, click **Start**, and then click **Run**.
2. Type **cmd** in the Run dialog box and press **Enter** to open a Command Prompt.
3. In the command prompt, change the working directory to the folder where the text files are present.
4. Run the following command to convert the files:
   ```
   EBCDICTool.exe /c <InputFile> <InputCharSet> <OutputFile> <OutputCharSet>
   ```

Where:

- InputCharSet and OutputCharSet can be in any one of the following formats:
  - EBCDIC V24-K
  - UNICODE
  - ASCII
  - EBCDIC V24+K(Japanese)
- InputFile is the file to be converted.
- OutputFile is the converted file.

The converted text file is saved.

Setting a Language

The EBCDIC tool enables the user to set a language by using the Options menu option of the EBCDIC tool menu. The two available languages are EBCDIC (V24-K) and Japanese (V24+K).

To change to a specific language, perform the following:

1. Open the Debugger EBCDIC Tool.

   The Debugger EBCDIC Tool menu appears.
2. Select **Settings** on the **Options** menu option of the EBCDIC Tool menu.

   The **Settings** dialog box appears.
3. Select a language from the displayed Character Set Mode.

   Note: *By default, the language is V24-K (EBCDIC).*
4. Click **OK**.

   The new language is set.

Accessing the Windows Menu Option

The windows menu option enables you to navigate to various open windows. It also enables you to close all the windows instantly. The windows menu option allows you to

- Close all the windows.
- Activate and close the opened windows.

Closing all Windows

To close all windows, perform the following:

1. Select **Close All Windows** on the **Window** menu option of the EBCDIC Tool menu.

   All open windows are closed.

   Note: *The application does not exit and only the opened windows are closed.*

Activating and Closing the Opened Windows

To activate and close the opened windows, perform the following:

1. On the **Window** menu option of the EBCDIC Tool menu, select **Windows….**

   The **Windows** dialog box appears.
2. To open a particular window, select a window from the **Windows** dialog box and click **Activate**.
3. To close a window, select a window from the **Windows** dialog box and click **Close Window(s)**.
4. To close the **Windows** dialog box, click **OK**.

EBCDIC Tool User Interface

The EBCDIC tool comprises the following,

- The Debugger EBCDIC Tool Menu.
- The Debugger EBCDIC Toolbar.

Debugger EBCDIC Tool Menu

The EBCDIC Tool menu is available on the Debugger EBCDIC Tool menu bar. The options available on the EBCDIC Tool menu enable you to access the various functionalities of the Debugger EBCDIC tool. The EBCDIC menu options are also available on the Debugger EBCDIC Toolbar. The various menu options are explained in the following sections.

EBCDIC Toolbar

The toolbar provides quick access to the various EBCDIC tool functionalities. The following table describes the toolbar options:

Database Toolbar

The following table describes the toolbar options available for the database menu.

Option	Icon	Description
Execute SQL	❗	Executes the SQL query.
Add	➕	Adds a new row in the selected table.
Delete	❌	Deletes the selected row.
Save	💾	Saves data entered in a database.
Move next	▶	Moves cursor to the next record.
Move previous	◀	Moves cursor to the previous record.
Move last	▶|	Moves cursor to the last record.
Move first	|◀	Moves cursor to the first record.

Extract File Toolbar

The following table describes the toolbar options available for the extract file menu.

Option	Icon	Description
Save File	💾	Saves the changes made to an existing file.
Save As	🔄	Saves the newly created file in the specified location.
Add Column Definitions	▥	Adds column definitions to a text file.

Building Applications

This section focuses on building AB Suite applications, AB Suite Client Framework applications, and AB Suite XML Framework applications.

Building is the process of producing the database definition language, the program source code and other files necessary to deploy your system from the business model that you have developed. This process is a one button operation that initiates a number of steps that Builder automatically completes without any further user intervention.

Builder is invoked from a Developer workstation and executes on that workstation. Note that the 'Developer workstation' could be an instance of Developer installed on a single-CPU or multi-CPU server. It is also possible that the platform hosting Developer could be the server upon which the application is deployed (Applicable only to Windows runtime).

Builder Overview

Builder in Agile Business Suite Developer builds and deploys applications based on the structure and contents of your Developer model, and creates or updates database schemas.

Note: *You must build the AB Suite Client Framework applications after generating the Client Framework projects for a specific client technology. Refer to Generating Client Framework Projectsfor more information on generating the Client Framework projects.*

There are four phases to building an application:

- Configuration – Where a user creates a set of properties or options associated with the application model which defines the deployment characteristics of the runtime application. These properties are directly analogous to the Generate Set capability maintained in the System View of previous Enterprise Application Developer releases.

- Generation (also Building) – The process of constructing (generating) the necessary source files from the structure and contents of the application model and then compiling and linking those files to create a set of executable components. This also includes the creation of a new or updated database schema in an appropriate format for deployment.
- Deployment – The process of configuring and installing the generated executable components onto a target server environment. This also includes the creation or reorganization / restructuring / migration of the application database.
- Access – When an AB Suite application is built, you can access an application through the Windows Forms Container on the host machine (for Windows operation), host terminal screens (MCP), Component Enabler interface (Windows, MCP). When an AB Suite Client Framework application is built, you can access an application through the WPF Client Container on the host machine, if the Client Technology property of the Technology folder is set to WPF. When an AB Suite XML Framework application is built, you can access it through MessengerClient.exe and submit individual XML messages to the application.

Configuration

Before building and deploying your application for the first time you need to configure Builder with the details of what to build and where to deploy it. You do this by placing the model elements you want to deploy in a deployable Element. You then specify configuration properties for the folder and the segment elements.

Note: *For a Windows platform, you need to first set up the Runtime database.*

Configure a Runtime Database Server for Windows

Before you attempt to build your application, you need to set up the database server that it uses.

Runtime for the Windows® operating system includes an Administration Tool, which provides an integrated interface to help you manage and control all the components that your application uses. You can use the Administration Tool to create a new database. This creates a basic database on the server. Other Runtime platforms which are available in Agile Business Suite use the proprietary tools for each platform. Refer to the *Agile Business Suite Installation and Configuration Guide* for each particular platform.

It is recommended, however, that you create your own database using the tools that come with your database software. This enables you to fully control the preparation, deployment, and configuration of the database. You must then use the Administration Tool to prepare the database you have created. Refer to the *Agile Business Suite Runtime for Windows® Operating System Administration Guide* for more information on how to do this.

To fully configure a system for deployment, the target host name, database name and the database server registration alias names must be known.

Creating a Set of Configuration Properties

You can create more than one set of configuration properties for your model. To create a new set of configuration properties, perform the following:

1. On the model's Property Pages dialog, click the **Configuration Manager...** .
2. From the Active Solution Configuration drop-down list, select **<New...>**.
3. Enter a Solution Configuration Name. Ensure that the **Create new project configurations** check box is selected.
4. From the **Platform** drop-down list select a platform for the deployment, or add a new platform to use. You can also remove an existing platform by selecting **<Edit...>**.

Note: *Having too many multiple configurations impacts the performance of Visual Studio especially in a multi user environment. You should delete unwanted configurations from the project at regular intervals. In a multi user environment, only one user should delete all the unwanted configurations across all the projects. After that, other users should delete their Visual Studio project folders only. After deleting the unwanted configurations, all users should copy the project folder to their respective locations.*

To delete multiple configurations, perform the following:

1. Select **Configuration Manager** from the **Solution Configurations** drop-down list on the standard toolbar.

   The **Configuration Manager** window opens.
2. Select **Edit** from the **Active solutions configuration** drop-down list.

   The **Edit Solutions Configuration** window opens.
3. Select the unwanted configurations from the **Name** list and click **Remove**.

   You can delete only one configuration at a time.
4. Click **Close** to close the **Edit Solutions Configuration** window.
5. Select **Edit** from the **Configuration** drop-down list in the **Project contexts** list.

   The **Edit Project Configurations** window opens.
6. Select the unwanted configurations from the **Name** list and click **Remove**.

   You can delete only one configuration at a time.
7. Click **Close** to close the **Edit Project Configurations** window.
8. Click **Close** to close the **Configuration Manager** window.
9. Save the project file and distribute the project folder to other users.

Creating and Configuring a Deployment Folder

To create and configure a deployment folder for an AB Suite model, perform the following:

1. Add a folder to the model that contains the segment you want to deploy.
2. Drag the segment you want to deploy into this folder.
3. Right-click the folder and select **Properties** from the context menu.

   The **Property Pages** dialog box appears.
4. Set the Deployable property to True. Thisenables all the relevant properties for editing.
5. Specify the deployment properties for the folder.

You can either create nested deployable elements so that parts of your application can be deployed separately for an AB Suite application. For example, you may need to deploy some reports separately from the main application. In Windows® runtime each deployable folder is built as a separate installation package. It is important that each deployable element has a unique package name within a project, particularly if you have multiple segments within a project.

To create and configure a deployment folder for an AB Suite Client Framework model, perform the following:

1. Add a folder to the model that contains the segment you want to deploy.
2. Drag the segment you want to deploy into this folder.
3. Right-click the folder and select **Properties** from the context menu.

   The **Property Pages** dialog appears.
4. Set the **Deployable** property to **True**.
5. Set the **Deploy Application Components** property to **True**.
6. Set the **Deploy Database** property to **True**.
7. Set the **Deploy SQL Views** property to **True**.
8. Enter the **Package Name** of the MSI file in which you want to package the deployable elements.

   By default, the value is the name of the folder.
9. Enter the **Deployment Host** name.
10. Browse to and select the **Package Installation Directory** where the deployableunit should be installed on the Runtime server.
11. Browse to and select the **Package Intermediate Directory** where the installer files should be transferred before the installation process starts.

Configuring the Segment

There are additional configuration properties that need to be set on the segment.

To configure a segment for an AB Suite application, perform the following:

1. Right-click the segment you want to deploy. It should already be in the configured deployable Element.
2. Select **Properties** from the context menu.

   The **<Segment Name> Property Pages** dialog box appears.
3. For Windows® platform, configure **Target Database** and **Database Server Registration** with the names used when the Runtime database was created.

To configure a segment for an AB Suite Client Framework application, perform the following:

1. Right-click the segment that you want to deploy. It should already be in the configured deployable Element.
2. Select **Properties** from the context menu.

   The **<Segment Name> Property Pages** dialog box appears.
3. Configure the **Database Schema Name**, **Database Server Registration**, and **Database Name** with the names used when the runtime database was created.

There are additional segment properties that can be changed at this point. Review the settings to ensure they are correct.

Generation

The overall sequence of events to generate the completed application model is as follows:

- The generation process is initiated from Developer when you select the Build command or the Rebuild command. Refer to Build Options for more information on Build Options.
- Builder first checks for any model structure validation error. If there are no errors, build proceeds to the next logic validation phase. In logic validation phase, Builder ensures that all the methods are validated. If any method is not validated, Builder performs the validation now. If the logic contains any errors, the generate process is cancelled. If model structure validation errors exists;
  - in any system component such as Ispecs or events, generation process is aborted.
  - in reports, builder excludes those reports and proceeds to the next logic validation process.
- Builder determines what files/code needs to be generated by reference to the previous deployment (if any) of the application using the selected configuration to the targeted server If change analysis detects that the element has changed since it was last built, Builder generates the source files for the element.

In Windows® platform, builder performs the following additional steps during generation:

- When all the elements have been validated and all the source files have been generated, the compilation phase begins. The generated C# source code files are input to the compiler to be compiled and linked to produce the deployable executable components. Compilation creates dynamic link libraries (DLLs), executable files, a configuration XML file for AB Suite models, a configuration RTXML file for AB Suite Client Framework models, and other files.
- A deployment package is constructed containing all of the deployable elements. This file is placed in a temporary location, specified in the Package Intermediate Directory configuration property. If this property is empty, by default, the Builder Output directory is used.
- The deployment service is remotely invoked and retrieves the deployment package from the package intermediate directory and deploys the application, making any necessary database changes.

Notes:

- *When the user selects the Rebuild option instead of the Build option, Builder performs a complete rebuild of the selected model--no change analysis is performed. Refer to Build Options for more information.*
- *If you are not able to build an AB Suite application, check whether the smbca.exe process is running in the task manager. You should stop the smbca.exe process manually to continue with the build process.*

Reports may be generated and deployed separately to the Segment generation and deployment or in conjunction with it.

No matter whether reports are generated and deployed separately or with the segment, you may choose to generate and deploy:

- A single report
- A group of reports
- All reports

Each report is generated and deployed as a separate component. However, if multiple reports are being generated and deployed at the same time they are batched up and deployed in a group.

The following table mentions the Build details for various platforms, which is applicable for an AB Suite application and AB Suite Client Framework application:

Field	Description
Start Deployment With	Set this field to Generate from the drop-down list. The build starts from the build phase specified. ***Note:*** *This is applicable for Windows® platforms.*

Field	Description
Stop Deployment After	Set this field to Install from the drop-down list. The build ends after the build phase specified. ***Note:*** *This is applicable for Windows® platforms.*
Retain Existing Database	Unchecking causes a new database to be created. ***Note:*** *This is applicable for Windows® and MCP platforms.*
Host	Name of the deployment server. ***Note:*** *This is applicable for Windows® and MCP platforms.*
User Name	Name of the user. ***Note:*** *This is applicable for Windows® platforms.*
Domain Name	By default, the domain name is set to dot (.). ***Note:*** *This is applicable for Windows® platforms.*
Password	The password set for the user name or user code. ***Note:*** *This is applicable for Windows® and MCP platforms.*
MultiThreaded Compilation	Checking this option enables the build process in large applications to identify the compilation streams that can be executed in parallel to speed up the overall build time on multi-CPU build machines. By default, this option is enabled and can be disabled in cases where an application must be compiled serially. In most small test applications, only a single compilation stream may be available. In these cases, the multithread compilation option does not effect on compilation times. ***Notes:*** • *This is applicable for Windows® platforms.* • *Multi-thread builds might result in a build operation failure with structured exception errors. This occurs in a few models for particular environment. If you experience such structured exception errors during multi-thread builds, you can work around the problem by running the build with a single-thread. For this, go to Visual Studio and set* ***Tools*** *>* ***Options*** *>* ***System Modeler*** *>* ***Builder*** *>* ***Number of Build Threads to 1****.*

Field	Description
Check for Inconsistencies between Online and Reports	You can select this option that allows you to check for inconsistencies of any methods, classes, attributes defined in a segment, which are used by a report. You can select this option only if you want to introduce this additional verification. If the elements defined in a segment are used by a report and these elements have been modified after the last online system build, a warning message appears in the build log. The warning message warns you that the elements in the segment that are referenced by the report have been modified, so building the report might fail or may result in report runtime errors. If any inconsistencies are detected then the build stops after completing this verification. You can deselect this option, if you want to continue with the build despite these warnings then you can submit this build again. Alternatively, you can build and deploy the online system toresolve these inconsistencies. ***Note:*** *This is applicable for Windows® platforms.*
Folder is	This is the deployable Element of the model. ***Note:*** *This is applicable for the MCP platform only.*
User Code	Name of the user code. This is the same as the User Name. ***Note:*** *This is applicable for the MCP platform only.*
Access Code	The access code set for the target host installation. ***Note:*** *This is applicable for the MCP platform only.*
Access Password	The password set for the Access Code. ***Note:*** *This is applicable for the MCP platform only.*
Charge Code	The charge code for the systems user code. ***Note:*** *This is applicable for the MCP platform only.*
Use host Details for FTP	Checking the box takes the default User Code name and password as mentioned in the User Code and password fields. ***Note:*** *This is applicable for the MCP platform only.*
FTP User Code	Name of the User code. This is the same as the user name. ***Note:*** *This is applicable for the MCP platform only.*
FTP Password	The password set for the FTP User Code. ***Note:*** *This is applicable for the MCP platform only.*
FTP Charge Code	The charge code for the FTP user code. ***Note:*** *This is applicable for the MCP platform only.*
Delay Compile and Deploy	Checking this box allows you to set the date and time when you want the system to generate. ***Note:*** *This is applicable for the MCP platform only.*

Field	Description
Deploy Date	The date on which you want the generate to commence. ***Note:*** *This is applicable for the MCP platform only.*
Deploy Time	The time at which you want the generate to commence. ***Note:*** *This is applicable for the MCP platform only.*

Build Options

Builder provides four build options, namely, Build, Rebuild, Clean, and Preview. The following table describes when to use each of these build options and what happens when they are used.

Build Options	When to Use	What Happens
Build	Used when you only want to generate the changed components.	Change analysis works out which of the components configured to be built need to be rebuilt due to changes to the model. Files associated with these components are generated in the Builder Cache.
Rebuild	Used when you want everything to be regenerated.	All components configured to be built are rebuilt. Files associated with these components are generated in the Builder Cache.
Clean	Used when you want everything to be removed before a subsequent Build. ***Note:*** *Clean option can be used only on Model Nodes.*	All files associated with the components configured to be built are removed from the Builder Cache, Builder output, and change analysis tables are cleaned. . Products of a build that may be in use, for example installed system, installed reports, Component Enabler files, Winform dlls are not removed with a clean. These files do not exist in the builder cache. If the selected model contains multiple segments with multiple top level folders, Builder cache, Builder output, and change analysis table for all the segments and folders are removed irrespective of the deployable option.

Build Options	When to Use	What Happens
Preview	Used to view details of elements that were modified since the last build along with the percentage of the build impact. A Preview is also used to determine if database reorganization is required.	Displays a list of model elements that are modified and require regeneration since the last time the application was generated. When built using Visual Studio, it also displays the percentage of the build impact for a full or partial build of the selected deployable folder. The output detail of this option helps you to analyze the impact of the current development on your deployed application. ***Note:*** *For MCP and CLR Configurations, this option is available as the Preview option via the Build menu.*

Notes:

- *The Build/Rebuild/Preview <Folder> menu options are enabled only when Deployable = True. These items are not visible when Deployable = False. In addition you must ensure that the outer folder containing the segment is deployable and Deploy Application Components is enabled.*
- *The Builder Cache Directory is used to store all files generated by a Build or Rebuild. It retains files generated by previous Builds or Rebuilds for use in subsequent Builds. In a multi-user environment it needs to be shared between the users. When the Build starts, the files that do not exist in Builder Output directory are copied from the Builder Cache folder. For efficiency purposes, it is recommended to install RoboCopy.*
- *The Builder Output directory is the location where files are generated to before they are compressed and moved to the Builder Cache Directory.*
- While building the ABSuite application in Windows runtime, you must enable 8.3 name creation on the drive for which the Builder Output and Builder Cache are configured.
  To enable 8.3 name creation, perform the following:
  1. *Open the Registry Editor.*
  2. *Browse to the following location:*
     `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem.`
  3. *Right-click **NtfsDisable8dot3NameCreation** and select **Modify...**.*
     *The **Edit DWORD Value** dialog box appears.*
  4. *In the **Value data** field, enter **0**.*
  5. *Click **OK** to close the dialog box.*

Following are the Build options for AB Suite applications and AB Suite Client Framework applications:

- The Build options for AB Suite applications can be categorized into Folder Only and normal Build options (non-Folder Only Build options). These are described as follows:

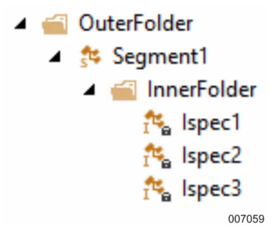
  During a Folder Only Build the target components (defined by the folder configuration properties) associated with the elements contained in the folder are built.

  A normal Build (non-Folder Only Build) can be viewed as a collection of Folder Only Builds, that is, a Folder Only Build of the selected folder and a Folder Only Build of all deployable folders contained within that folder. When the deployable targets of the outer folder and the subfolder are the same, then the element contained in the subfolder is built twice, because the elements that are members of subfolders are also members of the outer folders.

  For example, an outer folder has the Deploy Reports property set to true and its subfolder also has the Deploy Reports property set to true. When a build of the outer folder is initiated, it results in building the report twice: once in relation to the outer folder and again in relation to the subfolder.

  The following example explains the difference between a Folder Only and a normal Build in Windows for an AB Suite model.

  OuterFolder contains Segment1, and Segment1 contains an InnerFolder. The InnerFolder contains Ispec1, Ispec2, and Ispec3.



  OuterFolder is configured to deploy general application components, database components, and Windows Forms components, but not Component Enabler components. InnerFolder is configured to deploy Component Enabler components.

If you select OuterFolder and select **Build** or **Rebuild** from the context menu, the deployable outputs for both OuterFolder and InnerFolder are deployed as follows:

- General application components for Segment1, Ispec1, Ispec2, and Ispec3.
- Database components for Segment1, Ispec1, Ispec2, and Ispec3.
- Windows Forms components for Segment1, Ispec1, Ispec2, and Ispec3.
- Component Enabler components for Ispec1, Ispec2, and Ispec3.

If you select OuterFolder and select **Folder Only**, then select **Build** or **Rebuild**, the deployable outputs for OuterFolder are deployed, but not the deployable outputs (Component Enabler components) for InnerFolder, as follows:

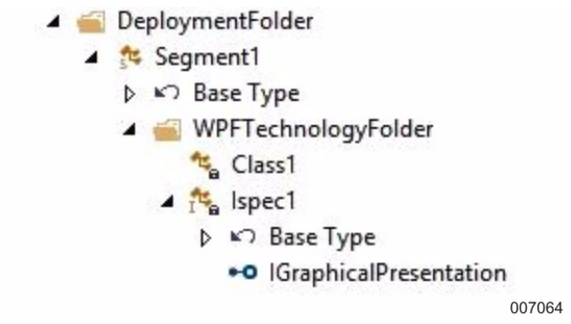
- General application components for Segment, Ispec1, Ispec2, and Ispec3.
- Database components for Segment, Ispec1, Ispec2, and Ispec3.
- Windows Forms components for Segment, Ispec1, Ispec2, and Ispec3.

• The Build options for AB Suite Client Framework applications can be categorized into Generate Access Layer and normal Build options. Refer to Generating Client Framework Projects for more information on generating the Technology folder.

A normal build is a collection of Technology folders build and deployable folders build.

The following example explains the difference between a Technology folder and a normal build in Windows® Runtime.

DeploymentFolder contains Segment1, and Segment1 contains a WPFTechnologyFolder. WPFTechnologyFolder contains Ispec1 and Class1, and the Ispec1 contains an IGraphicalPresentation node.



DeploymentFolder is configured to deploy general application components, database components, and WPF Client components. WPFTechnologyFolder is configured to generate Client Framework projects for user interface development.

If you select DeploymentFolder and select **Build** or **Rebuild** from the context menu, the deployable outputs for both DeploymentFolder and WPFTechnologyFolder are deployed as follows:

- General application components for Segment1
- Database components for Segment1, Ispec1, and Class1

Any changes identified by Change Analysis are also generated for the Technology folder.

Deployment

The deployment process is responsible for ensuring that the generated application is deployed with minimum disruption to your operation and at the time specified by you. The deployment process is normally initiated by the Configuration Builder after the generation and construction of the deployment package. You may also independently invoke the deployment service to re-deploy the generated application, for instance following a server failure, a server re-configuration, or to deploy the generated application to a different server.

Input to the deployment process comprises of the following:

- Generated and compiled programs/components.
- Database schema.
- Deployment instructions and configuration data.
- Information about the already deployed application if it has been previously deployed.

The deployment process performs the following operations:

- Verify that the deployment package is complete.
- If the application has previously been deployed, extract the database schema for comparison against the new database schema to determine if anything has changed.
- If the database has changed prepare and execute a strategy to change the database to the new format and migrate the existing data to it, if necessary, otherwise if it is a new database create the database using the generated schema.
- Install the generated executable components.
- Notify the Configuration Deployer that deployment is complete.

During the entire process, the build progress and error messages are displayed in the Output window. The user who initiates the build can monitor the process with progress indications for each stage, and obtain error reports, if an error occurs that halts the process.

Access

In Windows® platform deployment creates an application that clients can access. If the **Stop Deployment After** configuration property is set to **Run**, the application is automatically started on the runtime host as soon as deployment is complete. However, it may be necessary to back up the database after the application is deployed, so you may want to stop the build after the install stage.

Component Enabler

The standard application components that are deployed from Visual Studio run on Windows® platforms only. Component Enabler adds the ability to access standard Agile Business Suite applications from non-Windows platforms by using a graphical user interface.

To generate Component Enabler components, set the **Deploy Component Enabler User Interface** configuration property to **True**. Select the appropriate language for the **Default Translation** segment configuration property. The Component Enabler generator takes your ispec interface definitions and builds standard JavaBeans components and sample Java user interface applets.

Component Enabler sessions are established as TCP/IP connections using the Remote Access Server interface. This allows most client applications on most platforms to easily access your application.

Web-Enabling with Component Enabler

Component Enabler supports many ways of enabling access to user systems from Web browsers. One way is to run the generated sample interface application as an applet from a Web browser. More complex and dynamic Web pages can be built by using scripting languages and the Component Enabler component interfaces to access remote Agile Business Suite applications. Web servers can use server-side scripting and Component Enabler components to dynamically build Web pages that include live data from Agile Business Suite applications.

The ASP Generator can also be used to automatically generate Active Server Pages (ASPs) from an application defined in System Modeler. The ASPs that are generated are designed to create HTML pages that are similar to the graphical forms defined in the Painter. To provide the widest possible browser type and version support, some attributes of a graphical form are not included (for example, combo boxes). However, the customer can choose to modify the source of the ASP Generator directly, using the ASP Generator Customization Kit. This allows a customer to develop their own ASP generator that suits their specific needs.

Using the Winform Container for Windows® Platform

The WinForm Container operates through the WinForm Container executable file, and is controlled by an XML configuration file. The WinForm Container executable is run from a shortcut, or from the command line with the name of the XML file appended as a parameter. The XML file is read by the WinForm Container and the instructions in the XML file to direct the WinForm Container to the correct location and system. If the shortcut does not specify an XML configuration file, the WinForm Container executable returns an error message “Unable to find a CLSID for corresponding to the system name.”

The Agile Business Suite Developer installation includes a default XML file called config.xml which is located in the bin folder of the installation path, along with the WinForm Container executable. Also, a pre-configured XML file called ispec.xml is generated when each system is built by Builder. This file is located in your systems installation Bin folder.

A different XML configuration file can be specified by entering a path or URI after the WinForm Container.exe from a command line, or in a shortcut.

For example, “WinForm Container.exe” c:\AgileBusinessSuite\payroll.xml

If you specify a URI, the WinForm Container accesses the configuration file from a central location.

For example, “WinForm Container.exe” http://tardis/payroll.xml

You can setup a different shortcut for each system you may access, and for users who are required to access certain ispecs, reports and commands.

If you only access one system at a time, the configuration file should only have one system defined in it. If you access multiple systems via the SWITCH.TO command, you must include all systems that you may switch between in the one configuration file.

Creating the XML Configuration File

When a system is built by Builder, the generated files include an XML configuration file called ispec.xml, which is located in the Bin folder under the Package Installation Directory as specified in the Folder Properties, Installation category.

For example, <Installation Folder>\Release_Windows\SAMPLE\Core\Bin for 32-bit and <Installation Folder>\Release_Windows\SAMPLE\Core\Bin64 for 64-bit. This file is configured with most tags defined and includes all the ispecs and reports in the system.

You have two options for creating a custom XML configuration file for your users:

- Use the default config.xml file mentioned previously and using a text editor, remove the small number of existing sample entries and then add your own for your generated system.
- Use the generated ispec.xml file and using a text editor, modify the entries to suit your requirements. For instance you may wish to limit the number of ispecs or reports which can be accessed by a group of users and by removing some of the existing entries to create an individual XML file for that group.

Either of the above methods is acceptable, the decision depends entirely on which is most convenient. Once you have created your customized version of the XML configuration file, save it using a suitable name, for example SalesConfig.XML, and append the file name to the WinForm Container shortcut or command line.

Here is an example of a configuration file with a single system defined in it.

```
<?xml version="1.0" encoding="UTF-16"?>
<configuration xmlns="http://tempuri.org/config.xsd" version="1.0">
    <startupSystem name="Sample"></startupSystem>
<system name="Sample">
    <componentName>SAMPLE</componentName>
    <runtimeServer>localhost</runtimeServer>
<downloadServerURI>file://
C:\MySystem\Sample\Release_Windows\Sample\Interfaces\WinForms\Bin</
downloadServerURI>
    <localDownloadDirectory>.\</localDownloadDirectory>
    <defaultImageType>.jpg</defaultImageType>
    <defaultImageSubDirectory>images\</defaultImageSubDirectory>
    <defaultListType>.xml</defaultListType>
    <defaultListSubDirectory>lists\</defaultListSubDirectory>
    <disableConsole>true</disableConsole>
    <GIWCompatibleBehaviour>false</GIWCompatibleBehaviour>
```

```
        <enableLogging>true</enableLogging>
        <logFile>.\client.log</logFile>
        <windowState>Normal</windowState>
       <clientSize>800,600</clientSize>
       <useComputerNameForStation>false</useComputerNameForStation>
       <stationName></stationName>
        <ispecs>
        <ispec name="Menu" description="The description for the Menu class">
        </ispec>
    </ispecs>
        <reports>
        <report name="Report1" description="The description for the Report1 class">
        </report>
        <report name="Report2" description="The description for the Report2 class">
        </report>
        </reports>
        <commands>
        </commands>
        <languages>
        </languages>
    </system>
    </configuration>
```

Note: *For 64-bit, you need to modify the <downloadServerURI> entry with the following:*

```
<downloadServerURI>
            file://C:\ROC\Release\ROC18\Interfaces\WinForms\Bin64
            </downloadServerURI>
```

Refer to the table which describes the meaning of the XML tags.

Meaning of the XML Tags

The following table describes all the XML tags used in the XML configuration file:

XML Tag Pair	Description
<?xml version="1.0" encoding="UTF-8"?>	Do not modify this line, it contains information about the level and type of the xml format.
<configuration xmlns="http://tempuri.org/config.xsd" version="1.0">	Do not modify this line as it contains information about the level and type of the xml format.
<startupSyste" name="Sample"></startupSystem>	Identifies the <system> that should be started when the WinForm Container is run. If this line is omitted, then the first <system> block that is encountered is run. The name entered here must match the name given in one of the <system> blocks in the configuration file. Change it to be the same as the "system name" of the system you want to start up with. This is to allow for SWITCH.TO, where multiple systems are in one config file.
<system name="Sample">	Is the start of the block of data for the named system. In this case, MySystem. This 'name' is displayed on the title bar of the WinForm Container, and should be a user friendly name. The actual name of the runtime system is identified in a different line. Change this to the name of your system.

XML Tag Pair	Description
<componentName>SAMPLE </componentName>	This is the deployed COM+ system name of the generated runtime. The simplest way to find this is to open the 'component services' management console on the machine that the runtime was generated to, then Component Services/ Computers/My Computer/COM+ Applications. Find your generated system. The name of your COM+ application is what you should enter here. ***Note:*** *This name is case sensitive and must correspond to the case of the actual name of the folder.*
</componentName> <runtimeServer>localhost</ runtimeServer>	Identifies the machine name that the runtime system was deployed to. This can also be the IP address of the server.
<downloadServerURI>file:// C:\MySystem\ Sample\SampleDeploy\Release\ Sample\ Interfaces\WinForms\Bin </downloadServerURI> for 32-bit or <downloadServerURI>file:// C:\MySystem\ Sample\SampleDeploy\Release\ Sample\ Interfaces\WinForms\Bin64 </downloadServerURI> for 64-bit	The name of the computer the client should look to for updated or missing files. Language numbers are added automatically and do not need to be included as part of this entry. This can be left blank, but you need to manually copy the form control files to each end user computer. This may be a http: reference, or a file: reference. For example, http:// myWebServer/myAppFiles/
<localDownloadDirectory>.\</ localDownloadDirectory>	Identifies the directory on the local computer where the WinForm Container expects to find the individual forms, images and external list boxes. This is also the location that forms are downloaded to. This is normally the current directory. For example, .\
<defaultImageType>.jpg</ defaultImageType>	Identifies the file extension to use if an image name does not include one. This is generally the .jpg or .gif .jpg is best for photo type images, but .gif are better for graphic type images.
<defaultImageSubDirectory>im ages\ </defaultImageSubDirectory>	The sub-directory under the 'localDownloadDirectory' in which to place the images.
<defaultListType>.xml</ defaultListType>	Controls the file extension to use when downloading 'external' list files. External list files are list data files that are not created by the sendlist.static or sendlist.dynamic command, but are created by hand.
<defaultListSubDirectory>imag es\</defaultLis tSubDirectory>	The sub-directory under the 'localDownloadDirectory' in which to place external lists.

XML Tag Pair	Description
<disableConsole>true</ disableConsole>	Prevents a user for accessing the console dialog, and issuing colon commands. This option must be set to false if you want the end user to be able to run reports, or interact with reports. If this item is set to true, the Command Console, Select Commands and Run Reports menu items are disabled.
<GIWCompatibleBehaviour>tru e </GIWCompatibleBehaviour>	Alters the behavior of the WinForm Client to be more like that of a Graphical Interface Workbench client. This alters the way push button groups are handled, from a Windows standard way (no data sent unless a button in that group is pressed) to a GIW way (data is sent for every button group, unless a button in a different button group is pressed).
<enableKanji>true</enableKanji	Enables the correct handling of Kanji data in a Segment that has NationalString Nodal property under the National Support category set to “MultiByte”. In particular, it ensures that Kanji data sent to the Winform clients via the SendListDynamic and SendListStatic commands are properly encoded and displayed in list boxes.
<enableLogging>true</ enableLogging>	Controls logging. This is a diagnostic log, and should only be enabled when you are trying to track down a particular problem.
<windowState>Normal</ windowState>	Specifies the initial state of the Winform window upon the Winform Container being started. Valid values are “Normal”, “Maximized”, and “Minimized”.
<clientSize>800,600</ clientSize>	Specifies the initial size of the Winform window upon the Winform Container being started.
<useComputerNameForStation >false </ useComputerNameForStation>	Determines whether to use the local computer name as the station name for the Winform client session. By default, the Winform user’s Windows user ID is used as the station name.
<stationName>	Specifies the station name to use. If specified, this value overrides the default station name and/or the <useComputerNameForStation> setting above if True.
<logFile>.\client.log</logFile>	Identifies the location of the WinForm Container log file if the enableLogging tag is true in the config.xml file. The log file also contains masked data for attributes that have the EnableMaskDefinition property set to true and for attributes that have the ControlType property set to PasswordField.
<ispecs>	Do not modify. This tag identifies an <ispecs> block. It surrounds a list of the ispecs shown in the 'Open Form' dialog. If the <ispecs> block is empty, the select forms menu item is disabled in the WinForm client.
<ispec name=”Menu” description=” The description for the Menu class”>	Each <ispec> item identifies an ispec and description that appears in the list of forms in the WinForm Container. Add additional ispecs as required.

XML Tag Pair	Description
</ispec>	***Note:*** *The individual closing tag for each ispec may be omitted if each <ispec tag is terminated with the closing character of />.*
</ispecs>	Do not modify, as this tag identifies an <ispecs> block. It surrounds a list of the ispecs shown in the 'Open Form' dialog. If the <ispecs> block is empty, the select forms menu item is disabled in the WinForm client.
<reports>	Do not modify. This tag identifies a <reports> block. This block encompasses a list of reports the user is shown in the Report dialog. If the <reports> block is empty, the run reports menu item is disabled in the WinForm client.
<report name="report1" description= "The description for the report1 report">	Each <report> item identifies a report and description that appears in the run reports dialog. Add additional reports as required.
</report>	***Note:*** *The individual closing tag for each report may be omitted if each <report tag is terminated with the closing character of />.*
<report name="report2" description= "The description for the report2 report">	Each <report> item identifies a report and description that appears in the run reports dialog. Add additional reports as required.
</report>	***Note:*** *The individual closing tag for each report may be omitted if each <report tag is terminated with the closing character of />.*
</reports>	Do not modify. This tag identifies a <reports> block. This block encompasses a list of reports the user is shown in the Report dialog. If the <reports> block is empty, the run reports menu item is disabled in the WinForm client.
<commands>	Do not modify. This tag identifies a <commands> block.If the <commands> block is empty, the System Commands menu item is disabled in the WinForm client.
<command name="who" description="list of who is on the system">	Each <command> item identifies a colon command, and a description that appears in the System Command dialog. Add additional commands as required.
</command>	***Note:*** *The individual closing tag for each command may be omitted if each <command tag is terminated with the closing character of />.*
</commands>	Do not modify. This tag identifies a <commands> block. If the <commands> block is empty, the System Commands menu item is disabled in the WinForm client.

XML Tag Pair	Description
<languages>	Do not modify. This tag identifies a <languages> block. If this block is empty, the Change Language menu item is disabled. This block is only required if your end users need to change between languages.
<language name=”English” ID=”1033” / >	The languages must reflect the languages that you have defined in your AB Suite system. The name and ID should match the names and IDs of the languages you have defined in your runtime system. Add additional languages as required.
</language>	***Note:*** *The individual closing tag for each language may be omitted if each <language tag is terminated with the closing character of />.*
</languages>	Do not modify. This tag identifies a <languages> block. If this block is empty, the Change Language menu item is disabled. This block is only required if your end users need to change between languages.
</system>	Defines the end of this particular system. If you switch between more than one system using the SWITCH.TO command in you runtime logic, you must define a <system> block for each system that may be switched into.
</configuration>	Identifies the end of the configuration block.

Using the WPF Client Container for Windows Platform

The WPF Client Container operates through a WPF Client executable file and is controlled by an RTXML configuration file.

The WPF Client executable requires a start parameter to be supplied that contains the location of the executable file and the location of the RTXML file. A default version of the RTXML file is generated with each WPF Technology folder specified in the System Modeler. The RTXML file is placed in the “Access Layer API Deploy” folder, which is located under the Project folder for the WPF project, and is named <TechnologyFolderName>_Config.rtxml.

When the WPF Client executable starts, it reads the content in the <TechnologyFolderName>_Config.rtxml file and establishes a session with the host application.

The WPF Client Container executable is run through either of the following:

- A desktop shortcut
- A System Modeler solution

To run the WPF Client Container executable by using the desktop shortcut, perform the following:

1. Create a shortcut of the WPF Client Container executable on your desktop by performing the following,
   a. Right-click the **WpfClient.exe** file located in the Bin folder.
   b. Select **Create Shortcut** from the context menu that appears.

   The WPF Client Container shortcut icon appears on your desktop.
2. Right-click the shortcut and select **Properties**.

   The Properties window appears.
3. In the **Target** box, enter the location of the WPF Client executable and the location of the <TechnologyFolderName>_Config.rtxml file.

   For example, "<WPFClient Installation Path>\WpfClient.exe" "C:\Users\<UserName>\Documents\Visual Studio 2015\Projects\<TechnologyFolderName>\Access Layer API Deploy\WpfClient_Config.rtxml"

   Note: *If you do not specify the location of the WPF Client executable and the RTXML configuration file in the desktop shortcut, the error message "Configuration file missing the minimum required settings! Closing down..." appears.*
4. Click **OK**.

You can setup different shortcuts for each system you may access or for accessing only a certain ispec, report, or command.

To run the WPF Client Container executable using the System Modeler solution, perform the following:

1. Right-click the **<SystemName>.<TechnologyFolderName>.Views** project in the Solution Explorer window, and then select **Properties**.

   The <SystemName>.<TechnologyFolderName>.Views tab page appears.
2. From the left pane, select **Debug**.
3. Select the **Start external program** option, and then enter the location of the WPFClient.exe file in the corresponding box.

   For example, C:\Program Files (x86)\Unisys\AB Suite 6.1\Bin\WpfClient.exe.
4. In the **Command line segment** box, enter the location of the RTXML file.

   For example, C:\Users\appadminuser\Documents\Visual Studio 2015\Projects\<Project_Name>\Access Layer API Deploy\<TechnologyFolderName>_Config.rtxml

5. Right-click the **<SystemName>.<TechnologyFolderName>.Views** project in the Solution Explorer window, and then select **Set as StartUp Project**.

6. Right-click the **<SystemName>.<TechnologyFolderName>.Views** project, point to **Debug**, and then select **Start New Instance**.

   If you access only one system at a time, the RTXML configuration file should only have one "system" defined in it. If you access multiple systems, a switch directive is sent from a host application, which necessitates you to configure the required systems. This loads the required DataModel, DataViewModel, and Views assemblies that are configured for the system that you want to switch to. You must include all systems that you may switch between in the configuration file. Refer to the *Agile Business Suite Client Framework Programming Reference Manual* for more information on the WPF Client Container and the functions with the WPF Client.

When the WPF Client starts, it reads the specified <TechnologyFolderName>_Config.rtxml file to set up the connection parameters required to connect to the runtime system through the Access Layer API. The RuntimeServer locates the machine where the runtime system is deployed. The DownloadServerUrl and Assemblies configuration information locates the required DataModel, DataViewModel, and Views assemblies and dynamically loads them into the WPF Client application.

After a connection has been made to the runtime system, the toolbar controls and the Selection dialog box appear. The Selection dialog box displays a list of the available ispecs that you can select. Selecting an ispec from this dialog box executes a recall of the ispec in the host application, and then the corresponding screen appears in the WPF Client Container.

Customizing the RTXML Configuration File

You can create a custom RTXML configuration file by opening the default <TechnologyFolderName>_config.rtxml file by using a text editor, modifying the default entries that are generated for the AB Suite system, and saving the configuration file with a different name.

Note: *If you make custom changes to the RTXML file, it is important to save the file with a different name, because subsequent builds of the AB Suite system regenerates the file with default values and the custom changes made by you is lost. If there are changes in the AB Suite model that introduce additional content into the generated configuration file, you must merge these changes into your custom configuration file.*

The following is an example of the RTXML configuration file with a single system defined in it:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration xmlns="http://tempuri.org/config.xsd" Version="1.0">
  <StartupSystem Name="Sample" />
  <System Name="Sample">
    <ComponentName>SAMPLE</ComponentName>
    <RuntimeServer>localhost</RuntimeServer>
    <DownloadServerURI>C:\Users\Administrator\documents\visual studio
2015\Projects\WPF_Test\Access Layer API Deploy</DownloadServerURI>
    <DownLoadCredentials>
</DownLoadCredentials>
    <FileRepositoryDirectory>WPF_Test\Images</FileRepositoryDirectory>
    <Assemblies>WPF_Test.WPFClient.DataModels.dll,
WPF_Test.WPFClient.DataViewModels.dll, WPF_Test.WPFClient.Views.dll</Assemblies>
    <LocalDownloadDirectory>
    </LocalDownloadDirectory>
    <EnableKanji>false</EnableKanji>
    <DisableConsole>false</DisableConsole>
    <!--Valid values are Error, Info, Debug-->
    <LogLevel>Error</LogLevel>
    <LogFolder>c:\temp</LogFolder>
    <UseComputerNameForStation>false</UseComputerNameForStation>
    <StationName>WIN-IQFR1DJ1OU5</StationName>
    <SupportAutoTabbing>true</SupportAutoTabbing>
    <OffLine>false</OffLine>
    <IsAnonymous>false</IsAnonymous>
    <ForceLogin>false</ForceLogin>
    <UseGateway>false</UseGateway>
    <GateWayAddress>127.0.0.1</GateWayAddress>
    <ShowClosingDownMessageForBye>true</ShowClosingDownMessageForBye>
    <Reports />
<Languages>
    <Language Name="English" Locale="1033">
    <Language Name="Dutch" Locale="19">
</Languages>
<ATTRecord Enable="false" Mode="connect" Server="127.0.0.1" Port="8888">
      <File>C:\Temp\Recordings\Testcase.smtest</File>
      <RecordDynamicLists>false</RecordDynamicLists>
    </ATTRecord>
  </System>
</Configuration>
```

Meaning of the RTXML Tags

The following table describes all the RTXML tags used in the RTXML configuration file:

RTXML Tag Pair	Description
<?xml version="1.0" encoding="UTF-8"?>	Do not modify this line as it contains information about the level and type of the rtxml format.
<configuration xmlns="http://tempuri.org/config.xsd"version="1.0">	Do not modify this line as it contains information about the level and type of the rtxml format.
<startupSystem Name="Sample" />	Identifies the <system> that should start when the WPF Client Container is run, if there are multiple systems declared in the config file. If this line is omitted, then the first <system> block that is encountered is run. The name entered here must match the name given in one of the <system> blocks in the configuration file. Change the name to the "system name" of the system that you want to start.
<system Name="Sample">	This is the descriptive name of the system and appears on the title bar of the WPF Client Container window. It marks the start of the block of data for the named system. Change this to your system name.
<componentName> SAMPLE </componentName>	This is the name of the component installed in COM+ for the deployed system. It is typically the name of the segment defined in System Modeler. ***Note:*** *This name is case-sensitive and must correspond to the actual name of the folder.*

RTXML Tag Pair	Description
<runtimeServer>localhost </runtimeServer>	This is the machine name that the AB Suite application has been deployed to. This can also be the IP address of the server.
<downloadServerURI>C:\Users\Administrator\documents\visual studio 2015\Projects\WPFClient\Access Layer API Deploy</downloadServerURI>	This is the location that specifies from where the dependent components, such as DataModels, DataViewModels, or Views assemblies, are downloaded.
<downLoadCredentials> </downLoadCredentials>	This allows you to specify the FTP usercode and password, if you are using the FTP Server location for the DownloadServerURI.
<fileRepositoryDirectory>WPFClient\Images</fileRepositoryDirectory>	Specifies the location from where the dependent files such as images are retrieved. It can be on the same machine or on a remote server. The FileRepository interface of the Access Layer API transfers the dependent files from the specified location.
<assemblies> WPFClient.WPFClient.DataModels.dll, WPFClient.WPFClient.DataViewModels.dll, WPFClient.WPFClient.Views.dll </assemblies>	This defines the dependent assemblies required by the WPF Client application. This tag is pre-filled with the names of the DataModels, DataViewModels, and Views assemblies for a WPF Technology folder.
<localDownloadDirectory> C:\ProgramData\Unisys\ABSuite\6.1\WpfContainer Downloads\Sample </localDownloadDirectory>	This specifies the location of the local machine to which the dependent files are downloaded for use by the WPF Client application.
<enableKanji> false </enableKanji>	Enables the correct handling of Kanji data in a segment that has NationalString Nodal property under the National Support category set to “MultiByte”. In particular, it ensures that Kanji data sent to the WPF Clients through the SendDynamic and SendStatic commands are properly encoded and displayed in list boxes.
<disableConsole>false </disableConsole>	Prevents a user from accessing the console processing window in the WPF Client application. This option must be set to False if you want the end user to be able to run reports, or interact with reports. If this item is set to True, the Command Console, Select Commands, and Run Reports menu items are disabled.
<logLevel>Error</logLevel>	Sets the log level for the Access Layer API component. The valid values are Error, Info, and Debug.
<logFolder>C:\Temp </logFolder>	Specifies the location of the log file created by the Access Layer API logging process. By default, the location of the log file is C:\Temp.

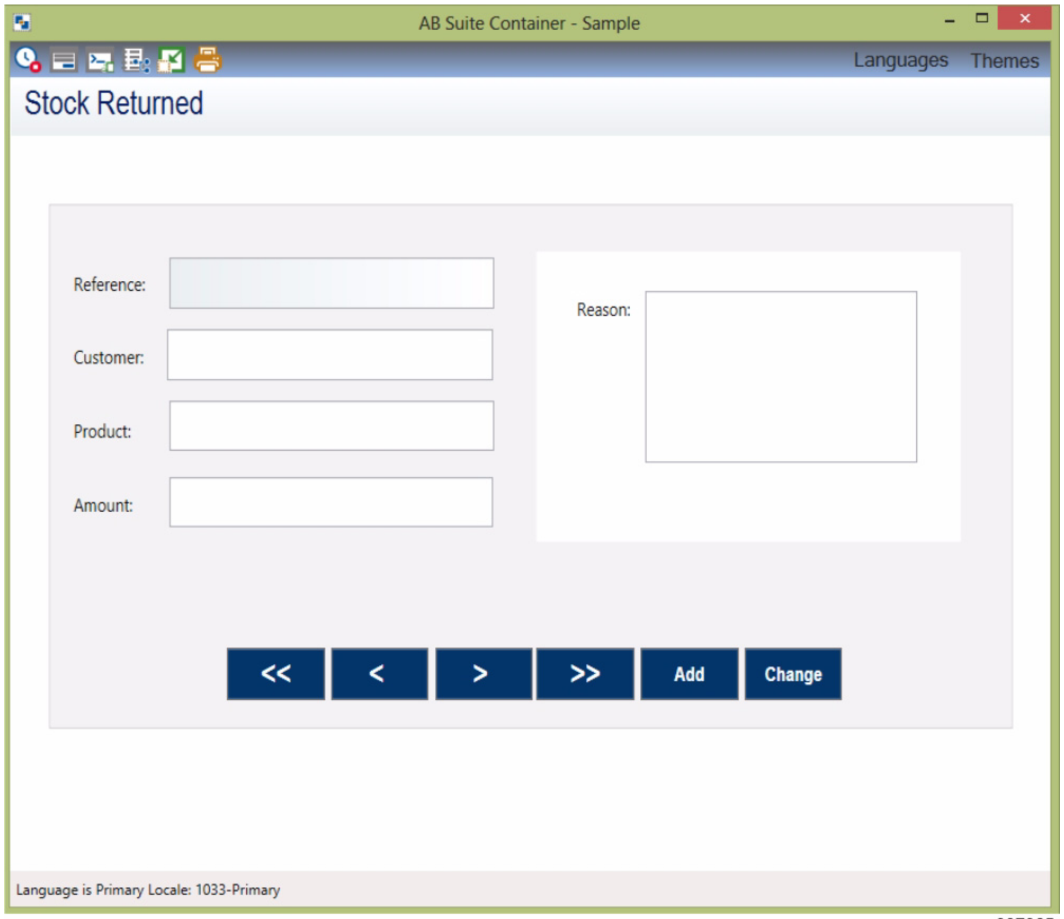
RTXML Tag Pair	Description
<useComputerNameForStation> false </ useComputerNameForStation>	Determines whether to use the local computer name as the station name for the runtime system. By default, the value is False.
<stationName>WIN-IQFR1DJ10U5</stationName>	Allows you to specify a station name for a machine running the WPF Client application. If specified, this value overrides the default station name (which is the current users Windows user name) and/or the <useComputerNameForStation> setting, if it is set to True.
<supportAutoTabbing>false </supportAutoTabbing>	Specifies whether the focus should move to the next field in the tab order when a field is filled according to the defined field length. By default, the value is False.
<offLine>false</offLine>	This allows you to run the WPF Client application in offline mode. It does not try to establish a connection with the runtime system if this tag is set to True. By default, the value is False.
<isAnonymous>false </isAnonymous>	Specifies whether the connection should be made anonymously. This allows you to run multiple WPF Client sessions to the runtime system simultaneously on the same machine. By default, the value is False.
<forceLogin>false </forceLogin>	An existing session for the same user can be overridden with a new connection to the runtime system if this tag is set to True. By default, the value is False.
<useGateway>false </useGateway>	Allows you to connect to the runtime system using the AB Suite WCF Gateway service, instead of using the COM/ DCOM directly from the WPF Client application. By default, the value is False.
<gateWayAddress>127.0.0.1 </gateWayAddress>	Specifies the address of the machine where the WCF Gateway is running.
<showClosingDownMessageForBye> true </ showClosingDownMessageForBye>	Specifies whether a message box should appear when an application is closed due to a Bye request received from the runtime system. By default, the value is True.
<reports> </reports>	Do not modify this tag as it identifies a <reports> block. This lists the reports generated for an application. The reports can be initiated from the WPF Client. If the <reports> tag is empty, the run reports menu item is disabled in the WPF Client.
<languages> </languages>	Do not modify this tag as it identifies a <languages> block. If this block is empty, the Change Language menu item is disabled. This block is only required if you need to switch between languages.
<language name="English" ID="1033" / >	Specifies the languages that you have defined in your AB Suite Client Framework application.

RTXML Tag Pair	Description
</language>	Defines the end of language block. ***Note:*** *The individual closing tag for each language may be omitted if each language tag is terminated with the closing character of /.*
</languages>	Defines the end of languages block.
<ATTRecord Enable="false" Mode="connect" Server="127.0.0.1" Port="8888">	This allows you to define the ATT Recording options to capture the ATT Test cases by using the WPF Client. The recorded test cases can be replayed using ATT through the Access Layer API.
<file>C:\Temp\Recordings\Testcase.smtest</file>	Specifies the path and name of the file to which the Test Case is recorded when ATT is in the Disconnect mode. For example, C:\Temp\Recordings\Testcase.smtest.
<recordDynamicLists> false </recordDynamicLists>	Specifies whether ATT record is Dynamic List Data or not. By default, the value is False, because this recording option adds an overhead to the ATT test case processing and should only be turned on if required.
</ATTRecord>	Defines the end of the ATT record block.
</system>	Defines the end of this particular system. If you switch between more than one system by using the SWITCH.TO command in your runtime logic, you must define a <system> block for each system that may be switched to.
</configuration>	Defines the end of the configuration block.

Using the WPF Client Container User Interface

The WPF Client Container displays the designed Views based on the DataViewModels generated by System Modeler when the Access Layer components are built for the WPF/XAML Client technology. The Views can be designed by using the WPF Designer (or Blend) and deployed for use by the AB Suite WPF Client application, which is a WPF Container solution provided by Unisys. The designed Views are compiled into a Views assembly that is loaded by the WPF Client at runtime. When a response is received from the host application for an ispec, the corresponding user interface is displayed in the Content Control of the WPF Client Container.

The following figure shows a sample WPF Client Container:



007065

The WPF Client Container comprises a title bar, a tool bar, and a status bar.

Title Bar

The title bar of the WPF Client Container displays

- The WPF Client icon
- The WPF Client Container name

  This name can be localized in the string resources of the WPF Client application.

  For example, The WPF Client Container name can be AB Suite Container.
- The runtime system name

  This is the system name that is specified in the <TechnologyFolderName>_Config.rtxml file. The name of runtime system in the above image is TestListSeg.

Toolbar

The toolbar displays various icons and menus that you can use to perform operations, such as opening or closing a session, running a report, and selecting a theme.

The icons present on the WPF Client Container tool bar along with their respective description are listed in the following table:

Icon	Name	Description
🕒	Open Session	Displays this icon when the WPF Client is connected to the runtime. This allows you to close the session.
🕒	Close Session	Displays this icon when you close the session. This allows you to open a new session for one of the systems configured in the config.rtxml file. When you open a session the Selection dialog box appears listing all the configured systems that are available. To initiate the connection with the host application, select a system from the **Selection** dialog box, and then click **OK**. ***Note:*** *When the WPF Client is not connected to a host system, all other Toolbar icons are hidden.*
▤	Select a Screen	Allows you to display a specific screen in the WPF Client Container. When you click this icon a Selection dialog box appears listing the screens that you can display. The Selection dialog box contains two tabs • Screen – This tab displays the list of ispecs specified for a WPF Client Technology folder. To open an ispec listed in the **Screen** tab, select an ispec, and then click **OK**. The selected ispec appears in the WPF Client Container window. • Teach – This tab displays the list of Help screens associated with the WPF Client Technology folder. To open a screen listed in the **Teach** tab, select a Help screen, and then click **OK**. The selected Help screen appears in a new window. You can use the Search box in the Selection dialog box to search for a specific ispec or Help screen.

Icon	Name	Description
	Show Console	Displays the Console window. The Console window contains two tabs • Cmd Output/Responses – This tab displays colon commands that can be sent to the runtime system. • Report Output – This tab displays the report output that arrives asynchronously from the runtime system. It displays video reports sent to the WPF Client. ***Note:*** *If you do not want to provide access to the console window you can choose not to display the Show Console icon by setting the DisableConsole tag to True in the <TechnologyFolderName>_Config.rtxml file.*
	Run Report	Displays the Selection dialog box with a list of reports that you can execute at runtime. This dialog box also allows you to • Search for a specific report in the list. • Set the device type for the report output. • Select a language for the report. • Enter report parameters required by the report. To run a report, select a report from the **Selection** dialog box, and then click **OK**. The report output appears in the Console window. ***Note:*** *The list of reports is currently extracted from the report entries configured in the <TechnologyFolderName>_Config.rtxml file.*
	Print Screen	Allows the user to print the screen in the current WPF Client Container window.
	Snap to Designed Size	Allows you to change the screen size back to the dimension of the originally designed View. The ContentControl that displays the View inside the WPF Client Container is contained within a ViewBox control. This allows you to resize the WPF Client Container window. When the WPF Client Container window is resized, all controls inside the ViewBox are automatically scaled so they appear larger or smaller, depending on the new dimensions of the window. ***Note:*** *If you resize the WPF Client Container window to a certain dimension, and then close and reopen the window, it appears with the resized dimension.*

The menus present on the WPF Client Container tool bar along with their respective description are listed in the following table:

Menu	Name	Description
Languages	Language	Allows you to select a language that has been implemented for an AB Suite Client Framework application. When you select a specific language, a request is sent to the runtime to switch to the specified language. The WPF Client application also switches to use resources specific to that language based on the locale of the specified language. The Language menu displaying the languages are created dynamically to show the languages included in the application. ***Note:*** *If you have configured only one language for the application, then the Language menu does not appear on the toolbar.*

Menu	Name	Description

Themes	Themes	Allows you to select a theme for the WPF Client. Selecting a theme loads the stylesheet associated with the theme. This formats the control types and other attributes in the screen as specified in the stylesheet.
		The WPF Client includes built-in themes for Dark and Light color combinations, which you can apply when running the application.
		When developing the Views for the WPF Client or when converting the AB Suite model to the AB Suite Client Framework model, you must customize the themes to suit your requirement or create new themes.
		To customize the Dark or Light themes, you must manually update the Theme_Dark.xaml or Theme_Light.xaml file.
		For example, if you want to apply the Light theme to the text box graphic control,
		1. In the Solution Explorer window, go to **WpfClient (WPF(Windows Presentation Foundation))** > **Sample.WpfClient.Views** > **Themes**.
		2. For the location, open the Theme_Light.xaml file.
		3. Specify a value for background (for example, Blue) as shown in the following code snippet:
		`<!- - Style for TextBox Controls- - >`
		`<Style x:Key = "TextBox_Style_01" Target Type = "{x:Type TextBox}" BasedOn = "{StaticResource BaseTextBox }">`
		`<Setter Property = "Background" Value="Blue" />`
		The textbox color changes to blue when you build the model and select the Light theme from the Themes menu.
		Note: *You need not customize the themes in the Sample Client Framework model as the selected theme loads the stylesheet associated with it and formats the graphic controls in the View as specified in the stylesheet.*
		The Themes menu displaying the themes are created dynamically to show the themes included in the Resources dictionary.
		Note: *If you have included only one theme in the Resources dictionary, then the Themes menu is not displayed on the toolbar.*

Status Bar

The status bar displays status text and error messages sent by the host application. If the host application sends multiple error messages, the status bar displays the message “Multiple Errors Received…”, and an Expand icon appears on the status bar. This allows you to expand the status bar upwards and view the error messages. However, if the host application sends a single error message, the Expand icon does not appear.

Note: *The status bar also displays unsolicited messages.*

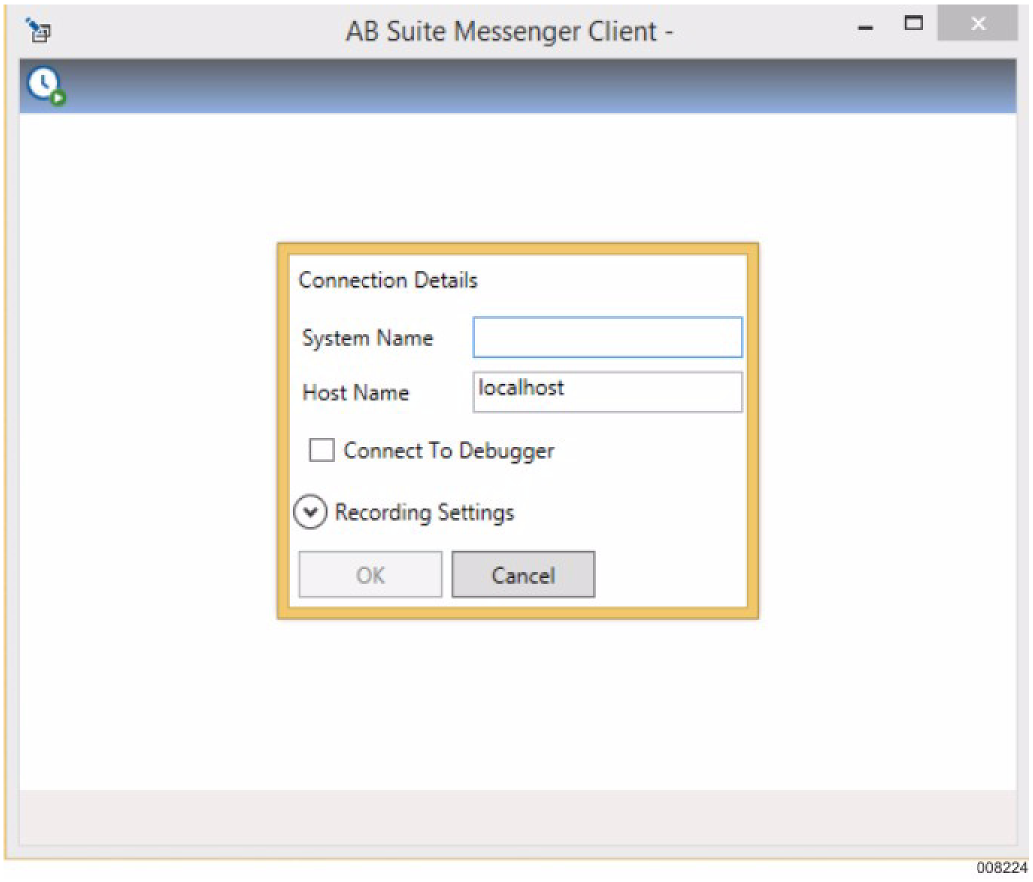
Using the Messenger Client for Windows Platform

The Messenger Client operates through a Messenger Client executable file. The Messenger Client executable file is a simple program that you can use to submit individual XML messages to the runtime system. The Messenger Client allows you to test the processing of XML messages during development.

You can run the Messenger Client either through a Graphical User Interface (GUI) or a command line interface.

The Messenger Client appears automatically when you build the XML Framework application after setting the debugger configuration property to either Messenger or External Application. You can also run the Messenger Client from the Start menu by pointing to **Start** > **Programs** > **Agile Business Suite 6.1** > **Development Environment** > **Messenger Client**.

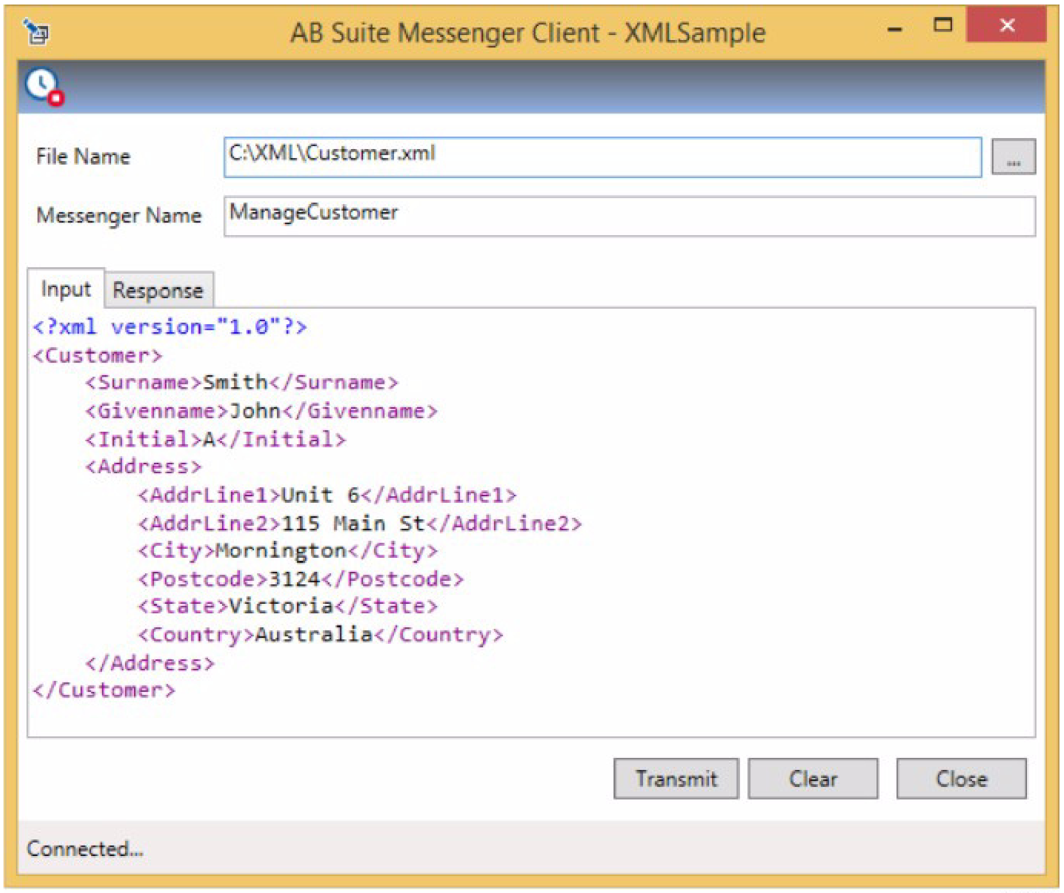
The AB Suite Messenger Client window appears.



To initiate a connection to the runtime system, perform the following:

1. In the **System Name** box, enter the system name.
2. In the **Host Name** box, enter the host name.
3. Select the **Connect to Debugger** check box if you have set the debugger configuration property to Messenger or External Component.
4. Click **OK**.

The AB Suite Messenger Client window appears.



C:\XML\Customer.xml

ManageCustomer

```
<?xml version="1.0"?>
<Customer>
    <Surname>Smith</Surname>
    <Givenname>John</Givenname>
    <Initial>A</Initial>
    <Address>
        <AddrLine1>Unit 6</AddrLine1>
        <AddrLine2>115 Main St</AddrLine2>
        <City>Mornington</City>
        <Postcode>3124</Postcode>
        <State>Victoria</State>
        <Country>Australia</Country>
    </Address>
</Customer>
```

008225

You can now submit XML messages. To submit XML messages, perform the following:

- In the **File Name** box, enter the location of the XML file or browse to and select the location of the XML file by clicking <add button>.

  The **Messenger Name** box defaults to the name of the root node when you open a file containing an XML message. This may or may not be the name of the Messenger class that you want to use.

You can directly input the XML message by performing the following:

1. In the text area of the Input tab, enter the XML message.

   Note: *You can edit the message before submitting it, if required.*

2. In the **Messenger Name** box, enter the Messenger class name.
3. Click **Transmit**.

   Note: *Click **Clear** to clear the text area. Click **Close** to close the Messenger Client window.*

The success of the transaction appears in the status bar of the AB Suite Messenger Client window. If the Messenger class returns a response XML message it appears in the Response tab.

You can run the MessengerClient.exe from the command line to submit XML messages without using the GUI interface. When the MessengerClient.exe is run from the command line all the input parameters can be supplied as command line parameters.

The syntax for running the MessengerClient.exe from the command line is as follows:

```
MessengerClient /s <System Name> /c <Class Name> /f <File Name>
```

You can include the following parameters:

- /s <System Name> – Name of the deployed system
- /h <Host Name> – Machine name of where the system is deployed
- /c <Class Name > – Name of the Messenger class to transmit
- /f <File Name> – File containing the data to transmit
- /d – Use Debugger
- /? – Show Help

Builder Functions

Builder is responsible for the following functions:

- Validate the consistency and semantics of the model.
- Verify that all applicable logic in the model has been validated and is error free.
- Determine what has changed in the model since the last time it was generated to the selected configuration.
- Schedule and execute the generation of the necessary files.
- Invoke the compilation process for the generated source code.
- In Windows® platform, builder transfers the deployment package to the target server.

To accomplish these tasks, Builder:

- Interacts with the Model database via the Model component.
- Interacts with the Developer (Visual Studio) environment via the Configuration Builder component.
- Interacts with the deployment service via the Configuration Deployer component.

Builder Architectural Elements

Builder contains the following architectural elements:

- Model validation
- Logic validation
- Change analysis
- Code optimization
- Generation management
- Task list management
- Code generation
- Database schema generation
- Deployment control-file generation as needed
- Compilation management and monitoring
- Deployment initiation and monitoring

While the general approach to the design of Builder is similar to Direct Generate in previous releases of Enterprise Application Builder, and uses many of the ideas used in that component, a new design builds on those ideas and introduces other concepts.

The most important of these design concepts is:

- Abstraction and separation of all specific language constructs to be generated from the code which generates them. This allows the generation process to become more generalized thus allowing easier adoption of new runtime processing paradigms and/or generation of different intermediate languages.

The design of the Generation process is based on a combination of our previous experience in generation and the results of investigations, which were carried out to determine a more effective ways to generate. Of significance is the need to target multiple languages and run-time technologies (in separate Generates). This is done using a single generate 'engine' by clear separation and enforcement of the boundaries between the target language code to be generated and the generation process.

The following principles guide the generation design:

- No target language code or assumptions are contained in the generate engine.
- Code to be generated is as minimal as possible.
- The same output artifact is not generated more than once in any generate.
- In association with the Parser, and other components which use it, a 'smart' Abstract Syntax Tree (AST) is designed which provides a more efficient means of generating and executing (debugging/testing) user logic.

MCP Builder

The MCP Builder is invoked when an AB Suite application is built using an MCP configuration. MCP Builder generates COBOL source files, ALGOL source files and other resources from the business model created in AB Suite Developer. While building an AB Suite application, the generated files and resources are transferred to the host machine where they are compiled and deployed into a MCP application. MCP Builder also generates the database schema of the AB Suite application.

MCP Build Process

The MCP Build process comprises of the following phases:

- Build
- Deployment

The Deployment phase comprises of the following sub-phases:

- Compile and Bind (for system and report builds)
- System Deployment (for system builds only)

The Deployment phase may be delayed by using the ‘Delay Compile and Deploy’ option on the Host Login dialog.

If required, the Deployment sub-phases may be restarted from the Application Builder Server on the host. The ability to restart the Deployment sub-phases is dependent on a successful Build phase.

MCP Build Options

When an AB Suite application is built using an MCP configuration, the Host Login dialog box displays deployment options specific to MCP Runtime. The build options are:

- Retain Existing Database
- Usercode and Password
- Accesscode and Password
- Chargecode
- Use Host details for FTP
- FTP Usercode and Password
- FTP Chargecode
- Delay Compile and Deploy
- Deploy Date
- Deploy Time

These options are used to specify whether to retain an existing database, where and when the application is deployed. The following table describes the different options:

Property	Function
Retain Existing Database	When Retain Existing Database is checked, database of the application being built is retained if it already exists. When Retain Existing Database is unchecked, a new database is created. If a database with the same name already exists, it is overwritten. You are prompted to confirm your action.
Usercode and password	Usercode specifies the host usercode under which the application is built. This usercode is used as the FTP account if the 'Use Host Details for FTP' box is checked.
Accesscode and password	If the host usercode under which the application is deployed has an accesscode associated with it, the accesscode and password must be entered to allow deployment to proceed.
Chargecode	If the host usercode under which the application is deployed has a chargecode associated with it, the chargecode must be entered to allow deployment to proceed.
Use Host Details for FTP	When 'Use Host Details for FTP' is checked, Usercode is used as the account for the transfer. When 'Use Host Details for FTP' is unchecked, 'FTP Usercode' is used as the account for the transfer.
FTP Usercode and password	FTP Usercode specifies the host usercode to be used as the FTP account if the 'Use Host Details for FTP' box is unchecked.
FTP Chargecode	If the 'FTP Usercode' has a chargecode associated with it, the chargecode must be entered to allow deployment to proceed.
Delay Compile and Deploy	When the 'Delay Compile and Deploy' box is unchecked, the Deployment phase starts immediately after the Build phase has finished. Builder continues to monitor the deployment activities. When the 'Delay Compile and Deploy' box is checked, Builder is suspended at the end of the Build phase until the date and time specified in the Deploy Date and Deploy Time options.
Deploy Date	Date on which deployment of the application starts. Format is dd/mm/yyyy.
Deploy Time	Time at which deployment of the application starts. Format is hh:mm:ss.

Building Applications

You can build the following:

- Deployable Folders
- The model

- A report contained within deployable folders configured for report deployment
- An ispec with presentation contained within deployable folders configured for component enabler interface generation

In Builder, an application is built using the Visual Studio Environment or Builder Application.

Building using Visual Studio Environment

You can use the Visual Studio Environment to build your application.

Only folders that contain deployable elements can be built.

Deployable elements include top-level classes (that is, Segment or vanilla classes), report classes, user interfaces, reports, or Technology folders. While a number of build scenarios are possible, all other element types cannot be built, and the **Build** option is disabled.

If a folder is selected for building, but does not contain any deployable elements, nothing is built. During a build, a small window appears before the start of the build where the elements are locked. Hence, changes cannot be made to elements. Once a safe point for the build is reached, the locks on the elements are released and other users can modify the elements while build is in progress. Note that no two users can build the same deployable element at the same time.

To build applications using the Visual Studio Environment, perform the following:

1. Open the Class View window and select the model, or elements you wish to build.
2. Using the configuration properties, specify the deployment generation properties.
3. Either right-click and select **Build**, or on the **Build** menu, select **Build**.

All logging information is displayed in the Output window, while errors encountered are displayed in the Error List window.

In Windows® runtime at the end of a build, a deployment package is created.

Logging information is displayed in the Output window, while errors encountered are displayed in the Error List window. You can navigate to the corresponding line of logic from the Output window and Error list window. You can view the Help on an error item by right-clicking an error item from the Error Window and selecting Show Help.

At the end of a build, a deployment package is created.

Restrictions

The following restriction applies:

In Windows® platform, if Builder is not configured to perform a full application build and deployment, you have to manually transfer the deployment package once it is created.

About Folders

A folder can contain any type of element, however its contents may be ignored during building, depending on its configuration settings. Folders can be used to group elements or deployment packages.

Technology Folders

A technology folder includes ispecs or classes for a particular client technology. If a technology folder is created with a given name, such as MyWPF, you should be aware that this name is used in conjunction with the model segment name to qualify project names and namespaces in the generated components. This allows you to include different technology folders in the same AB Suite application (maybe with a different set of ispecs in each). The chosen technology type can be the same or different for each technology folder.

Building Reports

Reports can be built and deployed separately, or with a segment.

Each report is built and deployed as a separate component. However, if multiple reports are built and deployed at the same time, they are deployed as one batch group.

Database Generation

Database generation determines the physical database schema based on the model's class/ispec inheritance structure, configuration properties set, and in Windows® platform determines the implicit limits imposed by the target Data Manager (SQL Server).

Files are built so that during deployment, files are created, replaced, or database reorganized for your Agile Business Suite application.

Note: *In windows® platform user-maintained classes can only exist as a base class (that is, they cannot participate in any inheritance structure).*

Build Scenarios

Following is a list of common build scenarios that can be performed in Builder:

- Build Preview
- Build a System
- Build Reports
- Build a Single Report
- Either right-click the folder and select Build, or from the Build menu, select Build.
- Build Component Enabler User Interfaces
- Build AB Suite Client Framework Applications

It is assumed that a system build includes building Application Components, Database and Winforms for a segment.

Build Preview

To determine the impact of a build and view the model elements that are modified since the last build, the Build Preview option can be used. Perform the following for a Build Preview:

1. Select the deployable folder.

   Note: *Ensure that the outer folder containing the segment is also deployable.*

2. From the Build menu, choose **Preview** or choose **Folder Only**, **Preview**.

   The Build Preview window appears. This window displays details as follows:

   - Project – Name of the system modeler project
   - Configuration – Type of configuration that you have specified for the desired runtime platform
   - Build Node – Name of the folder that is used for the preview.
   - Build Impact – Percentage of changed/updated element(s) that is built
   - Build Type – Displays the type of the build. If build type is Full, all the elements within the deployable folder(s) are built. If the build type is Partial, only a part of the application is built. If the build type is Deployment, none of the elements are built. Only those Builders that are required for deployment run.
   - Database – Displays the impact on a database. You need to create a new database if it does not exist. A database is reorganized if the following changes are made to the schema:
     - A new persistent attribute is added to the model
     - Existing persistent attributes are changed
     - The database files on the Host are deleted
     - All the system files are deleted from the Host
   - Logics – Number of logic elements that have errors or need to be validated
   - A summary of the number of Ispecs/Events/Reports/Classes/SQL Scripts that need to be built and the details of the elements that have changed since the last build are displayed.

The Details pane of the Build Preview window

- Displays the list of elements to be built for a particular deployable folder.
- Displays the list of elements that have errors or require validation if any.
- Displays a Database node with a link to the Database Comparison Report that contains details of the changes made to the database.

The build preview report is saved in the project folder as a BuildPreviewSummary <Folder name> <DD-MM-YYYY> <H-M>.xps file. For example, BuildPreviewSummary SampleDeploy 10-4-2012 6-34.xps. The file name is unique and you can print this report via File > Print.

Notes:

- *XPS is Microsoft's new electronic paper format, an alternative to the PDF format. The XPS Viewer comes preinstalled with Windows 7. You can download the XPS Viewer from http://msdn.microsoft.com/en-us/library/windows/hardware/dn641615(v=vs.85).aspx*
- *The debug and information messages of a build preview output are not recorded in the build log file unless you set logging at the DEBUG Level. To set the Build Log at Debug level, select Tools > Options > System Modeler > Builder > Logging > Log Build Debugging Information.*
- *If you have made significant changes in your model elements since the last build, executing this option might take time to gather and display all the necessary details.*
- *The Build Preview option is not available for a Debugger configuration.*

The Build Preview option is only available for deployment folders. It is not available for individual elements such as Ispecs or Reports.

Build a System

To build the system, perform the following:

1. Create a folder (outer folder).
2. Add the segment into the outer folder by dragging and dropping it into the folder, or on the **File** menu, select **Add Existing Item**.
3. Right-click the folder, select **Properties**, set the following properties to **True**:
   - Deploy Application Components.
   - For Windows® platform, deploy Winform User Interface.
   - Deployment Database.
4. Set other configuration properties as appropriate.
5. Select the outer folder.
6. Either right-click the folder and select **Folder Only**, then **Build**, or on the **Build** menu, select **Folder Only**, then **Build**.

Build Reports

To build reports, perform the following:

1. Create a folder under the Segment (Segment Class).
2. Copy the reports which you wish to be built, into the folder.
3. Right-click the folder, select **Properties** to display the Properties page, set the following properties to **True**:
   - Under the General section – set Deployable.
   - Under the Build Target Filters section – set Deploy Reports

4. Set other configuration properties as appropriate.
5. Click **OK** to close the Properties page.
6. Either right-click the folder and select **Build**, or from the **Build** menu, select **Build**.

Build a Single Report

To build a single report, perform the following:

Before building a report, ensure that the report you want to build is in a folder. If not, place the report in a folder.

1. Right-click the folder containing the report that you want to build. Select **Properties** to display Properties page.
2. Select the report you want to build from the folder in Class View.
3. Right-click the folder, select **Properties** to display the Properties page.
4. Set the following properties to **True** and click **OK**:
    - Under the General section – Deployable.
    - Under the Build Target Filters section – Deploy Reports
5. Now, to build the report, you can either–
    - Right-click the report you want to build, and select Build from the menu. Or,
    - Select the report you want to build, and select Build from the menu.

Build Component Enabler User Interfaces

To build Component Enabler user interfaces only, perform the following:

1. From the Class view select the Segment (Segment Class).
2. Right-click the Segment, and select Properties to display the Properties page.
3. Under the Component Enabler User Interface section complete the following properties:
    - Application Name
    - Package Prefix
    - CE Output Directory (optional)
4. Create a folder under the Segment (Segment Class).
5. Copy the ispecs for which you wish Component Enabler user interfaces to be built, into the folder.
6. Right-click the folder, select **Properties** to display the Properties page, set the following properties to **True**:
    - Under the General section – set Deployable.
    - Under the Build Target Filters section – set Deploy Component Enabler User Interface, and set the Translation property to the required language.

7. Under the Component Enabler User Interface section set the following optional properties:
    - Generate Getters and Setters
    - Generate Views
    - User Defined View Generator. If not specified the default generator is used.
    - CE Post Build Script
    - Ispec Models Source Language
8. Set other configuration properties as appropriate.
9. Click **OK** to close the Properties page.

Either right-click the folder and select **Build**, or from the **Build** menu, select **Build**.

Build the System, Reports, and Component Enabler User Interfaces

To build the system, reports, and Component Enabler user interfaces only, perform the following:

1. Create a folder (outer folder) as described in Build the system only . Do not initiate the last build step.
2. Create a Reports folder and Component Enabler folder as described in Build reports only and Build Component Enabler user interfaces only.
3. Drag the Reports folder and Component Enabler folder into the outer folder.
4. Select the outer folder.
5. Right-click the folder, select **Properties**, set the Deployable property to **True**.
6. Either right-click the folder and select **Folder Only**, then **Build**, or on the **Build** menu, select **Folder Only**, then **Build**.

Build AB Suite Client Framework Applications

To build AB Suite Client Framework applications, perform the following:

1. Create a deployment folder.
2. Add a segment into the deployment folder by dragging the segment into the folder. Alternatively, on the **File** menu, select **Add Existing Item**.
3. Set the configuration properties of the deployment folder.

    Refer to Creating and Configuring a Deployment Folder for more information on configuring the properties of the deployment folder.

4. Set the configuration properties of the segment.

   Refer to Configuring the Segment for more information on configuring the properties of the segment.

5. Right-click the deployment folder, and select **Build**. Alternatively, on the **Build** menu, select **Folder Only**, and then select **Build**.

Automating the MCP Build using Host Responder

While building an MCP application using MSBuild.exe command line interface, a waiting entry could show up asking the MSBuild.exe to provide a valid entry. For example, the following console text requires the user to enter a positive confirmation to overwrite the database:

```
14:57:19 Validating and converting Client supplied files
14:57:20 Generate verify is suspended
14:57:20 *** DATABASE ALREADY EXISTS, TO OVERWRITE ENTER AX YES ELSE NO.
14:57:20 Valid Reply: 5903 & one of AX {reply option},DS
14:57:20 ** For AX reply option(s) see prior messages **
```

To automate the build process so that a response could be sent to the host automatically without any human intervention you can use the Host Responder feature of AB Suite.

How: AB Suite Host Responder Works

The AB Suite Host Responder Interface uses standard console input and output which allows you to define your own handler application for host messages. Following is the sequence of events that occurs during a build:

1. The Host informs Builder that a waiting entry has been encountered which requires a response to continue.
2. The Builder calls a user-defined responder (if one has been defined and configured) to process the request.
3. If the responder can handle the response then it returns the string to pass back to the host. If it cannot, then it returns an empty string and Builder waits for the entry to be resolved using a standard interactive Host session.

The Host Responder is a set of code which can be read from the STDIN and written to the STDOUT. The responder class can be written in C#, Visual Basic .NET, Jscript.NET, Perl, or Java. For examples of Host Responder class, refer to Examples of Host Responder Class.

How To: Automate the Build Process for an MCP Platform

In this section we automate the build process of a model in an MCP platform using a Perlscript responder.

Using Builder Application

1. Create a host responder class and save the code file (for example Responder.pl) in the Bin directory of the AB Suite installation (C:\Program Files\Unisys\NGEN\Bin). For a sample Perl responder class, refer to Examples of Host Responder Class.
2. Copy the file MSBuildSettings.xml located in the Bin directory and save it in a temporary folder, for example in C:\Temp.
3. Open the MSBuildSettings.xml file from C:\Temp in a notepad and edit the tags in the MCP platform as follows:
   ```
   <ResponderAssembly>Perl.exel</ResponderAssembly>
   <ResponderClass>Responder.pl</ResponderClass>
   ```
4. Follow steps 1 to 5 from the section Build the system, reports and Component Enabler user interfaces only to configure a deployable Element for MCP.
5. Open command line interface, change the directory to C:\Program Files\Unisys\NGEN\Bin and type the following command:
   ```
       Builder.exe    /m    Meta    /re    MetaFolder    /c    "Release|MCP    /F
       "C:\Temp\MSBuildSettings.xml"
   ```
   Where,
   - Meta is the model name
   - MetaFolder is the deployable Element name
   - Release is the configuration name
   - MCP is the platform we are building to
   - C:\Temp\MSBuildSettings.xml is the path for the MSBuildSettings.xml file

Using Visual Studio Environment

1. Create a host responder class and save the code file (for example Responder.pl) in the Bin directory of the AB Suite installation (C:\Program Files\Unisys\NGEN\Bin). For a sample Perl responder class, refer to Examples of Host Responder Class.
2. Open Registry Editor.
3. Browse to HKEY_LOCAL_MACHINE\SOFTWARE\Unisys\System Modeler\Features\Builder and add two String Value keys and enter the values as follows:
   - ResponderClass – Enter Responder.pl in the Value data field
   - ResponderAssembly – Enter perl.exe in the Value data field
4. Follow steps 1 to 6 from section Build the system, reports and Component Enabler user interfaces only to build an MCP application.

Examples of Host Responder Class

Following are two code examples for a Host Responder class in C# and Perl.

Example 1 C# code for a DBOverwrite Responder

The following code example shows how to code a Responder class as part of a C# Console Application project:

```
// Copyright ⌐ 2002 Unisys Corporation. All rights reserved. UNISYS CONFIDENTIAL

// It contains two examples to check for the host prompts to confirm overwriting
// the database (auto-response set to YES) and reorganizing the database (auto-
response set to NO).

using System;
using System.Text.RegularExpressions;

namespace SampleResponder
{
/// <summary>
/// Summary description for SampleResponderClass.
/// </summary>
public class SampleResponderClass
{
    /// <summary>
    /// Main method reads log entries from standard input.
    /// If it finds any of interest it writes a response to Standard Output.
    //<DB name> CONTROL FILE IN USE -- WHEN DB CLOSED,ENTER  OK
    //Regex DBExistsPattern = new Regex("CONTROL FILE IN USE -- WHEN DB CLOSED,ENTER
OK");

    // Other messages to process...
    // ** WHEN DATABASE CLOSED , ENTER <MIX> OK
    // POP. OPTIONS WARNING - SEE CFG REPORT MIX
    /// </summary>
    [STAThread]
    static void Main(string[] args)
    {
        // Create regular expressions for the following two host queries.
        // Also create the regular expressions for the valid reply patterns
        // for these two host queries.
        //
        // 14:57:20 *** DATABASE ALREADY EXISTS, TO OVERWRITE ENTER AX YES ELSE NO.
        // 14:57:20 Valid Reply: 5903 & one of AX {reply option},DS
        //
        // NOTE: The \\* is simply a * because the star char is a Regexp special char
        // See the MS documentation on Regex class.
        Regex DBExistsPattern = new Regex( "\\*\\*\\* DATABASE ALREADY EXISTS, TO
OVERWRITE ENTER AX YES ELSE NO");
        // Note: the brackets () below indicate groups we want to extract from the
text
        Regex ValidDBExistsReplyPattern = new Regex( "Valid Reply: ([0-9]*) & one of
AX  {reply option},DS" );
        Regex DBReorgPattern = new Regex("\\*\\* DO YOU WISH TO RESTART THIS REORG,AX
YES        OR NO \\*");
        Regex ValidDBReorgReplyPattern = new Regex( "Valid Reply: ([0-9]*) & one of
AX          {reply option},DS" );

        bool DBExistQuestionMatch = false;
        bool DBReorgQuestionMatch = false;
        // YOU CAN EDIT HERE --> Initialize a boolean variable to indicate if
        // the message list consists the host query of interest.
        // bool HostQueryQuestionMatch = false;

        String sMsg = System.Console.ReadLine();
```

```
            // For each of the log lines check against the pattern(s) of interest
            while ( sMsg != null )
            {
                // Did we match the DB Exists Pattern?
                Match m = DBExistsPattern.Match( sMsg );
                if ( m.Success )
                {
                    DBExistQuestionMatch = true;
                }
                // Did we match the Valid Reply Pattern?
                m = ValidDBExistsReplyPattern.Match( sMsg );
                if ( m.Success )
                {
                    if ( DBExistQuestionMatch )
                    {
                    // Write response to standard output
                    System.Console.WriteLine(m.Groups[1].ToString() + " AX YES");}
                }

                // Did we match the DB Reorg pattern?
                m = DBReorgPattern.Match( sMsg );
                if ( m.Success )
                {
                    DBReorgQuestionMatch = true;
                }
                // Did we match the Valid Reply Pattern?
                m = ValidDBReorgReplyPattern.Match( sMsg );
                if ( m.Success )
                {
                    if ( DBReorgQuestionMatch )
                    {
                    // Write response to standard output
                    System.Console.WriteLine( m.Groups[1].ToString() + " AX NO" );
                    }
                }
            // read the next line from Standard Input
                sMsg = System.Console.ReadLine();
            }
            // If we didn't detect a response then return nothing allow manual response
            }
        }
    }
```

Example 2 Perl Code for a Host Responder

Here is similar code written as a Perl script:

```
    # This is a perl overwrite responder for the MCP host build

    # Read the log information from the host via STDIN - the standard input
    my(@LogLines) = <STDIN>;

    my($FoundDBExists) = 0;
    my($LogLine) = "";

    # Loop through the lines and match on loglines of interest...
    foreach $LogLine ( @LogLines )
    {
        # scan for DB exists message
        if( $LogLine =~ /\*\*\* DATABASE ALREADY EXISTS, TO OVERWRITE ENTER AX YES ELSE
    NO/ )
        {
            $FoundDBExists = 1;
            next;
        }

        if( $FoundDBExists )
        {
            if( $LogLine =~ /Valid Reply: ([0-9]*) & one of AX \{reply option\},DS/)
            {
                print "$1 AX YES";  # write response to STDOUT
                exit(0); # done!
            }
        }
    }
    # No Response
    print "";
```

Deploying Applications in Windows® Runtime

At the end of a build, a deployment package (msi file) is created, and depending on what has been set in Builder's configuration properties, transferred to the runtime server, and installed. Database reorganization may also occur during deployment.

A deployment package consists of one or more files, deployed to one or more runtime servers. Multiple deployment packages can also be deployed to a single server. A deployment package can contain separate components, or can be controlled to contain the elements you want. For diagnostic information collected during the deployment, view the Deployment.log file located on the runtime server:

```
<Agile Business Suite Directory>\Data\Public\Log
```

Note: *If a folder containing subfolders is deployed, deployment packages are created and deployed for each folder. Each deployment package is deployed one at a time, but in one single operation.*

Before you Deploy

Note the following considerations before deploying:

- Complete necessary installations – The Deployment Server and all runtime infrastructure must be installed on the server to which you wish to deploy. If it is not, and an application is manually deployed, deployment may appear to succeed, however is inoperable. The Builder factory for the target platform must also be installed on the Development environment.
- Deployment Impersonation – By default, the Deployment Server running under the Application Administrative User identity impersonates the user that invokes the deployment action to access the MSI package. This requires the invoking user to be given administrative privileges in the runtime machine and the application administrative user to be given the “Trust Account for Delegation” privilege at the domain level when the development environment and runtime environment are on different machines. Refer to the *Agile Business Suite Installation and Configuration Guide* for more information about the delegation privileges.

  An alternative method of deployment is by defining the values for the RuntimeServiceImpersonationLevel registry key where impersonation level is reduced. Refer to the *Agile Business Suite Installation and Configuration Guide* for more information on various impersonation levels.

- Ensure necessary files exist (applies to redeployments only) – Before rebuilding and redeploying your application, ensure that all files created with the previous deployment exist.
- Ensure the target database and database server registrations are created successfully on the runtime server.
- Stop the System – The system must be stopped before it is deployed. If the system is running when deployment is attempted, deployment aborts. Use the Admin Tool to stop a system.
- Enable/Disable the System – To enable your application after deployment, ensure that is enabled before deployment. Likewise, if it is disabled before deployment, it is disabled after deployment.
- Check database and application user details – To ensure a successful deployment, check that database and application user details are correct.

Checking Database and Application User Details

If deployment is initiated using Builder or the Administration Tool, database and application user details entered (such as name, password, domain, etc) are checked. If they are incorrect, deployment aborts.

If deployment is performed manually, no checks are performed. If database and application user details entered are incorrect, deployment appears to succeed, however the application is inoperable.

If details entered are correct, and the database or Segment COM+ component is being deployed, the application must be manually stopped and disabled before deployment proceeds. You are unable to connect to the application until deployment has completed.

Deploying with Reduced Impersonation Level

If the RuntimeServiceImpersonationLevel registry key option is used for deploying a system, you can access the deployment package at the following two levels:

- As an administrative user not having delegation privileges
- As an application administrative user having delegation privileges

When you opt to install a package with an administrative user not having delegation privileges, the administrative user can access the MSI package without impersonating the user that invokes the deployment. If the Package Intermediate Directory is a local path in the developer environment or on the runtime server, no prior settings are required. However, access permissions should be explicitly provided to the administrative user if the Package Intermediate Directory is a shared network path. In the network path, you must provide the following privileges to an administrative user at the network location:

- Permission to access (Read/Write) the shared builder output path from the network.

- Add to the "Network Access" group of the remote computer

  Note: *You can add a user to the Network Access group from* ***Control Panel*** *>* ***System and Security*** *>* ***Administrative Tools*** *>* ***Computer Management****. In the Computer Management window, navigate to* ***Local Users and Groups*** *>* ***Groups****. Double-click* ***Network Access*** *and then click* ***Add****.*

- Have "Access this computer from the network" privilege.

  Note: *You can grant a user "Access this computer from the network" privilege from* ***Control Panel*** *>* ***System and Security*** *>* ***Administrative Tools*** *>* ***Local Security Policy****. In the Local Security Policy window, navigate to* ***Local Policies*** *>* ***User Rights Assignment****. Double-click Access this computer from the network and then click* ***Add User or Group****.*

When you opt to install a package with application administrative user having delegation privileges, the deployment process is initiated by the application administrative user that invokes the deployment process. If the Package Intermediate Directory is located on the runtime server or located on a shared network path, you must provide the following privileges to the application administrative user:

- Permission to access (Read/Write) the shared package intermediate path from the network.
- Add to the "Network Access" group of the remote computer.
- Have "Access this computer from the network" privilege.

Note: *The privileges required for an administrative user without delegation privileges are same as those that are required for an application administrative user with delegation privileges, except the following privileges that needs to be provided:*

- *Enable the computer and user accounts to be trusted for delegation*
- *Impersonate a client after authentication*

Rebuilding or Redeploying

If you wish to rebuild or redeploy in the future, do not delete files created in your deployment system directory. If files are deleted and cannot be recovered, your application must be removed using the Windows Installer CleanUp Utility.

To redeploy your application manually on Windows hosts, before redeploying perform the following:

1. Stop any builds.
2. Disable the system.
3. Use the Windows Installer Cleanup Utility.
4. Modify the NGRuntime.XML to remove the system entry.
5. Delete the COM+ component.

How To: Control Deployment Package Contents

To control deployable outputs, perform the following:

1. Create a folder. Each deployable folder creates a separate file.
2. Copy the elements that you wish to build into the folder.
3. Configure the folder using the Build Target Filter properties .
4. Right-click the folder and select **Build**, or **Folder Only**, then **Build**.

Deploying Elements Separately

Application components (core system), databases, reports, user interfaces, and external files can all be deployed separately.

Use top-level folder configuration properties to specify which elements should be grouped together in a deployment package.

If a report class, or folder is selected for building, ensure the configuration properties set for the folder matches the top-level deployable Element properties. For example, ensure the deployment paths and database host matches the correct system.

Including External Installer files

Using Configuration Properties, you can bundle and execute external files within selected deployment packages. Builder typically binds external files as nested files.

Note: *External files must be capable of running without user input, otherwise deployment may appear to hang when it is actually waiting for input for an external package.*

Reorganizing a Database

Database reorganization involves adding, modifying, or deleting application specific database elements to produce new or modified database tables required to support your applications.

If your application is transferred to a remote server by using the Default method, database reorganization automatically occurs during application installation. Otherwise, databases can be manually reorganized. Errors encountered during database reorganization are logged to the DBReorg.log file. By default, this file is located in:

```
<NGEN directory>\Data\Public\Log
```

Since an existing schema does not exist for new deployments, database reorganization is responsible for creating required database elements.

Notes:

- *Before database reorganization occurs, it is recommended that you create a full database backup.*
- *To reorganize a database, you must have sufficient user privileges on both the database reorganization module, and the targeted database.*

User-maintained Tables

A user-maintained table provides database performance features that may not be available with Developer.

It is your responsibility to ensure that user-maintained tables are consistent and reorganized with corresponding class definitions.

Restrictions

A consistency check cannot be performed when database reorganization is in progress.

Manual Reorganization

To manually reorganize a database, use the following command line argument:

```
DBReorgEXE.exe <alias> <dbName> <owner> <retainMode> <execMode> <folder>
```

Where,

- alias is the name of the Database Server Registration for the system database, as created by the database administrator using the Administration Tool (for example, default).
- dbName is the name of the database to be reorganized.
- owner is the schema owner's user name.
- retainMode is whether the existing database is to be retained for the operation (YES or NO).
- execmode is the function you want to perform on the database:
  - CHECK – performs a consistency check on the database.
  - PLAN – builds and prints the reorganization plan. The reorganization plan is a collection of actions that reorganize the database.
  - REORG – completes the reorganization of the database.
  - DROP – removes the database tables.
- UMTCHECK – runs a user maintained consistency check on the database
- folder is the directory where system files such as DatabaseSchema.xml are located.

Once a database is reorganized, one of the following values is displayed:

- 0 – reorganization has started, and has successfully completed.
- 1 – reorganization has started, and is unsuccessful.
- 2 – reorganization has failed to start.

Note: *Stop the system before attempting to manually reorganize a database.*

How To: Allow Recovery from Failed Reorganization

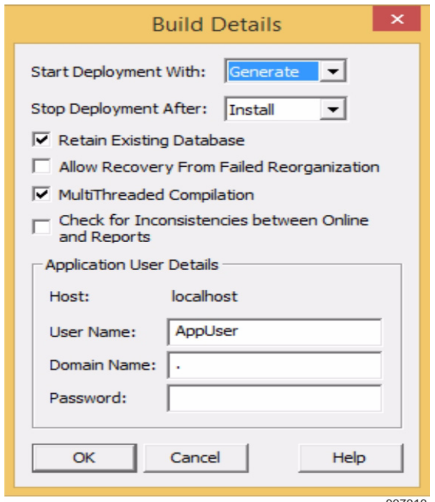
Recovery from a failed Runtime database reorganization has been implemented for Debugger and Windows® Runtime with Microsoft SQL Server. When the option is selected on the Build Dialog (Windows) or Debugger Configuration Property (Debugger), a backup of the Runtime database occurs before the reorganization occurs.

If the reorganization fails, the Runtime database is restored to the state prior to the reorganization.

This allows the error causing the reorganization failure to be corrected and the system built successfully. If the option ‘Allow Recovery From Failed Reorganization’ is checked, should the database reorganization fail, the Runtime database is restored to the state immediately before the build was done.

Otherwise, if the ‘Allow Recovery From Failed Reorganization’ is unchecked, should the reorganization fail, the Runtime database may be rendered unusable.

By default, the option is unchecked for Windows® Runtime and is set to true for Debugger.


Build Details

Start Deployment With: Generate

Stop Deployment After: Install

[✓] Retain Existing Database

[ ] Allow Recovery From Failed Reorganization

[✓] MultiThreaded Compilation

[ ] Check for Inconsistencies between Online and Reports

Application User Details

Host: localhost

User Name: AppUser

Domain Name: .

Password:

007010

Performance

The performance overhead is dictated by the Microsoft SQL Server backup procedures and the size of the database.

Note: *This feature should not be considered a backup facility for your data, as the backup files are not retained permanently after a successful reorganization. If you require your data to be regularly backed up, SQL Server's backup functionality is recommended to maintain a regularly updated backup set.*

Transferring a Deployment Package

Deployment packages are transferred to runtime servers by:

- Using the default method in Developer.
- Manually copying deployment packages from the Staging area to the runtime server.

Once your deployment package has been transferred, it can be installed .

Note: *Win Form and Component Enabler Java deployment packages cannot use the default method. Use manual transfer instead.*

Default Method

Use default deployment method by using the **Build**, or **Folder Only**, **Build option** in Developer. Once an application is built, deployment and installation automatically occurs on the runtime server (where the Stop Deployment After configuration properties includes transfer and install phases).

Manual Method of Deployment

To manually transfer a deployment package, you can either copy the deployment package from the Staging area to the runtime server, or type the command line argument. Once transferred, it can be installed .

Alternatively, there are other methods to transfer your deployment package.

Installing a Deployment Package

A deployment package in Windows is a Microsoft Installer (msi) package. It is normally found in the Package Intermediate Directory folder, specified in the Deployable Element configuration properties. While other msi packages can usually be installed by double-clicking on the file, Agile Business Suite deployment packages require additional command line arguments, which are not passed to the package if it is invoked by double-clicking on the file.

It is recommended, where a default deployment operation is not feasible, manually deploy the package and install it using MSIEXEC command line arguments as described in Using MSIEXEC command line arguments section.

Notes:

- *Each application must be installed in its own directory. Do not install multiple applications in the same directory.*
- *An application cannot be installed to several different databases on the same server. If an application is updated (that is, it is deployed to a server that hosts a previously deployed installation), and the target database of the new application is different to the database used by the installed application, the application creates a new database. The old database is not removed. Any existing data it contains must be migrated manually.*
- *Once a deployment package has been deployed, do not delete any deployment created files.*

Using MSIEXEC Command Line Arguments

The /i parameter of MSIEXEC command line is used to install the Agile Business Suite deployment package.

Syntax

```
msiexec /i  "<PackageGUID> or <complete MSI Name with its path>" <UserName>
<Password> <Domain>
```

Following table lists the various parameters of MSIEXEC command and their description:

Parameter	Description
/i	Installs the Agile Business Suite deployment package. You can either pass the msi file along with its path or the GUID of the deployment package. If the system is already installed, you are prompted to repair or remove the installed system. **Examples:** msiexec /i C:\Temp\StagingArea\Sample.msi msiexec /i {D95F933C-4F0B-11DD-867E-444553544200}
/x	Uninstalls an installed system. You can either pass the msi file along with its path or the GUID of the deployment package. If system is not already installed, the following message is displayed - “This action is only valid for the products which are currently installed”. **Examples:** msiexec /x C:\Temp\StagingArea\Sample.msi msiexec /x {D95F933C-4F0B-11DD-867E-444553544200}
/passive	It can be passed with /i or /x. The /passive parameter installs, repairs or removes the installation in passive mode. The progress bar is visible during the installation and uninstallation. **Examples:** msiexec /passive /i C:\Temp\StagingArea\Sample.msi msiexec /passive /x C:\Temp\StagingArea\Sample.msi

Parameter	Description
/q	Installs and uninstalls in quiet mode. **Examples:** msiexec /q /x C:\Temp\StagingArea\Sample.msi msiexec /q /x C:\Temp\StagingArea\Sample.msi
/j	Installs the system for all the users of the machine but the user must have logged on as administrator to install the system. **Example:** msiexec /j {C:\Temp\StagingArea\Sample.msi}
/norestart	Do not restart once the Installation/Uninstallation is over.
/a	Creates a network image of the installable msi in the path mentioned in the folder property "Package Installation Directory" **Example:** msiexec /a C:\Temp\StagingArea\Sample.msi
/forcerestart	Prompts you to restart after the Installation/Uninstallation. **Example:** Msiexec /forcerestart /i C:\Temp\StagingArea\Sample.msi
DropDB	Specifies whether or not to retain the existing database. It can be set to a value of TRUE or FALSE. By default, the value for this parameter is FALSE. TRUE indicates retain existing database and False indicates do not retain existing database. **Example:** C:\Temp\StagingArea\Sample.msi userCode=appuser domain=. password=app1user# DropDB=TRUE
EnableAfterDeploy	Specifies that the COM+ system generated by AB Suite would be enabled if it is set to true. **Example:** C:\Temp\StagingArea\Sample.msi userCode=appuser domain=. password=app1user# enableAfterDeploy=TRUE
REPORT	Specifies that the whether the report should be deployed or not. It can be set to a value of 0 or 1. 0 indicates that deploy reports is false. 1 indicates deploy reports is true. **Example:** msiexec.exe /forcerestart /i C:\Temp\StagingArea\Sample.msi userCode=appuser domain=. password=app1user# REPORT=0

Repairing and Uninstalling

While the Deployment Package is installed via Windows Installer, it cannot be normally Repaired or Removed due to dependency complexities – especially when subsequent report deployments are made. It is highly recommended that you use the Admin Tool to uninstall systems where possible. Where this is not possible, use the Microsoft Fix it from the Microsoft Installer SDK to remove the systems. Note that you need to manually delete the COM+ component as it can be left behind if locked by residual database operations (such as updating system status) during the uninstallation. Refer to Removing an Invalid System for more information.

Note: *If your application is uninstalled, it must first be stopped and disabled.*

To create specific uninstallation instructions, use Builder's configuration properties . For example, leave user-maintained tables untouched during uninstallation.

Removing an Invalid System

You can remove an invalid system from the AB Suite environment by using Microsoft Fix it.

Warning

It is not recommended to use this cleanup procedure as there is a substantial risk of environment corruption if you are not familiar with the cleanup procedure. However, if you still want to perform the cleanup procedure, ensure to back up the runtime data and any custom files before running Microsoft Fix it.

To remove an invalid system, perform the following:

1. Run Microsoft Fix it.

   Read information about running Microsoft Fix it at the following website:

   http://support2.microsoft.com/mats/program_install_and_uninstall
2. Delete the COM+ component from Component Services. To do this open Runtime Administration Tool, Console Root ,Component Service , My Computer, COM+ Application and delete the COM+ Application from the list.
3. Right-click the **AB Suite Runtime Manager** and click **Shutdown**.
4. Delete the database via SQL Server Enterprise Manager.
5. Delete SQL Login for the system via SQL Server Enterprise Manager (must delete corresponding DB first).

Administrative Installation

An administrative installation allows multiple users to install your application from a network drive.

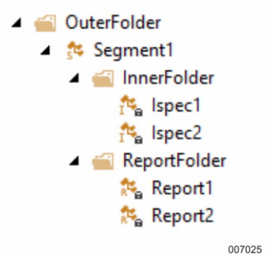
To install, users simply locate the deployment package, double-click it, then complete the installation wizard. No registry entries are made, shortcuts created, and the application cannot be launched.

Note: *You need to supply the same user name and password credentials for Administrative installations, as they do for default and manual installations.*

Programmatic Use of Deployment Server Component

In certain circumstances, direct use of the deployment server through its COM interface is preferable over the use of Developer when deploying systems and reports. The deployment server can be invoked to deploy generated application MSI files, and requires the creation of the DeploymentServer object. A DeploymentServer object is used to access properties and methods on the server. Following example explains the creation of a DeploymentServer object.

An AB Suite system contains an OuterFolder, an InnerFolder (including all the ispecs), and ReportFolder with all the reports.



If you do a generate-install, the system is deployed and two MSIs are created in the Package Intermediate Directory if specified or in the packages folder of the Builder output directory. Now, if you make any changes to the ispecs in the painter and do a generate-generate on the OuterFolder, MSIs are created in the Builder Output directory. To make these changes made to the ispecs available without affecting the reports, you can use the following VB script.

Deploying a System

To determine the PROGID of the deployment server, use the Administrative Tool and locate the Prog id for the DeploymentServer object.

```
dim SIretcode
‘Create an object of Deployment Server’
set DSobj = CreateObject(DEPLOYMENTSERVER_PROGID)
SIretcode = DSobj.StartInstall("ApplicationFolder", "<Package Intermediate
Directory>\ApplicationFolder.msi", "userCode=uuu domain=ddd password=ppppp
dropBS=TRUE", nothing)
If SIretcode = 0 then
WScript.Echo "Deployment Success"
Else
WScript.Echo "Deployment Failed with error code" & SIretcode
End If
```

where,

- ApplicationFolder – Name of the deployable folder that contains the segment.
- Package Intermediate Directory – The folder in which the generated application MSI files are located. This value must correspond to the Package Intermediate Directory configuration setting in the deployable Element.
- userCode =<uuu> –”<uuu>” is the Application User account's name that is used as the deployed application's identity.

- domain=<ddd> – "<ddd>" is the Application User account's domain that is used as the deployed application's identity.
- password=<ppp> –"<ppp>" is the Application User account's password that is used as the deployed application's identity

If changes are made to the reports (adding a label) and ReportFolder is built, an MSI for reports, ReportsFolder.MSI is created. Use the following VB script to deploy the reports MSI (without affecting the application folder).

Deploying a Report

To determine the PROGID of the deployment server, use the Administration Tool and locate the Prog id for the DeploymentServer object.

```
dim SIretcode
'Create an object of Deployment Server'
set DSobj = CreateObject(DEPLOYMENTSERVER_PROGID)
SIretcode = DSobj.StartInstall("ReportsFolder",
"<ReportsMsiPath>\ReportsFolder.msi", "userCode=uuu domain=ddd
password=ppp", nothing)
If SIretcode = 0 then
WScript.Echo "Deployment Success"
Else
WScript.Echo "Deployment Failed with error code" & SIretcode
End If
```

where,

- ReportsFolder – Name of the report folder in the Application Folder.
- ReportsMsiPath – The folder in which the generated report MSI files are located. This value must correspond to the path where the report MSI files are located.
- userCode =<uuu> –"<uuu>" is the Application User account's name that is used as the deployed application's identity.
- domain=<ddd> – "<ddd>" is the Application User account's domain that is used as the deployed application's identity.
- password=<ppp> –"<ppp>" is the Application User account's password that is used as the deployed application's identity.

Notes: *Follow the instructions to execute either of the scripts given above.*

1. *Create a text file, paste the code for deploying a system, and save the file with extension .vbs.*
2. *Change the directory paths used in the script to the directory paths and MSI names as applicable for your machines.*
3. *From the command prompt, navigate to the path where the script file is saved.*
4. *Execute the script with the following command:*
   ```
   C:\>cscript myVBScript.vbs
   ```

The deployment server script determines whether to make a partial or full installation, and accordingly execute MSIEXEC command by passing suitable parameters.

Deploying an Application in MCP Runtime

MCP Builder communicates with the Application Builder server (NGEN27/APPL_BLD_61) to deploy an application to the selected host. Refer to the *Agile Business Suite Runtime for ClearPath MCP Administration Guide* for more information on configuring and running the Application Builder server.

Before you Deploy

For an MCP configuration, ensure the following before deploying an application:

- Ensure that MCP Runtime is installed on the host.
- If Interim Corrections have been released, ensure that an Interim Correction (IC) compatible with the MCP Builder client has been installed.
- If there are multiple MCP Runtime installations, ensure the Application Builder server for the specified MCP Runtime installation is running on the host.
- For re-deployments where a reorganization of the database occurs, ensure that you are able to stop the application when requested. The exception to this is a database reorganization using a reorganization type of REORGDB with the AUTOSWAP property set to true.

Build and Deployment Configuration Properties

Following is a list of configuration properties sections that affect build and deployment processes:

- Build Target Filter
- Component Enabler User Interface
- Configure/BNA
- Connection
- DASDL
- Default DASDL
- Debugging
- Environment
- External
- General Configuration
- Installation
- National Support
- NAP Direct Interface
- OLTP
- Pack Allocation

- Persistence
- Profile DASDL
- Remote Database
- Runtime Options
- Runtime Transfer Utility
- Subsystem
- Winform User Interface

All Agile Business Suite configuration properties are listed in this documentation, but some may be associated with only certain platforms, and may only be available for the platform to which you are generating. All configuration properties can be reset to default or initial values. You can reset the value of a simple entry field, predefined entry field, and an entry field with a user interface. A predefined entry field includes all the configuration properties where you can select the values from the available list.

Note: *All property values appear in bold in the Property Pages window, if they are different from the defined default values.*

Configuration properties are directly associated with elements in the Class View window in Visual Studio.

To edit an element's configuration properties, perform the following:

1. Select an element from the Class View window.
2. Right-click the element, then select **Properties**.

To reset a simple and predefined configuration property field value to a default or initial value, perform the following:

1. Select the property in the right pane of the Property Pages window that needs to be reset. For example, **Data Set Buffers**, **Runtime Behavior of Unavailable Commands**, property of a Segment. Refer to Default DASDL, Runtime Options for more information on Segment properties.

   The property field is highlighted and a list appears on the value field. The property value appears in bold.
2. Select list option; **inherit from parent or default** to reset the property to a default value.
3. Click **OK** or **Apply**.

The property field value is reset to the default value.

To reset a configuration property that includes a user interface, perform the following:

1. Select the property field in the right pane of the Property Pages window that includes a user interface. For example, **Internal DASDL Options** property of a Segment. Refer to DASDL for more information on DADSL Segment properties.

The property field is highlighted and a list appears on the value field with two options, **inherit from parent or default** and **Edit...**. You can either:

a. Select list option; **inherit from parent or default** to reset the property to a default value.

b. Select list options, **Edit...** to open the relevant user interface and edit the value of this property.

2. Click **OK** or **Apply**.

Note: *The drop-down list gets enabled only for the properties which has a User Interface defined.*

Build Target Filter

Property	Function
Deploy Application Components	Specifies whether Application (core system) Component(s) are deployed. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment.
Deploy Component Enabler User Interface	Specifies whether Component Enabler user interface(s) are deployed. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders contained within a segment.
Deploy Database	Specifies whether the database is deployed. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment.
Deploy Legacy Text Interface	Specifies whether character screens or Legacy Text interface(s) are deployed. By default, this property is set to True. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Deploy Reports	Specifies whether Report(s) are deployed. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Deploy SQL views	Generates SQL views for all the persistent components (for example, Events, Profiles, etc). When set to True, it sets the view for all the persistent components.

Property	Function
Deploy Winform User Interface	Specifies whether Winform user interface(s) are deployed. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders only.
External Installer Files	Specifies the location of external installer files to be included in the deployment. Type the full path names for any files to be included in the folder package. Use semi-colons to separate multiple path names. For example, \\network\my folder\installer1.msi; C:\mylocalfolder\installer2.msi By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is not set to Generate. This property is specific to Folders only.
Generate Runtime Transfer Utility File	Specifies whether to build the MCP Runtime Transfer Utility (RTU) file. By default, this property is set to False. This property is enabled if Deployable is set to True. This property is disabled while any of the other deploy properties are enabled. This property is specific to Folders only.
Translations	Specifies a list of languages that can be used by the system. The MCP platform usesonly the first 15 of these. There is a wrapper for this so that users can edit the languages. Semi-colons are used to separate translations. By default, this property is set to the primary language. This property is enabled if Deployable is set to True. This property is specific to Folders only.

Component Enabler User Interface

Property	Function
Application Name	Specifies the name of the Component Enabler User Interface application. The following naming conventions apply: • Component Enabler reserved words cannot be used. • The first character must be an alphabet or an underscore. Subsequent characters can be alphanumeric or underscore. • Maximum length is 30 characters. By default, this property is empty. This property is specific to Segments only.

Property	Function
CE Output Directory	Specifies the location of the output directory to which the generated files are written. The default value, if no directory is specified, is the Temp directory in the TMP Environmental Variable. This property is specific to Segments only.
CE Post Build Script	Specifies the location of an optional script or batch file to run on completion of the build. This property is specific to Folders contained within a segment.
Generate Getters And Setters	Specifies whether Getters and Setters are used. Getters and Setters are extra methods on the field objects and take the form get<fieldname> and set<fieldname>. By default, this property is set to False. This property is enabled when: • Deployable is set to True. • Deploy Component Enabler User Interface is set to True. This property is specific to Folders contained within a segment.
Generate Views	Specifies whether presentation files (For example, Java, Visual Basic etc) are generated. By default, this property is set to True. If set to False, only IspecModel files are created and compiled. This property is enabled when: • Deployable is set to True. • Deploy Component Enabler User Interface is set to True. This property is specific to Folders contained within a segment.
Ispec Models Source Language	Specifies the source language in which to generate the Component Enabler IspecModels, ComponentList and PublicMethodList. The options are Java, C#, or both. By default, this property is set to C# and Java. This property is specific to Folders contained within a segment.
Package Prefix	Specifies the name and type of company in World Wide Web format. For example, com.unisys. The following naming conventions apply: • Component Enabler reserved words cannot be used. • The first character must be alpha, or an underscore. Subsequent characters can be alphanumeric, period, or an underscore. • Maximum length is 60 characters. By default, this property is empty. This property is specific to Segments only.

Property	Function
User Defined Public Segment Method Generator Java Class	Specifies the Java Class name of the generator that is used to generate interface classes for a segment's public methods. If a name is not specified, the default generator is used to generate interface classes. The following naming conventions apply: • Component Enabler reserved words cannot be used. • The first character must be alpha, or an underscore. Subsequent characters can be alphanumeric, period, or an underscore. • Maximum length is 120 characters. By default, this property is empty. This property is enabled when: • Deployable is set to True. • Deploy Component Enabler User Interface is set to True. This property is specific to Folders contained within a segment.
User Defined View Generator	Specifies the .NET based view generator dynamic linked library that is used to generate "other" presentation files. If a name is not specified, a default Presentation Client generator is used, and IspecView.Java files are created and compiled. If a name is specified, "other" presentation files are created and IspecView files and classes are not created and compiled. The following naming conventions apply: • Component Enabler reserved words cannot be used. • The first character must be alpha, or an underscore. Subsequent characters can be alphanumeric, period, or an underscore. • Maximum length is 120 characters. By default, this property is empty. This property is enabled when: • Deployable is set to True • Deploy Component Enabler User Interface is set to True. • Generate Views is set to True. This property is specific to Folders contained within a segment.

Configure/BNA

Property	Function
Usercode	Specifies the usercode for the BNA Generate job. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.

Property	Function
Access Code	Specifies the access code for the BNA Generate job. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.
Charge Code	Specifies the charge code to be used for application accounting purposes. All resource usage by the BNA Generate is debited to this charge code. This property is limited to a maximum length of 45 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.
Configure Set Type	Provides the user with a single option to specify what the configuration is used for. The following options are available for this property: • BUILD – Indicates that the configure set is used for a normal build • CONFIGURE – Indicates that the configure set is used for configuring a target system with Runtime Transfer. • RDB – Indicates that the configure set is used to specify a Remote DB secondary database and system. • BNA Generate – Indicates that the configure set is used to create a BNA Generate system. (Not implemented in this release). This property is specific to Segments only.
Configure Work Pack	Specifies the name of the pack on which the Configuration work files are located on the target host. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is disabled when Configure Set Type is set to BUILD. This property is specific to Segments only.
DMSII Pack	Specifies the name of the pack where an alternative version of DMS II software is located. This location and the value entered in the DMSII Usercode property identify the alternative version of DMS II to be used by the BNA Generate. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.

Property	Function
DMSII Usercode	Specifies the usercode for use with an alternative version of DMS II software. This usercode and the value entered in the DMS II Pack field identify the alternative version of DMS II to be used by the BNA Generate. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.
Generate Family	Specifies the desired family specification to ensure that the required DMSII files and compilers are visible to the generate process. You must make an entry in this field if: • The system software required during the generation is not resident on either the primary or alternate pack family of the WFL usercode family. • Some or all of your installed host software is installed on a pack family called DISK This property is limited to a maximum length of 44 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to BNA Generate. This property is specific to Segments only.
Generate Work Pack	Specifies the location for use with BNA generate. All files produced by the BNA generate reside on this location. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled if Configure Set Type is set to BNA Generate/Configure. This property is specific to Folders only.
MODEL Base Configuration	Specifies the configuration name of the Base System, when using the DMS Model facility. The Base System configuration must be a valid configuration. The Base System is the system on which the modeled database is modeled or based. This property is limited to a maximum length of 10 alphanumeric characters. By default, this property is empty. This property is enabled when Configure Set Type is set to CONFIGURE. This property is specific to Segments only.

Property	Function
Retain Existing Database	Specifies whether to retain an existing database, or initialize a new database. The following options are available for this property: • Retain Existing – Retains the existing database already on the target host • No Retain Existing – Initializes a new database • Configure New Clone – Initializes a new cloned secondary database, if you are configuring a standby System to support a secondary database. • Retain Existing Cloned – Retains existing cloned secondary database, if you are configuring a standby System to support a secondary database. By default, this property is set to Retain Existing. This property is disabled when Configure Set Type is set to BUILD. This property is specific to Segments only.
Target Host Name	Specifies the name of the target host to which the application is transferred and configured. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is disabled when Configure Set Type is set to BUILD. This property is specific to Segments only.
Host Builder Usercode	Specifies the usercode of the Runtime software on the target host. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is disabled when Configure Set Type is set to BUILD. This property is specific to Segments only.
Host Builder Pack	Specifies the name of the pack on which the Runtime software is located on the target host. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is disabled when Configure Set Type is set to BUILD. This property is specific to Segments only.

Connection

Property	Function
Accesscode	Specifies a valid ClearPath MCP accesscode set on the target host. ***Note:*** *You must enter the password associated with the specified accesscode when you request a generate. This password for the accesscode is stored in a tmp file for the duration of your NEGN session.* This property is limited to a maximum length of 17 alphanumeric characters. The first character must be an alpha or a numeric character. By default, this property is empty. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Chargecode	Specifies a valid ClearPath MCP charge code set on the target host. This property is limited to a maximum length of 60 alphanumeric characters. This property can contain the slash character. By default, this property is empty. This property is enabled if Deployable is set to True. This property is specific to Folders only.
FTP Chargecode	Specifies the chargecode for the FTP Server software on the specified target host. This property is limited to a maximum length of 60 alphanumeric characters. This property can contain the slash character. By default, this property is empty. This property is enabled if Deployable is set to True. This property is specific to Folders only.
FTP Usercode	Specifies a valid Usercode for FTP connection to the host. This Usercode must be defined as a valid user in the FTP Server software on the specified target host. Before you generate a System or Report, you must first log into the Builder Server using MCP login details. The information required to log in is determined by the target host. In addition, an FTP session must be established for all hosts and the FTP login details must be specified. ***Note:*** *You should enter the password associated with the specified FTP usercode when you request a generate. This password for the FTP usercode is stored in a tmp file for the duration of your NEGN session.* This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled if Deployable is set to True. This property is specific to Folders only.

Property	Function
MCP FTP Site Command	It is used to customize how the generated files are FTP'd to the host. The user can enter either of the two commands for text data: QUOTE SITE MAPIN TEXT <text options> TEXT <text options> Users can use all options except FILekind and RECordlength, which are controlled by the generate process. Users may use the MCP site MAPIN command to specify the character set translation used in the file transfer. Refer to the *TCP/IP Distributed Systems Services Operations Guide* in the Support website for more information on the MAPIN command.
MCP Kana Mode	Specifies the Kana mode for MCP Japanese hosts only. This property is set to one of the following values according to the character set used on the target ClearPath MCP host. • V24PlusK • V24MinusK By default, this property is set to V24PlusK. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Port Number	Specifies the port number used for logging on to Builder Server. This number should match the port configured for the Builder Server. Before you generate a System or Report, you must first log into the Builder Server. The information required to log in is determined by the target host. By default, this property is set to 8600. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Usercode	Specifies a valid ClearPath MCP user code set on the target host. ***Note:*** *You should enter the password associated with the specified Usercode when you request a generate. This password for the Usercode is stored in a tmp file for the duration of your NEGN session.* This property is limited to a maximum length of 17 alphanumeric characters. The first character must be an alpha or a numeric character. By default, this property is empty. This property is enabled if Deployable is set to True. This property is specific to Folders only.

DASDL

Property	Function
$-Options (DASDL Compiler)	Specifies the DASDL compiler options. The value is a free format string field that allows the user to specify compiler options that is included in the DASDL source. Text entered in this field is inserted in the DASDL source file, after the default $-. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for a list of DASDL compiler options. This property is limited to a maximum length of 63 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.
Addresscheck Required	Specifies whether a DMSII addresscheck statement is generated. The following options are available: • DMSII Default (default setting): 'ADDRESSCHECK' not generated. • Yes: 'ADDRESSCHECK SET' is generated. • No: 'ADDRESSCHECK RESET' is generated. If you attempt to reset this property for an application that has previously been set and generated, a warning message is displayed. Changing the property value results in a DASDL compile error. By default, this property is set to DMSII Default. This property is specific to Segments only.
Allowed Core	Specifies the amount of memory (in number of words) needed for database buffers. By default, this property is set to 80000. The numbers range from 2000 to 268435455. This property is specific to Segments only.
Areasize in Records	Specifies the areasize in records. By default, this property is set to 0. The numbers range from: • 0 to 1048575 for Classes. • 1 to 9999 for Profiles. This property is specific to <<Ispec>> Classes, Vanilla Classes, EventSets and Profiles only.

Property	Function
Audit Sections	Specifies the number of audit trail sections for a segment and generates a statement for the defined sections. This property is enabled when the Extended Edition property of a segment is set to True. The following rules apply to the value of this property: • If the **Audit Sections** property is greater than one and the **Extended Edition** property of a segment is set to True, a statement for the section is generated, such as SECTIONS = nn. • If the **Audit Sections** property is equal to zero or one or the **Extended Edition** property of a segment is set to False, a statement for the section is not generated. By default, this property is set to 0. The numbers range from 0 to 63. This property is specific to Segments only.
Blocksize in Records	Specifies a value for the Blocksize to override the default defined by the host. By default, this property is set to the calculated default value. The numbers range up to 4095. This property is specific to <<Ispec>> Classes, Vanilla Classes, and EventSets only.
Control Point	Specifies the DMSII Control Point. By default, this property is set to 2. The numbers range from 1 to 4095. This property is specific to Segments only.
DumpEncrypt	Specifies whether it enables automatic tape encryption during a DMUTILITY DUMP operation. If DumpEncrypt is set to True and EncryptType property is not specified, the default EncryptType, TDES, is used. By default, this property is set to False. This property is specific to Segments only. ***Note:*** *The **Obfuscate Level** configuration property must be set to 0 before DumpEncrypt can be used.*
EncryptType	Specifies the encryption algorithm that the DMUTILITY and COPYAUDIT software use when copying files to tape. The following options are available for this property: • TDES • AES256 • Blank By default, this property is set to Blank. This property is specific to Segments only.

Property	Function
Filler Size	Specifies the size of the filler area to be allocated at the end of the database structure. This size reduces as attributes are allocated which uses the filler space. You may optionally specify a filler area to be allocated at the end of each database record on a structure relating to the Event Set or an <<ispec>> class or a vanilla class. This filler area means that new Attributes can be added to an <<ispec>> class without causing database reorganization. Each new Attribute is allocated space in the filler area and the total size of the record does not change. This property is limited to a maximum length of four digits in the range of 0 to 2047. By default, this property is set to 0. This property is specific to <<Ispec>> and Vanilla Classes, and EventSets only.
Filler Size at Last Generate	Specifies the amount of space in the filler area remaining that could be allocated for new attributes as at the last generate. This is purely a calculated field (non enterable). When generate is complete, the Filler size at last generate should match the Filler space remaining. This property is Read-Only. This property allows a maximum length of four digits in the range of 0 to 2047. By default, this property is empty. This property is specific to <<Ispec>> and Vanilla Classes, and EventSets only.
Filler Space Remaining	Displays the amount of space remaining in the allocated filler area. If the space in the filler area is exceeded the name of this field changes to Filler Space Exceeded By. By default, this property is empty. This property is Read-Only. This property is specific to <<Ispec>> and Vanilla Classes, and EventSets only.
Internal DASDL Options	Specifies the definition for the population and packs for structures in the following categories: • Internal • ROC • HDBA You can use the Internal DASDL dialog box to add or modify structures definitions. Click the browse button to access the dialog box. This property is specific to Segments only.

Property	Function
Keycompare Required	Specifies whether key rechecks are required for added protection against writing corrupt information. The key check is done by the DMSII software to validate whether the key value entered to retrieve a record matches the value in the record. By default, this property is set to False. This property is specific to Segments only.
Locked File	Specifies whether the LOCKEDFILE attribute is set on audit files, dump tapes, and database files (when initialized). The following rules apply to the value of this property: • If the **Locked File** property of a segment is set to True a statement for the section is generated, such as LOCKEDFILE. • If the **Locked File** property of a segment is set to False no statement is generated. By default, this property is set to False. This property is specific to Segments only. ***Caution:*** *Locked File causes the LOCKEDFILE attribute on audit files, dump tapes/files, and database files (when initialized) to be set to True. The LOCKEDFILE attribute on these files must be set to False before they can be removed or overwritten.*
Lock Program	Specifies whether the database stack and RDBSUPPORT are initiated as locked processes through the use of the LP (Lock Program) MCP system command. If the database stack and RDBSUPPORT are initiated as locked processes, they are prevented from being discontinued. The following rules apply to the value of this property: • If the **Lock Program** property of a segment is set to True, a statement for the section is generated, such as LOCKPROGRAM. • If the **Lock Program** property of a segment is set to False, no statement is generated. By default, this property is set to False. This property is specific to Segments only.
Log Access	Specifies the database access logging setting for the class and generates a statement for the defined sections. The following rules apply to the value of this property: • If the **Log Access** property of the class is set to True, a statement for the section is generated, such as LOGACCESS = TRUE. • If the **Log Access** property of the class is set to False, a statement for the section is not generated. By default, this property is set to False. This property is specific to <<Ispec>> classes, EventSets, and Vanilla classes only. ***Note:*** *To enable this property, the* ***System Log*** *option for Major Type must be set to 1 and Minor Type must be set to 35.*

Property	Function
Memory Resident	Specifies whether the record you access remains in memory. By default, this property is set to False. This property is specific to <<Ispec>> classes, EventSets and Vanilla classes only.
Number of Sections	Specifies the number of sections for an Event set or Component set. If you enter a value of 0 or 1 the Events or Components are not sectioned. This property can be set if the Extended Edition property is enabled. This property is limited to a maximum length of three digits in the range of 2 to 255. By default, this property is set to 0. This property is specific to Classes only.
Obfuscate Level	Specifies the value for masking the data content. The following options are available for this property: • 0 – Indicates that AB Suite does not apply data masking to attributes in the database. • 1 – Indicates that the entire database uses the same methodology of masking data. • 2 – Indicates that each structure in the database uses a different method to scramble the data content. • 3 – Indicates that each record of the structure uses its own methodology to scramble data. This type of obfuscation is only allowed when all datasets/tables in the database have the Extended Edition property set to True. By default, this property is set to 0. This property is specific to Segments only. Refer to the *Agile Business Suite Runtime for ClearPath MCP Administration Guide* for more information and restrictions on the use of this property.
Overlay Goal	Specifies the DMSII Overlay Goal. This parameter controls the rate at which buffers are overlaid to the disk. The value for Overlay Goal must be an integer or decimal number in the range 0 through 100. For example, 50.00000000, 9.00001000. By default, this value is set to 1.00000000. This property is specific to Segments only. ***Note:*** *This value must have at least one digit before the decimal point and exactly eights digits after the decimal point.*

Property	Function
Reapply Completed/ Independent Trans	Specifies whether to provide generated application with recovering capability. If you attempt to reset this property after it is once set and generated, a warning message is displayed and the change is disallowed. The warning message also states that the Two Phase Commit, Extended Report Recovery or Enable OLTP options are set and are not compatible with this property being reset. By default, this property is set to True. This property is specific to Segments only.
Resident Limit	Specifies the DMSII Resident limit. Resident Limit is a DASDL PARAMETERS option. The Resident Limit option limits the amount of memory used for MEMORY RESIDENT buffers. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the RESIDENT LIMIT option. By default, this property is set to 250000. The numbers range from 0 to 268435455. This property is specific to Segments only.
Resident Limit - Use DMSII Default	Specifies whether the DMSII default value is used for the DMSII Resident limit, or the value defined in the Resident Limit property. Resident Limit - Use DMSII Default is a Boolean property that indicates whether to use the DMSII default or not. By default, this property is set to True. Valid values are True and False. This property is specific to Segments only.
Revert to standard layout at next generate	Specifies whether to revert the field order layout for the <<ispec>> class in the DASDL file to the standard alphabetical sequence. This means the Attributes in the filler area become part of the complete sequence. If this property is set to False, the Attributes in the filler area remain in the order they are entered and the position of the existing Attributes does not change. By default, this property is empty. This property is specific to <<Ispec>> classes, EventSets and Vanilla classes only.
Statistics	Specifies whether to enable the DMSII software to accumulate basic statistics on physical and logical access to all the database structures. By default, this property is set to True. This property is specific to Segments only.

Property	Function
Statistics Location	Specify the location of the statistics report. The following options are available for this property: • Blank • Location By default, this property is set to Blank. This property is specific to Segments only. Refer to the *Enterprise Database Server for ClearPath MCP Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the STATISTICSLOC option.
Memo Data – Pack	Specifies the pack on which the data set for Memo Data is to be stored. When the StoreIfPresent property is set to True for one or more persistent attributes of an <<ispec>> class, vanilla class or <<Event>> class, a separate data set (table) is created for the data of these attributes. This data set is called Memo Data. The Memo Data - Pack property specifies the pack on which this data set is located. By default, this property is set to the Default Pack. This property is specific to <<Ispec>> and Vanilla Classes, and EventSets only.
Memo Data – Expected Number	Specifies the maximum number of entries you expect to be created for Memo Data, which exists if the StoreIfPresent property is set to True for one or more persistent attributes of the <<Ispec>> class. This property is enabled only if Memo Data – Population Estimation Method is set to Use Expected Number Method. This property’s value ranges from 1 to 268,435,455. By default: • This property is set to 100 for <<Ispec>> and Vanilla classes. • This property is set to 1000 for <<Event>> classes. This property is not applicable to <<Ispec>>, <<Event>> and Vanilla classes: • If the <<Ispec>> or vanilla class inherits from another class. • If the <<Event>> class’s EventSet property is not spaces (that is, an AutoPersist dependency with another <<Event>> class has been defined). This property is specific to <<Ispec>> classes, EventSets and Vanilla classes.

Property	Function
Memo Data – Percentage	Specifies the expected number of entries for Memo Data as a percentage of the Default Calculation. This property is enabled only if Memo Data– Population Estimation Method is set to Use Percentage Method. This property allows a percentage in the range of 0 to 100. By default, this property is set to 100. This property is not applicable to <<Ispec>>, <<Event>> and Vanilla classes: • If the <<Ispec>> or vanilla class inherits from another class. • If the <<Ispec>> or vanilla class inherits from another class (that is, an AutoPersist dependency with another event class has been defined). This property is specific to <<Ispec>> classes, EventSets and Vanilla classes.
Memo Data – Population Estimation Method	Specifies the population estimation method for Memo Data. When the StoreIfPresent property is set to True for one or more persistent attributes of an <<ispec>> class, vanilla class or <<Event>> class, a separate data set (table) is created for the data of these attributes. This data set is called Memo Data. The Memo Data - Population Estimation Method property specifies the method by which the data set population is calculated. The following options are available for this property: • Use Default Calculation – Select this option for the target builder to calculate the population for the Memo Data Table using default calculations. • Use Expected Number – Select this option for the target builder to use the value in the Expected Number property of the <<Ispec>> or vanilla class. • Use Percentage Method – Select this option for the target builder to calculate the population for the Memo Data Table using the percentage value in the Memo Data– Percentage property of the <<Ispec>> or vanilla class. By default, this property is set to Use Default Calculation. This property is specific to <<Ispec>> classes, EventSets and vanilla classes. ***Caution:*** *If the value specified for the expected number for Memo Data results in a population that exceeds 268,435,455, you must ensure that Sections have been defined for the parent ispec/ class and associated profiles. Refer to the* Agile Business Suite Runtime for ClearPath MCP Adminstration Guide *for more information on how to prepare related classes and profiles. If you do not prepare the ispec and associated profiles before using a large population, syntax errors may occur during compilation of the DASDL.*

Property	Function
Sync Point	Specifies the DMSII Sync Point. A sync point is a point in time at which all users of the database are out of transaction state. By default, this property is set to 500. The numbers range from 0 to 99999. This property is enabled if Reapply Completed/ Independent Trans is set to True. This property is specific to Segments only.
Sync Wait	Specifies a value that determines the period of time that DMSII allows a program to be held waiting for a sync point. By default, this property is set to 4. The numbers range from 0 to 99999. This property is enabled if Reapply Completed/ Independent Trans is set to True. This property is specific to Segments only.

Internal DASDL Dialog Box

Use the Internal DASDL dialog box to define or modify the definition of population and packs for structures in the following categories:

- Internal
- ROC
- HDBA

To display this dialog box, click **Browse** in the **Internal DASDL Options** property under the **DASDL** category for Segments.

Set the definitions for the structures in each category using the Internal tab, the ROC tab and the HDBA tab. These tabbed pages contain the following fields:

Property	Function
Display Population Increase	Specifies whether to display a message when an automatic population has occurred. By default, this property is set to True.
Display Population Warning	Specifies whether to display a warning when the population reaches the percentage of the set limit. By default, this property is set to True.
Extended	Specifies whether to enable the use of the DMSII EXTENDED option. By default, this property inherits its value from the parent property on the segment.

Property	Function
No Fine Table Lock	Specifies the usage of fine table locking for the set or subset associated with the internal data set when the entries are being deleted. By default, this property is set to False. The setting of the No Fine Table Lock property is ignored for the sets of GLB-DIALOGINFO and GLB-GLIOFFINFO, and for ROC-O-SSET and ROC-O-HSET.
Pack	Specifies the pack for the selected structure. By default, this property is set to different values for different structure. Refer to the following table for more information.
Population	Specifies the number of entries for the selected database structure. By default, this property is set to different values for different structure. The numbers range from 0 to 268435455. Refer to the following table for information.
Population Increase	Specifies the percentage by which the population increases if it exceeds its limit for the selected database structure. By default, this property is set to 0. The percentage ranges from 0 to 100.
Population Warning	Specifies the percentage of the population at which a warning message is issued for the selected database structure. By default, this property is set to 0. The percentage ranges from 0 to 99.
Sections	Specifies the number of sections. The Extended property must be set to True. By default, this property is set to 0. The numbers range from 0 to 255. The sections are ignored for GLB-DIALOGINFO and GLB-GLIOFFINFO.
Structure	Lists the database structures contained in the selected category (Internal and ROC). Refer to the following table for more information. For each selected structure, the corresponding Population, Population Increase, Population Warning, Pack, Extended, Sections and No Fine Table Lock attributes are displayed and allowed to be changed. By default, this property is set to GLB-HUB.

The following table lists the structures in each category, their corresponding default Population, and default Pack attributes values:

Category	Structure	Default Population	Default Pack
Internal	GLB-HUB	100000	SYSTEM
	GLB-CRITIC	2048	
	GLB-SECURITY	20000	
	GLB-DIALOGINFO	65536	
	GLB-GLIOFFINFO	4000	
	AUDITAREA	1024	

Category	Structure	Default Population	Default Pack
ROC	ROC-H	800	SYSTEM
	ROC-O	5600	
	ROC-S	8000	
	ROC-T	99000	
HDBA	GLB_DBSCHEMA	20000	SYSTEM

Setting Internal Structure Property Values

To modify the value of any property for an existing structure, perform the following:

1. Click the desired Category tab. The structures for that category are displayed in a list.
2. Click the property you wish to change for a structure.
3. Specify the value for the property.

To reset the value for a particular structure or set of structures to its default, select any structure row and right-click in the column, then select Reset or Reset All from the context menu.

Profile DASDL

Property	Function
Duplicates order	Specifies the order in which records with duplicate key values are retrieved. This property is enabled only if the Duplicates Allowed property of the Profile is set to True. By default, this property is set to Last. The range of values includes: First, Last, or No specific order. This property is specific to Profiles and <<ispec>>, <<Event>>and vanilla class automaint profiles only.
Load Factor	Specifies a percentage value to define how full an index table can be. By default, this property is set to 67 percent. The numbers range from 1 to 99. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only.

Property	Function
Memory Resident	Specifies the index tables that remains in memory once they are accessed. These tables are not removed from memory until either the database is closed or the option is dynamically reset. The following options are available for this property: • Coarse – Indicates that the coarse index tables remain in memory. • All – Indicates that both coarse and fine index tables remain in memory. • None – Indicates that normal buffer allocation/deallocation algorithms apply. By default, this property is set to None. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only.
No Fine Table Lock	Specifies the usage of fine table locking for a profile when the entries are being deleted. Set the No Fine Table Lock property to True to prevent fine table locking when deleting the entries. This property is enabled when the • Extended Edition property of the class is set to True. • DuplicatesAllowed property of the profile is set to False. The following rules apply to the value of this property: • If the **No Fine Table Lock** property of a profile is set to True, the **Extended Edition** property for the class is set to True, and the **DuplicatesAllowed** property for the profile is set to False, the DMSII NOFTLOCK option is generated for the profile. • If the **No Fine Table Lock** property of a profile is set to False, the **Extended Edition** property for the class is set to False, or the **DuplicatesAllowed** property for the profile is set to True, the DMSII NOFTLOCK option is not generated for the profile. By default, this property is set to False. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only.
Profile Sections	Specifies the definition of the key ranges to section the Profile. This property allows a maximum length of 825 characters. The following rules apply to the value of this property: • Users can enter data like K1 = 10, K2 = 100, K3 = 2000, K4 = "ABCD"; up to a maximum of 825 characters. • Names of Attributes must be in uppercase and alpha literals must be in quotes. You can do either of the following: • Select the **<inherit from parent or default>** option from the list to reset the property to default key. • Select the **<Edit...>** option from the list to edit the value of the property in the Profile Sections – Configurations dialog box. Refer to Profile Sections – Configuration Dialog Box for more information. By default, this property is empty. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only.

Property	**Function**
Physical Profile Sections	Specifies the Physical or Logical type of section for a profile and generates a statement for the defined profile sections. This property is enabled when the • Profile Sections property of a profile is not blank. • Extended Edition property of the class, which the profile spans, is set to True. The following rules apply to the value of this property: • If the **Profile Sections** property is not blank and the **Physical Profile Sections** property is set to False, a statement for logical profile sections is generated, such as SECTIONS(… . • If the **Profile Sections** property is not blank and the **Physical Profile Sections** property is set to True, a statement for physical profile sections is generated, such as PHYSICAL SECTIONS(… . By default, this property is set to False. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only. ***Note:*** *The **UsePhysicalSectionsOnly** registry setting overrides this property. For example, if the **Profile Sections** property is not blank, the Extended Edition property of the class, which the profile spans, is set to True, and the **UsePhysicalSectionsOnly** registry setting is present, PHYSICAL SECTION(is generated for the profile, regardless of the **Physical Profile Sections** property setting.*
Set Buffers	Specifies the number of set buffers. By default, this property is set to 5. The numbers range from 0 to 1048575. This property is specific to Profiles only. ***Note:*** *In the Buffer group properties, you cannot set values for all the properties as not all combinations of properties are valid. Only the following combinations of nonzero values are valid:* • *Buffers* • *Buffers and Buffers per User* • *Buffers and Buffers per random User* • *Buffers, Buffers per Random User, and Buffers per Serial User* • *Buffers and Buffers per Serial User* Although a value of 0 is valid for the Set Buffers property, AB Suite generates a value of 5 in the set physical options in the DMS file for it. Refer to the definitions of Buffers per User, Buffers per Random User, and Buffers per Serial properties for more information. The property on the segment specifies the default number of system buffers. The property on Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles inherits the value of the property on the segment by default.

Property	Function
Set Buffers per Random User	Specifies the number of buffers per random user. By default, this property is set to 1. The numbers range from 0 to 254. This property is enabled if the Set Buffers per User property is set to 0. This property is disabled if the Set Buffers per User property is not set to 0. When it is disabled, the Set Buffers per Random User property is reset to 0. Refer to **Note** in the definition of Buffers property. The property on the segment specifies the default number of buffers per random user. The property on Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles inherits the value of the property on the segment by default.
Set Buffers per Serial User	Specifies the number of buffers per serial user. By default, this property is set to 2. The numbers range from 0 to 254. This property is enabled if the Set Buffers per User property is set to 0. This property is disabled if the Set Buffers per User property is not set to 0. When it is disabled, the Set Buffers per Serial User property is reset to 0. Refer to **Note** in the definition of Buffers property. The property on the segment specifies the default number of buffers per serial user. The property on Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles inherits the value of the property on the segment by default.
Set Buffers per User	Specifies the number of buffers per user. By default, this property is set to 0. The numbers range from 0 to 254. This property is enabled if the Set Buffers per Random User property and the Set Buffers per Serial User property is set to 0. This property is disabled if the Set Buffers per Random User property is not set to 0 or the Set Buffers per Serial User property is not set to 0. When it is disabled, the Set Buffers per User property is reset to 0. Refer to **Note** in the definition of Buffers property. The property on the segment specifies the default number of number of buffers per user. The property on Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles inherits the value of the property on the segment by default.
Tablesize	Specifies the number of entries that can be made to an index table. By default, this property is set to 0. The numbers range from 0 to 4095. This property is specific to Profiles and <<ispec>>, <<Event>> and vanilla class automaint profiles only.

Profile Sections – Configuration Dialog Box

The Profile Sections – Configuration dialog box allows you to specify sections for a profile (set or subset). For example

K1 = 10, K2 = 3000,

K1 = 30, K2 = 5000,

K1 = 60, K2 = 8000,

K1 = 100, K2 = 9000,

For logical sectioning, set the **Physical Profile Sections** property to **False**; for physical sectioning, set the **Physical Profile Sections** property to **True**.

Refer to Sets, Subsets, and Accesses in the *Enterprise Database Server for ClearPath MCP Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information about Sections and on correctly specifying ranges.

Default DASDL

Property	Function
Check Sum	Specifies whether to enable the DMS II Checksum feature for added protection against writing corrupt information. Using the Checksum feature, the DMS II software performs a hash total of the words written in each physical IO. The hash total is written as an extra word at the end of the IO buffer. By default, this property is set to True. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.

Property	Function
Data Set Buffers	Specifies the number of data set buffers. A Dataset is a table in the database. The Dataset Buffers options are used to generate the BUFFER option for Datasets in the DASDL. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the BUFFER option. This property's value ranges from 0 to 1048575. By default: • This property is set to 4 for Segments • This property assumes the value of owner for <<Ispec>> classes, EventSets and Vanilla classes. This property is not applicable for <<Ispec>>, <<Event>> and Vanilla classes: • If the class inherits from another class • If the class is an <<event>> class whose EventSet property is not spaces (that is, an AutoPersist dependency with another event class has been defined). ***Note:*** *In the Data Set Buffer group properties, you cannot set values for all the properties as not all combinations of properties are valid. Only the following combinations of non-zero values are valid:* • *Data Set Buffers* • *Data Set Buffers and Data Set Buffers per User* • *Data Set Buffers and Data Set Buffers per random User* • *Data Set Buffers, Buffers per Random User, and Data Set Buffers per Serial User* • *Data Set Buffers and Data Set Buffers per Serial User* Although a value of 0 is valid for the Data Set Buffers property, AB Suite generates a value of 5 in the data set physical options in the DMS file for it. Refer to the definitions of Data Set per User, Data Set Buffers per Random User, and Data Set per Serial properties for more information. This property is specific to <<ispec>> and vanilla classes, EventSets and Segments only.

Property	Function
Data Set Buffers per Random User	Specifies the number of data set buffers per random user. For a Random user, the reading or updating operation on the Dataset is done randomly. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on how buffers are allocated to random and serial users. This property's value ranges from 0 to 254. By default: • This property is set to 1 for Segments • This property assumes the value of owner for <<Ispec>> classes, EventSets and Vanilla classes. This property is enabled if the Data Set Buffers per User property is set to 0. This property is disabled for <<Ispec>>, <<Event>> and Vanilla classes: • If the class inherits from another class • If the class is an <<event>> class whose EventSet property is not spaces (that is, an AutoPersist dependency with another event class has been defined). This property is disabled if the Data Set Buffers per User property is not set to 0. When it is disabled, the Data Set Buffers per Random User property is reset to 0. Refer to **Note** in the definition of Data Set Buffers property. This property is specific to <<ispec>> and vanilla classes, EventSets and Segments only.
Data Set Buffers per Serial User	Specifies the number of data set buffers per serial user. For a Random user, the reading or updating operation on the Dataset is done randomly. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on how buffers are allocated to random and serial users. This property's value ranges from 0 to 254. By default: • This property is set to 2 for Segments • This property assumes the value of owner for <<Ispec>> classes, EventSets and Vanilla classes. This property is enabled if the Data Set Buffers per User property is set to 0. This property is disabled for <<Ispec>>, <<Event>> and Vanilla classes: • If the class inherits from another class If the class is an event class whose EventSet property is not spaces and is not set to a value that is the same name as itself (that is, an AutoPersist dependency with another event class has been defined). This property is disabled if the Data Set Buffers per User property is not set to 0. When it is disabled, the Data Set Buffers per Serial User property is reset to 0. Refer to **Note** in the definition of Data Set Buffers property. This property is specific to <<ispec>> and vanilla classes, EventSets and Segments only.

Property	Function
Data Set Buffers per User	Specifies the number of data set buffers per user. This property's value ranges from 0 to 254. By default: • This property is set to 0 for Segments. • This property assumes the value of owner for <<Ispec>> classes, EventSets and Vanilla classes. This property is enabled if the Data Set Buffers per Random User property and the Data Set Buffers per Serial User property is set to 0. This property is disabled for <<Ispec>>, <<Event>> and Vanilla classes: • If the class inherits from another class. • If the class is an <<event>> class whose EventSet property is not spaces (that is, an AutoPersist dependency with another event class has been defined). This property is disabled if the Data Set Buffers per Random User property is not set to 0 or the Data Set Buffers per Serial User is not set to 0 and also reset the Data Set Buffers per User property to 0. Refer to **Note** in the definition of Data Set Buffers – Buffers property. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.
Display Population Increase	Specifies whether to display a message when an automatic population has occurred. This property is enabled only if the Population Increase property is set. By default, this property is set to True. This property is specific to <<ispec>> and vanilla Classes, EventSets, and Segments only.
Display Population Warning	Specifies whether to display a warning when the population reaches the percentage of the set limit. This property is enabled only if the Population Warning property is set. By default, this property is set to True. This property is specific to <<ispec>> and vanilla Classes, EventSets, and Segments only.

Property	Function
Dumpstamp	Specifies whether to enable the use of the DUMPSTAMP option in DASDL for all data sets. The DUMPSTAMP option provides the facility for incremental backups by adding a timestamp word to each block of the affected structure. You can only use this property if you have the required license key on the target host. If you enable this property and you do not have the license key, a dialog is displayed at the beginning of the build, giving you the option to continue without the Dumpstamp option or cancel the build. Refer to the *DASDL documentation* for more information on the DUMPSTAMP option. The following options are available: • DMSII Default: 'DUMPSTAMP' not generated. • Yes: 'DUMPSTAMP = TRUE' is generated. • No: 'DUMPSTAMP = FALSE' is generated. At the segment, the default setting is DMSII Default. At the class level, the default setting is <inherit from parent or default>. This property is specific to <<ispec>> and vanilla Classes, EventSets, and Segments only.
Extended Edition	Specifies whether to enable the use of the DMSII EXTENDED option. Using the DMSII EXTENDED option causes the dataset to have records that contain two additional words at the beginning of the record: Record Serial Number and an internal Transtamp field. These provide improvements to dataset access. Refer to the EXTENDED option in the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information. To use this option, you must have the DMSII Extended Edition key installed on your MCP machine. ***Caution:*** *Setting the Segment Extended Edition property to True for an existing system that has already been generated with Extended Edition set to FALSE, or vice versa, may cause the REORGANIZATION program to exceed size limits for large systems.* By default, this property is set to False. This property is specific to <<ispec>> and vanilla Classes, EventSets, and Segments only.
Guardfile Information – Name	Specifies the file name for Guardfile information. The Guardfile information designates the name and location of the guard file, if any, that controls access to the database files through the Accessroutines. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the GUARDFILE option. This property value is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.

Property	Function
Guardfile Information – Pack	Specifies a valid pack name for Guardfile information. This property value is limited to a maximum length of 17 alphanumeric characters. Select a pack from the list of available packs. By default, this property is empty. This property is specific to Segments only.
Guardfile Information – User	Specifies a valid username for Guardfile information. This property value is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.
Number of Sections	Specifies the number of sections for an Event set or Component set. If you enter a value of 0 or 1, the Events or Components are not sectioned. This property can be set if the Extended Edition property is enabled. This property is limited to a maximum length of three digits in the range of 2 to 255. By default, this property is set to 0. This property is specific to <<Ispec>> classes, EventSets, and Vanilla classes only.
Optimize Blocksize to VSS-2	Specifies whether to enable the blocksize feature. A new blocksize value, in records, is calculated. This option is used to optimize the calculation of DASDL blocksize attributes. By default, this property is set to False. If you wish to use the populations in excess of 268,435,455 with classes or ispecs, set **Optimize Blocksize to VSS-2** to True and create a DWORD registry key called ActivateVSS2blocking in the following location: • For 32-bit: HKEY_LOCAL_MACHINE?\SOFTWARE\Unisys\System Modeler\Features\Builder • For 64-bit: HKEY_LOCAL_MACHINE?\SOFTWARE\Wow6432Node\Unisys\System Modeler\Features\Builder ***Caution:*** *Setting* ***Optimize Blocksize to VSS-2*** *to True and creating a DWORD registry key called ActivateVSS2blocking causes a database reorganization with the next system build. Likewise, removing the ActivateVSS2blocking registry key after building a system with it causes a database reorganization with the next system build.* This property is specific to <<ispec>> and vanilla classes, EventSets and Segments only.

Property	Function
Population Increase	Specifies a percentage by which the population increases if it exceeds its limit. Values greater than 0 percent allows an automatic increase in the population by up to that percentage if the population exceeds its current limit, up to the system limit of 1000 areas. By default, this property is set to 0. The numbers range from 0 to 100 in percentage. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.
Population Warning	Specifies a percentage of the population at which a warning message is issued. Values greater than 0 percent causes the system to issue a warning message when the population reaches that percentage of its limit. This property is limited to a maximum length of two digits in the range of 0 to 99 percent. By default, this property is set to 0. The numbers range from 0 to 99 in percentage. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.
Reblock Factor	Specifies the Reblock Factor. The REBLOCKFACTOR allows the user to specify the factor by which the blocksize on a Dataset is increased during serial access. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the REBLOCK and REBLOCKFACTOR options. This property is limited to a maximum length of two digits in the range of: • 1 to 17 for Segments. • 1 to 60 for Classes By default, this property is set to 1. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.
Securityguard Information – Name	Specifies file name for Securityguard information. Securityguard Information is used to set the SECURITYGUARD option in the DASDL. The SECURITYGUARD option sets the SECURITYGUARD file attribute to the name and location of the guard file that controls direct access to the database files by programs other than the Accessroutines. Refer to the *Data and Structure Definition Language (DASDL) Programming Reference Manual* for more information on the SECURITYGUARD option. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.

Property	Function
Securityguard Information – Pack	Specifies a valid pack name for Securityguard information. Select a pack from the list of available packs. By default, this property is empty. This property is specific to Classes and Segments only.
Securityguard Information – User	Specifies valid username for Securityguard information. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to <<ispec>> and vanilla Classes, EventSets and Segments only.
Sections	Specifies the number for a class and generates a statement for the defined sections. This property is enabled when the **Extended Edition** property of the class is set to True. The following rules apply to the value of this property: • If the **Sections** property is greater than one and the **Extended Edition** property of a segment is set to True, a statement for sections is generated, such as SECTIONS = nn. • If the **Sections** property is equal to zero or one or the **Extended Edition** property of the class is set to False, a statement for sections is not generated. By default, this property is set to 0. The numbers range from 0 to 511. This property is specific to <<Ispec>> Classes, vanilla classes and EventSets only.
Use Calculated Buffers	Specifies how buffer statements are generated for individual Profiles. By default, this property is set to True. When this property is set to True, the buffer statements for individual profiles use a calculated value. When this property is set to False and the buffer values for the profile match the Set Buffer values of the segment, no buffer statements are generated for the profile (the profile uses the default values specified for the segment). When set to false and the buffer values of the profile differ from the Set Buffer values of the segment, buffer statements are generated using the profile settings. This property is specific to Segments only.
VSS Warning	Specifies whether to issue a warning when the blocking attributes of the structure are not optimized for the operational characteristics of the storage device. This property provides a runtime indication of discrepancy between the blocking attributes of a structure and the storage device type upon which it resides. By default, this property is set to False. This property is specific to Segments only.

Debugging

Property	Function
Code Set	Specifies the Code Set in which you want LRSS to interpret data that is being sent. This property is limited to a maximum length of 20 alphanumeric characters. By default, this property is empty. This property is enabled when: • Deployable is set to True • Enable Remote CALL Statements is set to True This property is specific to Folders only.
Data Translation Routines Exist	Specifies whether to allow user-developed hook routines to be invoked, to translate Windows character sets to the matching character representation on the target host. This option prompts Developer to look for hook routines to invoke before sending data to the remote host, and immediately after retrieving data from successfully called functions on the host. By default, this property is set to False. This property is enabled when: • Deployable is set to True. • Enable Remote CALL Statements is set to True. This property is specific to Folders only.
Enable Host Database Access	Specifies whether to enable Debugger to access the deployed database (target host database) for the application. Host Database Access (HDBA) allows you to access data directly from a generated target platform. This means that when you execute a database command, such as Determine, Lookup, or ForEach, the records are retrieved from the DMSII database on the targeted MCP platform instead of the local SQL Server database. ***Caution:*** *The HDBA functionality is for debugging purpose only. Do not use it with your deployed production databases, or you destroy the integrity of your production database, risk losing data, and risk corrupting your database.* By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders only.

Property	Function
Enable Remote CALL Statements	Specifies whether to enable Debugger to access the subroutine on the target host. If enabled, the following Remote CALL Statement Attributes is also enabled: • Data Translation Routines Exist • Language • Code Set • Primary Disk • Secondary Disk By default, this property is set to False. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment.
Language	Specifies the language in which you want the Remote Subroutine Server (LRSS) to return error messages. This property is limited to a maximum length of 20 alphanumeric characters. By default, this property is empty. This property is enabled when: • Deployable is set to True • Enable Remote CALL Statements is set to True This property is specific to Folders only.
Port Number (HDBA)	Specifies the port number of the Host Database Access (HDBA). By default, this property is set to 1871. The numbers range from 1 to 9999. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment.
Port Number (LRSS)	Specifies the port number of the Remote Subroutine Server (LRSS). By default, this property is set to 6004. The numbers range from 1 to 9999. This property is enabled if Deployable is set to True. This property is specific to Folders only.
Primary Disk	Specifies the disk on which LRSS is to initially locate the call subroutine. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when: • Deployable is set to True • Enable Remote CALL Statements is set to True This property is specific to Folders only.

Property	Function
Secondary Disk	Specifies the disk on which LRSS is to locate the call subroutine if it cannot be found on the Primary Disk. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is enabled when: • Deployable is set to True • Enable Remote CALL Statements is set to True This property is specific to Folders only.

Environment

Property	Function
Areas	Specifies the number of areas for the extract file. If the number of areas is 0, value for areas is calculated.
Area Size	Specifies the area size for the extract file. If area size is 0 or less than the block size (RecordsPerBlock times the size of the maximum record length in the extract file), value for area size is calculated.
Block Size Algorithm	Specifies which algorithm to use while calculating the block size (Records Per Block = 0). The following options are available for this property: • Default – This algorithm is equivalent to the EAE block size algorithm. The resultant blocksize is similar to one calculated by EAE. • Large – This algorithm calculates the largest optimal blocksize for the extract file. • Small – This algorithm is equivalent to the LINC block size algorithm. This option is provided to those who still need the block size that is produced by this algorithm. The large block size algorithm is recommended for extract files with a large Expected Number.

Property	Function
Copy Audit to Tape	Specifies whether to copy the Audit files of a DMS II job to a magnetic tape. The following options are available for this property: • None – Indicates that the Audit files remain on pack until physically removed. • QIC Tape – Indicates that the Audit files are copied to a 1250 BPI QIC tape device. • GCR Tape 2200 – Indicates that the Audit files are copied to a GCRTAPE 6250 BPI (2200 ft) tape device. • GCR Tape 3200 – Indicates that the Audit files are copied to a GCRTAPE 6250 BPI (3200 ft) tape device. • HS8500 Cartridge – Indicates that the Audit files are copied to a HS8500 (BPI11000) 8mm cartridge device. • User Defined – Indicates that you can define your own tape device or specify the specific size of your audit files on pack. • TAPE 3800 – Indicate that the Audit files are copied to a TAPE 3800 BPI (1/2 inch cartridge) device. • TAPE 1600 – Indicates that the Audit files are copied to a TAPE 1600 BPI (2200 ft) tape device. By default, this property is set to None. When a value other than None is selected, a DMS II job DATABASE/WFL/COPYAUDIT is initiated whenever a DMS audit trail is closed and a new audit trail is started. This job copies your Audit files to magnetic tape and then deletes the files from the pack. If User-defined is selected, you can define your own tape device or specify the specific size of your audit files on pack. The specific attributes can be set using the User Defined Copy Audit Details property that is enabled. This property is specific to Segments only.
Expected Number	Specifies the expected number of records for the corresponding extract files. This property’s value ranges from 1 to 99,999,999,999. By default, this property is set to 0. This property is enabled when **Kind of File** property is set to Disk. This property is specific to Extract files (classes inheriting from GLB.FILE).
Expected TPH	Specifies the expected number of transactions per hour (TPH), expressed in thousands per hour. The expected number of transactions per hour is used for calculating the Input Queue Memory Size for the COMS program delimiters. By default, this property is set to 5. The numbers range from 5 to 999. This property is specific to Segments only.

Property	Function
External DASDL File	Specifies the path and name of your user-created DASDL source file name. If specified, Developer checks whether the file exists. The following naming conventions apply to the DASDL source file: • For a Build Configuration (the segment configuration property Configure Set Type is set to Build, for generates), the file name must be in Windows format and the path must be visible to Developer. The directory path can be on the same machine or a mapped directory on another machine. For example, F:\GENERATE\DASDL. • For a Configure Configuration (the segment configuration property Configure Set Type is set to Configure, for Runtime Transfer), the file name should be in Host MCP format and the path must be the host directory where the file is located. The file name should not include any usercode or packname for MCP hosts. For example, (MYUSER)CONFIGURE/DASD ON MYPACK. This property is limited to a maximum length of 80 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.
External Database - Name (DB1) External Database - Name (DB2)	Specifies the name of the external database. ***Note:*** *You can specify the name details for up to two external databases For example, details of external databases, DB1 and DB2.* This property is limited to a maximum length of 17 characters. By default, this property is empty. This property is specific to Segments only.
External Database - Pack (DB1) External Database - Pack (DB2)	Specifies the name of the pack for the external database. ***Note:*** *You can specify the pack details for up to two external databases For example, details of external databases, DB1 and DB2.* Select a pack from the list of available packs. By default, this property is empty. By default, this property is empty. This property is specific to Segments only.
External Database - User (DB1) External Database - User (DB2)	Specifies the user name of external database. ***Note:*** *You can specify the user details for up to two external databases For example, details of external databases, DB1 and DB2.* This property is limited to a maximum length of 17 characters. By default, this property is empty. This property is specific to Segments only.
Generate as Coroutine	Specifies whether the Report is to be generated as a coroutine for invocation by a method within the system or by an external program. By default, this property is set to False. This property is specific to Reports only.
Generate Full LINCOFF	Specifies whether to generate the LINCOFF file. By default, this property is set to False. This property is specific to Segments only. Refer to the *Agile Business Suite Runtime for ClearPath MCP Administration Guide* for more information on generating the LINCOFF file and compiling NOF programs.

Property	Function
High Frequency	Specifies whether faster access is to be provided at runtime. By default, this property is set to False. This property is enabled if Ispec Type is set to Table only. This property is specific to Ispecs and classes only. ***Note:*** *Excessive use of this facility can result in wasteful use of memory resources.*
Ispec Type	Specifies the database usage by the ispec or vanilla class. The following options are available for this property: • Default – Select this option for normal database usage. • Table – Select this option for high frequency access. • Direct – Select this option to create a direct dataset in the DASDL. By default, this property is set to False. This property is specific to <<ispec>>, <<event>>, and vanilla classes only.
Kind of File	Specifies whether the extract file is a tape or disk file. Extract files created and read on tape are single files on single or multi-reel tapes. By default, this property is set to Disk. ***Note:*** *If this value is set to Tape, the corresponding Records Per Block (Tape) and Tape Volume Label properties must be set to valid values and cannot be set to zero.* This property is specific to Extract files (classes inheriting from GLB.FILE).
POF Duplicate Family	Specifies the name of a duplicate pack family for the Protected Output File, if you want a duplicate file for recovery purposes. The duplicate file takes the name you have entered in the POF Name property. This property is limited to a maximum length of 17 characters. By default, this property is empty. This property is specific to Segments only.
POF Family	Specifies the name of the pack family for the Protected Output File. This property is limited to a maximum length of 17 characters. By default, this property is empty. This property is specific to Segments only.
POF Name	Specifies the name of the Protected Output File (POF). This file is used by COMS to manage the recovery of output messages. This property is limited to a maximum length of 10 characters. By default, this property is empty. This property is specific to Segments only.
Protected Input	Specifies whether to implement COMS input protection. By default, this property is set to False. This property is specific to Segments only.

Property	Function
Record Format	Specifies the format (Internal or Generic) of an extract file. The following options are available for this property: • I – Specifies to extract the file in the Internal format of the host you are extracting from. • G – Specifies to extract the file in a Generic format so that the file can be loaded into a system running on a different host type. ***Notes:*** • *The Generic format only applies to Ispec extractions; the format for frame extractions is not affected by this setting.* • *The current release of AB Suite does not support the sharing of generic extract files between platforms when the records within them contain aggregated instances.* By default, this property is set to I This property is specific to Extract files (classes inheriting from GLB.FILE).
Records Per Block	Specifies the number of records per block for the corresponding extract file. This property's value ranges from 1 to 99999999. By default, this property is set to the default. You should be aware of the minimum and maximum physical block sizes for the device being used, and ensure that your specified block/record size falls within those limits. This property is specific to Extract files (classes inheriting from GLB.FILE).
ROC uses Database	Specifies whether to store Report output in a ROC database. By default, this property is set to False. This property is specific to Reports and Segments only. For Reports, this property is enabled if the System Uses ROC property for the Segment is set to True for the application. If this property has been set for the Segment, then the Reports should inherit the value for this property from the Segment. Otherwise, you can set this option for each report using this property.
Sort Disk Size	Specifies the required sort disk size in kilowords. You can significantly improve the performance of SORT logic commands by making more disk space available to the sort operation. This property's value must be in the range 1 through 999999. By default, this property is empty. If no value is entered, the default is calculated in the Clearpath MCP environment. Refer to Calculation of Default for more information. This property is specific to Reports only.
Sort Memory Size	Specifies the memory size in kilowords for SORT logic command. The value of this property ranges from 0 through 99999999. By default, this property is set to 0. Refer to Calculating Sort Memory Size for more information. This property is specific to Reports only.

Property	Function
Subsystem	Specifies the subsystem to where it belongs. By default, this property is set to the default subsystem. This property is limited to a maximum length of 10 alphanumeric characters. This property is specific to Classes only.
Tape Volume Label	Specifies to use labeled tapes. This property is enabled when **Kind of File** property is set to Tape. The following options are available for this property: • True – If files are labeled • False – If files are not labeled By default, this property is set to True. This property is specific to Extract files (classes inheriting from GLB.FILE).
User Defined Copy Audit Details	Specifies the details of the tape device or the details of the audit files on pack as defined by the user. This property is enabled if the Copy Audit to Tape property is set to User Defined. This property is limited to a maximum length of 10 lines of 66 characters or 660 characters. By default, this property is empty. This property is specific to Segments only.
User Privilege	Specifies the user terminal access to an Ispec at runtime. Each terminal has an assigned security level which is defined as Integer3 Installation Data record in the COMS Configuration File in the ClearPath MCP-based COMS utility. Refer to the *Enterprise Application Runtime for ClearPath MCP Administration Guide* for more information. In Ispec logic at runtime, the privilege level of the terminal currently accessing the System is contained in the System Attribute GLB.PRIV. By default, the value is 1. The privilege ranges 1 from 15. This property is specific to Ispecs and Events only.
Transferable	Specifies whether the Report is to be transferred to another host using the All Transferable Reports option on the Selected Transfer screen of the Runtime Transfer Utility (RTU). This option relates specifically to the MCP Runtime Transfer process, during which options may be selected to decide which reports are transferred to the target system. By default, this property is set to False. This property is specific to Reports only.

Calculation of Default for Sort Disk Size

If no value is entered for the Sort Disk Size property, the default is calculated in the ClearPath MCP environment.

Default sizes are applied as follows:

- If no Extract files to be sorted have a specified Expected Number value (on the Extract File Options page), a value of 20 modules (16,384 words of memory) is used. Otherwise, the following formula is calculated for every Extract file to be sorted that has a value expected specified in its **Expected Number** property. The value maxframe is the largest record size for Frames in the Extract file.

  (Expected / 1000) * maxframe * 2.25) / 10800
- If the largest value for these Extract files is 20 or more, that number of modules is used.
- If the largest value for these Extract files is less than 20 and all Extract files to be sorted have been considered (that is, they all have a value in the Expected Number property), then that number of modules is used.
- If the largest value for these Extract files is less than 20 but not all Extract files to be sorted have been considered (that is, they do not all have a value in the Expected Number property), then a value of 20 modules is used.

Example 1

A Report has several Frames to be extracted and sorted in Extract file A. The largest Frame record size is 240, and Extract file A has an expected number of 1,000,000. If you do not enter a value for the **Sort Disk Size** property, the default is 50 modules, calculated as:

(1,000,000 / 1000) * 240 * 2.25) / 10800 = 50

Example 2

A Report has one Frame to be extracted and sorted in Extract file B. The Frame record size is 80, and Extract file A has an expected value of 10,000. If you do not enter a value in the **Sort Disk Size** property, the default is 1 module, which is the rounded-up result from the calculation:

(10,000 / 1000) * 80 * 2.25) / 10800 = 0.17

Note: *If any other Extract files that were to be sorted did not have an expected number, the default becomes 20 modules.*

Calculation of Sort Memory Size

The following formula is used to calculate the required memory size for each of the sorts executed in the report:

((((<Maxrecsize>/6) + ((<key1> + <key2> +...+ <keyN>)/6) + 3) * <number of records>)/1000) + 1

Where,

- <Maxrecsize> is the length of the maximum frame or class record size extracted to the extract file that is sorted.
- <key1>, <key2>… <keyN> are the lengths of the keys used in the sort.
- <number of records> is the value in the Expected Number property of the extract file.

You must first calculate the memory size for each sort and then enter the largest value in the Sort Memory Size configuration property. It is recommended to specify minimum of two kilowords and maximum of 200 kilowords for memory sort.

Refer to the ClearPath Enterprise Servers System Software Utilities Operations Reference Manual for details on the limitations with memory sort.

External

The external configuration properties of the Builder are used to determine and establish a connection with an external resource on a Windows® platform.

The **Component Type** property of an external class broadly defines the component type of the external resource that implements its functionality. You must specify the component type because AB Suite generates different external class wrappers for different component types.

To set the configuration properties, right-click an external class, and select **Properties**.

For an imported data source, set the **Data Source Location** property of the external class to either **Defined Locally** or **Location**. An error message is displayed if you enter an invalid value other than Defined Locally or an existing Location name.

Note: *You can select an existing **Location** to specify predefined connection information or set the properties within the class configuration with the **Defined Locally** option. Refer to **Data Source** properties in System Modeler Locations for more information on setting the properties for the Defined Locally option.*

Property	Function
Component Type	Specifies the type of external component to be interfaced with an external class. For example, .NET, COM, DLL, EXE, Java, or Shell. By default, this property is set to **EXE**.
Identifier	Specifies the ID of the external component: • Prog ID for COM objects • Name for .dll files, .exe files, and shell programs • Qualified name for .NET objects By default, this property is set to the name of the external class.

Property	Function
.NET Assembly	Specifies the full path of the assembly containing the .NET component type. This can be the compiled DLL for the .NET component. This property is only available if the Component Type is set to **.NET**.
Default Interface	Specifies the name of the dispatch interface that contains the methods to be invoked. You must define an external class for every interface if you use methods from different interfaces of the component. ***Note:*** *Interfaces can only be used if they are in the same namespace as the class. This property can contain alphanumeric and '_ ' characters.* This property is enabled if the Component Type is set to either **.NET** or **COM**.
Type_Library	Is optional. If specified, AB Suite uses the type library to retrieve the GUIDs of the COM-Visible classes and methods of the external component. Otherwise, AB Suite generates its own type library through the tlbexp and regasm calls. The type library is used if the interface of the component is defined in another library. You can select **Make assembly COM-Visible** in the **Assembly Information** window to create a COM-Visible assembly for an external component. This property is enabled if the Component Type is set to **COM** or **.NET**.

Note: *When you build an application in a Debug Mode, you must ensure that the following rules are met to avoid errors during build:*

- *The .NET Assembly must be COM Visible, which can be set globally in the Assembly Information properties window.*
- *The interface of the assembly must be defined as Public.*

However, these settings are not required when you build an application for Windows® *runtime, and it builds the application with a warning message.*

General Configuration

Property	Function
Access Code	Specifies the access code for the Generate job. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Folders only.
As Coroutine	Specifies whether the external component is a MCP co-routine.
Build Dependencies	Specifies whether classes (that are dependencies of the current selected build) are automatically built. By default, this property is set to True. This property is specific to Segments only.

Property	Function
Builder Cache Location	Specifies the location of the builder cache folder. This property value overrides the builder cache location from Tools > Options > System Modeler > Builder > General. By default, the value of the property is empty. ***Note:*** *This property is not applicable to debugger.*
By Function	Specifies whether the external MCP library is called by its function name or title.
Charge Code	Specifies the charge code to be used for application accounting purposes. All resource usage by the Generate is debited to this charge code. This property is limited to a maximum length of 45 alphanumeric characters. By default, this property is empty. This property is specific to Folders only.
COM Prog ID	Specifies the COM object's Prog ID. This property can contain alphanumeric and '_' characters. By default, this property is the name of the Segment or Report. This property is specific to Segments and Reports only.
Deployable	Specifies whether a folder represents a deployable unit. By default, this property is set to False. This property is specific to Folders only.
Description	Specifies a description of the folder. By default, this property is empty. This property is specific to Folders only.
Environment Name	The name of the environment for an Application Server. (for example, JBOSS1, JBOSS2, etc). This must be one of environment names configured in Deployment configuration of NGServer.
Exclude From The Build	Specifies whether the element is excluded from the Build. By default, this property is set to False. This property is specific to Segments and Reports only.
Generate As Name	Specifies a name of a report when generated from this folder that is different from the default name of a report. The Generate As Name property takes precedence over Alternate Name, which in turn takes precedence over Alias or Model Name. • This property is enabled if the Deploy Reports property is set to **True**. You can either: • Select the **<inherit from parent or default>** option to reset the property to default name. Or • Select the **<Edit...>** option to edit the value of the property in the Generate As Dialog Box. This property is specific to the folders contained within a Segment.

Property	Function
Glb.SysVersion	Specifies the current version of the system. This number increments with every build. By default, this property is empty. This property is read-only. This property is specific to Folders only.
Library UserCode	Specifies the usercode for the external MCP library or co-routine. This property is disabled if the **By Function** property is **True**.
Library Name	Specifies the title or the function name of the external MCP library, or the name of the external MCP co-routine.
Library Pack	Specifies the location used to connect to the external MCP component. This property is disabled if the **By Function** property is **True**.
Start Deployment With	Option to allow control over the generate, transfer and install stages of a deployment. • Generate – The phase of deployment where an MSI package is compiled and created. • Transfer – Transfer the package to the Package Intermediate Directory • Install – Install the package in the Package Installation Directory • Compile and Package – The phase of deployment where a package is compiled and created. • Deploy – Generate, compile, package and deploy the application. By default, this property is set to Generate. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment. ***Note:*** *This setting can also be overridden later, just before the Build using the Build Details dialog box.*

Property	Function
Stop Deployment After	Option to allow control over the generate, transfer, install and run stages of a deployment. • Generate – The phase of deployment where an MSI package is created. • Transfer – Transfer the package to the Package Intermediate Directory. • Install – Install the package in the Package Installation Directory. • Run – Runs the deployed application from the installation directory. • Compile and Package – The phase of deployment where a package is compiled and created. • Deploy – Generate, compile, package and deploy the application. • Restart Application Server – Deploy the application and restart the application Server. This property is enabled if Deployable is set to True. This property is specific to Folders not contained within a segment. ***Note:*** *This setting can also be overridden later just before the Build using the Build Details dialog box.*
Usercode	Specifies the usercode for the Generate job. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Folders only.

Generate As Dialog Box

The **Generate As** dialog box enables you to view and modify information on reports. The report information is available in a table of four columns for a specific folder configuration. The table displays the following columns:

- Report Name: Specifies the name of a report. This value is read-only.
- Default Name: Specifies the report configuration Alternate Name, if defined. If this value is not defined, the default name property contains the Alias for an MCP configuration or Model name for a Windows configuration. This value is read-only.
- Generate As: Specifies the generated name of a report. You can edit this value and ensure that the value does not exceed 10 alphanumeric characters for MCP configurations.

  Note: *There is no such restriction for Windows® configurations.*
- Status: Specifies the status of individual reports. This value is read-only. If a report does not exist directly under the folder, a status of "Missing" appears against the report; else, the status is blank. A status of "Conflict" appears against a report along with a conflict message in the lower pane when a conflict occurs.

A conflict can occur when

– The Generate As value is the same for two or more reports

– The Generate As value of a report is blank and the default name of the report is the same as the Generate As value of another report.

In such situations, you must rename the Generate As value to resolve the conflicts.

Notes:

- *If the length of the generated name is less than 100, the **Generate As Remaining Characters** displays the number of characters in red.*
- *If the Generate As value is not yet defined for a report, a value of Not Specified appears against those reports.*

Installation

Property	Function
Additional Install Script	This script is called after the AB Suite Deployment process. It is run after the application has been successfully deployed and is used to run user-specific post-deployment configuration steps. By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is not set to Generate. This property is specific to Folders only. An example would be to copy Winform configuration files to a file server to automate Winform distribution.
Additional Uninstall Script	This script is called after the AB Suite application is deleted, usually from the Admin Tool. It is run after the component has been removed and any DBReorg operations. It is used to run user-specific post-uninstallation clean up steps. By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is not set to Generate. This property is specific to Folders only. An example would be to de-register 3rd party external components that are no longer used or needed after the AB Suite application is removed.
Clone Database	Specifies whether to clone the database for use as a secondary remote database. This property is only valid if you set a valid name in the RDB Configuration property. By default, this property is set to False. This property is specific to Segments only.

Property	Function
Deployment Host	Specifies the name of the deployment server. For Unix, it is the DNS name or the IP address of the machine where the application is to deployed to. By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is set to Install or Run. ***Note:*** *For a Windows host only, precede the name with a double backslash \\, unless the entry is localhost* This property is specific to Folders not contained within a segment.
Deployment Interface Type	Currently only one deployment interface is supported, the default windows deployer. By default, it is Default Windows Deployer. This property is specific to Folders only.
Deployment Port Number	For Unix, it is the port number of the host where the application is to be deployed.
DMSII Pack	Specifies the name of the pack (location) where an alternative version of DMS II software is located. This pack and the value entered in the DMS II Usercode field identify the alternative version of DMS II to be used when building the DASDL source and by the deployment phase when compiling the database. Select a pack from the list of available packs. By default, this property is empty. This property is specific to Segments only.
DMSII Usercode	Specifies the usercode for use with an alternative version of DMS II software. This usercode and the value entered in the DMS II Pack field identify the alternative version of DMS II to be used when building the DASDL source and by the deployment phase when compiling the database. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.
External Installer File Deployment Script	Specifies a script that is executed during installation to install anything from the External Internal Files property. That is, to install anything that is not from Agile Business Suite. By default, it is empty. This property is specific to Folders only.
Number of Parallel Compiles	Specifies the number of compile threads (multithreading) that run concurrently. By default, this property is set to 1. The numbers range from 1 to 99. This property is specific to Segments only.
Number of Reorg Tasks	Specifies the maximum number of reorganization tasks that can occur simultaneously. By default, this property is set to 1. The numbers range from 1 to 9999. This property is specific to Segments only.

Property	Function
Package Installation Directory	Specifies the directory where the msi file installs, when it is run. Type a valid DOS path. For example, C:\my folder By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is not set to Generate. This property is specific to Folders not contained within a segment.
Package Intermediate Directory	Specifies where the package installer file(s) transfer to before the installation process begins. By default, this property is empty. This property is enabled if Deployable is set to True, and Stop Deployment After is set to Transfer, Install or Run. This property is specific to Folders only.
Package Name	Specifies the name of the msi file. By default, this property is set to the folder name. Package names must be unique within a project. This property is enabled if Deployable is set to True. This property is specific to Folders only.
RDB Configuration	Specifies the name of an RDB Configuration that is to be used in conjunction with this Configuration to setup/maintain a secondary database. Refer to the Backup, Recovery, and Remote Database section of the *Agile Business Suite Runtime for Clearpath MCP Administration Guide* for guidelines on this feature. By using Remote Database Backup, you can maintain an up-to-date copy of your production database on a different host for the purposes of disaster backup/recovery. If you wish to clone the database the next time the system is transferred, set the Clone Database property to True. This property is limited to a maximum length of 17 characters. By default, this property is empty. This property is specific to Segments only.
Reorg Allowedcore	Specifies the amount of memory that you require for database reorganization. The value is specified in words and is factored by 100,000 (1 = 100,000). If you enter a value greater than the amount of the available memory, the value is changed to eighty percent of the available memory. This ensures that reorganization does not fail. The property value ranges between 1 and 5,497,558 inclusive. By default, this property is set to zero. If zero or no value is entered in this property, the default value of this property is set to 100,000. This property is specific to Segments only.

Property	Function
Reorg Preverify	Specifies when to use the DMSII PREVERIFY option with a database reorganization. If you set Reorg Preverify to True, a pre-verification task is initiated for each generated data set before reorganization actually begins. This task locks the structure, preventing changes until the reorganization starts. It then reads all records in the data set to ensure the verify condition for records (for example, that required items are not null). PREVERIFY is not permitted with the REORGDB option. You have to ensure that the Reorg Preverify property is set to False before setting the Reorganziation Type property to REORGDB. Refer to the *Enterprise Database Server for ClearPath MCP Utilities Operations Guide* for more information. By default, this property is set to False. This property is specific to Segments only.

Property	Function
Reorganisation Type	Specifies the reorganization options. The following options are available for this property: • DMS Offline – Indicates that you do not want reorganizations to be audited, but you want to use the DMS II OFFLINE or NORESTART feature. When the reorganization is complete you are prompted to perform a database dump. Refer to the *Enterprise Application Runtime Administration Guide for ClearPath MCP* for more information on the Offline and NORESTART features. • Offline NoPostDump – Indicates that you do not want reorganizations to be audited, but you want to use the NOPOSTDUMP feature. This option is an extension of the DMS Offline option, but you arenot prompted to perform a database dump. • Online Reorg – Indicates that you want reorganizations to be audited. This may slow down your reorganization time. • ReorgDB – Indicates that you want to use the DMSII REORGDB option with the next reorganization. Refer to the *Enterprise Database Server for ClearPath MCP Utilities Operations Guide* for more information on usage and constraints of the REORGDB option. You do not need to shut down your database during the reorganization if AutoSwap is set to True and no user intervention is required. However, if AutoSwap is set to False, you need to shut down your system briefly at the time the reorganized structures are swapped. This allows you to install software changes at the same time. When you are asked to enter AX SWAPNOW, you must bring your system down before responding to the request. The generate operation continues as normal after that, and you may restart your system after the generate operation completes. When this option is selected, the following REORGDB properties are enabled: – REORGDB DB Title – REORGDB CopyTo – REORGDB AutoSwap – REORGDB TotalCopyCore – REORGDB AllowedCore – REORGDB KeepRSN – REORGDB NoRestart Regardless of the AutoSwap setting, if you use ReorgDB, ensure that you terminate any active reports prior to the deployment if they reference any database structure that is reorganized. By default, this property is set to DMS Offline. This property is specific to Segments only.

Property	**Function**
REORGDB AllowedCore	Specifies the amount of core memory that can be used for the reorganized database copy. The value is specified in words and is factored by 10,000 (1 = 10,000). This option for the REORGDB mode of reorganization serves the same purpose as the database ALLOWEDCORE parameter that is specified in DASDL. ***Note:*** *The DMSII REORGDBALLOWEDCORE option serves the role of database ALLOWEDCORE parameter only during the processing of updates. It does not have a role during the background OFFLINE reorganization.* By default, this property is set to 1000 (10,000,000 words). The numbers range from 1 to 99,999. This property must be set if Reorganization Type property is set to ReorgDB. This property is specific to Segments only.
REORGDB AutoSwap	In this release of Agile Business Suite Runtime for Clearpath MCP, this property is ignored. The DMSII REORGDBAUTOSWAP option is always set to FALSE. This property must be set if Reorganisation Type property is set to ReorgDB. By default, this property is set to False. This property is specific to Segments only.
REORGDB CopyTo	Specifies the database location where the production database structures are captured and applied during a REORGDB reorganization. When the REORGDB CopyTo option is set to spaces, the pack name of the existing structure is used. This property may be set if Reorganisation Type property is set to ReorgDB.
REORGDB KeepRSN	Specifies whether the KEEPRSN option is used with a REORGDB reorganization. Set this property to True to preserve the RSN value of new records created during a REORGDB reorganization. This property is enabled when Reorganization Type is set to REORGDB. By default, this property is set to False. This property is specific to Segments only.
REORGDB NoRestart	Specifies that the reorganization cannot be restarted. Select this option to enable the reorganization of the database copy designated by the REORGDBTITLE to proceed without performing excessive input/output operations that can slow the process. This property is enabled when Reorganization Type is set to REORGDB. By default, this property is set to False. This property is specific to Segments only.

Property	Function
REORGDB Title	Specifies the name of the database copy during a REORGDB reorganization. This property must be set if Reorganisation Type property is set to ReorgDB. This property allows a maximum length of 50 characters. By default, this property is empty. This property is specific to Segments only.
REORGDB TotalCopyCore	Specifies the value to be used with the REORGDB TOTALCOPYCORE option. The value is specified in words and is factored by 10,000 (1 = 10,000). This property must be set if the Reorganization Type property is set to ReorgDB. By default, this property is set to 5000 (50, 000, 000 words). The numbers range from 1 to 99,999. This property is specific to Segments only.
Sort Sets on Reorg	Specifies whether Agile Business Suite Runtime for Clearpath MCP includes explicit statements in the BUILDREORG deck for reorganizing datasets (<<Ispec>>/vanilla classes) and their sets/ subsets (Profiles) when the deployment process has determined that a structure needs reorganizing. For large structures, this can be a more efficient reorganization of the dataset and its sets. By default, this property is set to False. This property is specific to Segments only.
Suggested Number of Class DLLs	Defines the number of classes that is included in each DLL. The number of DLLs is determined by dividing the number of classes by this property value. For example, if Classes per DLL is set to 50 and you have 250 classes, you end up with 5 DLLs. Every group class is counted as a class.
User Deployment Script	This property allows specific deployment processes to be developed, tailored to the user's requirements and deployment context. This script is run after the generated deployment package is produced and placed in the package intermediate directory. It can be used, for example, to manually copy and deploy the application across non-domain network scenarios, or to run specific pre-deployment configuration steps. By default, this property is empty. If left empty, default process is used to control installation. This property is enabled if Deployable is set to True, and Stop Deployment After is set to Install or Run. This property is specific to Folders only. ***Note:*** *To run user-defined scripts, remote scripting must be enabled. Refer to the* Visual Studio Online Help *on how to set up remote Windows script hosts for more information.*

Property	Function
WFL Family	Specifies the MCP Family statement to be used by the generate process for storing your system and system software files. This property requires an entry if either of the following situations apply: The system software required during the installation (for example, the compilers) is not resident on either the primary or alternate pack family of the WFL usercode family. To address this situation, enter the statement (where pack is the pack family of system software): DISK = pack ONLY To ensure all work files are kept on your primary family, enter (where pack is the pack family of system software and primary is the primary family): DISK = primary OTHERWISE pack You are placing some or all of the installed host software onto a pack family called DISK. To address this situation, enter the statement: DISK = DISK ONLY Or, if the system software is not on the pack family called DISK, enter (where pack is the pack family of system software): DISK = DISK OTHERWISE pack ***Note:*** *If you specify an alternate pack, that is used as the alternate pack of the FAMILY statement associated with LSS and the Update programs within COMS, and therefore be the alternate pack for files accessed by those programs (for example, WFLs referenced by a START; command).* This property is limited to a maximum length of 44 characters. By default, this property is empty. This property is specific to Segments only.

NAP Direct Interface

Property	Function
Enable NAP Direct Interface	Specifies whether an application can communicate through a direct interface to external applications. The direct interface is the Network Application Platform (NAP). By default, this property is set to False. This property is specific to Segments only.
NAP GSDs – AIM Buffer	Specifies the name of your NAP AIM buffer Group attribute. This attribute must be a Group attribute, and cannot be part of another Group attribute. If you change the value in this property all attributes are regenerated. This property is enabled when the Enable NAP Direct Interface is set to True. This property is limited to a maximum length of 18 characters. By default, this property is empty. This property is specific to Segments only.

Property	Function
NAP GSDs – Global Work	Specifies the name of the NAP Global Work Group attribute. This attribute must be a Group attribute, and cannot be part of another Group attribute. If you change the value in this property all attributes are regenerated. This property is enabled when the Enable NAP Direct Interface is set to True. This property is limited to a maximum length of 18 characters. By default, this property is empty. This property is specific to Segments only.

National Support

Property	Function
Internationalization Language	Specifies the national language that is available on the host targeted by this Configuration. This language to be used as the collating sequence for a national item This property enables you to set your DASDL definition to specify a collating sequence (other than the default EBCDIC sequence) for the national Attributes stored on your database. This ensures that all text comparisons and sort sequences use the same collating sequence as the DMSII database. It also applies to your generated classes and methods. This property must be specified if any attributes of the Segment are defined as national items. A National Attribute in Agile Business Suite is a National String under the following conditions • National Support = Single-byte • Internationalization Language not set to NONE or spaces • Collating Sequence not set to NONE or spaces. The following rules apply to the value of this property: • If a segment contains some national items and the value of this property is empty then generate fails (with errors). • If a segment contains some national items and the value of this property is Ignore National then generate proceeds as normal (without errors). • If a segment does not contain any national items and the value of this property is empty or Ignore National then generate proceeds as normal (without errors). This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. It also has predefined value of Ignore National. This property is specific Segments only.

OLTP

The OLTP information is only relevant to Agile Business Suite systems for the MCP and Windows® platform.

Property	Function
Accept OLTP Transaction	Specifies whether the <<Ispec>> class is able to accept OLTP transactions. The <<Ispec>> class only receive transactions if you have set the Enable OLTP property to True and specified a service name in the OLTP Service Name property for the <<Ispec>> class. The OLTP Reply property is automatically set to True when you set the Accept OLTP Transaction property to True. These options can also be used independent of each other. By default, this property is set to False. This property is specific to Ispecs only. ***Note:*** *<<Copy Ispec>> classes cannot receive OLTP transactions.*
Default Service Name	Specifies the default OLTP service name. This property is specific to Segments only.
Enable OLTP	Specifies whether to enable the use of OLTP to access any services for elements of the Segment. By default, this property is set to False. If this property is set to True, additional OLTP properties are enabled. This property is specific to Segments only.
Load OLTP Configuration Data	Specifies whether to load the OLTP configuration data. By default, this property is set to True. This property is specific to Segments only.
Number of Concurrent OLTP Reports	Specifies the initial number of OLTP Interface Programs (system/OLTP/IP) to be configured for the application. By default, this property is set to 1. The numbers range from 1 to 99. This property is specific to Segments only.
OLTP Client	Specifies whether to enable this <<Ispec>> or <<Report>> class as an OLTP Client (accepts OLTP transactions). This property is enabled for Reports if OLTP Participation is set to OLTP Client and Server. By default, this property is set to False. This property is specific to <<Ispec>> and <<Report>> classes only.
OLTP Participation	Specifies whether the System participates as: • Both OLTP Client and Server, • OLTP Client only, or • OLTP Server only. By default, this property is set to OLTP Client and Server. This property is specific to Segments only.

Property	Function
OLTP Server	Specifies whether to enable this <<Ispec>> class as an OLTP Server (accepts OLTP transactions). By default, this property is set to False. This property is always set to True if OLTP Client is set to True. This property is specific to <<Ispec>> classes only.
OLTP Service Name	Specifies the OLTP service name for the <<Ispec>> class. The property has the following naming conventions: • Limited to a maximum length of 15 alphanumeric characters • Case sensitive • Must start with an alpha character and support local case when specified • Allows a maximum of 10 OLTP service Name to be specified. This field is enabled only if the Enable OLTP option is set to True. By default, this property is empty. You can leave this property blank to use the default OLTP service name for the subsystem. This property is specific to <<Ispec>> classes only.
OLTP Reply	Specifies whether to enable the <<Ispec>> class to reply to OLTP transactions. An <<Ispec>> class buffer definition is also generated for definition to OLTP. The OLTP Reply property is automatically set to True when you set the Accept OLTP Transaction property to True. These options can also be used independent of each other. An <<Ispec>> Class buffer definition is also generated for definition to OLTP By default, this property is set to False. This property is specific to <<Ispec>> and <<Event>> classes only. ***Note:*** *<<Copy Ispec>> classes cannot receive OLTP transactions.*
Report Timeout	Specifies the report timeout in seconds. By default, this property is set to 450. The numbers range from 1 to 9999. This property is specific to Segments only.
Transaction Timeout	Specifies the OLTP transaction timeout in seconds. By default, this property is set to 300. The numbers range from 1 to 9999. This property is specific to Segments only.

Pack Allocation

Property	Function
Audit Pack	Specifies the storage location for the Database Audit files. Select a pack from the list of available packs. If you do not specify an entry in this property, the Dictionary Pack entry is used. The Audit files should not be directed to the pack specified as the Default Pack. The entry in this field is ignored for audit purposes if you enter User Defined in the Copy Audit to Tape property under the Environment category of the Segment Configuration properties. However, the pack you enter in the Audit pack field is always used to hold the secondary copy of the GLB.UNIQUE file if that pack is different from the Dictionary Pack. ***Note:*** *The Audit Pack name must be different to the Default Pack name.* This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Dictionary Pack. This property is specific to Segments only.
Database Pack	Specifies the storage location for the Application Database structures. Select a pack from the list of available packs. If you do not specify an entry in this property, the Dictionary Pack entry is used. Database structures that do not have explicit packs specified for them are placed on the pack that you specify in this property. If you intend using the MCP-based Pack Mirroring feature, it is essential that you use this property to define one pack for all your Database structures. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.
Default Pack	Specifies the storage location for all packs in the Configuration. Select a pack from the list of available packs. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to SYSTEM. This property is specific to Segments only.

Property	Function
Dictionary Pack	Specifies the storage location for the following system and DMS files: • DMS II description and control file • DMS II database, unless one of the following options is taken: – A Database pack is specified. – Components, Events, and Audit files are redirected. – Structures and GLB-DIALOGINFO datasets are redirected. • GLI layout file for your system. • Screen layouts file for your system. • Control file for your system. Select a pack from the list of available packs. This property allows a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.
DuplicateAudit Pack	Specifies the storage location for the second copy of the DMS II database audit files. Select a pack from the list of available packs. If you specify an entry in this property, a copy of the audit files is created and is maintained on the pack specified in this property. These files are called the secondary or duplicate audit files. If you do not specify an entry in this property, secondary audit files are not maintained. The entry in this property is ignored for audit purposes if you enter User Defined in the Copy Audit to Tape property under the Environment category of the Segment Configuration properties. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is empty. This property is specific to Segments only.
Event Pack	Specifies the storage location for the Event Set data. This is also the default pack for all Profiles using Events. Select a pack from the list of available packs. If you do not specify an entry in this property, the Database Pack entry is used. If the Database Pack is not specified, then the Default Pack entry is used. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. All Profiles using Events reside on the Database Pack, or if that is not specified, they reside on the Default Pack. This property is specific to Segments only.

Property	Function
Extract Pack	Specifies the storage location for the Report Extract files. Select a pack from the list of available packs. This property allows a maximum length of 17 alphanumeric characters. By default, this property is set to DEFAULT which indicates that the Dictionary Pack entry is used and if that is not specified, the Default Pack entry is used. This property is specific to Segments only. Specifies the name of the pack (location) to where the extract data is to be stored. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the default pack. For example, SYSTEM. This property's value is used only for the first generate and is not used for Events. This property is specific to Reports only.
Log Pack	Specifies the storage location for the Log files. Select a pack from the list of available packs. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to Default Pack. This property is specific to Segments only.
Object Pack	Specifies the storage location for the object code of the system, its associated Reports, and for any temporary work files. Select a pack from the list of available packs. If you do not specify an entry in this property, the Dictionary Pack entry is used. If the Dictionary Pack is not specified, then the Default Pack entry is used. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.
Pack	Specifies a storage area (location) on the host for the application components. Select a pack from the list of available packs. This property is limited to a maximum length of 10 alphanumeric characters. By default, this property is set to: • Default pack for Classes. This property value is used only for the first generate and is not used for Events. • Is empty (which equates to default pack inside the application) for Profiles. This property is specific to Classes and Profiles only.

Property	Function
Reorg Pack	Specifies the storage location for structures created by the reorganization process of a retained database. Select a pack from the list of available packs. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to Default Pack. This property is specific to Segments only.
ROC O/P file Location	Specifies the storage location for the flat files on which Report Output Control files (ROC) files are stored. Select a pack from the list of available packs. If you do not specify an entry in this property, the ROC DB Location entry is used. If that is not specified, then the Default Pack entry is used. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.
ROCDB Pack	Specifies the name of the database family in which Report Output Control (ROC) files are stored. Select a pack from the list of available packs. If you do not specify an entry in this property, the Database Pack entry is used. If the Database Pack is not specified, then the Default Pack entry is used. This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.
StationInfo Pack	Specifies the storage location for the GLB-DIALOGINFO information. Select a pack from the list of available packs. If you do not specify an entry in this property, the Database Pack entry is used. If the Database Pack is also not specified, then the Default Pack entry is used This property is limited to a maximum length of 17 alphanumeric characters. By default, this property is set to the Default Pack. This property is specific to Segments only.

Persistence

Property	Function
Alternate Name	Specifies an alternate name to be used in database and system definitions. This property can contain alphanumeric and '_' characters. By default, this property is empty. This property is specific to Segments, persistent Classes, Ispecs, Events, Copy Ispecs, Copy Events Profiles, Persistent Attributes and Methods only.

Property	Function
Cluster Index	Specifies the profile to be used as the clustered index for the class's database table. By default, this property is empty. This property is specific to persistent classes only.
COMS Window Name	Specifies the name for the COMS window under which your generated systemis identified to COMS. This property is limited to a maximum length of 10 alphanumeric characters. By default, this property is set to the Segment name. This property is specific to Segments only.
Database Name	Specifies the name of the database. The name is case sensitive and should match the name of the database exactly. By default, this property is empty. This property is specific to Segments only.
Database Schema Name	Specifies the name of the database schema. This schema name is created as a database user. Type a name in uppercase letters. By default, this property is set to the Segment name. This property is specific to Segments only. Tables of each system are owned by the user derived from the database schema name. If two systems use the same database schema name, then there could be a possibility that the other system could be disrupted when one system is being reorganized. Hence using a unique database schema name is recommended.
Database Server Registration	Specifies the name of the runtime's database server registration that refers to the database server to which you want to deploy the generated application. The name is case sensitive and should match the name of the database registration exactly. By default, this property is empty. This property is specific to Segments only.
Database Tablespace	Specifies the location of the object's database table. By default, this property is empty. This property is specific to persistent classes and Profiles only. Maximum of 30 characters, the first of which must be alphabetic.

Property	Function
Expected Number	Specifies the expected population for ispec and vanilla classes, expected monthly volume for event classes, expected number/ monthly volume for profiles and the expected number of records for extract files. By default, this property is set to 100. For profiles, this property is enabled if Population Estimation Method is set to Use Expected Number. This property is specific to persistent classes and Profiles only. For MCP Applications: The value of this property ranges from 1 to 545,755,813,887. An exception to this is direct ispec/classes and their profiles, for which the range is 1 to 268,435,455. By default: • This property is set to 100 for <<Ispec>> and vanilla classes. • This property is set to 1000 for <<Event>> classes. ***Caution:*** *If the value specified for the expected number exceeds 268,435,455, you must ensure that sections have been defined for the ispec/class and associated profiles. Refer to the* Agile Business Suite Runtime for Clearpath MCP Adminstration Guide *for more information on how to prepare related classed and profiles. If you do not prepare the ispec and associated profiles before using a large population, syntax errors may occur during compilation of the DASDL.*
Percentage of Expected Number	Specifies the maximum number of database records expected to be created for the Profile as a percentage of the Default Calculation. By default, this property is set to 100. This property is specific to Profiles only. This property is enabled when Population Estimation is set to Specify Percentage of class's Expected Number.

Property	Function
Population Estimation Method	Specifies the method used to specify the maximum number of database records expected to be created for the Profile. The following options are available for this property: • Use Default Calculation – Select this option to use the default calculations for the Expected Number/Monthly Volume. For a Profile owned by an <<Ispec>>, <<Event>>, or vanilla class, the Default Calculation is the Expected Number of the owner class. For a Profile owned by an EventSet and spans more than one <<Event>> class, the Default Calculation is the sum of the Expected numbers of all <<Event>> classes included in the Profile's conditions. • Use Expected Number Method – Select this option for the target builder to use the value in the **Expected Number** property for the profile. • Use Percentage Method – Select this option for the target builder to calculate the population for the profile as a percentage (ProfileUsePercentage) of the Default Calculation By default, this property is set to Use Expected Number. This property is specific to Profiles, ispec class, event class, vanilla class, extract file configuration properties only.
Records Per Block	Specifies the expected number of records for the corresponding extract files. This property's value ranges from 1 to 99999999. By default, this property is set to the default. You should be aware of the minimum and maximum physical block sizes for the device being used, and ensure that your specified block/record size falls within those limits. This property is specific to an Attribute that inherits GLB.File (extract files).
User Maintained Table	Specifies whether a database table is reorganized by the user or by system during deployment. By default, the value is False. Applies to persistent classes.
View Alternate Names	Displays an alternative name of a report. You can use the **View Alternate Names** dialog box to view information about reports. Click the ellipsis button (…) to access the **View Alternate Names** dialog box. This dialog box includes a read-only list of data in the following columns. • Folder Name: Displays the folder in which the report exists. • Default Name: Specifies the report configuration Alternate Name, if defined. If this value is not defined, the default name value contains the Alias for an MCP configuration or Model name for a Windows® configuration. • Generate As: Displays the generated as value of a report.

Remote Database

Property	Function
ACK Rate	Specifies the number of blocks of audit files to be sent between your primary and secondary databases, before your application checks for an acknowledgment. This property enabled when • Configure Set Type is set to RDB. • File Transmission Mode is set to Audit block write. By default, this property is set to 0. The numbers range from 0 to 99. This property is specific to Segments only.
Delay AuditFile Removal	Specifies whether to delay the audit files being removed before RDB has finished sending them. This property is enabled when Configure Set Type is set to RDB. By default, this property is set to False. This property is specific to Segments only.
Dump Tape Cycle	Specifies the cycle number for the dump tape. This property is only enabled when • Configure Set Type is set to RDB. • Dump Medium is set to Tape. By default, this property is set to 0. The numbers range from 0 to 999999. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*
Dump Tape Density	Specifies the density of the dump tape. This property is only enabled when • Configure Set Type is set to RDB. • Dump Medium is set to Tape. By default, this property is set to 0. The numbers range from 0 to 999999. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*
Dump Name	Specifies the name of the dump file. This property is only enabled if Configure Set Type is set to RDB. By default, this property is empty. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*

Property	Function
Dump Medium	Specifies the medium on which the dump file is kept. The following options are available for this property: • Pack – Indicates that the dump file is on pack • Tape – Indicates that the dump file is on Tape By default, this property is set to Tape. This property is enabled if Configure Set Type is set to RDB. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*
Dump Pack	Specifies the name of the pack if your dump file is on pack. You can either enter the name or select it from the list. This property is limited to a maximum length of 17 alphanumeric characters. This property is only enabled when • Configure Set Type is set to RDB. • Dump Medium is set to Pack. By default, this property is empty. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*
Dump Tape Serial Number	Specifies the serial number of the dump tape. This property is only enabled when • Configure Set Type is set to RDB. • Dump Medium is set to Tape. By default, this property is set to 0. The numbers range from 0 to 999999. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*
Dump Tape Version	Specifies the version number for the dump tape. This property is only enabled when • Configure Set Type is set to RDB. • Dump Medium is set to Tape. By default, this property is set to 0. The numbers range from 0 to 999999. This property is specific to Segments only. ***Note:*** *You can specify up to three sets of dump details.*

Property	Function
Error Handling	Specifies the method of communication error handling. The following options are available for this property: • Connection Dropped • Operator Intervention Refer to the *Remote Database Backup, Planning and Operations Guide* for more information on these options. This property is enabled when: • Configure Set Type is set to RDB. • File Transmission Mode is set to Audit block write. This property enabled if the File Transmission Mode property is set to Audit block write. By default, this property is set to Connection dropped. This property is specific to Segments only.
File Transfer Option	Specifies the file transfer rate. The following options are available for this property: • FTRapid – transfers files at a faster rate. • Native – transfer files at a standard rate. By default, this property is set to Native. This property is enabled when: • Configure Set Type is set to RDB. • File Transmission Mode is set to Audit block write. This property is specific to Segments only.
File Transmission Mode	Specifies the mode of communication between your primary and secondary databases. This property is enabled when Configure Set Type is set to RDB. The following options are available for this property: • Audit block write • File Switch • ServerCapable • Not ServerCapable Refer to the *Remote Database Backup, Planning and Operations Guide* for more information on these options. By default, this property is set to Audit block write. This property is specific to Segments only.
Number of Workers	Specifies the number of dump workers that you want to use, if your primary database is to be dumped to tape or pack. The number of dump workers is the number of reels dumped in parallel. This property is enabled when Configure Set Type is set to RDB. By default, this property is set to 1. The numbers range from 0 to 99. This property is specific to Folders only.

Property	Function
Pack Mappings	Specifies the mapping of the Packs for the selected Configuration to the equivalent Packs for the remote database on the secondary host. For example, PACK1=PACK2, PACK3=PACK4, etc. By default, this property is empty. This property is specific to Segments only.
Port Timeout From Primary	Specifies the time, in seconds, for which the primary application waits for a response from the secondary database before timing out. This property is enabled when Configure Set Type is set to RDB. By default, this property is set to 0. The numbers range from 0 to 9999. This property cannot be set to 0, when: • Error Handling property is set to Connection Dropped • File Transmission Mode is set to Audit Block Write • Port Timeout From Secondary property is set to zero. This property is specific to Segments only.
Port Timeout From Secondary	Specifies the time, in seconds, for which the secondary application waits for a response from the primary database before timing out. This property is enabled when Configure Set Type is set to RDB. By default, this property is set to 60. The numbers range from 0 to 9999. This property cannot be set to 0, when: • Error Handling property is set to Connection Dropped • File Transmission Mode is set to Audit Block Write • Port Timeout From Primary property is set to zero. This property is specific to Segments only.
Synch Restart Interval	Specifies the interval between when the System detects that a Catchup condition has occurred (when your Secondary system has fallen behind the processing of your Primary system) and when it initiates the Catchup process. This property is enabled when Configure Set Type is set to RDB. By default, this property is set to 0. The numbers range from 0 to 999. This property is specific to Segments only.

Runtime Options

Property	Function
Active Month Number	Your system uses Active Month Number and Expected Number to reserve space for Event data. Active Month Number specifies the number of months. Events are stored before they may be removed. ***Note:*** *It is the user's responsibility to remove the old Event transactions, not Agile Business Suite. This property is used to specify how much space to reserve.* By default, this property is set to 1. The numbers range from 1 to 99. This property is specific to Segments only.
Binary	Specifies the Binary options for numeric. The following options are available for this property: Binary Coded Decimal, Binary, or None. By default, this property is set to None. This property is specific to Attributes painted in a report frame only. This property is disabled if Default Device is set to Enterprise Output Manager (EOM) Generated Reports.
DataReader Capable	Specifies whether a runtime system is using the Datareader SQL Server feature to retrieve data through the Determine commands. This feature modifies the process of data retrieval and subsequently optimizes the performance of AB Suite runtime. Refer to the *Agile Business Suite Runtime for Windows® Operating System Administration Guide* for more information about the Datareader feature. ***Note:*** *Most of the AB Suite runtime systems have a performance improvement by using the Datareader feature. However, the Datareader feature can be inefficient for some runtime systems and this property should be set to false for those systems.* By default, this property is set to True. This property is specific to Segments only.
Def Bounds Checking	Specifies whether a bound checking is set using the MOVE logic command or assignment operation. By default, this property is set to False. This property is specific to Segments only.
Default Translation	Specifies the translation used for the System GUI at runtime. If this property is empty, the Session language is used. By default, this property is empty. This property is specific to Segments and Reports only.
Extend Report Recovery	Specifies whether failed Reports store recovery information. Failed Reports can be rerun once offending problems have been fixed. By default, this property is set to False. This property is specific to Segments only.

Property	Function
Extract File Location	Specifies the full path name of the Extract File. By default, this property is empty. This property is specific to Classes and Attributes that inherit from Glb.File only.
Extract File Name	Specifies the name of the Extract File. By default, this property is set to the name of the extract file object. This property is specific to Classes and Attributes that inherit from Glb.File only.
Fireup Ispec	Specifies the name of the <<ispec>> class that automatically displays when the deployed system is run. By default, this property is empty. This property is specific to Segments only.
Generate Optimized Initialization Code	Specifies generating an optimized code for performing initialization of the entire primitive and group attributes that are defined as members of a Segment. Normally all the primitive and group attributes in a Segment are initialized as part of every ispec transaction. If this property is set to True, then additional code is generated to perform this initialization in an optimized manner that initializes only those attributes that are used in each ispec transaction. If your system has a large number of attributes as members of the Segment class, such as greater than 5,000, the optimized initialization code may provide a noticeable performance improvement. Always remember that additional time is taken during the build process to generate the additional initialization code, during the generation of the <segment>.cs file. So, this property provides a tradeoff between build time and runtime performance. ***Note:*** *This setting is subjective and some runtime systems might experience a performance benefit. If you have more than 10,000 primitive and group attributes defined as members of the Segment class, it is recommended to set this property.* By default, this property is set to False. This property is specific to Segments only.
Glb.Param Size	Specifies the size of System Attribute Glb.Param. By default, the size is 2,000. The maximum size of the System Attribute Glb.Param is 262,000. This property is specific to Segments only.
Global Work Size	Specifies the size of the global work buffer. By default, this property is set to 128. The numbers range from 128 to 4095. This property is specific to Segments only.
Has External Interface	Specifies whether the selected class is represented by an external interface. For example, Web Service, USER, OFFLINE, or GLI. By default, this property is set to False. This property is specific to Ispecs, Events, Copy Ispecs and Copy Events only.

Property	Function
High Frequency	Specifies whether faster access is to be provided at runtime. By default, this property is set to False. This property is enabled if Ispec Type is set to Table only. This property is specific to <<Ispec>> and vanilla classes only. ***Note:*** *Excessive use of this facility can result in wasteful use of memory resources.*
Ispec Type	Specifies the database usage by the <<ispec>> or vanilla class. The following options are available for this property: • Standard – Select this option for normal database usage. • Table – Select this option for high frequency access. • Direct – Select this option to create a direct dataset in the DASDL (you must have defined only one key in the class: Primitive = number and length < 12). By default, this property is set to Standard. It is recommended that you keep the same setting across all MCP configurations. This property is specific to<<Ispec>> and Vanilla classes only
KanjiSpace Conversion	Determines whether the Kanji space-ASCII space conversions are done while reading and writing extract files. When Kanji Space property is set to TRUE, one Kanji space is converted to two ASCII spaces while reading and writing extract files. By default, the configuration property KanjiSpaceConversion is not visible. To make this property visible on Segment Property pages, set the NationalString property on the Segment to MultiByte and Validation to Kanji. By default, the configuration property KanjiSpaceConversion is set to False. Set this property to True. ***Note:*** *KanjiSpaceConversion configuration property is not available for MCP Platform.*
Line Spacing	Specifies the line spacing. The following options are available for this property: Single, Double, or Triple. By default, this property is set to Single. This property is disabled when device is set to Enterprise Output Manager (EOM) Generated Reports or DI. This property is specific to Reports only.
Log Activities	Specifies whether to write details of non-transaction activities (such as signing on and off) to a log file. By default, this property is set to False. This property must be set to True if Log Transactions is set to True. This property is specific to Segments only.

Property	Function
Log File Location	Specifies the directory that contains the runtime log file. Type a valid DOS path. For example, C:\my folder By default, this property is empty. This property is specific to Segments only.
Log File Size	Specifies the size of the log file. By default, this property is set to 4000. The numbers range from 1000 to 999999. This property is enabled when Log Activities is set to True. Once you set the logging, the Log File Size defaults to 4000 or as specified or changed by the user. If Log File Size is less than 1000, then, it defaults to 1000 minimum. This property is specific to Segments only.
Log Transactions	Specifies whether to write details of transaction activities to a log file. By default, this property is set to False. This property must be set to True if Log Activities is set to True. This property is specific to Segments only.
Preserve Session Data	Use this property to control the physical I/0 overhead associated with saving and retrieving session data records. It specifies whether to retain data unique to a user across sessions or when the System terminates. By default, this property is set to False. When this property is set to True, you must take care when using the System Attributes: • TRANNO If this property is set to False, the overheads incurred by retaining this information in the database are removed. This property is specific to Segments only.
Primary GSDs/DADs FTU Translatable	Specifies whether primitive segment attributes and attribute captions (DADs) can be translated through the Forms Translation Utility (FTU) only. By default, this property is set to True. This property is specific to Segments only.

Property	Function
Report Messages Sent to Client	Specifies whether the messages generated in Reports are sent to the client who started them. ***Note:*** *This refers specifically to reports that are initiated from logic via the RUN LDL+ command.* Reports produce a number of messages – including standard messages like BOJ/EOJ messages and all user-defined messages resulting from Message commands in the logic. If Report Messages Sent to Client is set to All Messages then these messages are displayed back to the user session that started the report (they display on the status line or the Console window pop-ups to display the messages). If Report Messages Sent to Client is set to No Messages then the user does not see any messages from the running report. This property is valid on Windows® Configurations only.
Reset Report Count When System Is Generated	Specifies whether to reset the incrementing version number for Glb.RepVersion to 1, when the system is built. By default, this property is set to False. This property is specific to Segments only.
Reset Version Number If Identifier Is Changed	Specifies whether to reset the incrementing version number for Glb.SysVersion to 1, when the Runtime Version Identifier field is changed. By default, this property is set to False. This property is specific to Segments only.
ROC Output Location	Specifies the location that contains Report Output Control (ROC) files. Type a valid Location name or select a valid name from the list. Refer to Locations in the *Agile Business Suite Developer Online Help* for more information on locations. By default, this property is empty. This property is specific to Segments only.
ROC Uses Database	Specifies whether ROC output is kept in files or in the database By default, this property is set to False for Segments and it assumes the value of owner for Reports. This property is specific to Segments and Reports only.
Runtime Behavior Of Unavailable Commands	Specifies what occurs at runtime when unavailable commands are executed. By default, this property is set to Warning Message. This property is specific to Segments only.

Property	Function
Runtime Version Identifier	Specifies the Runtime Version on the host, and provides the value for System Attribute Glb.VersionId. For every value of Glb.VersionId, a corresponding version number for Glb.SysVersion exists which increments with every Build. Type an identifier to a maximum of 20 characters. By default, this property is set to 1. This property is specific to Segments only.
Standard Heading	Specifies whether the standard heading is to be printed on each page of Report output. The standard heading consists of the Report name, a page number, the time at which the Report was run, and the time of its last compilation By default, this property is set to True. This property is disabled when device is set to Depcon Generated Reports or DI. This property is specific to Reports only.
System Uses ROC	Specifies whether the Report Output Control (ROC) facility must manage the output from all Reports. By default, this property is set to True. This property is specific to Segments only.
Two Phase Commit	Specifies whether the two phase commit process is used for updating databases. This occurs with external Automatic Entries. With two phase commit, the sending and receiving systems are able to roll back to a synchronized point if both have not successfully completed their related transactions. By default, this property is set to False. This property is specific to Segments and Reports only.
Use Data Invocation	Specifies whether to set data invocation. Data invocation enables the values of Attributes of a Profile to be accessed without accessing the Ispec record. This can reduce the number of database accesses if used appropriately. If it is used inappropriately, database access is increased. By default, this property is set to False. This property is specific to Segments only.
Use No-Domain Usercode As Station Name	Specifies the option to allow the domain name as part of the session identification in GLB.STATION. If set to False, then GLB.STATION (and GLB.STN) contains the full usercode including the domain prefix. If set to True, then GLB.STATION (and GLB.STN) contains just the usercode without the domain prefix. This property is valid on Windows® Configurations only.

Property	Function
User Privilege	Specifies the user terminal access to an <<Ispec>> class at runtime. Each terminal has an assigned security level which is defined as Integer3 Installation Data record in the COMS Configuration File in the ClearPath MCP-based COMS utility. Refer to the *Agile Business Suite for ClearPath MCP Administration Guide* for more information. In Ispec logic at runtime, the privilege level of the terminal currently accessing the System is contained in the System Attribute GLB.PRIV. By default, the value is 1. The privilege ranges 1 from 15. This property is specific to <<Ispec>> and <<Event>> classes only.
Video Capable	Specifies whether to direct a report's output to a video device. By default, this property is set to False. This property is disabled when device is set to Enterprise Output Manager (EOM) Generated Reports or DI. This property is specific to Reports only.

Runtime Transfer Utility

Property	Function
Runtime Transfer Utility File Name	Specifies the name of the Runtime Transfer Utility (RTU) file. By default, this property is empty. This property is enabled if Generate Runtime Transfer Utility File is set to True. This property is specific to Folders only.
Location to FTP the RTU File To	Specifies where to FTP RTU file on the host. By default, this property is empty. This property is enabled if Generate Runtime Transfer Utility File is set to True. This property is specific to Folders only.
RTU Configure Set	Specifies the name of the Configure Set for the RTU file. By default, this property is empty. This property is enabled if Generate Runtime Transfer Utility File is set to True. This property is specific to Folders only.
RTU RDB Configure Set	Specifies the name of the RDB Configure Set for the RTU file. By default, this property is empty. This property is enabled if Generate Runtime Transfer Utility File is set to True. This property is specific to Folders only.

Subsystem

Property	Function
Subsystem	Specifies a Subsystem for the <<Ispec>> or <<Event>> class. ***Note:*** *Subsystems only apply to systems generated on ClearPath MCP-based hosts. On MCP-based runtime hosts, <<Ispec>> and <<Event>> classes are grouped into subsystems. For each subsystem, there is a corresponding COMSTP program with the file name and the program name the same as the corresponding subsystem.* By default, this property is empty. This property is specific to <<ispec>> and <<Event>> Classes only.
Subsystems Definition	Defines a list of subsystems for the Segment. You can use the Subsystems Dialog Box to add, modify, or delete Subsystems. Click **browse** to access the dialog box. This property is specific to Segment only.

Subsystem Dialog Box

Use the Subsystem dialog box to add, update, or delete a Subsystem.

Note: *Subsystems only apply to systems generated on ClearPath MCP-based hosts. Ispecs are grouped into subsystems.*

For each subsystem, there is a corresponding COMSTP program with the file name system/COMS_LINC_TP and the program name the same as the corresponding subsystem.

To display this dialog box, click the Browse button in the **Subsystems Definition** property under the **Subsystem** category for Segments.

By default, the Subsystems PRIMARY, QUERY, ROC, and LSS are created when a Segment is first added to the Model. You can add Ispecs to a subsystem by specifying the Subsystem property for each Component and Event. The Ispec you enter as the Fire.up Ispec should be in the PRIMARY subsystem.

The default Subsystems are numbered in the following way:

- Subsystem number 1 is always PRIMARY.
- QUERY and ROC initially derive values from PRIMARY and are given the subsystem number (1).
- LSS always has the subsystem number LSS.

The following table illustrates the relationship between Subsystem name and Subsystem number:

Subsystem Name	Subsystem Number
PRIMARY	1
User-defined	2-10
Query	A
ROC	C
LSS	LSS

- The Subsystem dialog box displays the properties listed in the following table in a list box:

Property	Function
Number	Specifies the unique Subsystem number. A maximum of nine subsystems may be defined by the user. These can be numbered from 2 to 10. Subsystem numbers 1, A, C, and LSS are reserved and cannot be allocated to any other Subsystem. Subsystem 1 is called PRIMARY, A and C can be QUERY or ROC. When defining a new Subsystem, only unused Subsystem numbers are displayed in the drop-down list. You can derive values for QUERY and ROC from other Subsystems. When defining values for QUERY and ROC all user-defined subsystem numbers are displayed plus 1, A, and C are also displayed if they have not been used. This property is limited to a maximum length of 10 alphanumeric characters. The first character cannot be a numeric or '-' characters. By default, this property is empty. This property is specific to Segments only.
Name	Specifies the name of the Subsystem. The Subsystem name, GLOBAL, is reserved and cannot be used. The names PRIMARY, QUERY, ROC, and LSS cannot be deleted or changed. This property allows a maximum length of 10 alphanumeric characters. The first character cannot be a numeric or '-' characters. By default, this property is empty. This property is specific to Segments only.
Priority	Specifies the priority at which a COMSTP program of a Subsystem runs. By default, this property is set to 50 for the predefined Subsystems: PRIMARY, QUERY, ROC, and LSS. The numbers range from 1 to 99. This property is specific to Segments only.

Property	Function
Min Copies	Specifies the minimum number of copies of the COMSTP program of a Subsystem to start when the System is running. By default, this property is set to 0. The numbers range from 0 to 255. A value of zero means that the COMSTP program for that Subsystem is not started until one of its Ispecs is invoked. This property is specific to Folders only. ***Note:*** *This field is not applicable to the LSS Subsystem.*
Max Copies	Specifies the maximum number of copies of each COMSTP program of a Subsystem that can be running. By default, this property is set to 1; except for the PRIMARY Subsystem which has a value of two. The numbers range from 0 to 255. This property is specific to Folders only. ***Note:*** *This field is not applicable to the LSS Subsystem.*
Idle Timeout	Specifies an idle timeout period for an individual COMSTP program. The value entered in this property represents the elapsed time before an idle COMSTP program or Subsystem is terminated. By default, this property is set to 0. The numbers range from 0 to 99. You can enter 99 for no timeout. A value of zero means that the maximum timeout period of 60 minutes is used. Enter a time in minutes of 0 through 60, or The LSS Subsystem timeout is used for the ClearPath MCP-based :TIM command. Enter a value of 0 through 98, or enter 99 for no timeout. A value of zero keeps the current: TIM value. ***Note:*** *If you use the :TIM system command to provide a system-wide timeout, this takes effect only after the COMSTP program have closed down.* This property is specific to Segments only.

To Add a Subsystem, perform the following:

1. Select an empty row on the Listview, right-click and Select **Add** from the context-menu. A Subsystem with the next available number and default settings and name is added to the list.
2. Click each property and make the appropriate entries, if needed.

To Delete a Subsystem, perform the following:

Deleting a user-defined Subsystem causes its Ispecs to be allocated to the PRIMARY Subsystem. The Subsystems PRIMARY, QUERY, ROC, and LSS cannot be deleted.

1. Highlight the subsystem you want to delete from the list.
2. Right-click and select **Delete** from the context-menu. The subsystem is removed from the list.

To Reset the Settings of a Subsystem, perform the following:

1. Highlight the subsystem for which you want to reset the values back to default from the list.
2. Right-click and select **Reset** from the context-menu. The properties of the subsystem is reset back to its defaults.

Winform User Interface

Property	Function
Download URI	Specifies the location where clients can download Winform user interfaces from. By default, this property is empty. This property is enabled when Deployable is set to True, and Deploy Winform User Interface is set to True. This property is specific to Folders only.
Final Location	Specifies the location where Winform items are located. Type a valid Location name or select a valid name from the list. Refer to Locations in the *Agile Business Suite Developer Online Help* for more information on locations. By default, this property is empty. This property is enabled when Deployable is set to True, and Deploy Winform User Interface is set to True. This property is specific to Folders only.

Build Settings

To edit the general settings of the Builder, perform the following:

1. From the **Tools** menu, select **Options**.
2. In the left pane, expand **System Modeler**, and then click **Builder**. Alternatively, enter the keyword Builder in the **Search** box.

The following topics describe properties applicable to each section:

- General
- Component Enabler
- Error Handling
- Logging
- MCP

General

Property	Function
Location of Build Output	Specifies the root location of output files generated by Builder. By default, this property is set to "<tmp directory>\BuilderOutput".
Location of Cache for generated files	Specifies the root location of Builder's cache. In a multi-user development where the model repository is shared between the users, the Cache folder must be shared between the users using a network drive. Map the network drive to the local host. By default, this property is set to "<tmp directory>\BuilderCache".
Number of Build threads	Specifies the number of build threads that runs concurrently. Type a number from 1 to 64. By default, this property is set to 1.
Number of Dependency Threads during the generate phase	This setting determines the number of threads used early in the build while many Builder threads are busy generating files. This value can be zero to leave all dependency logging until Builder has less to do on the client side. This implies that, more memory is required to keep all the dependency information until it is logged.
Number of Dependency Threads after the generate phase	This setting determines the number of threads used later in the build where Builder is doing less work on the client side. This value must be greater than or equal to Number of Dependency Threads during the generate phase.
Rebuild Threshold	This setting controls the automatic rebuild mechanism. The automatic rebuild mechanism optimises the build to ensure it completes as quickly as possible. The Rebuild Threshold drop-down list includes the following options: • Lowest • Low • Medium • High • Highest The higher the setting, the greater the amount of change required to an application before an automatic rebuild is triggered. By default, this option is set to **Medium**. ***Note:*** *This setting should be modified only when build takes significantly longer than rebuild, or an automatic rebuild is being triggered frequently.*

Note: *Dependency Logging Thread count is separate to the Number of Build Threads. Since the "Number of Dependency Threads during the generate phase" setting controls the number of threads that can be working in the generate phase of the build, the "Number of Build threads" setting may need to be adjusted for performance benefits. If a large number of dependency threads are used, depending on the number of CPUs on the build machine, there would be a greater number of total threads running.*

Component Enabler

Property	Function
Class Path	Specifies the directory path for LINCViewer.jar. LINCViewer.jar contains Java classes required by components, and GUI forms. It is omitted if the compiler does not require a specific directory path, or if the .jar file is part of the system class path. By default, this property is empty. ***Note:*** *This property is used when it is referred to in the Compiler String property. If it is not specified in the Compiler String property, the Class Path under the system Class Path is used.*
Compiler String	Specifies the Java compiler. By default, this property is set to javac -d "%2" "%3". Specify the bin directory of the jdk on the system path, or change the default value to specify the full path for the javac.

Error Handling

Property	Function
Stop Build At First Generate Error	Specifies whether generation should stop when an error occurs, or stop once source generation has completed, but before compilation begins. By default, this check box is unchecked.
Stop Build If Generation Warnings Occur	Specifies whether compilation should begin if source generation warnings have occurred. By default, this check box is unchecked.
Error Type for Unsupported Commands – Element Related	Specifies the error handling policy for potential runtime error situations when commands used, are not available in the context. For example, SLEEP in an ispec class. By default, the Warning radio button is selected.
Error Type for Unsupported Commands – Platform Related	Specifies the error handling policy for potential runtime error situations when commands used, are not available for the target platform. By default, the Warning radio button is selected.

Logging

Property	Function
Log Build Debugging Information	Specifies whether build debugging information displays in the Output pane in Visual Studio. By default, the check box is unchecked.
Log Build Information	Specifies whether build information displays in the Output pane in Visual Studio. By default, the check box is checked.

Property	Function
Log Build Warnings	Specifies whether build warnings displays in the Output pane in Visual Studio. By default, the check box is checked.

MCP

Property	Function
Number of AsynchronousFTP threads	Asynchronous processing of MCP targetbuilder FTP tasks increases the performance and reduces the build time. By default, the value of this property is set to 5.

Exporting and Importing Model Elements

Model elements are exported and imported using the Import and Export wizards.

The wizards can be accessed from the **File** menu by selecting either **Export** or **Import**. To access the wizard externally, go to **Start** > **Apps** > **Agile Business Suite 6.1** > **Model Exporter** or **Model Importer**.

Elements exported from Agile Business Suite models are contained in XML interchange files. You can import both XML interchange files and the MDL files created from Enterprise Application Environment.

Note: *Importing parts of an application may create Unresolved Elements in the model.*

Export and Import Wizards

The Export and Import wizards offer a convenient way to export elements from your model, and to import export files into your model.

The wizards can be accessed in the following ways:

- In Developer, on the **File** menu, select either **Export** or **Import**.
- Go to **Start** > **Apps** > **Agile Business Suite 6.1** > **Model Exporter** or **Model Importer**.

The wizards can also be invoked from the command line, refer to Exporting and Importing from the Command Line. Alternatively you can run an entire import or export from the command line without the wizards being displayed.

The following information includes:

- Export Wizard
- Import Wizard
- Exporting and Importing from the Command Line

Export Wizard

The Export wizard gives you several options for exporting elements from your model. You can export:

- The entire model to an export file.
- Individual elements to an export file.
- Those elements that have changed since a selected point in time.

Export Options

- Export Settings
- Advanced Settings
- Logging Settings

Export Settings

By default, the export settings are displayed when the wizard is invoked.

Select **Advanced Settings** to display the advanced export options.

Source Server

Select the name of the server that contains the model to be exported.

By default, when invoked from Developer, the name of the server selected in Class view is displayed.

Source Model

Select the name of the model to be exported.

By default, when invoked from Developer, the name of the model selected in Class view is displayed.

Destination File

Enter or browse for the path of the file into which the model database is to be exported.

Select Elements

Select the elements you want to export. An element tree is displayed once a valid Source Server and Source Model are selected.

Advanced Settings

These options are displayed when you select Advanced Settings on the Export dialog box. Select Export Settings to display the basic export options.

Changes Since

This option allows the user to export the elements which were changed after specified time from the model. The export wizard offers the following four different “Change Since” options:

Last Migration Database Baseline – Select this option to export only the elements that are imported after the last migration database baseline operation.

Last Import – Select this option to export the changes made since the last import was performed.

Last Export – Select this option to export changes made since the last export was performed.

Date – Select this option to export changes made since the specified date.

Logging Setting

These options are displayed when you select Logging Settings on the Export dialog box. Select Export Settings to display the basic export options.

Logging to file settings

Append Log File – Select to specify that the log entries are to be appended to the end of the file rather than replacing the existing contents.

Log File – Enter, or browser for, the name of the log file in which export results are written.

Messages (in output window only)

Errors – Select this check box to view only error messages in the output window.

Warnings – Select this check box to view only warnings in the output window.

Information – Select this check box to view the general logging information in the output window.

Register Public Model Feature

This option is used to register a Public Model Export license provided by Unisys. To register the Public Model Export license, click the **Register Public Model Feature** link. In the Register Public Model Feature dialog box that appears, enter a valid license provided by Unisys, and then click **OK**.

Refer to Using Public Model File in the Documentation Libraries page on the Product Support site for more information about the license.

Note: *Registering the Public Model License allows you to export the Public Model files.*

Import Wizard

The Import wizard enables you to import interchange files created by the Export Wizard.

Note: *The Debug Mode configuration properties are stored in the Visual Studio Solution files and not the database. Hence, these properties cannot be exported.*

Partial Import

Note: *Importing parts of an application may create* ***Unresolved Elements*** *(refer to Unresolved Elements in the System Modeler in the* Visual Studio Online Help*).*

During partial migration, only the elements that are loaded is migrated. However, if an element is referencing another element that is not part of the load, the migration is not successful. Therefore, ensure that you load even the referencing element in partial migration. Refer to Partial Import for more information on performing a partial import.

Version Control during Migration

During migration, the Import wizard creates versions of all elements with the exception of elements that are not Namespaces. For example, folders and diagrams are not namespaces. These excluded elements are versioned as part of their parent.

The Import wizard creates a version file for each element that is versionable in Enterprise Application Developer release 3.3. It sets the Version File property to <SegmentName>\<ElementName>.model to ensure that the name is unique within the model.

Import Options

- Import Settings
- Advanced Import Settings
- Logging Settings
- Addressing Import/Export Issues

This topic includes the issues you might face while Exporting and Importing.

- Addressing Importing Issues

Import Settings

By default, the Import Settings are displayed when the wizard is invoked.

Select **Advanced Settings** to display the advanced import options.

Select **Logging Settings** to display the settings for logging during import.

Destination Server

Enter, or select, the name of the server containing the model into which you want to import the interchange file.

Destination Model

Enter or select the name of the database into which you want to import the interchange file. Select the Create Database option if this is a new database.

Source File

You can import more than one file in a single import. Click the Add button and then either enter the path of, or browse for, the required file to import.

The selected files can be Agile Business Suite Developer interchange files, or control (.BCH) files.

Use the Remove button to delete files from the list.

Filters

None – Select this option if you do not want to use any filters. If this option is selected the end user can import the model and the configurations

Configurations Only – Select this option to import only the configurations and to ignore all other elements in the incoming model file. This option can be selected only when importing a .model file to an existing database.

Options

Create Database – Select this option to create the required database before importing the selected file.

Rollback on Uncoverable Error – Select this option to roll the database back to its state before the import, if an unrecoverable error occurs.

Validate After Import – Select this option to automatically validate logic once the file is imported.

Migration database – Select this option to create a migration database.

The UnderMigration Property

The Migration-Only Database may be opened in System Modeler. The majority of its properties is read-only. One of the exceptions to this is the UnderMigration property, which for a Migration-Only Database has a value of True

The UnderMigration property may be changed to a value of False, which converts the Migration-Only Database into a regular (editable) database. However, this operation is irreversible and should not be done without due consideration. When you change the UnderMigration property to false, a warning message is displayed to advise you of this and request confirmation to proceed.

Once you change the property value to false, the UnderMigration property isno longer displayed in the list of properties in the Properties Window. The other properties displayed in the Properties Window is enabled, indicating the database is no longer read-only.

Restrictions

If an element is deleted in the 3R3 model and it is partially imported, the corresponding element never gets deleted in the AB Suite after the partial import.

Note: *A model file with XML Framework enabled can only be imported into another model that has the same setting. That is, you cannot import a classic model into an XML Framework model or vice-versa. Similarly, you cannot import a Client Framework model into an XML Framework model or vice-versa.*

Advanced Import Settings

These options are displayed when you select Advanced Settings on the Import dialog box. Select Import Settings to access the basic import options.

Specify New Owner

Select or enter the name of the new owner of the imported model elements. The name of the new owner must be a qualified name with a dot separator. For example, where the new owner is salesSegment.cashSaleIspec, the model elements are imported under the cashSaleIspec element of the segment salesSegment. Alternatively, click browse to display the Select Owner Element navigator. Browse to and select the element you want to become the parent of the import and click OK.

Note: *The value entered in the New Owner file is only applicable when importing a .model file.*

On Element Clash

Conflicts occur if an element being imported has the following:

- The same identifier as an element that already exists in the model.
- The same name and is in the same position in both the interchange file and the model.

Select one of the following options to specify how to deal with conflicts that may arise:

Ask – Select this option to display the Conflict Resolution dialog box each time a conflict is encountered. You can use this dialog box to deal with conflicts on a case by case basis.

Replace – When a conflict occurs, this option replaces the existing element with the one from the interchange file. Any element at the same level in the model, that were not in the interchange file are removed. Where, the element being imported is not in the existing model, but has the same name as an existing element within that namespace, the element from the file is renamed and an error is reported.

Skip – Select this option to retain the existing element in the model and ignore the incoming element.

Substitute – Select this option to replace the values of the existing element in the database with those in the interchange file. Any elements at the same level in the model that were not in the interchange file remains. Where the element being imported is not in the existing model, but has the same name as an existing element within that namespace, the elementis renamed and an error is reported.

Override Language Conflict

The Override Language Conflict provides alternative options to deal with language conflicts

- If a complete model is imported, the name and locale values of the language are automatically updated.
- If a partial model is imported, the values are not updated. You can update the language by explicitly selecting the update model option on the Import Wizard window.

Notes:

- *The built-in primary language for a model is defined when the model is created.*
- *By default, the primary language value is Primary.*

Conflicts occur if a model being imported has a conflict in the following:

- Name of the language
- Locale of the language

Select one of the following options to specify how to deal with conflicts that may arise:

Ask – Select this option to display the Override Language Conflict dialog box each time a language conflict is encountered. You can use this dialog box to deal with conflicts on a case by case basis.

Update Model – Select this option to overwrite the existing language with the incoming language when a language conflict is encountered.

Keep Existing – Select this option to ignore the incoming language and retain the existing language in the Model when a language is encountered.

Abort – Select this option to abort the import when a language conflict is encountered.

Additional Options

Create New Identifiers – In the model, each element has a globally unique identifier. Selecting this option gives each of the elements being imported a new globally unique identifier. The import maintains the structure defined in the model file.

Attempt Import in Exclusive Mode – Select this option to attempt to enter exclusive mode in the database. If successful, no other user can access the database and the database remains locked until the import is complete. This can improve the performance of the import.

The switch to exclusive mode fails if any other user is logged into the database. A message is displayed and you can select to continue or cancel the import.

Version Controlled

The Version Controlled option applies only to LCIF files not for AB Suite .model files.

Migration Granularity

Elements in AB Suite are able to be versioned individually or collectively in a single file. LCIFImport allows for the granularity of the version control files generated to be specified. The two options available are:

Components – All Non-Primitive objects is set into their own separate version control file.

Segment Members – All elements owned by with the Model, or the Segment, (plus profiles) is set into their own separate version control file.

Baseline Migration Database

This option allows you to baseline the Migration Database prior to the import of partial EAE extract file(s) such that doing an export using Changes Since Last Migration Database Baseline would only export elements changed by that partial import.

This option would be available for selection only if the file being imported is a partial EAE extract file and the import is being carried into a pre-existing migration database.

The Migration Database is also base lined prior to the import of a full EAE extract into a migration database, which has the segment being imported already present. Doing an export using the Changes Since Last Migration Database Baseline option would only export elements changed by that import. This does not require an explicit selection of the option to Baseline Migration Database.

Automatically Enlarge Small Controls

EAE allowed users to create controls that are smaller than the minimum size for AB Suite. Controls that are too small may not display correctly in AB Suite Runtime. The AB Suite Import provides a new option Automatically Enlarge Small Controls in Advanced Setting Page of the Importer Main Form.

This option allows you to specify whether small controls should be resized automatically. This option is applicable to import of an EAE MDL Files. By default, this option is not selected. If this option is not selected, the controls could imported with the size as specified in the LCIF Extracted File. Controls that are too small may not display correctly in AB Suite.

If this option is selected, the small controls is enlarged automatically to AB Suite minimum size. The enlarged controls may overlap to surrounding controls.

Migrate Display Item Sizes As Seen In

EAE Developer synchronizes the sizes of display items to fit the desired text thereby taking into account the font properties. It is however possible to create display items that are smaller in appearance than the required size. This situation arises when the display item is getting its font from the segment. If the font on the segment is changed, EAE does not automatically update the size of all the labels that rely on this font. If the EAE segment is extracted after such a change, the .mdl file contains incorrect sizes for display items.

Until now, AB Suite migrator was importing display items with the size specified in the .mdl file. To address the issues with labels being out of sync, AB Suite Import Wizard has the extra option to import display items with the size defined in the .mdl or calculate a new size.

To choose between EAE Developer and EAE MDL File, the AB Suite Import provides an option in advanced settings of Importer Main Form:

- EAE Developer – Select this option to allow the AB Suite Developer to import the sizes of the display item as seen in EAE Developer. The sizes of the display items are calculated based on its Font metrics (that is Font name, Font size) and text of the display item.
- EAE MDL File – Select this option to allow the AB Suite developer to import the sizes of the display items as they are defined in LCIF file. By default, the AB Suite importer imports the sizes of the display items as seen in LCIF Extracted File.

Configurations To Include

All the configuration related options are applicable only when importing a .model file.

All – Select this option to create configurations in the incoming model file, if they do not exist in the database. It is applicable only when importing a .model file.

Existing – This option is set as default to import the configurations existing in the database.

None – Select this option to ignore all the configurations in the incoming .model file.

Import Only Configurations

Select this option to import only configurations and ignore all other elements in the incoming model file. It is applicable only when importing a .model file. By default, configurations are just updated if they already exist in the database during import.

LDL+ Migration Settings

Previously, during migration, the StoreOrSend() method was added to all the ispecs. This method contained logic to test the value of system Attributes like GLB.DESTINATION, GLB.DESTHOST, and so on, and then perform a Store() or a Send() on the ispec based on these values. Correspondingly, commands like AUTO; WRITE and AUTO; WRITE&CLEAR were migrated as a call to the method StoreOrSend().

This could result in unnecessary performance overhead for systems that do not use the external HUB feature. To handle this, an LDL+ migration option has been provided within the Import utility to control this behavior (in Import, select **Advanced Settings** and click **LDL+ Migration Settings**).

The options available are:

- Default
- As Ispec.Send
- As Ispec.Store
- File Specifying ispecs to be migrated as Store() or Send()

Default – Select this option to migrate all AUTO; WRITE (and AUTO; WRITE&CLEAR) commands as Ispec.Store() or Ispec.Send() or Ispec.StoreOrSend() depending on the ispec properties (for example, IsExternal, HasPresentation).

As Ispec.Send() – Select this option to migrate all AUTO; WRITE (and AUTO; WRITE&CLEAR) commands as a simple call to Send().

As Ispec.Store() – Select this option to migrate all AUTO; WRITE (and AUTO; WRITE&CLEAR) commands as a simple call to Store() rather than the more complex logic to handle the different behavior for Store() and Send(). This has the dual benefit of the logic being migrated as simpler code (easier to read) and it performs better.

File Specifying ispecs to be migrated as Store() or Send() – Select this option to further fine grain control over the migration at an individual ispec level. Here the user can select a text file that contains a separate list of ispecs that have to be migrated as Store() and ispecs that have to be migrated as Send(). For example:

Consider a model containing six ispecs – ISPC1, ISPC2, ISPC3, ISPC4, ISPC5 and ISPC6. We want to control the migration such that AUTO.WRITE; for ISPC4 and ISPC3 needs to migrate as ISPC4.Store() and ISPC3.Store() respectively. Similarly we want to migrate AUTO.WRITE; for ISPC1 and ISPC5 as ISPC1.Send() and ISPC3.Send(). Hence, the file format would be:

[Ispecs As Store]

:List of ispecs for which AUTO.WRITE; (and AUTO.WRITE&CLEAR) are to be migrated as Ispec.Store()

ISPC4

ISPC3

[Ispecs As Send]

:List of ispecs for which AUTO.WRITE; (and AUTO.WRITE&CLEAR) are to be migrated as Ispec.Send()

ISPC1

ISPC5

The remaining ispecs in the system (ISPC2 and ISPC6) is migrated with the default behavior.

Conflict Resolution

This dialog box is displayed if you select the **Ask** option from the advanced import settings. It enables you to deal with conflicts on an individual basis and to change your conflict resolution strategy for the remainder of the import process.

Different options are available depending on what sort of conflict occurs, either a name or identifier conflict.

Existing Item

This group enables you choose an option that applies to the existing element in the model:

- Change the name of the existing element.
- Delete the existing element and import the element from interchange file.
- Substitute the values in the database with the values in the interchange file.

Incoming Item

This group enables you to choose an option that applies to the element in the interchange file:

- Change the name of the incoming element.
- Use the Delete button to retain the existing element and ignore the incoming element.

On Clash

Use these options to change your conflict resolution strategy for the remainder of the import. You can select:

- Replace
- Substitute
- Skip

Language Conflict

This dialog box is displayed if you select the Ask option for Override Language Conflict from the Advanced Import Settings. It enables you to deal with language conflicts on an individual basis.

The options available on the Language Conflict Dialog are:

Update Model – Select this option to overwrite the existing language with the incoming language when a language conflict is encountered.

Keep Existing – Select this option to ignore the incoming language and retain the existing language in the Model when a language conflict is encountered.

Abort – Select this option to abort the import when a language conflict is encountered.

Logging Settings

These options are displayed when you select Logging Settings on the Import dialog box. Select Import Settings to access the basic import options.

Logging to file settings

Append Log File – Select this option to specify that the log entries are to be appended to the end of the file rather than replacing the existing contents.

Log File – Enter, or browse for, the name of the log file in which import results are to be written. By default, the log file name is <ModelFilename>.log and is created in the same directory as the model file.

Messages

Select from the following options the type of messages to be displayed in the output window during the import. All messages are recorded in the log file regardless of this setting.

- Errors
- Warnings
- Information

Addressing Import/Export Issues

This topic includes the issues you might face while Exporting and Importing.

Addressing Importing Issues

While importing a model from other environments to Agile Business Suite, you may encounter some issues and behavioral differences. You must follow the guidelines given below for some of the importing issues:

Logically Deleted Records and the LookUp Command

- When executed within an Ispec, LOOKUP; command ignored logically deleted records (that is, component records that had been deleted, so that MAINT was set to 'D').
- When executed within a Report, LOOKUP; command read logically deleted records.

In Agile Business Suite, to make the behavior of the LOOKUP; command more uniform and consistent, the LOOKUP; command always retrieves logically deleted records.

To maintain the existing behavior in migrated systems, some additional LDL+ logic is included in LOOKUP; commands coded within insertables and segment methods and ispecs. This logic causes logically deleted records to be rejected or skipped by LOOKUP; commands.

There are two situations where the LOOKUP; command can be used.

1. Looping forms of LOOKUP; command coded within ispecs, insertables, and segment methods read logically deleted records into memory, but the added logic results in them being skipped or ignored. However, if a looping form of LOOKUP; command reads only logically deleted records, Glb.Status is set to spaces because at least one record has been read into memory.
2. The additional logic to skip logically deleted records is added to LOOKUP; commands within ispecs, insertables, and segment methods. When a Report calls an insertable or segment method that performs a Lookup, the logically deleted records are skipped (due to the included logic).

Auto Entry Ispec and its instance

During migration of an Auto Entry Ispec from EAE, the import process creates an instance of the Ispec with the name _AutoIspec appended to Ispec name. This instance is then placed under the folder named AutosForIspecs. For example, if CUST is an Auto Entry Ispec in your model, the instance Cust_AutoIspec is created and placed under AutosForIspecs folder after migration.

Addressing Export Issues

Auto Entry Ispec and its Instance

- If you export only the Auto Entry Ispec without selecting its instance and import the resultant model file to a new model database, then only the Auto Entry Ispec is available in the new model database.

  Note: *If CUST is an Auto Entry Ispec in your model, the instance Cust_AutoIspec is created and placed under AutosForIspecs folder.*

- If you export only the instance of the Auto Entry Ispec and then import the resultant model file to a new model database, the instance is available in the new model database. Since the Auto Entry Ispec is referred by its instance but is not present in the model file and the database, so an unresolved element under the Segment with a green question mark is created.

  Note: *If both the Auto Entry Ispec and its instance are exported and imported to a new model database, then these elements are added correctly under their respective owner.*

Exporting and Importing from the Command Line

In addition to the Export and Import wizards, you can also access the import and export facilities from the command line.

The command line options give access to all the same functionality as is available from the Import and Export wizards.

- Exporting from the Command Line
- Importing from the Command Line

Exporting from the Command Line

You can export elements from the model using the following syntax:

```
Export.exe [-q] [-s DatabaseInstance] -m ModelName [-a] [-l LogFileName] -f Filename
ElementName [-nc] [-nu] [-nd] ElementName2 [-nc] [-nu] [-nd] ElementName3 ...
```

You can export the changed elements from the model using the following syntax:

```
Export.exe {-se|-si|-sd “DATE”} [-q] [-s DatabaseInstance] –m ModelName [–a] [–l
LogFileName] –f Filename ElementName [-nc] [-nu] [-nd] ElementName2 [-nc] [-nu] [-nd]
Elementname3 ...
```

Where:

Syntax	Description
[ ]	Denotes an optional argument.
-a or /a	Denotes an optional argument. Specifies that the log entries are to be appended to the end of the log file rather than replacing the existing contents.
-b BatchFileName or */b BatchFileName*	Specifies the full path of the export batch file. Specifies the qualified names of the elements to be exported in the batch file. ***Note:*** *Specify flag #BATCH_EXPORT as the first line and only one qualified element name at each line in the export batch file.*
ElementName	Specifies the qualified names of the elements to be exported from the model.

Syntax	Description
-f Filename or */f FileName*	Specifies the full path of the export file.
-l *LogFileName* or /l *LogFileName*	Specifies a different log file name to use rather than the default.
-m *ModelName* or /m *ModelName*	Specifies the name of the model within the database instance. This is a required parameter.
-nc or /nc	Specifies that the children of the element should not be exported.
-nd or /nd	Specifies that the documentation of the element should not be exported.
-nu or /nu	Specifies that the UML diagrams of the element should not be exported.
-q or /q	Specifies that the export should be quiet, that is, the progress dialog box is not displayed. If running in quiet mode, all required parameters must be given. The export exits with an error if all required parameters are not defined.
-s *DatabaseInstance* or /s *DatabaseInstance*	Specifies the SQL Server database instance. If this parameter is omitted, the file is exported from the local machine.
-sd "DATE" or /sd "DATE"	Specifies that changes made since a specified date are to be exported. The format for the date is mm/dd/yyyy hh:mm:ss.
-se or /se	Specifies that changes made since the last export are to be exported.
-si or /si	Specifies that changes made since the last import are to be exported.
-smb or /smb	Specifies that elements that have been imported after the Migration Database has been base lined are to be exported.
-vf or /vf	Specifies the version file to be exported. The -vf option is available only on the export command line and should be used with the version control feature.

For example, at the command prompt type:

To write export messages to a log file:

```
Export -s <server name> -m <model name> -f <filename.model> -l <log file path name>
<Segment Name>
```

Where, Export –s localhost –m Meta –f Meta.model –l Meta.log MYSEG

Note: *The log file is created only if error or warning messages are generated. If the export runs to completion without generating any error or warning messages then no log file is created.*

To export those elements which have changed since a specific date:

```
Export -s <server name> -m <model name> -f <filename.model> -sd <date> <Segment Name>
```

Where, Export -s localhost -m Meta -f Meta.model -sd 07/27/2006 MYSEG

Note: *The date string format is determined by the regional settings currently in effect.*

To export those elements which have changed since the last export:

```
Export -s <server name> -m <model name> -f <filename.model> -se <Segment Name>
```

Where, Export -s localhost -m Meta -f Meta.model -se MYSEG

To export those elements which have changed since the last import:

```
Export -s <server name> -m <model name> -f <filename.model> -si <Segment Name>
```

Where, Export -s localhost -m Meta -f Meta.model -si MYSEG

Note: *The –sd, -se and –si options are mutually exclusive.*

The Export window is displayed which shows the progress of the Export action.

Note: *By default, the directory should be /NGEN/Bin/*

To export multiple elements in batch by using export batch file:

```
Export -s <server name> -m <model name> -f <filename.model> -b <filename.bch>
```

Note: *For example, the content of the export batch file (filename.bch) might look like below:*

#BATCH_EXPORT

Segment1.Ispec1

Segment1.Ispec2

Importing from the Command Line

You can invoke the import facility using the following syntax:

```
Import [-cs|-ci|-cr] [-q] [-v] [-x] [-c] [-a] [-l LogFileName] [-r] [-s
DatabaseInstance] -m ModelName FileName1 FileName2 ...
```

Where:

Syntax	Description
[ ]	Denotes an optional argument.
-a or /a	Specifies that the log entries are to be appended to the end of the log file rather than replacing the existing contents.

Syntax	Description
-aw MigrationMethod or /aw MigrationMethod	Select this option to specify how the AUTO; WRITE or AUTO; WRITE&CLEAR commands should be migrated to AB Suite. The parameters available are: • Send – migrates all AUTO;WRITE (and AUTO;WRITE&CLEAR) commands as a simple call to Send(). • Store – migrates all AUTO;WRITE (and AUTO;WRITE&CLEAR) commands as a simple call to Store(). • Default – migrates all AUTO; WRITE (and AUTO; WRITE&CLEAR) commands as Ispec.Store() or Ispec.Send() or Ispec.StoreOrSend() depending on the Ispec properties.
-awf FilePath or /awf FilePath	Select this option to specify a file containing a list of Ispecs to indicate whether the AUTO;WRITE commands should be migrated as Store() or Send() method for each Ispec.
-bmd or /bmd	Specifies that the Migration Database should be base lined prior to the import of the file.
-c	Create new database for partial import.
-c or /c	Specifies that the model database should be created if it is not currently present.
-ci or /ci	Specifies that the skip conflict resolution option is to be used if conflicts occur. Retains the existing element in the model and ignores the incoming element.
-cm	Manual conflict resolution. For every clash found, the user is asked how to proceed.
-cr or /cr	Specifies that the replace conflict resolution option is to be used if conflicts occur. Deletes the existing element in the model and replaces it with the incoming element from the interchange file.
-cs or /cs	Specifies that the substitute conflict resolution option is to be used if conflicts occur. Replaces the values of the existing element in the database with the values in the interchange file.
-esc or /esc	Specifies that automatic resize of small controls is done if controls are too small to display on AB Suite clients correctly.

-eds or /eds	Specifies that sizes of the display items are imported as seen in EAE developer.
	If the option is –eds is given, the sizes of the display items are imported as seen in EAE Developer. For example:
	`c:\import.exe -eds -m testsizes displayItems.mdl`
	If this option is not given. The sizes of the display items are imported as seen in LCIF Extracted File. For example:
	`c:\import.exe -m testsizes displayItems.mdl`

Syntax	Description
-ele or /ele	Select this option to import the colors as they are defined in an LCIF extracted .MDL file.
Filename	Specifies the names of the files to import.
-inc	Specifies that all the configurations in the incoming .model file should be created if they do not exist in the database. You must specify this option to import all the configurations if the incoming .model file is partially exported from another AB Suite database. If this option is not specified, new configurations are imported (only if the incoming .model file is fully exported) from another AB Suite database.
-l *LogFileName* or /l *LogFileName*	Specifies a different log file name to use rather than the default.
-m Meta	Name of AB Suite database (model) for partial import.
-m *ModelName* or /m *ModelName*	Specifies the name of the model within the database instance. This is a required parameter.
-n NewOwner	Specifies that all imported elements should be imported with a new owner. The name of the owner is provided by the <NewOwner> parameter.
-nc	Specifies that all the configurations in the incoming .model file should be ignored.
-oc	Imports only the configurations and ignores all other elements in the incoming .model file. It is applicable only when importing a .model file.
-inc	Specifies that all the configurations in the incoming .model file should be created if they do not exist in the database. You must specify this option to import all the configurations if the incoming .model file is partially exported from another AB Suite database. If this option is not specified, new configurations are imported (only if the incoming .model file is fully exported) from another AB Suite database.
-ni	Generates new unique identifiers for all imported elements.
-oc	Specifies this option to import only configurations and ignore all other elements in the incoming model file. It is applicable only when importing a .model file. By default, configurations are just updated if they already exist in the database during import.
-p	New database is a Migration-Only Database for partial import. ***Note:*** *You must use this option in conjunction with the -c option and only load files with extensions of .mdl or .bch.*
-q or /q	Specifies that the import should be quiet, that is, the progress dialog box is not displayed. the importer automatically exits after an import. If running in quiet mode, all required parameters must be given. The import exits with an error if all required parameters are not defined.
-r or /r	Specifies that if an error prevents completion or the import is cancelled, the model database should be rolled back to the state at which it was when the last file was begun.

Syntax	Description
-s (local)	Database server for partial import.
-s *DatabaseInstance* or /s *DatabaseInstance*	Specifies the SQL Server database instance. If this parameter is omitted, the file is imported on the local machine.
-tca or /tca	Abort the import if the existing language and the language in the .model file are different.
-tci or /tci	Retain the existing language and ignore the language in the .model file.
-tcm or /tcm	Prompt the user for an action when the existing and incoming languages are different.
-tcr or /tcr	Update the existing language with the language in the incoming .model file.
-v	Validate language elements after import for partial import.
-v or /v	Specifies that validation be performed following the import.
-vc or /vc	Indicates that all nonprimitive objects should be set into their own separate version control file.
-ve or /ve	Indicates that all the elements owned by the Model or Business Segment including profiles should be set to their own separate version control files.
-vf or /vf	Specifies the version file to be imported. The -vf option is available only on the import command line and should be used with the version control feature.
-vn or /vn	Specify this option to skip the version control file setting for all the elements owned by the Model or Business Segment. You should not specify this option if you want to use the vc or ve option because they are incompatible.
-x or /x	Specifies that exclusive mode is to be used during the import. Provides increased performance during the import.

Unresolved Elements

Importing small segments of a larger model may introduce unresolved elements into the structure. This can happen when all the following conditions are met:

1. When there exists an element in the model that refers to another, either through inheritance, ownership or in a folder.
2. The first element is imported into the model.
3. The second element does not already exist in the model and is not imported.

This is most likely to occur when using source control where the two elements are in separate version files. Checking out a file imports it into the model. The unimported class is flagged as unresolved during the import and must be resolved before you can build or debug the part of the model containing the unresolved element. An Unresolved tab is displayed in the model that lists all elements that have yet to be resolved.

The IsResolved property is used to indicate that there are elements in the model that refer to elements that do not currently exist in the imported model. This flag is only visible when set to False and an element is unresolved. The import log warns that an imported element refers to another element that isn't in the model or interchange file. Changing the IsResolved property from False to True resolves the element by re-creating it. This should be done with care, as it creates a new Id for the element, so you lose any "relation" to the element is was referencing to. This property can only be changed from False to True. Once resolved, an element cannot be made unresolved again. The best way to resolve a unresolved element is by importing this element

Refer to Resolving Elements for more information.

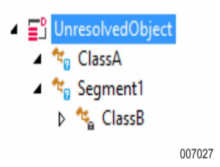
Note: *You are unable to build or debug a part of an application that contains an unresolved element without first resolving the element.*

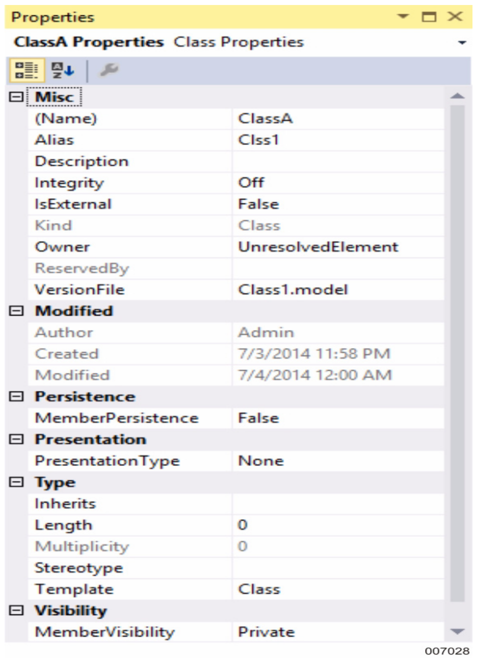
Resolving Elements

There are many ways in which unresolved elements can be created. One of the most common is through inheritance. This simple example describes how such an unresolved element can be created and resolved.

A developer working on a part of an application creates a class, called Class B which inherits from Class A. That is, B has a dependency on A. Class B is in another part of the application and therefore has a different VersionFile from Class A. The developer checks in the classes for version control. A second developer wants to work on Class B and checks it out. During the import of the class a warning is shown that the target for the

'Inherits' property of the Class 'ClassB' is required but does not exist in the Model or in the Interchange File. Class B is displayed in the model showing that it inherits from Class A which is listed as unresolved.





Misc	
(Name)	ClassA
Alias	Clss1
Description	
Integrity	Off
IsExternal	False
Kind	Class
Owner	UnresolvedElement
ReservedBy	
VersionFile	Class1.model
Modified	
Author	Admin
Created	7/3/2014 11:58 PM
Modified	7/4/2014 12:00 AM
Persistence	
MemberPersistence	False
Presentation	
PresentationType	None
Type	
Inherits	
Length	0
Multiplicity	0
Stereotype	
Template	Class
Visibility	
MemberVisibility	Private

The parameters of Class A (including its stereotype) aren't known, so Class B only has those parameters that override Class A's parameters. The second developer has three choices:

- Continue working and leave the class unresolved. The developer won't be able to build or debug the part of the application that contains the unresolved class. However, Class B can be checked back in, and the model may still work correctly (provided that the developer hasn't changed anything relating to Class A and Class B).
- Do a get latest recursive on the model. Provided that Class A is already checked in, the class is resolved.

  Note: *If you know Class A's VersionFile you could do a get latest of Class A only (again, provided it was already checked in). This would also resolve the class.*
- Create the unresolved class by changing class A's IsResolved property to True. The class is created in the local model and assigned the default properties for the class. This new class has the same ID as that in the source code bank. The developer can now build and debug the part of the application that contains the dependent classes, but may get conflicting results because the created class may not have the correct properties defined for it. Class B can still be checked back in, and the model may still work correctly (provided that the developer hasn't changed anything relating to Class A and Class B). Or, the developer can subsequently check out Class A from the source code bank overwriting the created class and restoring the correct properties to Class B.

Migrating System Modeler Database

The Model Migrator is a standalone application that assists in migrating an Agile Business Suite System Modeler database from a previous version to the current version. For example the Model Migrator would be used when upgrading your existing version of Developer to a newer version. Upgrading with the Model Migrator does not perform an Export and Import. This is a different function and should be used where appropriate, refer to the *Agile Business Suite Developer Online Help* for more information.

The Model Migrator is accessed from the Start menu and runs as a Migration Wizard.

The wizard guides you through each step of the migration process. At the conclusion of the migration the wizard displays a summary of whether or not the migration was successful.

You must be a database administrator to run the Migrator, and the database cannot be use by any other application.

When you run the wizard, each of the following property pages are displayed in sequence.

- Settings Page
- Overview Page
- Migration Page
- Summary Page

Settings Page

The Settings page is used to select the database that needs migration, and configure migration conditions. You can select multiple databases if you wish. The following fields should be completed:

Server – Lists the SQL Server visible to the current machine. Specify the name of server containing the database, or databases, to be migrated. The default is the current Model server machine.

Databases – Lists the available databases on the selected server that require migration. Check each database that you wish to migrate.

Back up selected database(s) – Specifies whether to back up the databases before the migration. If you check this box and the migration fails, the database automatically gets restored.

Directory – Specifies the name of the directory in which the database is backed up

Once you have completed the above fields, click Next to continue with the wizard and display the Overview Page. Click Cancel to cancel the migration.

Overview Page

The Overview page lists the selections made on the Settings page. If you are satisfied that the details are correct, click Next to continue to the Migration page, or click Back to return to the Settings page and change your selections.

Migration Page

The Migration page displays the progress of the migration of the selected database migration. You can click Cancel at any time to terminate the migration.

Summary Page

The summary page displays a log of the upgrade activity, and identifies if the upgrade was successful, or if it failed. If you selected multiple databases in the Settings page the log identifies each database upgrade activity separately.

Section 4
Managing Applications

You can manage the lifecycle of an application by using Team Foundation Server (TFS). By using a suite of tools, you can manage different versions of AB Suite elements, Folders, and Dictionaries; develop, build, and test your applications.

TFS provides version control, a build system, and helps you to define and manage test plans for system modeler projects.

Version Management

Source Control is an AB Suite feature that provides source control of versionable AB Suite elements, Folders, and Dictionaries. Source Control stores each version of an element so that you can access and use earlier versions. With Check–in Policies, you can enforce restrictions upon the type of changes before committing a Class, Folder into the source control server. System Modeler supports any source control tool that is supported by Microsoft Visual Studio IDE. The VS IDE supports two types of Source Control Integrations:

- Source Control Plug-in API (MSSCCI API providers) – This provides the basic Source Control functionality similar to Microsoft Visual Source Safe.
- VS Source Control Package – This provides a new, deep-integration with Visual Studio Package. It is suitable for Source Control integration that demands a high level of sophistication and autonomy in its Source Control model, similar to Team Foundation Server (TFS). Refer to Getting Started with Team Foundation Server in the *Visual Studio Online Help* for more information.

The hub of Source Control is the Source Control Bank. The Source Control Bank is a storage facility that includes controlled copies of elements, but it does not replace the Model. The Source Control Bank supports Source Control services, such as locking and maintaining the history of an element, while the Model functions as a work area where copies of the controlled elements are modified and tested before you move them back into the Source Control Bank. The elements in a Model are automatically protected by Source Control when it is active. You can add individual elements or a group of related elements in the bank as one file. For example, you can add a <<Segment>> Class to the Source Control Bank along with all its children such as <<Ispec>> Classes, <<Report>> Classes, Attributes.

Once you add an element to the Source Control Bank, if you want to modify that element and keep track of its versions, you must check it out. The Check Out procedure exports a copy of the element from the Source Control Bank and then imports it into the Model, where you can modify it. That element is locked under your userid in Source Control.

When you check in a modified element, a new version of the element is created in Source Control. If you attempt to modify an element that is under Source Control without checking it out, it is automatically checked out irrespective of the setting that you may have selected from the **Tools**, **Options** setting in Visual Studio under the Source Control folder.

You can also 'Get' an element, which is to take a copy without checking it out, but the element is not locked and can be checked out and modified by another user. The current version in a Model is replaced with the version that is the subject of the "Get" action.

Source Control also allows you to 'Undo'pending changes, thereby discarding any changes made to an element. Only the user, who owns the workspace or the Source Control Administrator is permitted to undo the pending changes.

Source Control relies on the Source Control Services in Visual Studio and is subject to the special conditions of Source Control in System Modeler. You can control the functionality of Source Control by setting a number of user options.

Viewing Specifications, Differences, and Merging

You can view the specifications of an element in the Model, and an Agile Business Suite format file. Using the System Modeler Compare and Merge Tools, you can also compare two elements, or files side by side.

With the additional merge utility you can combine two elements, or files by reviewing each difference between the two items and selecting which variant is included in the final merge item.

You can also create a report on the differences between models, versions in the source control bank, and EXML files.

These utilities can be accessed from the development environment by using the System Modeler Tools or directly from the Source Control Utilities.

Note: *A source control tool must be installedto report on differences in versions in a source control bank.*

System Modeler Tools

The view, merge, and report differences utilities can be accessed from within the development environment from the Tools, System Modeler Tools menu.

You can view the differences between elements or files, merge them into a third output file or create a report that identifies the differences between two files.

Note: *A source control tool must be installed to report on differences between versions in a source control bank.*

Viewing Differences between Two Specifications

The View Differences utility allows you to view the differences between the specifications of two elements or files.

To view differences, perform the following:

1. Open a System Modeler project in Visual Studio.
2. Select the model or element in the Class View.
3. On the **Tools** menu, select System Modeler Tools, and then click View Differences menu item to open the Compare Elements/Files Wizard.

Compare Elements/Files Wizard

The Compare Elements/Wizard allows you to view differences between the specifications of two elements or files.

The Compare Elements/Files Wizard displays two dialog pages; First Compare Options, Second Compare Options. These pages allow you to select the elements, files that form the basis of the compared output, and select suitable export formatting methods.

First Compare Options

This page allows you to select the first model elements, file that form the basis on which the compared output is generated, and contains two radio buttons with associated edit controls. Select one of the radio buttons; the edit controls are enabled depending on which radio button is selected.

- **Model Element**
  1. Use the browse button to select a model element
  2. Once you have selected the model element, select from the list of available Export Format.

     Following are the list of choices by which the export can be performed.

     – **Normal** – Exports the selected object and all its “child” elements.

     – **VersionControl** – Exports the selected element and all the “child” elements that form part of the selected element.

       Additionally, if the selected element is version controlled as a child of an element that has not been selected, the export includes that “parent” element as well.

       If any of the “child” elements of the selected element are versioned separately from the selected element, these “child” elements are excluded from the export.
  3. Click **Next** to move to the “Second Compare Options” page of the dialog box.

- **File Name**

  1. Use **browse** to select the .model file to be included in the selection. These files are usually created by performing an export of an existing model, or its element(s).
  2. Click **Next** to move to the "Second Compare Options" page of the dialog box.

Second Compare Options

The Second Compare Options page allows you to select the second model element that forms the comparison based on which the compared output is generated. The fields should be completed in a similar way to those in the "First Compare Options" page.

Click **Finish** to open the Compare View window.

Merging differences

The merge utility combines two elements, or files by reviewing the difference between the two items, and selecting which variant is included in the final merged item.

You can merge the differences between elements and files into a third output file.

To merge differences, perform the following:

1. Open a System Modeler project in Visual Studio.
2. Select the model or element in the Class View.
3. On the **Tools** menu, select **System Modeler Tools**, and then click **Merge Differences** menu item to open the Merge Elements/Files Wizard.

Merge Elements/Files Wizard

The Merge Elements/Files Wizard allows you to combine two elements or files to generate the final merged file.

The Merge Elements/Files Wizard displays three dialog pages; First Compare Options, Second Compare Options, and Merge Options. These pages allow you to select the files that form the basis of the merged output file, specify a location to where the merged output file is saved, and select suitable export formatting methods.

First Compare Options

This page allows you to select the first model elements, files that form the basis on which the final merged file is generated, and contains two radio buttons with associated edit controls. Select one of the radio buttons; the edit controls are enabled depending on which radio button is selected.

- **Model Element**
    1. Use **browse** to select a model element.
    2. Once you have selected the model element, select from the list of available Export Format.

       Following are the list of choices by which the export can be performed.

       – **Normal** – Exports the selected object and all its "child" elements.

       – **VersionControl** – Exports the selected element and all the "child" elements that form part of the selected element.

         Additionally, if the selected element is version controlled as a child of an element that has not been selected, the export includes that "parent" element as well.

         If any of the "child" elements of the selected element are versioned separately from the selected element, these "child" elements are excluded from the export.
    3. Click **Next** to move to the "Second Compare Options" page of the dialog box.
- **File Name**
    1. Use the **browse** to select the .model file to be included in the selection. These files are usually created by performing an export of an existing model, or its element(s).
    2. Click **Next** to move to the "Second Compare Options" page of the dialog box.

Second Compare Options

The Second Compare Options page allows you to select the second model element that forms the comparison based on which the merged output file is generated. The fields should be completed in a similar way to those in the "First Compare Options" page.

Click **Next** to move to the "Merge Options" page of the dialog box.

Merge Options

1. Enter or browse for a file that holds the merged output of the two files.
2. Select **Import Merge Result**, if you want the merged file to be imported into your model immediately.
3. Click **Finish** to open the **Merge Options** window.

Creating a Differences Report

The Report Differences utility allows you to produce a report on the differences between two models, two labeled versions of elements in the source control bank, or the .model files in two different directories. You can also create a report on any mixture of those options, such as a model and a set of .model files in a directory. However, you should ensure that the differences reported upon are of value as the wizard does not determine the suitability of the objects being selected for the report.

To compare labeled versions of elements in the source control bank, perform the following:

1. Perform a 'Get Latest' or 'Get By Label' by using the Source Control Client.

   This places all the files of the project in the target directory.
2. Invoke the Report Differences Generation Wizard to compare the target directory with an AB Suite model or another directory.

You can use the Tools, System Modeler Tools, Report Differences menu to produce a report in XML format that displays the differences between two files, or two models.

To create a differences report, perform the following:

1. Open a System Modeler project in Visual Studio.
2. Select the model in the Class View.
3. On the **Tools** menu, select **System Modeler Tools**, and then click **Report Differences** to display the Report Differences Generation Wizard.

Report Differences Generation Wizard

The Report Differences Generation Wizard allows you to produce a report on the differences between two models, two labeled versions of elements in the source control bank, or the .model files in two different directories.

The Report Differences Generation Wizard displays three dialog pages; First Compare Options, Second Compare Options, Report Format Options. These pages allow you to select the files that form the basis of the differences report, select suitable formatting of the report, and specify a location to where the report is saved.

First Compare Options

The First Compare Options page allows you to select the first object that forms the basis of comparison on which the report is based, and contains two radio buttons with associated edit controls. Select one of the radio buttons; the edit controls are enabled depending on the radio button selected.

- **Model Database**

  If you want to compare a complete Model database, select this radio button and perform the following:

  1. In the Server Name list box, select from the list of available servers the location of the server on which the model database is located.

     Note: *The server initially displayed is the one on which the current model is stored.*
  2. Once you have selected the server, select from the list of available databases, the one which you want to be part of the differences report.

     Note: *The Model name is automatically displayed based on the database name selected.*

3. Click **Next** to move to the Second Compare Options page of the dialog box.

- **Folder**

  1. Use **browse** to select a folder that either contains a model, labeled version of an element in the source control bank, or the .model file to be included in the selection. The .model files are usually created by performing an export of an existing model, or its element(s).
  2. Click **Next** to move to the "Second Compare Options" page of the dialog box.

Second Compare Options

1. The Second Compare Options page allows you to select the second object that forms the comparison on which the report is based. The fields should be completed in a similar way to those in the "First Compare Options" page.
2. Click **Next** to move to the "Report Format Options" page of the dialog box.

Report Format Options

The Report Format Options page allows you to specify the output formatting of the report and the output file.

- **Report Title** Group
  - Standard: If you want the title of the report to contain the standard title, select this radio button.
  - Custom: If you want to specify a different title, select this radio button and enter the text for the title in the edit box.
- **Stylesheet** Group
  - Standard: If you want the report to use the standard XML stylesheet for formatting, select this radio button. The standard stylesheet is located in the bin directory of the installation folder.
  - Custom: If you want to specify a custom stylesheet for the report formatting, select this radio button and browse for you custom stylesheet.

    Note: *The custom stylesheet must have been previously created and available on your system.*
- **Result** Group
  - FileName: The Filename edit box allows you to specify the location and output file name to which you want the report file to be saved. You can also use the Browse button to select the location and the output file name.
  - View Report: If you want to view the report immediately after it has been created, select the **View Report** check-box, otherwise the report file is saved and you can view it later.
  - Display Files Full Path in the Report: If you want the report to include the full path name where the report is created, select this check-box.

3. Click **Finish** to view the differences between the compared objects in XML format.

Source Control Services in Visual Studio

System Modeler allows you to perform most of the operations in Source Control within Visual Studio. There is an exception to the list of operations, which is listed in the later subsection. The exception is primarily because System Modeler is a model-based system as opposed to the common file based system with which Visual Studio usually interacts.

Visual Studio allows integration of a variety of version control tools with its Source Control services. Options are available to specify how Source Control services should respond to operations invoked by the user. These are available from the **Tools**, **Options** menu within the Source Control folder. Refer to the *Visual Studio Online Help* for overview information on the Visual Studio Options dialog box.

For options specific to using Source Control for System Modeler, refer to Setting User Options in Source Control.

The *Visual Studio Online Help* provides extremely detailed information on the Visual Studio Source Control services, for example, on TFS and describes the working and the operations that can be carried out with it. You should refer to this information if you are not familiar with the concepts of Source Control, prior to using the System Modeler Source Control and its documentation.

The information in this section of the System Modeler help includes the functionality specific to using Source Control with System Modeler.

Source Control in System Modeler

Irrespective of the tool that is used to supply Source Control services, all functions provided by System Modeler are accessed using the Visual Studio Source Control and TFS project collection from menu item under the File menu, Team menu, or through context menus. Refer to the online tutorial of Visual Studio, Getting Started with Team Foundation Server for connecting to Team Foundation Server and creating a Team Project. Due to the structural difference in System Modeler, the following menu item is not suitable and therefore is not available within a System Modeler project, or it exhibits a different behavior.

Menu	Accessed from...	Description
Exclude <<Element Name>> From Source Control.	File, Source Control menu when an element is selected In Class view or Solution Explorer.	Not available for a Model based system.

Context Menus

In addition to the menu items accessed from the **File**, **Source Control** menu, you can select an element and right-click to display a context menu. These offer the same source control functions within System Modeler as any other Visual Studio project.

Source Control Utilities

To be able to view differences, or merge, elements directly in the System Modeler Source Control Utilities, they must be exported so that the elements are available in the correct format. The Export utility provides a means to perform this action.

To export an element, perform the following:

1. From Class view or the Members pane, select the element or group of elements that you wish to export.
2. From the **File** menu, select **Export**.

   The Export Wizard with the selected elements checked on the tree view is displayed.

Refer to the I*mport/Export Wizard Online Help* for more information on using the wizard.

To access the wizard externally, go to **Start** > **Apps** > **Agile Business Suite 6.1** > **Model Exporter**.

Elements exported from Developer models are contained in the Agile Business Suite XML file.

Once the elements have been exported into System Modeler Export Files you can open the files in the Version Control Utilities.

The Differences utility can also be invoked from the command line by using the following command and parameters:

```
Compare <file name 1> <file name 2>
```

Where <file name 1> is the first file to view in the Compare View window, and <file name 2> is the second file to view in the Compare View window.

The Merge utility can also be invoked from the command line by using the following command and parameters:

```
Merge <file name 1> <file name 2> <file name 3>
```

Where <file name 1> is the first file to view in the Merge View window, <file name 2> is the second file to view in the Merge View window, and <file name 3> is the name of the resulting merged Interchange file.

Setting Version Files

The Version Files utility allows you to set the VersionFile for versionable elements like Namespaces and Diagrams. This utility sets the VersionFile property to ensure that the element is unique within the model.

Note: *The element with a version file name specified holds all the elements in this domain that do not have the version file name specified.*

To invoke the Set Version Files utility, perform the following:

1. Open a System Modeler project in Visual Studio.
2. Select the model in the Class View.
3. On the **Tools** menu, select **System Modeler Tools**, and then click **Set Version Files** menu item to open the Set Version Files Wizard.

Set Version Files Wizard

The Set Version Files Wizard allows you to set the VersionFile for versionable elements like Namespaces and Diagrams. This utility sets the VersionFile property to ensure that the element is unique within the model.

Granularity of Versioning

The elements in AB Suite can be versioned individually or collectively in a single file. This wizard allows for the granularity of the elements that does not have a version control file set, to be set.

The two options available are:

- Complex Structures/Component – All Non-Primitive objects is set into their own separate version control file.
- Primitive and Complex Structures/Segment Members – All elements owned by the Model, or the Segment, (plus profiles) is set into their own separate version control file.

File Name Format

The Set Version Files wizard creates a version file for each element that is versionable. It sets the Version File property to either <ElementName>.model or <SegmentName>/<ElementName>.model to ensure that the name is unique within the model.

Click **Finish** to set the version file of the model elements.

Setting User Options

Several user options are available for Source Control within the Options dialog box. You can also set the **VersionFile** Property for newly created Classes, Folders, Dictionaries, or Diagrams in the **Policies** page.

To access the **Source Control** page, select **Options** from the **Tools** menu and select the **Source Control** page. From this page, you can select the following options:

- Plug-in Selection
- Environment
- Team Foundation Server

Plug-in Selection

Current source control plug-in – This field specifies the source-control plug-in to use with Microsoft Visual Studio and allows changes to plug-in specific options.

Environment

You can control default values for environment settings of version control by using a set of following options.

Get everything when a solution or project is opened check box – This option enables a get operation and downloads the files to your workplace when you open a solution or project file.

Check in everything when closing a solution or project check box – This option prompts you to check-in elements when you close a solution or project.

Display silent check out command in menus check box – This option displays the **Check Out for Edit Now command** on the File menu and suppresses the **Check Out** dialog box.

Keep items checked out when checking in check box – This option specifies that when you check in items to update the Source Control store, the items should remain checked out to you.

Team Foundation Server

You can configure Team Foundation version control to use a proxy server, which caches copies of version control files in the location of a distributed team. This significantly reduces bandwidth requirements for remote developers by using a proxy server. To perform this procedure, you must be a member of the Administrators or Users security group on the computer where Visual Studio is installed.

Use proxy server for file downloads check box – This option allows you to specify the server name and port to configure an AB Suite client to use Team Foundation proxy server.

Note: *Team Foundation Server Proxy listens for client requests on port 8081.*

To access the System Modeler Policies page, select **Options** from the **Tools** menu and choose the **Policies** page. From this page, you can select the following options:

Confirm Move on owner change – This option prompts you to confirm when an element is moved to a new owner. The settings available are **Ask**, **Don't Ask**, and **Ignore**.

Set the VersionFile Property on creation – This option enables you to set the **VersionFile** Property for newly created Classes, Folders, Dictionaries, or Diagrams.

The settings available are **Ask**, **Don’t Ask**, and **Ignore**.

- Ask: Prompts you whether to assign a value to the **VersionFile** Property or not?
- Don't Ask: Assigns a value to the **VersionFile** Property
- Ignore: No value is assigned to the **VersionFile** Property.

You can additionally set the VersionFile Property of the following:

- All Classes
- All Classes and Folders
- All Classes owned by the Model or a Segment

All Classes and Folders owned by the Model or a Segment

Versionable Objects Control Status

You can make all elements within the System Modeler versionable. The Solution Explorer and Class View windows display the source control status of elements by using the following signal icons:

Signal Icons

Signal icons indicate information about the status of your elements.

🔒 Checked In

Element is checked into a source control database.

✓ Checked Out Exclusive

Element is checked out from Source Control to one developer only. Other developers cannot access this element. Select a checked out element and select the context menu item, Source Control Properties to display information of the user who has the element checked out.

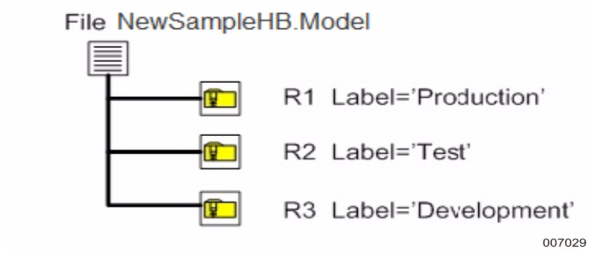
Note: *Shared check out is not available in System Modeler due to restrictions of the model-based system.*

+ Pending Addition

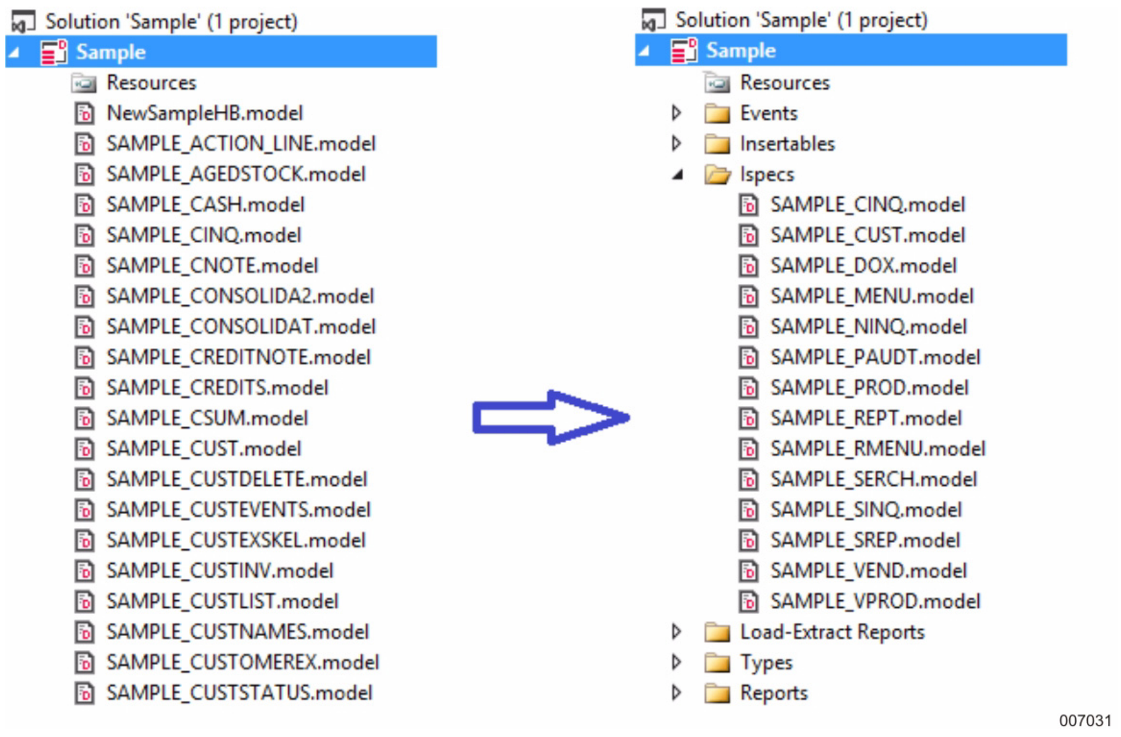
An element is displayed with a plus (+) symbol that denotes a pending addition. The element is added to the source files list in the **Pending Changes** window.

Setting up the Source Control Bank in an AB Suite Environment

In an AB Suite Development environment, always ensure that files representing Dev, Test, and Prod environment are present. The Source Control Bank and the respective databases are populated with the three model files. The aim of the following steps is to create a file history for each versionable element. For example, the Sample model has three model files, Production, Test, and Development.



For performance reasons, you should include the version files in folders. Visual Studio prefers the files to be hierarchically arranged in folders within the project node. Thereby, try reorganizing the files of the Sample model as shown in the right pane of the following figure.



Solution 'Sample' (1 project)
- Sample
  - Resources
  - NewSampleHB.model
  - SAMPLE_ACTION_LINE.model
  - SAMPLE_AGEDSTOCK.model
  - SAMPLE_CASH.model
  - SAMPLE_CINQ.model
  - SAMPLE_CNOTE.model
  - SAMPLE_CONSOLIDA2.model
  - SAMPLE_CONSOLIDAT.model
  - SAMPLE_CREDITNOTE.model
  - SAMPLE_CREDITS.model
  - SAMPLE_CSUM.model
  - SAMPLE_CUST.model
  - SAMPLE_CUSTDELETE.model
  - SAMPLE_CUSTEVENTS.model
  - SAMPLE_CUSTEXSKEL.model
  - SAMPLE_CUSTINV.model
  - SAMPLE_CUSTLIST.model
  - SAMPLE_CUSTNAMES.model
  - SAMPLE_CUSTOMEREX.model
  - SAMPLE_CUSTSTATUS.model

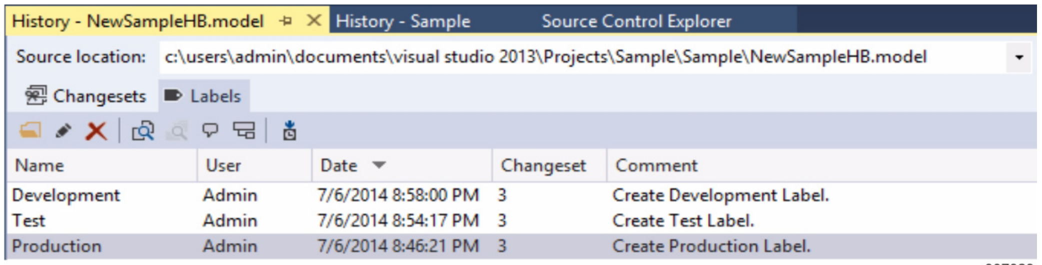
Solution 'Sample' (1 project)
- Sample
  - Resources
  - Events
  - Insertables
  - Ispecs
    - SAMPLE_CINQ.model
    - SAMPLE_CUST.model
    - SAMPLE_DOX.model
    - SAMPLE_MENU.model
    - SAMPLE_NINQ.model
    - SAMPLE_PAUDT.model
    - SAMPLE_PROD.model
    - SAMPLE_REPT.model
    - SAMPLE_RMENU.model
    - SAMPLE_SERCH.model
    - SAMPLE_SINQ.model
    - SAMPLE_SREP.model
    - SAMPLE_VEND.model
    - SAMPLE_VPROD.model
  - Load-Extract Reports
  - Types
  - Reports

You can move or add version files when project and files are managed with Source Control, but the act of moving a file to a folder, is a 'Remove' and 'Add' in to Source Control.

To add or move version files, perform the following:

1. **Import** Prod.model into a new database.
2. Create folders and add files to the folders.
3. Add Label, 'Production' to all the files. Refer to Using Labels in Source Control to add labels.
4. Check out all the files.
5. **Import** the Test.model into the new database created in step 1.
6. Check in all the files and add Label, 'Test' to all the files. Refer to Using Labels in Source Control to add labels.
7. Check out all the files.
8. **Import** the Dev.model into the new database created in step 1.
9. Check in all the files and add Label 'Development' to all the files. Refer to Using Labels in Source Control to add labels.

Always ensure that the three model files have the following output.



Source location: c:\users\admin\documents\visual studio 2013\Projects\Sample\Sample\NewSampleHB.model

Name	User	Date	Changeset	Comment
Development	Admin	7/6/2014 8:58:00 PM	3	Create Development Label.
Test	Admin	7/6/2014 8:54:17 PM	3	Create Test Label.
Production	Admin	7/6/2014 8:46:21 PM	3	Create Production Label.

007023

Using Source Control

Other than the exceptions identified in Source Control in System Modeler, using Source Control to manipulate versioned elements in System Modeler is the same as in any other Visual Studio project. Refer to Visual Studio Source Control services for more information on Source Control in Visual Studio. The following information covers the basic operations information covers the basic operations.

- Adding Elements, Folders, Dictionaries to the Source Control Bank
- Checking In an Element
- Checking Out an Element
- Getting an Element
- If you do not want to synchronize the workspace and the database automatically, perform the following:
- Using Command Line Interface
- Comparing Versions

Before you can work with files that are in Source Control, you must create a workspace on your local computer to which you copy the files. Refer to Creating a Workspace with your Team Project in Source Control to create a workspace in TFS.

Adding Elements, Folders, Dictionaries to the Source Control Bank

All elements, Folders, Dictionaries are versionable. Any element that is added in Source Control as part of its parent can be individually versioned separately from its parent. For instance, this may be a requirement if you have a number of <Ispec> classes within a segment which different developers wish to work on independently.

To prevent one developer locking the entire set of elements by checking out one element, you can separately version the child elements, and break them away from their parent element. You can move or add version files when project and files are managed with Source Control, but the act of moving a file to a folder, is a 'Remove' and 'Add' in Source Control.

The elements can either exist in the Source Control Bank individually or as part of some ancestor that has been versioned. The top node (Model) is always versioned, and so everything in the Model is initially versioned as part of the Model, unless you have set the VersionFile Property of specific Elements in the Model. In that case, the source bank contains versioned files representing the Model and each of the Elements with their **VersionFile** Property set. The **VersionFile** Property of an element in a Model be assigned a value when created depending on the settings in the **Policies** page. If the **VersionFile** Property of elements in a Model is set and the newly added elements are not initially versioned as part of the Model, the Model is not locked. Note that everything in a Model belongs to some versioned file in the source bank, be it the Model node file or another file.

To make an Element, Folder, Dictionary versionable, perform the following:

1. Select the element, Folder, Dictionary in **Class View**, or **Members Tab** of the document window.

   Note: *Select the element to be versioned separately, if you want to version an element separately from the parent.*

2. If it is not already open, open the **Properties** window from the **View**, **Properties** Window menu item.
3. Navigate to the **VersionFile** Property.
4. Type a name for the file that contains details of the element in the Source Control Bank. The file name must be unique across the entire Model.

   The versioned element(s) file name(s) is now displayed in the **Solution Explorer** window. Although, the versioned element is not yet added to the source control bank, but is in a "checked out" state. The red tick symbol next to the element icon indicates the checkout state from both **Class View** and **Solution Explorer** windows.

   Note: *When you add an initial element to Source Control, the entire Model is also added. This is due to the dependency that elements have with each other. On importing the .smproj file from Source Control, the Import wizard allows you to import all the model files along with the .smproj file in one go.*

When an element is separately versioned from its parents, ensure to note the following behavior:

- The parent of the separately versioned element is also checked out. The parent is checked out to create the separation of the element and the parent. The parent should be checked back into the Source Control to protect its integrity.
- The now separately versioned element includes its children as part of the separated group.
- The element can be the subject of all Source Control operations independently from the parent.
- The parent of the element can be the subject of all Source Control operations independently from the separated element, and independently from other separately versioned elements.

If Source Control detects a newly added or removed versionable element that is added or removed by a different user, the project file and Source Control information for the user and session should be updated. The project file is checked out and you are prompted to add the new versionable elements to the Source Control.

Following are the two options available when prompted for adding the newly detected elements:

- Select **Cancel**, if you did not add the new elements.

  The project file is checked out and you can see the new elements in Solution Explorer, Class View.
- Select **OK**, if you are not sure whether you or some other user added the new elements. If the file or element exists, then a warning message appears that a file or element with the name already exists.

  The project file is checked out and you can see the new elements in Solution Explorer, Class View.

  Note: *Since the checkout procedure of the project file may reload the project, you might need to do the operation again that added or removed the new elements.*

The parent and all siblings of the new element are checked out prior to including the new element, unless one or more of the siblings was separately versioned. This ensures that the controlled parent and possible siblings of the new element are associated with it.

Note that when a project is added to Source Control, all the files are added to one single folder. AB Suite System Modeler allows users to add or remove folders to the Solution Explorer. A folder created in the Solution Explorer is also represented by a folder on the hard disk.

Files can be moved to and from a folder (in the Solution Explorer) by:

- Selecting files and performing a cut or paste operation.
- Using the **Add New Item** dialog on a folder node in the Solution explorer.

When a file is moved to or from a folder and the file exits on disk, the file on disk also moves to the new location. If the file is under Source Control, the history of the file is lost, because the file is deleted from the Source Control bank in the old location and added to the Source Control bank at the new location. Read the user manual of the Source Control tool of your choice on how to recover a deleted file and how to move files if you want to preserve this history.

Migrating an existing version control bank

If you have an existing version control bank, you can migrate the elements in the bank by using the History Migrator Utility. Refer to the *History Migrator Online Help* for more information on this tool.

After migration, three model files representing Dev, Test, and Prod environments exist.

Checking In an Element

Checking in an element exports the element from the Model and creates a new version of the file containing the selected element in the Source Control Bank. Team Foundation version control files are checked in to the source control server by using the **Pending Changes** window.

Any load errors that you encounter during check in are written to the Visual Studio Source Control Output Window.

To display and check in pending changes, perform the following:

1. View pending changes by performing any one of the following:
   - In **Source Control Explorer**, right-click any element that contains pending changes (indicated by this symbol: ✓), and then click **Check In Pending Changes**.
   - In **Solution Explorer**, right-click one or more solutions, projects, or files, and then click Check In.
   - On the **View** menu, point to **Other Windows**, and click **Pending Changes**.
2. In the **Source Files** channel, clear the check boxes for any elements that you do not want to check in, and type any applicable comments in the **Comment** box.
3. If you select more than one element, the version file name does not appear in the Menu.
4. Click **Check In**.

Checking Out an Element

Checking out an element exports a copy of the selected element from the Version Control Bank and then imports it into the System Modeler, where you can modify it. When you check out a file, it is reserved under your userid.

To remove the reservation on an element, the TFS Administrator must undo the checkout performed by another user. This can be easily done by a command line utility available with Team Explorer. The command line can be accessed by launching the Visual Studio Command Prompt window. Refer to the MSDN documentation for using the TFS command line.

You can then remove the reservation on the file by using the Reservation Tab. Refer to Reservation Tab for more information.

Any load errors that you encounter during check out are written to the Visual Studio Source Control Output Window.

To check out an element, perform the following:

1. Select the element(s) in the **Solution Explorer** or **Class View** window.
2. Select **File**, **Source Control**, **Check Out** <version file name> menu item, or **Check Out** from the context menu. The element is checked out and the status icon changes from a lock to a red tick mark.

If you select more than one element, the version file name does not appear in the Menu.

Getting an Element

'Getting' a version of an element from the Version Bank makes a copy of the latest version without locking the element. When you get a file, it is not locked under your userid and another user can check out the file and modify it, unless you have previously checked out the file. In such a scenario, it silently discards any changes you have made since checking out.

There are three Get commands in Version Control:

- **Get Latest Version** – Loads the latest version of an element and its versioned children from the Source Control Bank without locking the element.

  Any load errors that you encounter while getting elements are written to the Visual Studio Source Control Output Window.

  #### Getting the latest version of an element

  To get the latest version of an element, perform the following:

  1. Create and identify a workspace in TFS. Refer to Creating a Workspace with your Team Project in Source Control.
  2. Select the element(s) in the **Solution Explorer** or **Class View**.
  3. Select **File**, **Source Control**, **Get** <version file name> menu item, or **Get** from the context menu.

     The latest version of the selected element is copied to the workspace from Source Control.

Note: *If you select more than one element, the version file name does not appear in the Menu*

- **Get Specific Version** - Retrieves a specific version of all elements from the Source Control Bank without locking any element.

Getting a specific version of an element

To get a specific version of an element in a Model, perform the following:

1. Create and identify a workspace in TFS. Refer to Creating a Workspace with your Team Project in Source Control.
2. Select the element in the **Solution Explorer**.
3. Select **File**, **Source Control**, **Get Specific Version** menu item, or **Get Specific Version** from the context menu.

   The specified version of the selected element is copied to the workspace from Source Control.

- **Get Latest Version (recursive)** - Loads the latest version of all elements from the Source Control Bank without locking any element. This menu command is only available at the Model level of the tree structure. It effectively gets an entire model and all its versioned elements, even where some of the element may be separately versioned.

WARNING

This silently discards all changes made to any checked out element since the checkout.

Getting the latest version of an element recursively

To get the latest version of all elements in a Model recursively, perform the following:

1. Create and identify a workspace in TFS. Refer to Creating a Workspace with your Team Project in Source Control.
2. Select the Model in the **Solution Explorer**.
3. Select the **File**, **Source Control**, **Get Latest Version (Recursive)** menu item, or **Get Latest Version (Recursive)** from the context menu.

The workspace and the database get synchronized automatically and the latest version of the selected model is copied to the workspace from Source Control.

If you do not want to synchronize the workspace and the database automatically, perform the following:

1. In the Solution Explorer window, right-click the project and select **Unload Project**.
2. Right-click the project, and then select **Edit <AB Suite Project Name>.smproj**.
3. In the <AB Suite Project Name>.smproj code editor set the <SccMaintainFiles> and <SccMaintainModel> tags to **False**.

Or

1. Open the <AB Suite Project Name>.smproj file in text editor, such as Notepad.
2. Set the <SccMaintainFiles> and <SccMaintainModel> tags to **False**.

Note: *If you set the <SccMaintainFiles> to True and <SccMaintainModel> tags to False, the File Changes Detected window appears when you select Get Latest Version (Recursive) option. The File Changes Detected window displays the file names that are added, modified, or deleted. You can get the latest version of only the files that you want by selecting the check box next to the file names.*

If you do not want to synchronize the workspace and the database automatically, perform the following:

Viewing Pending Changes

The revisions to files in Team Foundation version control are saved to your local workspace until you check in these changes. Such changes are referred to as pending changes. You can view and manage pending changes by using the **Pending Changes** and **Check In** windows. From the **Pending Changes** window, you can:

- View and check in pending changes in your workspace.

  Refer to Checking In an Element in Source Control for more information on viewing and checking in pending changes.

- Undo pending changes.

  To undo pending changes in Source Control Explorer, perform the following:

  1. In **Source Control Explorer**, right-click any element for which you want to undo pending changes, and click **Undo** or **Undo Pending Changes**.
  2. In the **Undo Pending Changes** dialog box, select the check box of each file for which you want to undo pending changes, and click **Undo Changes**.
     - In Source Control Explorer, the pending change type is removed in the **Pending Change** column.
     - If you undo an edit, your copy of the file is replaced with an unmodified version you checked out.
     - If you undo a delete, the version you deleted is restored (unless you have performed a get operation since pending the delete; in which case the version you last tried to get is downloaded).
     - If you undo an add, the file is left undisturbed.

- Compare elements in a pending change to a different version

To compare files in a pending change to a different version, perform the following:

1. On the **View** menu, click **Other Windows**, and then click **Pending Changes**.

   The Pending Changes window is displayed.

2. In the **Pending Changes** window, right-click the element that you want to compare to another version, click **Compare** and then click **With Unmodified Version**, **With Workspace Version** or **With Latest Version**.

3. If you are prompted with the **Source Control** dialog box, click **Save and continue** to save your changes to the disk before you continue with the compare.

   The two elements are presented with their differences highlighted, or if the elements are the same, a dialog box appears that shows that the elements are identical.

Refer to View and Manage All Pending Changes in your Workspace in *Visual Studio Online Help* for more information on performing the operations on pending changes window.

Using Command Line Interface

You can accomplish common Source Control operations by using a command line interface. This allows you to carry out the Source Control operations without being in the System Modeler environment, or the Source Control application. The Command Line Interface from the Source Control provider works in conjunction with the Import and Export tools. A Source Control provider provides a Command Line Interface to perform a Get, CheckIn or CheckOut of files. You can use AB Suite Export or Import command to either Export a version file ‘element’ or Import a version file ‘element’.

The following commands that are permitted in Command Line Interface are:

1. CheckIn – Check in the indicated elements.
2. CheckOut – Check out the indicated elements.
3. GetLatest or Get by Label – Get the latest version of the indicated elements.

Note: *The security implementation of the standard Microsoft Visual Studio Source Control API does not allow passing of passwords. This means that even though these are command line functions, they still display an authentication dialog when accessing the source control bank. The dialog varies depending on the source control provider.*

CheckIn

The CheckIn command line interface enables you to perform Check in functions from command prompt or within a script.

The following command line syntax is used to export the versioned elements from AB Suite Model and CheckIn versioned elements into a Source Control bank:

1. Type Export.exe at the command prompt to export the version file elements from an AB Suite Model.
   ```
   Export.exe -m FooDev -f C:\Temp\Class2.Model -vf Class2.model
   ```
   Note: *The -vf parameter indicates the version files to export when using Export.exe command.*
2. Use the Team Foundation Command line utility to check-in the file(s)
   ```
   tf checkin [/author:author name] [/comment:("comment"|@comment file)]
   [/noprompt] [/notes:("Note Name"="note text"|@notefile)]
   [/override:(reason|@reasonfile)] [/recursive] [/saved] [/validate] [itemspec] [/
   bypass] [/login:username,[password]]
   ```

where:

Parameter	Description
/noprompt	Suppresses any prompts for input from you..
/recursive	Checks in all elements in the specified or implied working folder and subfolders.
itemspec	Specifies a file or folder to check in.

Refer to Checkin Command in the *Visual Studio Online Help* for more information on parameters.

For example; to CheckIn Sample model by suppressing all prompts, use the following command.

```
tf checkin Sample.model  /noprompt
```

For help on these parameters enter tf parameter /? at the command prompt.

If you wish to capture and view a specific version of a file in a temporary folder for logging purposes, use the following View command as an example:

```
c:\temp>tf view c:\temp\input.txt > c:\temp\checkin.txt
```

CheckOut

The CheckOut command line interface enables you to perform Check out functions from command prompt or within a script.

The following command line syntax is used for Checking out versioned elements into Source Control Bank and import versioned elements from Source Control Bank.

1. Use the Team Foundation Command line utility to check out the file(s)
   ```
   tf checkout [/lock:(none|checkin|checkout)] [/recursive]
   ```

2. Type **Import.exe** to import the version file element(s) from an AB Suite Model file(s)
```
Import.exe -m FooDev  -vf  Sample.Model
```

Note: *The -vf parameter indicates the version files to import when using import.exe command.*

where:

Parameter	Description
/recursive	Checks out all files that match the itemspec in the current directory and in all the subfolders of the current directory.

Refer to Checkout Command in the *Visual Studio Online Help* for more information on parameters.

For example; to check out Sample model, use the following command:

```
tf checkout Sample/Sample.model
```

Get Latest or Get by Label

The Team Foundation Command line utility provides a command line interface to perform Get latest functions.

1. Perform a ‘Get Latest’ or ‘Get by Label’ by using the following Team Foundation Command line utility.
```
tf get [itemspec] [/version:versionspec] [/all] [/overwrite] [/force]
[/preview] [/recursive] [/remap] [/noprompt] [/login:username,[password]]
```
2. Type **Import** command to import all files.
```
Import.exe -m FooDev -vf Sample.smproj
```

Note: *The -vf parameter indicates the version files to import when using import.exe command.*

where:

Parameter	Description
/recursive	Recursively retrieves all elements that match your itemspec.
/nonprompt	Suppresses any dialog boxes that would have been displayed during this operation.

Refer to Get Command in the *Visual Studio Online Help* for more information on parameters.

To get complete information about Team Foundation command-line tool, tf, refer to Team Foundation Command-Line Reference in the *Visual Studio Online Help*.

Comparing Versions

You are able to compare the latest revision of an element in the Source Control bank with the current copy of the element in the model.

With the additional System Modeler Merge Export File utility, you can combine two elements or revisions by reviewing each difference between the two items and selecting which variant is included in the final merged file. You can also create a report on the differences between objects, revisions, and export files.

The System Modeler Compare and Merge Tools that is accessed from **File**, **Source Control**, and **Compare** menu item provide these functions. You can also access them from the **Compare Versions** item on the context menu. Refer to Compare Two Files in the *Visual Studio Online Help* for more information on how to compare different versions of an element.

Note: *The Compare Revisions function is only available when you select a System Modeler element within* ***Solution Explorer****, or the* ***Class View*** *window.*

Using Source Control with TFS

The following information gives an overview of the operations specific to TFS.

- Connecting to Team Projects
- Creating a Workspace with your Team Project
- Viewing Historical Data about an Element
- Using Labels

Connecting to Team Projects

You can organize your projects into groups by creating team project collections. Before connecting to team project, you should create a team project from the Team Explorer.

To connect to a team project collection, point to **Team** menu and click **Manage Connections** > **Connect to Team Project**. Refer to Creating a Team Project and Connecting to a Team Project Collection in the *Visual Studio Online Help* for more information.

To create a team project, right-click the team project collection and select **New Team Project**. Refer to Creating a Team Project and Connecting to a Team Project Collection in the *Visual StudioOnline Help* for more information.

The new team project appears in **Team Explorer**.

Creating a Workspace with your Team Project

A workspace includes folders on the local machine that is mapped to version-controlled folders on the Team Foundation version control server. You can create this workspace by using Visual Studio Application Lifecycle Management (ALM) or by opening a Command Prompt window. After you create a workspace, you typically download all of the files from the version control server into your workspace.

Refer to Create a Workspace and Get Files in the *Visual Studio Online Help* for more information on creating a workspace by using ALM.

To create a workspace manually, perform the following:

1. On the **File** menu, click **Source Control**, and then click **Workspaces**.
2. In the **Manage Workspaces** dialog box, click **Add**.

   Note: *You can also click the default workspace and then click Edit if you want to reuse the default workspace.*
3. For each folder that you want to map, perform the following:
   a. Under **Working folders**, click the first empty row in the **Source Control Folder** column, and then click the ellipses (...).
   b. In the **Browse for Folder** dialog box, click the folder on the server that contains the files that you want to work with, and then click **OK**.
   c. Under **Working folders**, click the first empty row in the **Local Folder** column, and then click the ellipses (...)
   d. In the **Browse for Folder** dialog box, click the local folder into which you want to copy the files.
   e. (Optional) Click **Make New Folder**, and type a name for the new folder where your local copies should be stored.
   f. Click **OK**.

Viewing Historical Data about an Element

Team Foundation version control maintains historical data related to every version of every element that was checked in. You can view the History window by right clicking and selecting **View History** in the Source Control Explorer. Refer to using the History Window in the *Visual Studio Online Help* for more information about the data displayed in this window or the actions that you can take.

Using Labels

Labels allow you to take a snapshot of your files so that later, you can refer back to that snapshot. You can

- Add a label
- List, view, find, edit, rename, remove labels
- Roll back a large set of files to a labeled version

Refer to Use Labels to Take a Snapshot of Your Files in the *Visual Studio Online Help* for more information on applying and using labels.

Source Control Operations

The Source Control wizard is activated when you open a solution under Source Control. It enables you to either use an existing model or create a new model.

To associate a project with the AB Suite model database, perform the following:

1. Open the solution file from the workspace.

   The files are automatically added to the projects directory.

   The System Modeler Project wizard appears that includes the following two options to select from:

   - Select **New** to create a new database.
     a. From the **Server Name** list, select a server.
     b. In the **Database Name** box, enter the database name.
     c. In the **Model Name** box, enter the model name.
     d. Click **Finish**.

   On completion, the model files in the workspace are imported automatically to the database.

   Or

   - Select **Existing** to synchronize the files with an existing database.
     a. From the **Server Name** list, select a server.
     b. From the **Database Name** list, select the database name.
     c. In the **Model Name** box, enter the model name.
     d. Click **Finish**.

In case of any differences between the workspace and database, the File Detection dialog box appears to resolve the differences.

Creating AB Suite Environment

This subsection describes the steps to create or update AB Suite environments such as, Development, Test, or Production with a Source Control database.

Set up a Development Environment

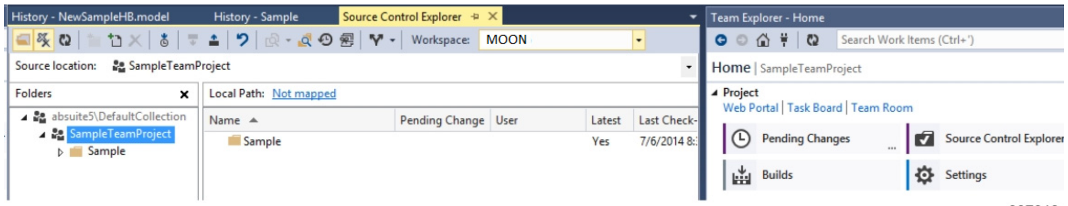
The steps to set up a Development Environment include, perform the following:

1. Connect to Team Projects by using **Team Explorer**.

   Refer to Connecting to Team Projects in Source Control for more information on connecting to Team Projects by using Team Explorer and other clients.

2. In **Team Explorer**, double-click the **Source Control** folder node that is associated with the team project you want to work with.

   Source Control Explorer opens with the respective team project highlighted.



Workspace: MOON

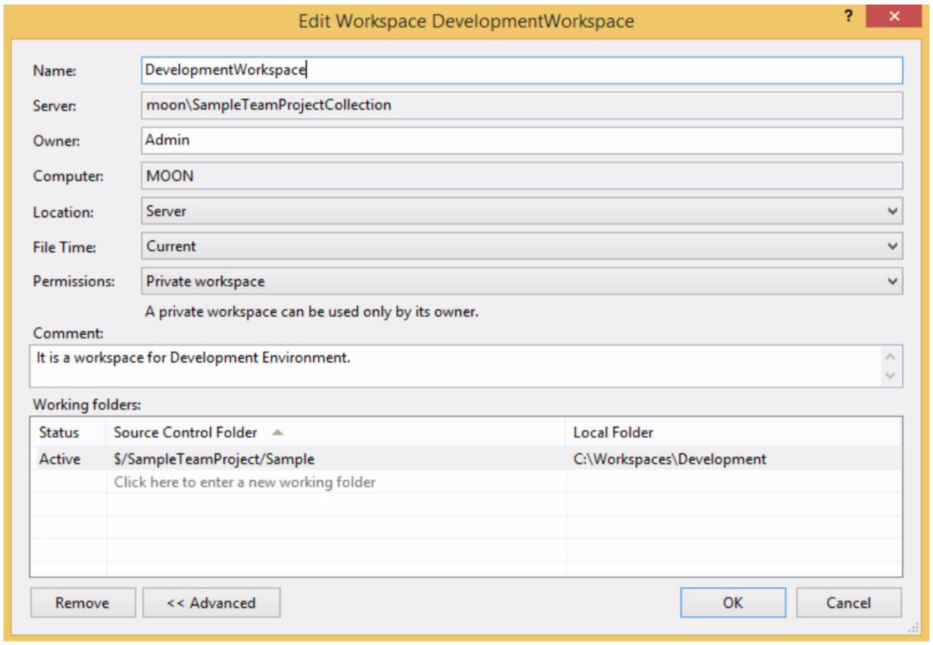
Source location: SampleTeamProject

Local Path: Not mapped

Name	Pending Change	User	Latest	Last Check-
Sample			Yes	7/6/2014 8:

007013

3. Create a workspace and select it as your workspace for Development Environment. Refer to Creating a Workspace with your Team Project in Source Control for more information on creating and updating a workspace.



Name: DevelopmentWorkspace

Server: moon\SampleTeamProjectCollection

Owner: Admin

Computer: MOON

Location: Server

File Time: Current

Permissions: Private workspace

A private workspace can be used only by its owner.

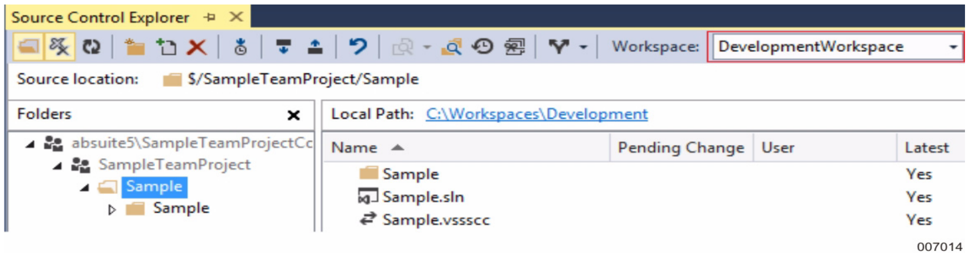
Comment:

It is a workspace for Development Environment.

Working folders:

Status	Source Control Folder	Local Folder
Active	$/SampleTeamProject/Sample	C:\Workspaces\Development
	Click here to enter a new working folder	

008206


Source Control Explorer
Workspace: DevelopmentWorkspace
Source location: $/SampleTeamProject/Sample
Folders
absuite5\SampleTeamProjectCc
SampleTeamProject
Sample
Sample
Local Path: C:\Workspaces\Development
Name
Pending Change
User
Latest
Sample
Yes
Sample.sln
Yes
Sample.vssscc
Yes
007014

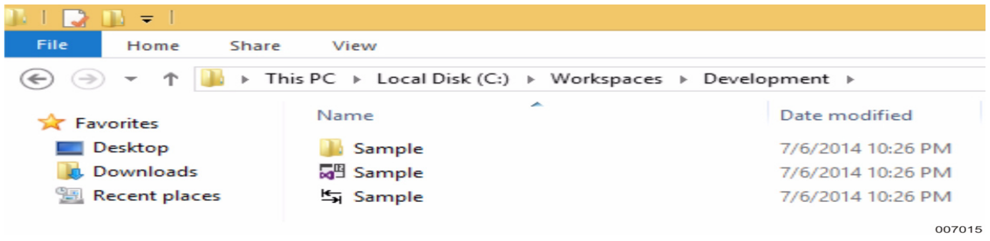
4. Select the working project in **Source Control Explorer**, and select **Get Specific Version** and **Get by Label** "Development". Refer to Getting an Element in Source Control for more information on getting specific version or getting by label.

   This copies all the files to the local folder. For example, C:\Workspaces\Development.

   Following is the folder structure and files in the local machine.

   The Get by Label "Development" step can also be achieved by using the Team Foundation command line utility. You can navigate to the working folder with the CD command and use the following command to get the files that are labeled as 'Development'. Refer to Command Line Interface in Source Control for more information on using Get Latest/Get by Label command line utility.

   ```
   C:\Workspaces\Development>tf get /version:Ldevelopment
   ```



   You can get a copy of the latest files in the local folder by running the following command in the command line:

   ```
   Import -m <model name> -vf <filename>
   ```

5. Open Visual Studio development environment, and then open the Solution or the Project file.

   The **Application Project Wizard** appears.

6. Select **New** or **Existing**.

   Note: *Select **New** if you want to associate an existing project, which is under source control, with a new database. Select **Existing** if you want to associate an existing project, which is under source control, with an existing database.*

   - If you select **New**, perform the following:

     a. From the **Server Name** list, select a server name.

     b. In the **Database Name** box, enter a database name.

     c. In the **Model Name** box, enter a model name.

- If you select **Existing**, perform the following:
    a. From the **Server Name** list, select a server name.
    b. From the **Database Name** list, select a database name.

8. Click **Next**.
9. Click **Finish**.

    Note: *When you associate an existing project with a new database, the database is initialized by importing all the version files that are present in the project file. However, when you associate an existing project with an existing database, it is assumed that you are dealing with a multi-user environment and the database is up-to-date.*

After performing these steps, you can perform the CheckIn or CheckOut operations.

Update or Refresh an Existing Development Environment

Following are the two ways to update an existing development environment with the latest changes:

- Using the VS IDE
- Using the Source Control Client with the import utility

Updating by using the VS IDE

1. Open the Solution or Project file.
2. Select the solution or project in the **Solution Explorer**.
3. Right-click the solution or project and select **Get Latest Version (Recursive)** from the context menu.

    This uploads all the changed files.

Updating by using the Source Control Client

1. Perform step1 through step 3 from subsection Setting up a Development Environment.
2. Right-click your working project in **Source Control Explorer** and select **Get Latest Version**.

    All out-of-date files are copied in the local folder. For example, C:\Workspaces\Development
3. In the Import Wizard, select the project file (smproj) to be imported.
4. Select the database to be updated.
5. Select **Replace** to avoid resolution conflicts.
6. Click **OK** to complete the import.

You can also perform step 3 through step 6 by using a batch file with the following commands:

```
C:\Workspaces\Development >tf get
Import.exe  -c -m <databaseName> -s <hostName>
C:\Workspaces\Development\Sample\Sample.smproj
```

Note: *The above steps only imports the changed files while importing a .smproj file.*

Set up a Test Environment

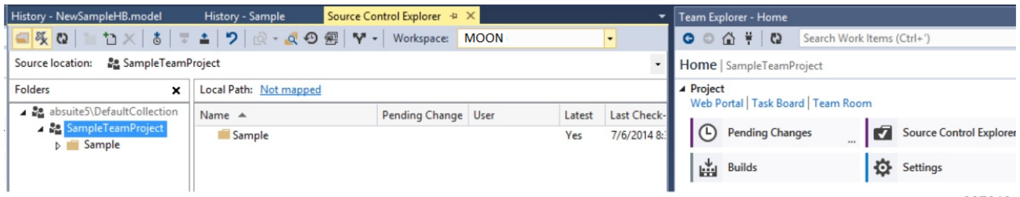
To set up a Test environment include, perform the following:

1. Connect to Team Projects by using **Team Explorer**.

   Refer to Connecting to Team Projects in Source Control for more information on connecting to Team Projects by using Team Explorer and other clients.

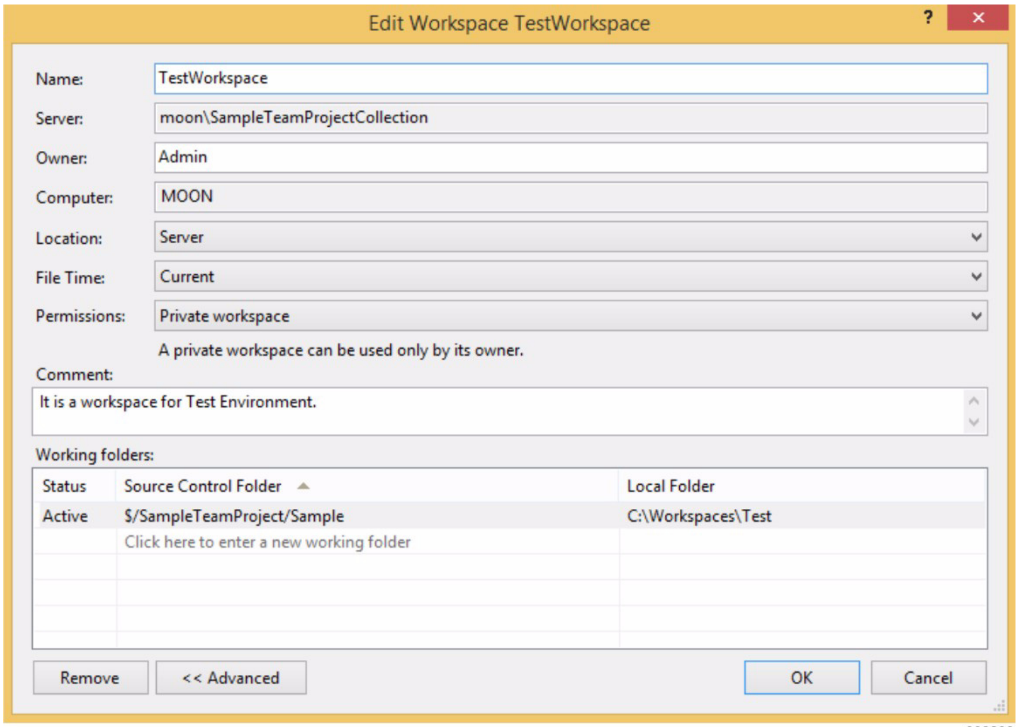
2. In the **Team Explorer**, double-click the **Source Control** folder node that is associated with the team project you want to work with.

   Source Control Explorer opens with the respective team project highlighted.



007013

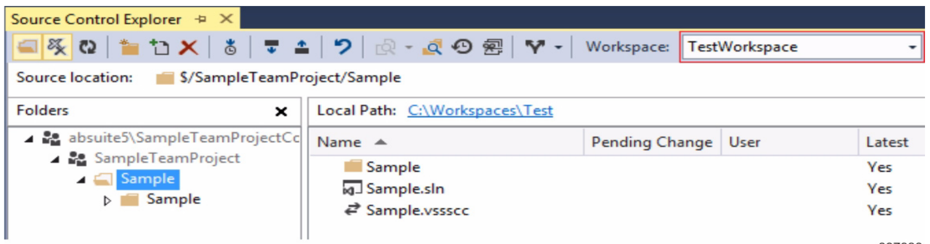
3. Create a workspace and select it as your workspace for the Test Environment. Refer to Creating a Workspace with your Team Project in Source Control for creating and updating a workspace. Following is the folder structure and files in the local machine.



Name: TestWorkspace
Server: moon\SampleTeamProjectCollection
Owner: Admin
Computer: MOON
Location: Server
File Time: Current
Permissions: Private workspace
A private workspace can be used only by its owner.
Comment:
It is a workspace for Test Environment.
Working folders:

Status	Source Control Folder	Local Folder
Active	$/SampleTeamProject/Sample	C:\Workspaces\Test
	Click here to enter a new working folder	

008208



Workspace: TestWorkspace
Source location: $/SampleTeamProject/Sample
Local Path: C:\Workspaces\Test

Name	Pending Change	User	Latest
Sample			Yes
Sample.sln			Yes
Sample.vssscc			Yes

007030

4. Select the working project in **Source Control Explorer**, and select **Get Specific Version** and **Get by Label** “Test”. Refer to Getting an Element in Source Control for more information on getting specific version or getting by label.

   You can get a copy of the latest files in the local folder by running the following command in the command line:

   ```
   Import -m <model name> -vf <filename>
   ```

5. Open Visual Studio development environment, and then open the Solution or the Project file.

   The **Application Project Wizard** appears.

6. Select **New** or **Existing**.

   Note: *Select **New** if you want to associate an existing project, which is under source control, with a new database. Select **Existing** if you want to associate an existing project, which is under source control, with an existing database.*

- If you select **New**, perform the following:
   a. From the **Server Name** list, select a server name.
   b. In the **Database Name** box, enter a database name.
   c. In the **Model Name** box, enter a model name.
- If you select **Existing**, perform the following:
   a. From the **Server Name** list, select a server name.
   b. From the **Database Name** list, select a database name.

7. Click **Next**.
8. Click **Finish**.

   Note: *When you associate an existing project with a new database, the database is initialized by importing all the version files that are present in the project file. However, when you associate an existing project with an existing database, it is assumed that you are dealing with a multi-user environment and the database it up-to-date.*
9. Click **Build** to build and deploy the system.

Note: *The above steps only imports the changed files while importing a .smproj file.*

Set up a Production Environment

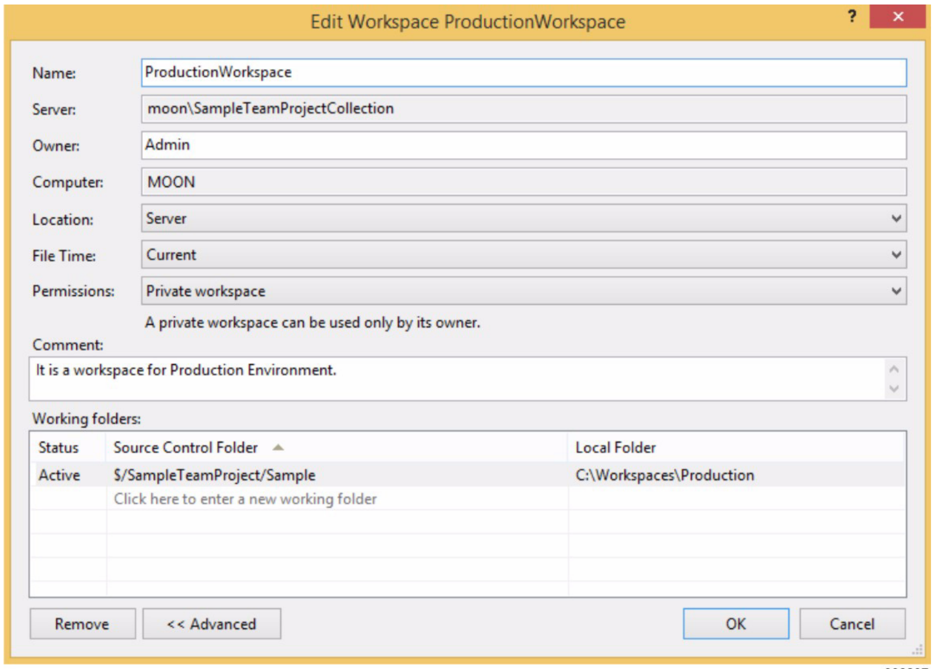
The steps mentioned in the Production environment are necessary if you perform builds in this environment and do not use the runtime transfer files from the Test environment.

1. Connect to Team Projects by using **Team Explorer**.

   Refer to Connecting to Team Projects in Source Control for more information on connecting to Team Projects by using Team Explorer and other clients.
2. In the Team Explorer, double-click the **Source Control** folder node that is associated with the team project you want to work with.

   Source Control Explorer opens with the respective team project highlighted.

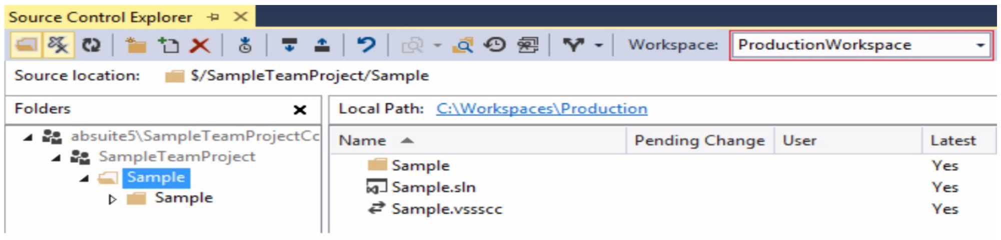
3. Create a workspace and select it as your workspace for Production Environment. Refer to Creating a Workspace with your Team Project in Source Control for more information on creating and updating a workspace.



Name: ProductionWorkspace
Server: moon\SampleTeamProjectCollection
Owner: Admin
Computer: MOON
Location: Server
File Time: Current
Permissions: Private workspace
A private workspace can be used only by its owner.
Comment:
It is a workspace for Production Environment.
Working folders:

Status	Source Control Folder	Local Folder
Active	$/SampleTeamProject/Sample	C:\Workspaces\Production
	Click here to enter a new working folder	

008207



Workspace: ProductionWorkspace
Source location: $/SampleTeamProject/Sample
Local Path: C:\Workspaces\Production

Name	Pending Change	User	Latest
Sample			Yes
Sample.sln			Yes
Sample.vssscc			Yes

007024

4. Select the working project in **Source Control Explorer**, and select **Get Specific Version** and **Get by Label “Production”**. Refer to Getting an Element in Source Control for more information on getting specific version or getting by label.

   You can get a copy of the latest files in the local folder by running the following command in the command line:

   ```
   Import -m <model name> -vf <filename>
   ```

5. Open Visual Studio development environment, and then open the Solution or the Project file.

   The **Application Project Wizard** appears.

6. Select **New** or **Existing**.

   Note: *Select **New** if you want to associate an existing project, which is under source control, with a new database. Select **Existing** if you want to associate an existing project, which is under source control, with an existing database.*

- If you select **New**, perform the following:
  a. From the **Server Name** list, select a server name.
  b. In the **Database Name** box, enter a database name.
  c. In the **Model Name** box, enter a model name.
- If you select **Existing**, perform the following:
  a. From the **Server Name** list, select a server name.
  b. From the **Database Name** list, select a database name.

7. Click **Next**.
8. Click **Finish**.

   Note: *When you associate an existing project with a new database, the database is initialized by importing all the version files that are present in the project file. However, when you associate an existing project with an existing database, it is assumed that you are dealing with a multi-user environment and the database it up-to-date.*
9. Click **Build** to build and deploy the system.

Release Management

In addition to giving you a more meaningful way to refer to versions, version labels can be used to build a product release. Version labels allow you to do a Get or Checkout operation on all versions with a specified label. By assigning a release-specific label to the versions of all elements required for a release of your product, you can retrieve all of these elements in one operation.

To build a product release, perform the following:

1. Ensure that all versions of elements to be included in the release have been labeled with the version label identifying them for that release, for example WindowsRel2V1.
2. Perform a Get or Checkout operation to retrieve all versions with version label WindowsRel2V1 from the Source Control Bank.

When you perform the operation, you can load all the retrieved versions into the Model, then use Builder to define a System and generate directly to a target host. Refer to Building a Product Release by Labels for more information about release management.

Building a Product Release by Labels

Assigning Version Labels

A version label can be assigned to one version at a time, but to make labeling of a number of versions easier, Version Control also allows you to assign a label to:

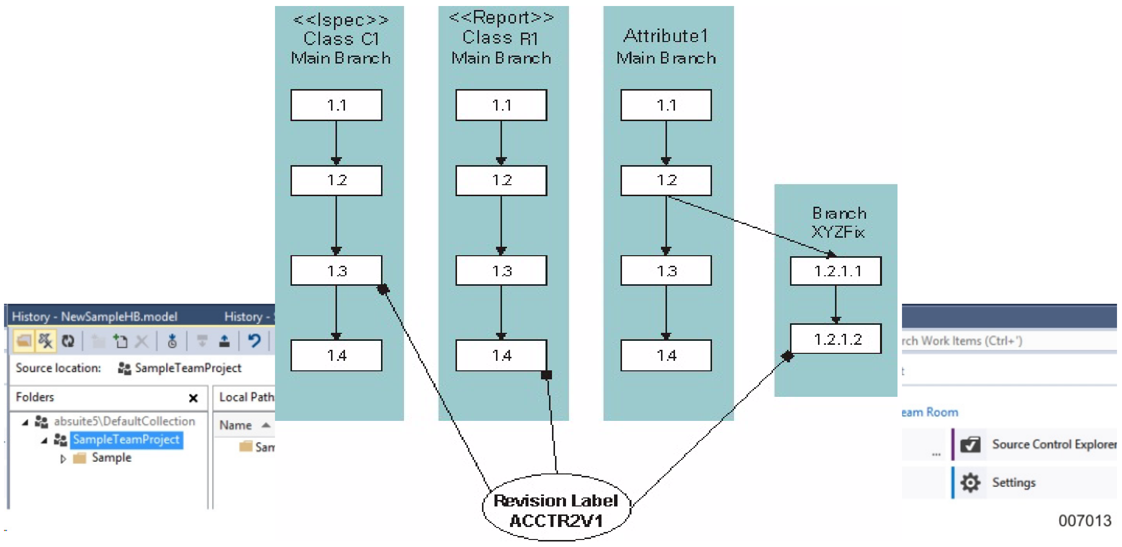
The current versions in the Model, for all elements or for selected elements in a Model.

The latest version for all elements or for selected elements in a Model.

Refer to the documentation provided with your source control tool for more information on how to assign labels.

Assembling the Release

To assemble the versions required for a release, perform a Get or Checkout on a Version by Label operation and load a copy of all versions with the specified version label into the Model. In the diagram below, version 1.3 for <<Ispec>> Class C1, version 1.4, for <<Report >> Class R1, version 1.2.1.2, for Attribute1 version 1.2.1.2 is retrieved to build the release.

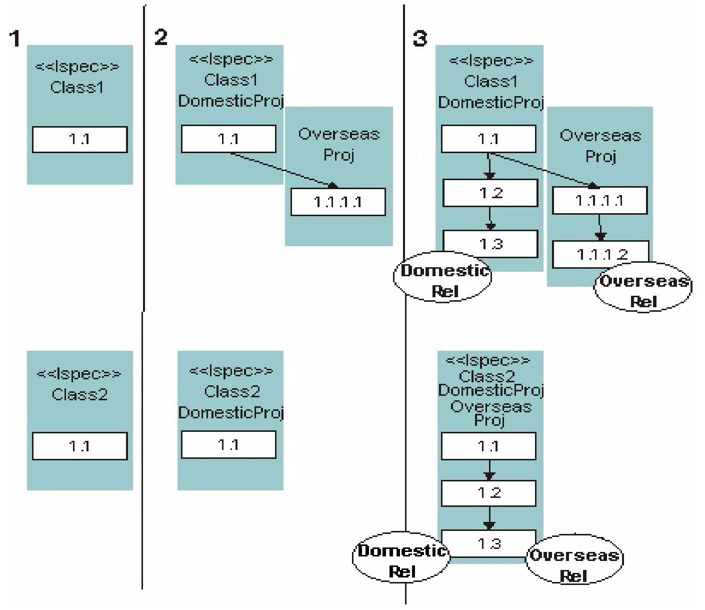


Delivering the Release

After the versions required for the release have been copied to the Model, use Builder to define a System and generate directly to the target host.

Example

The diagram below gives an example of two <<Ispec>> Classes that are added to the Source Control Bank and modified for two projects.



(1) <<Ispec>> Class1 and <<Ispec>> Class2 are added to the Source Control Bank, creating version 1.1 on a main branch for each element.

(2) A decision is taken to develop variants of <<Ispec>> Class1 for the Domestic Release of the product and Overseas release. Version 1.1 is checked out, rewritten for Overseas, and checked in to a new branch called OverseasProj. The main branch is given a branch label DomesticProj.

<<Ispec>> Class2 is the same for both Domestic Release and Overseas release. Its main branch is given two labels: DomesticProj and OverseasProj.

(3) <<Ispec>> Class1 and <<Ispec>> Class2 are modified.

Version label OverseasRel is assigned to <<Ispec>> Class1 version 1.1.1.2 and to <<Ispec>> Class2 version 1.3. A Get or Checkout by Label operation is performed to build Overseas Release 1.

Version label DomesticRel is assigned to <<Ispec>> Class1 version 1.3 and to <<Ispec>> Class2 version 1.3. A Get or Checkout by Label operation is performed to build Domestic Release 1.

Integrity Management

Integrity Management is an inherent feature designed to ensure that the Developer Model is synchronized with the Source Control Bank. This is accomplished by protecting controlled elements, that is, elements which have been added to the Source Control Bank. Protection is provided by identifying any controlled element that has not been checked out from the Source Control Bank as being read-only.

Note: *There is a performance overhead when using Integrity Management. The size of the impact depends on the number of dependencies on the element that is being changed, and the granularity of the versionable objects.*

Refer to Working with Integrity Management for more information about this feature.

Working with Integrity Management

Each time you perform an operation that might change a versionable element, Integrity Management checks to see what other elements might be affected by the change and takes remedial action.

Some examples of how Integrity Management affects Source Control are included below.

Updating

When modifying an element, which is not checked out, it is checked out automatically before any modification can be made. When an element is already checked out, Developer locks the element until the modification is complete. When other elements are impacted by this modification, they are also checked out automatically.

Checking Out

For a Check Out operation, you arenot prompted to confirm the overwrite of changed elements in the Model. The Check Out will go ahead and overwrite what is in the Model.

Adding

When a new element is added to the model, it is versioned by default as part of its parent, if the parent has the version file property set.

Deleting

When deleting an element, it checks out its parent if the parent has the version file property set, or if this child element has the version file property set. The version file in the source control tool isnot marked as deleted. It is your responsibility to perform regular purges.

Any dependent elements arealso checked out, as is the case with updates described earlier.

Automated Test Tool

Automated Test Tool (ATT) menu within System Modeler can be used to unit test Agile Business Suite (AB Suite) applications and AB Suite Client Framework applications. This tool is integrated with the AB Suite development environment. You can record, play back, and change the inputs to transactions at runtime to verify the changes in an AB Suite application. It is also used to verify the working of an application when there is a change in the AB Suite model or a model is migrated from one release of AB Suite to another.

You can use the System Modeler functions while an application is running and transactions are being performed. The transactions executed at runtime are dynamically recorded as a series of test steps.

ATT menu allows you to perform the following for each test step:

- View the data exchanged during a transaction and verify it at the time of recording.
- Pause the recording as transactions are executed. This is useful for large test runs, where it might be error prone to revisit a long test sequence and insert proper verifications.
- Play back test steps to validate the application.

The Visual Studio File menu lets you create a Unit Test Project to organize test cases. You can access the ATT menu from the Visual Studio menu bar to edit, validate, and play back tests. The Visual Studio Test menu provides the functionality required to debug and run test cases. Refer to References for more information on the ATT menu user interface.

Recording Test Cases

Before you start recording transactions with System Modeler, you need to perform the following:

1. Configure the application configuration file.

   In the application configuration file, you have to first enable recording and specify the mode of recording. Test cases can be recorded in the Connect or Disconnect mode. You can also specify the port number and the server IP used to connect to the runtime application.

2. Create a Unit Test Project.

   After you configure the application configuration file, you need to create a test case in a Unit Test Project to record the transactions. Refer to Creating a Unit Test Project for more information on creating a unit test project.

Configuring to Record

In addition to the default Winforms client, System Modeler supports recording with Component Enabler (CE) clients. Refer to Configuring Component Enabler Clients for more information on configuring the CE clients.

Configuring to Record with Winforms

1. Open the Winform configuration file used for your session. The default Winform configuration file is *Ispec.xml* in the location <Package Installation Directory>\<Solution Configuration>_<Solution Platform>\<Segment name>\Core\Bin.
2. Find the entry < ATTRecord ></ ATTRecord> and specify the following parameters:

Parameter	Description
ATTRecord	Specifies whether recording is enabled. Set this to true to enable recording.
ATTRecordMode	Specifies whether you want to record in the Connect or Disconnect mode.
ATTRecordServer	Specifies the socket server being used to connect to the runtime application. Configure this value based on your socket server setting.
ATTRecordPort	Specifies the Port being used to connect to the C# test project within System Modeler when recording a test in "connect" mode. This should correspond to the Listener Port Number defined within the global settings file, of your test solution in System Modeler.

Note: *The <xmlfile> </xmlfile> tag specifies the path and name of the recorded file when you record in the Disconnect mode.*

The following is an example of configuring the Winform config file, such as ispec.xml file to enable recording in the Connect mode with the Winforms client:

```
<ATTRecord enable ="true" mode="connect" server="127.0.0.1" port="8888">
<XmlFile>C:\Testcase.smtest</XmlFile>
</ATTRecord>
```

Creating a Unit Test Project

After you enable recording and set the recording mode, you must create a test case in a Unit Test Project. A test case is a collection of test steps and is saved with the .smtest file extension. The transactions executed at runtime are recorded as a series of test steps in a SMTest file. Test steps are listed in the order that they are recorded and indicate the type of transactions that were performed at runtime. Refer to Transaction Types for more information on the types of transactions that can be recorded.

Unit Test Project

A System Modeler Test Project is a test solution that holds the recorded and playback test cases. A Unit Test Project can contain multiple recorded and playback test cases. The Unit Test Project is listed in Solution Explorer. It is recommended that you create a new test solution when you start recording for the first time.

To create a new Unit Test Project in Microsoft Visual Studio 2015, perform the following:

1. Click **File** > **New** > **Project...**
2. From the left pane, select **Test** from the **Visual C#** project types in the **Add New Project** window.

   Note: *System Modeler supports only the C# unit test project types.*
3. In the New Project dialog box, select **.NET Framework 4.6.1** or higher from the drop-down list or you can change the Target Framework field after the unit test project is created. Refer to Step 7 to set the .NET Framework after the unit test project is created.

   Note: *You need to set the .NET Framework to 4.6.1 or higher for the current application to that ensure ATT works correctly.*
4. Select **Unit Test Project**.
5. Enter a project name in the **Name** field.

   This is the .sln solution file used to hold all the tests. The **Location** field is automatically populated.
6. Click **OK**.

   A Unit Test Project is created.

To change the .NET Framework after the Unit Test Project is created, right-click the Unit Test Project, select Properties, and modify the Target Framework field to .NET Framework 4.6.1 or higher.

Recording Test Cases in the Connect Mode

Ensure that you have configured the application configuration file and created a unit test project before you start recording transactions. Refer to Configuring to Record and Creating a Unit Test Project for more information.

Note: *The global settings file is automatically generated on performing System Modeler operations, such as recording or generating a C# file.*

In the Connect mode, start recording in System Modeler first and then launch the client application. You can refer to the recorded steps in the New SMTest window in this mode.

To record a new test case in the Connect mode, perform the following:

1. Modify the application configuration file and ensure that the recording Port Number is configured and the recording mode is set to Connect. Refer to Configuring to Record for more information.
2. Create a new Unit Test Project for the test case. Refer to Creating a Unit Test Project for more information.
3. From the ATT menu, select **Record** to start recording the transactions.

4. Start the client application and perform transactions. For every transaction, System Modeler adds a new test step to the SMTest case and records the input data.
5. From the ATT menu, select Stop when you complete the transactions to be recorded.

Configuring and Playing Back Test Cases

After you record a test case, you can play back the recorded test case. System Modeler establishes a connection to the host application and the test is processed by stepping through the recorded test steps sequentially. However, you must ensure that the system connection information is configured in the global settings file and is selected prior to playback.

Configuring for Playback

Before you play back using System Modeler, you need to configure the System Modeler environment and connection information of the client application to enable the System Modeler to communicate with the runtime system and client application. While configuring the System Modeler environment and connection information, you can specify the skipped fields, log details, and session details.

Notes:

For AB Suite applications

- *Ensure to generate the CE Bundle before you deploy the application for playback.*
- *Ensure that you create a view for the deployed client application via the Runtime Administration Tool.*

You must configure the following for playback in System Modeler:

- General
- Session Details
- Connection Details

These configuration options can be specified in the global settings file of the Solution Items. These settings are globally applicable to all the test cases in the Unit Test Project.

Notes:

- *For AB Suite applications, you must configure the Session details and for AB Suite Client Framework applications, you must configure the Connection details.*
- *For AB Suite Messenger Client applications, configuration of Session details and Connection details is not required.*

Configuring General Details

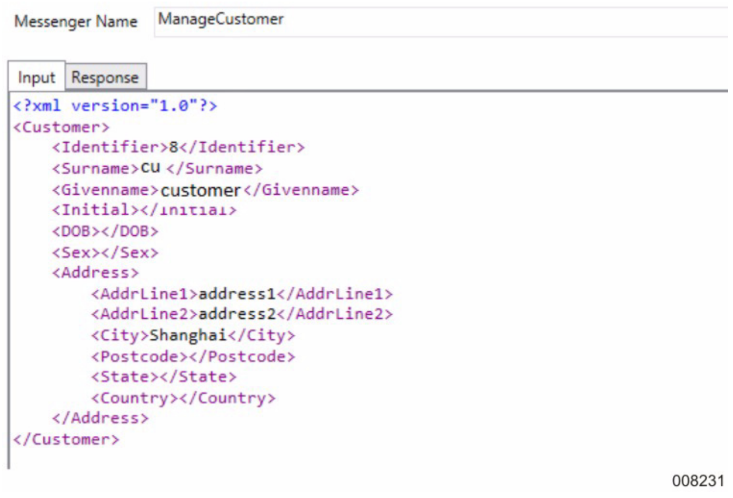
In the General Details tab, you can configure the common settings for Session and Connection. You must configure the Port Number and optionally you can specify the fields to be skipped before you start recording. The information for these fields are not recorded and are skipped during playback.

To configure the general details for test cases, perform the following:

1. In the **Listener Information** pane
   - In the **Port Number** field, enter the port used for playback.
2. In the Skipped Fields pane
   - In the **Name** field, enter the name of the field to be skipped.
   - In the **Application** field, enter the application name which the field belongs to.
   - In the **Description** field, provide a description of the skipped fields.

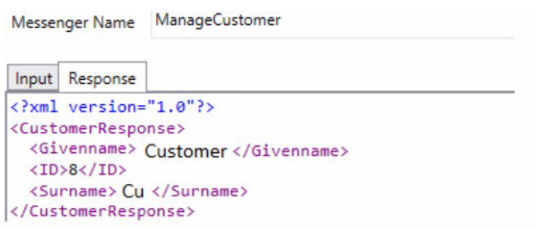
Notes:

- *The field names added to the skipped fields list are case-sensitive.*
- *If you want to skip the xml data element recorded using messenger client, you should specify the XPath structure in the Field Name field. Consider the following example:*



```
<?xml version="1.0"?>
<Customer>
    <Identifier>8</Identifier>
    <Surname>cu </Surname>
    <Givenname>customer</Givenname>
    <Initial></Initial>
    <DOB></DOB>
    <Sex></Sex>
    <Address>
        <AddrLine1>address1</AddrLine1>
        <AddrLine2>address2</AddrLine2>
        <City>Shanghai</City>
        <Postcode></Postcode>
        <State></State>
        <Country></Country>
    </Address>
</Customer>
```

008231



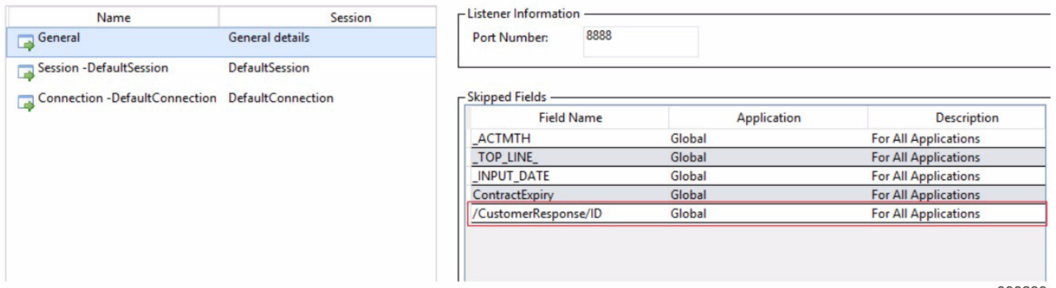
Messenger Name ManageCustomer

Input Response

```
<?xml version="1.0"?>
<CustomerResponse>
  <Givenname> Customer </Givenname>
  <ID>8</ID>
  <Surname> Cu </Surname>
</CustomerResponse>
```

008232

If you want to skip the xml element, **<ID>8<ID>**, you should enter /CustomerResponse/ID in the **Field Name** of the **Skipped Fields** pane as shown in the following. When you playback the recorded messenger client test case, the xml element ID value under CustomerResponse is ignored.



Name	Session
General	General details
Session -DefaultSession	DefaultSession
Connection -DefaultConnection	DefaultConnection

Listener Information

Port Number: 8888

Skipped Fields

Field Name	Application	Description
_ACTMTH	Global	For All Applications
_TOP_LINE_	Global	For All Applications
_INPUT_DATE	Global	For All Applications
ContractExpiry	Global	For All Applications
/CustomerResponse/ID	Global	For All Applications

008233

Configuring Session Details

In the Session Details tab, you can add multiple sessions and specify connection information for each session that is required to connect to the runtime system during playback. You must configure at least one session for playback.

To configure session details for test cases, perform the following:

1. Click **Session** in the left pane.
2. In the **Sessions** table, enter a name and description of a session in the **Name** and **Description** columns, respectively. You can define multiple sessions here.
3. In the **Session Details** pane
   - In the **Application** field, enter the name of the application deployed.
   - In the **Bundle** field, enter the name of the deployed Bundle. For example, order_entry.
   - In the **URL** field, enter the URL of the runtime server specified in the configuration file of the deployed application.
   - In the **View** field, enter the name of the View that is configured in the Administration Tool to connect to the client application.

     Note: *You can either select the value from the drop-down list or enter it manually.*

- In the **Package Prefix** field, enter the package prefix that was specified in the deployment properties. For example, com.unisys.
- In the **Username** field, enter the name of the Application User.
- In the **Password** field, enter the password for the Application User.
- In the **Domain** field, enter the domain of the Application User.
- In the **Timeout** field, enter the number of seconds for the client application to time out if the server does not respond to a transaction request within the specified time.
- In the **CE Output** field, select the CE output directory. By default, the value is 'C:\NGEN_CE\Classes'.
- Click **Test Connection** to test whether the connection has been established.

  Note: *A warning message appears if the connection fails.*

4. In the **General Logging** and **CE Logging (during playback)** panes:
   - Select the log level number from the **Level** list.

     Refer to the *Agile Business Suite Runtime for Windows® Operating System Administration Guide* for more information on log levels.
5. Enter the location and name of the log file in the **Filename** field.

Configuring Connection Details

In the Connection Details tab, you need to configure the connection information for Client Framework applications, which is required to connect to the runtime system during playback.

1. In the **Session Details** pane
   - In the **Session Name** field, enter the name of the connection.
   - In the **System** field, enter the name of the application deployed.
   - In the **Host** field, enter the host name used to deploy the application.
   - In the **Station Name** field, enter the station name used for connection
   - In the **LogLevel** field, enter the log level used to trace the test process.
   - In the **LogFolder** field, enter the default path present in the global settings file, to store the log files generated during test.
   - Click **Test Connection** to test whether the connection has been established.

     Note: *A warning message appears if the connection fails.*

   Note: *The Playback through Debugger is used to specify whether the connection is using a Debugger session. If checked, then the connection can playback through a debugger session. If not the connection cannot playback through a debugger session.*

Playing Back Test Steps

The recorded test steps generate C# test methods similar to the existing C# methods. You can play back the C# Test methods that have been generated for the recorded test cases. The playback submits the C# test method to the server and processes the results returned by the server.

The following playback options are available within the System Modeler:

- Playback using Visual Studio Environment
- Playback using Team Foundation Server (TFS)
- Playback using Command Line

Note: *The C# test cases can be configured and played back using the Microsoft Test Manager (MTM). Refer to the MSDN documentation for more information.*

Test Explorer Window

The SMTests in a unit test project can be viewed in the Test Explorer window. The Test Explorer window displays items that correspond to the C# test methods. You can group tests and initiate playback. The lower pane of the Test Explorer window displays a summary of the test results after a test executes. You can open the Test Output window and visit the link to the XML file to view details of the playback steps. Select the **CheckIn** option in the Team Explorer window.

Playback using Visual Studio

This feature allows you to play back a complete sequence in System Modeler. Perform the following in Microsoft Visual Studio 2015 to playback all the steps:

1. Configure the connection information in the global settings file.

   Refer to Configuring for Playback for more information on configuring for playback.
2. Click **Save** on the toolbar.
3. To build the solution, click **Build** on the **Build** menu.
4. Select **Test** > **Windows** > **Test Explorer** to open the Test Explorer window.
5. Select the C# test method from the test explorer window, right-click, and select **Run Selected Tests**.

A summary of the playback session is displayed in the lower pane of the Test Explorer window. If the playback fails, the System Modeler displays the reason for the failure of a step.

Playback using TFS

You can playback test steps by using TFS. To playback test steps from TFS, you must create a build definition.

Note: *Refer to Building Applications by Using TFS for more information on establishing a connection to TFS.*

Adding Solution to the TFS Server

You must add the solution to the TFS server to enable playing back test cases using TFS. To add the solution to the TFS server, perform the following:

1. Select the solution on the **Solution Explorer** window.
2. Right-click and select **Add Solution to Source Control**.

   The **Add Solution to Source Control** dialog box appears.
3. Click **Ok**.
4. Right-click the solution and select **Check In**.
5. Select the **CheckIn** option in the Team Explorer window.

   The solution gets added to the TFS server.
6. Create a build definition. Refer to Creating a Build Definition for information on creating a Build Definition.
7. Queue the created build definition. Refer to Using Build Definitions for information on Queueing the build definition.

Playback using Command Line Interface

You can play back an SMTest through the Visual Studio 2015 Command Prompt. The mstest command runs tests in test files. You can execute the playback command from the command line and view the outcome. The test file name and configuration file can be specified through arguments passed to the command. Refer to the MSDN documentation for more information on the mstest command.

Setting the Low Account Month

The Low Account Month (LAM) value is used to prevent an application from accepting a transaction with a date (contained in the built-in attribute ActMth) that is older than the date when the application was deployed. By default, the value is the year and month in which the application was initiated.

An error that the Account Month is not open is displayed if you play back tests that were recorded before the current system was deployed. It is recommended that you set the LAM to a value that is less than the LAM of such tests.

To set the Low Account Month for the Windows® Runtime, perform the following:

1. Open the **Administration Tool**.
2. Right-click the deployed system, click **All Tasks** > **Configure...**
3. On the **Customize** page, enter a value in the **Low** field in the **High and Low Account Months** pane.
4. Click **OK**.

To set the Low Account Month (LAM) for the MCP Runtime:

Ensure that you are logged on to the system from the MCP host with the Controller security access level.

From the home position of page 2 of the session where you are logged onto your system,

Enter :**LAM [mmyy]**

For example, :LAM 9806 sets the Low Account Month to June 1998. If no value is entered, the current value of LAM is displayed.

Refer to Administration Commands in the *Agile Business Suite Runtime for ClearPath MCP Administration Guide* for more information on setting the Controller security level.

Migrating Test Case Recordings

The Recordings created with AB Suite 2.0, 3.0, 4.0, or 5.0 can be migrated to AB Suite 6.1 by adding the existing files to a C# test project.

Migrating Test Cases and Test Suites to AB Suite 6.1

PTo migrate a test case or a test suite to AB suite 6.1, perform the following:

1. Right-click the Unit Test Project in the Solution Explorer view and click **Add** > **Existing Item...**

   The Add Existing Item dialog box is displayed.

2. Browse and select the test cases or test suite you want to migrate, and click **Add**.

   The Test cases or test suite is migrated and an SMTest file is automatically generated. The SMTest file appears in Solution Explorer.

   Notes:

   - *To generate the C# file, you need to select the SMTest file and click **Save**. The C# files are generated automatically.*
   - *Multiple test cases can be migrated simultaneously.*

References

ATT Menu User Interface

The ATT Menu User Interface comprises of the ATT menu options:

Note: *The ATT menu is available on the Visual Studio menu bar once you create a System Modeler Test Project.*

ATT Menu Options

The ATT menu is available on the Visual Studio menu bar. The options available on the ATT menu let you navigate through all the functions of System Modeler ATT. The ATT Menu provides access to the various System Modeler ATT functionalities. The following table describes the menu options:

Option	Icon	Description
Record	●	Starts recording a new test case and System Modeler ATT creates a new SMTest case, and adds the recorded step to it.
Pause/Resume	⏸	Pauses or Resumes recording of test steps.
Stop	■	Stops recording of test steps and generates the related SMTest files.
Cancel	⊘	Cancels recording of test steps.

Configuring Component Enabler Clients

System Modeler supports recording with the following Component Enabler clients:

- VB.Net winforms
- ASP.Net webforms
- Presentation Client
- WPF Client
- Messenger Client

Note: *It is assumed that you have generated a Component Enabler (CE) bundle and deployed the runtime application. The CE bundle is required for playback. Refer to the* Agile Business Suite Component Enabler User Guide *for more information on creating application bundles for different clients.*

Configuring to Record with VB.Net Winforms

1. Open the *config.xml* file from the \bin folder under the VB.NET client project directory, for example, C:\NGEN_CE\ VB.NETClient\Sample\bin
2. Find the entry < ATTRecord ></ ATTRecord> in the config.xml file and specify the following parameters:

Parameter	Description
Enable	Specifies whether recording is enabled. Set this to true to enable recording.
Mode	Specifies whether to record in the Connect or Disconnect mode.
Server	Specifies the socket server used to connect to the runtime application. Configure this value based on your socket server setting.
Port	Specifies the Port used to connect. Configure this value based on your port setting.

Note: *The <xmlfile> </xmlfile> tag specifies the path and name of the recorded file when you record in the Disconnect mode.*

The following is an example of configuring the config.xml to record in the Disconnect mode with the VB.Net client.

```
<ATTRecord enable ="true" mode="disconnect" server="127.0.0.1" port="8888">
<XmlFile>C:\NGEN_CE\VB.NET Client\Sample\bin\TestCase.smtest</XmlFile> </ATTRecord>
```

Configuring to Record with ASP.NET Web Forms

1. Open the *Web.Config* file from the Views folder of the ASP.NET client project directory.
2. Find the entry <Plugin> </Plugin> and specify the following parameters:

Parameter	Description
ATTRecord	Specifies whether recording is enabled. Set this to true to enable recording .
ATTRecordMode	Specifies whether you want to record in the Connect or Disconnect mode.
ATTRecordServer	Specifies the socket server used to connect to the runtime application. Configure this value based on your socket server setting.
ATTRecordPort	Specifies the Port used to connect to the runtime application. Configure this value based on your port setting.
ATTRecordXMLFilePath	Specifies the path and name of the file in which the test case is recorded when the record mode is set to Disconnect.

The following is an example configuration in the Web.Config file to record in the Connect mode with the ASP.Net client:

```
<add key="ATTRecord" value="true"/>
<add key="ATTRecordMode" value="connect"/>
<add key="ATTRecordServer" value="127.0.0.1"/>
<add key="ATTRecordPort" value="8888"/>
<add key="ATTRecordXMLFilePath" value="C:\NGEN_CE\TestCase.smtest"/>
```

Configuring to Record with a Java-Based Presentation Client

There are two ways to configure the Java-based Presentation Client.

- Open and edit the *config.xml* file directly using a text editor.
- Open the Configuration Assistant and edit the configuration file. The Configuration Assistant is installed with the Component Enabler (CE) Development Environment with the path, C:\NGEN_CE.

Note: *Although the Presentation Client generates Java IspecModel files to record, always ensure that C# IspecModel files are generated to play back the recorded files. Since C# IspecModel files are used to play back, it is a good practice to generate Java and C# IspecModel files to avoid any problem.*

Using the Configuration File

Find the entry < ATTRecord > </ ATTRecord> and set the following parameters:

Parameter	Description
enable	Specifies whether recording is enabled. Set this to true to enable recording.
mode	Specifies whether you want to record in the Connect or Disconnect mode.
server	Specifies the socket server being used to connect . Configure this value based on your socket server setting.
port	Specifies the Port being used to connect to. Configure this value based on your port setting.
xmlFile	Specifies the path and name of the file in which the test case is recorded when the record mode is set to Disconnect.

Note: *The <xmlfile> </xmlfile> tag specifies the path and name of the recorded file when you record in the Disconnect mode.*

The following is an example of the configuration set in config.xml to enable recording in the Connect Mode with the Presentation Client:Using the Configuration Assistant

```
<attRecord>
<enable>true</enable>
<mode>connect</mode>
<server>127.0.0.1</server>
<port>8888</port>
<xmlFile>c:\temp\testcase.smtest</xmlFile>
</attrecord>
```

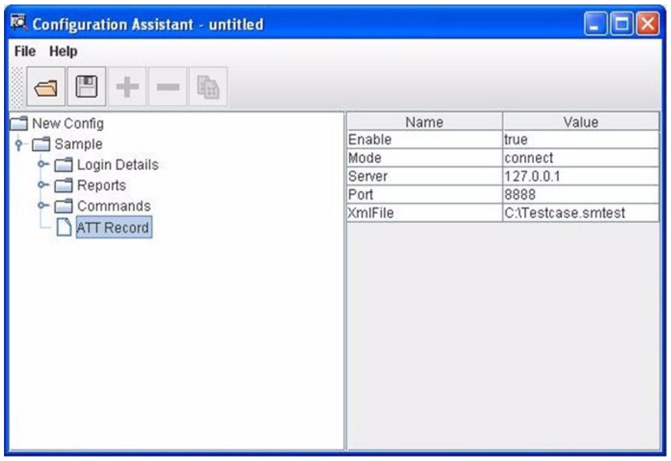
The Configuration Assistant is installed with the Component Enabler Development Environment and lets you edit the application configuration file.

1. Point to the bottom-left corner of the screen for Windows Server 2012 R2 and click the **Start**.

   The Start screen appears with the list of applications.

2. Click **Configuration Assistant** or type Configuration Assistant on the desktop.

   You can also right-click the Start screen and click **All Apps** for the list of applications.



Name	Value
Enable	true
Mode	connect
Server	127.0.0.1
Port	8888
XmlFile	C:\Testcase.smtest

Configuring to Record with a WPF Client

1. Open the *WpfClient_Config*.rtxml file from the Access Layer API Deploy folder.
2. Find the entry < ATTRecord ></ ATTRecord> and specify the following parameters:

Parameter	Description
ATTRecord	Specifies whether recording is enabled. Set this to true to enable recording.
ATTRecordMode	Specifies whether you want to record in the Connect or Disconnect mode.
ATTRecordServer	Specifies the socket server being used to connect to the runtime application. Configure this value based on your socket server setting.

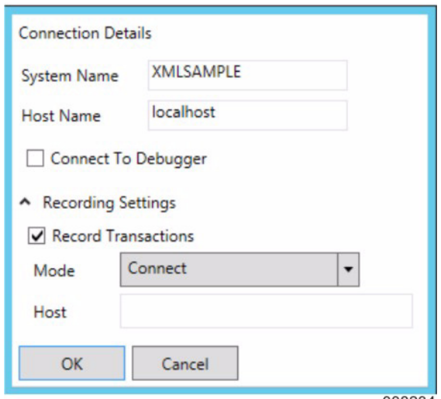
Parameter	Description
ATTRecordPort	Specifies the Port being used to connect to the runtime application. Configure this value based on your port setting.
File	Specifies the path and name of the recorded file when you record in the Disconnect mode.
RecordDynamicLists	Specifies whether the lists sent through the SendDynamic command can be recorded. Set this to true to enable recording of the lists sent through this command.

The following is an example configuration in the WpfClient_Config.rtxml file to record in the Disconnect mode with the WPF client:

```
<ATTRecord enable ="true" mode="disconnect" server="127.0.0.1" port="8888">
<File>C:\Temp\Recordings\Testcase.smtest</File>
<RecordDynamicLists>True</RecordDynamicLists>
</ATTRecord>
```

Configuring to Record with Messenger Client

1. Open the Messenger Client.


Connection Details

System Name: XMLSAMPLE

Host Name: localhost

☐ Connect To Debugger

Recording Settings

☑ Record Transactions

Mode: Connect

Host:

008234

2. Enter the following parameters.

Parameters	Description
Record Settings	
Record Transactions	Select this option to set record settings.
Mode	Specifies whether to record in the Connect or Disconnect mode.
Host	Specifies the host used to connect when you record in the Connect mode.
File Name	Specifies the path for the recorded file when you record in the Disconnect mode.

Recording Test Cases in the Disconnect Mode

Recording in the Disconnect mode is optional. It is simple and useful if you want to record directly without starting Visual Studio. You can execute transactions on the client applications which are then recorded in the XML file specified in the configuration file. You can later edit the test case in System Modeler Test List Editor. The recording progress is not visible in System Modeler.

To record a test case in the Disconnect mode, perform the following:

1. Modify the application configuration file and ensure that the recording mode is set to Disconnect. Refer to Configuring to Record for more information.
2. Start the client application and perform transactions on the application.
3. Close the session or directly close the application.

System Modeler automatically creates a test case file that records the transactions and input information as test steps. The test file is saved in the location specified in the application configuration file.

Note: *Multiple sessions cannot be recorded for a client application.*

Using the SampleTest Scripts

The SampleTest application enables you to run the SampleTest scripts placed within the AB Suite 6.1 Bin directory.

To run the sample scripts, perform the following:

1. Restore the SampleTest application by performing the following:
   a. On the **File** menu, point to **New**, and then click **Project**.

      The **New Project** dialog box appears.

   b. In the Project Types pane, expand **Templates**, expand **System Modeler**, and then select **AB Suite Application from Backup**.

      The System Modeler Project wizard appears.

   c. In the Selection field, click the **...** button to browse for the batch file.

      Note: *The SampleTest.bck file is present in the following location:*

      C:\Program Files (x86)\Unisys\AB Suite 6.1\Bin\

   d. Click **Next**.

      The project creation confirmation message appears in the wizard.

   e. Click **Finish**.

      The sample application is restored.

   f. Click **Close**

      The sample AB Suite SampleTest application appears in the Solution Explorer window.

2. To build the SampleTest application, in the Solution Explorer window, select the SampleTest application, right-click and select **Build**. Alternatively, on the **Build** menu, select **Build**.
3. To playback the test scripts in the Test Explorer window, right click the test scripts and select **Run Selected Tests**.

   Note: *Before you can run the test scripts, ensure that the WPF client is installed.*

   The test result appears in the lower pane of the Test Explorer window.

You can backup any System Modeler recording as a test project (.bck file) and then restore it into a System Modeler environment on a different machine. Refer to Performing Backup of AB Suite Solutions and Restoring the AB Suite Solutions for more information.

Transaction Types

System Modeler Test List Editor

An SMTest is displayed in a New SMTest window. Each node represents a transaction executed on the runtime system via the client. A test step is recorded in the same order it occurs, the latest one being the last node

There are three types of test steps based on the type of transactions:

- Colon Command
- SubmitIspec
- SelectIspec

For each test step, the details of the transaction are recorded. The details include information, such as Ispec name, status messages, return codes, field names and their values, and test comments. These details can be viewed and edited according to the test scenario requirement.

Colon Command

- This section describes the details for a colon command. Every colon command records information under five categories, as follows:Name – The name given to the test step that consists of the colon command.
- Send Data – Records the colon command executed.
- Received Data – Records the result of the executed colon command.
- Validation Data – Conditions that specify how the colon command is validated.
- Documentation – Describes the colon command.

Validation Data

Information	Description
Condition	You can specify the validation condition using this option. Depending on the test requirement, you can set the validation condition to one of the following: • '=' – Checks if the input is equal to the value specified in the expected value field. • '<>' – Checks if the input is not equal to the value specified in the expected value field. • 'Include' – Checks if the input contains the string specified in the expected value field.
Received Messages/ Accepts	This is the expected value against which all subsequent playback values are checked for when the test case is run.

SubmitIspec

This section describes the details for a SubmitIspec test step. Every SubmitIspec test step records information under five categories:

- Name – The name given to the SubmitIspec test step.
- Send Data – Records the details of the previous SubmitIspec step executed.
- Received Data – Records the details of the current SubmitIspec step executed.
- Validation Data – Conditions that specify how the SubmitIspec is validated.
- Documentation – Describes the SubmitIspec test step.

A detailed description of the fields in each category is given below.

Send Data

Category	Information	Description
General Result	Name	This is the name of the previous SubmitIspec or SelectIspec Selection test step.
Ispec details	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name.
	Comments	You can add comments against each of the displayed fields.

Received Data

Category	Information	Description
General Result	Name	This is the name of last Ispec of current SubmitIspec test step.
	Status	This holds status line messages and errors passed back to the client by the host.
	Errors	These are system-generated error messages added as part of the LDL+ logic.
Ispec details	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name.
	Comments	You can add comments against each of the displayed fields.

Validation Data

Category	Information	Description
General Validation	Name	This is the name of the Ispec processed in the current SubmitIspec step.
	Status	This holds the expected status line messages and errors.
	Errors	This holds the expected errors.
	Copy From Data	This displays the copy ispec.
Ispec details	Skip	Select this check box to ignore this field during playback.
	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name. This is the expected value for playback. The condition for comparison is determined by the value set in the Condition column.
	Condition	This column specifies the validation condition. Depending on the test requirement, you can set the validation condition to one of the following: • '=' – Checks if the input is equal to the value specified in the value column. • '<>' – Checks if the input is not equal to the value specified in the value column. • 'Include' – Checks if the input contains the string specified in the value column.

Note: *The values displayed in the Ispec details fields are the values as recorded in the database. They do not display the values as presented on the presentation.*

For example, consider a list box with the following screen values:

Screen Values	Stored in Database as...
Asia Pacific	AP
North America	NA
South America	SA
Europe	EU

The values displayed in the Ispec details table are as stored in the database.SelectIspec - Transaction Details Area

This section describes the details for a SelectIspec selection test step. Every test step records information under five categories:

- Name – Name given to the SelectIspec test step.
- Send Data – Records the details of the previous SelectIspec step executed.
- Received Data – Records the details of the current Ispec processed in the test step.
- Validation Data – Conditions that specify how the SelectIspec step is validated.
- Documentation – Describes the SelectIspec test step.

A detailed description of the fields in each category is given below:

Send Data

Category	Information	Description
General Result	Name	This is the name of the previous SelectIspec test step.
Ispec details	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name.
	Comments	You can add comments against each of the displayed fields.

Received Data

Category	Information	Description
General Result	Name	This is the name of the current Ispec processed in the test step.
	Status	This holds any status line messages and errors passed back to the client by the host.
	Errors	These are system generated error messages added as part of the LDL+ logic.

Category	Information	Description
Ispec details	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name.
	Comments	You can add comments against each of the displayed fields.

Validation Data

Category	Information	Description
General Validation	Name	This is the name of the Ispec processed in the current SubmitIspec step.
	Status	This holds the expected status line messages and errors.
	Errors	This holds the expected errors.
	Return Code	This holds the expected return code.
Ispec details	Skip	Select this check box to ignore this field during playback.
	Property	This lists all the fields contained within the Ispec.
	Value	This displays the value corresponding to the Field Name. This is the expected value for playback. The condition for comparison is determined by the value set in the Condition column.
	Condition	This specifies the validation condition. Depending on the test requirement, you can set the validation condition to one of the following: • '=' – Checks if the input is equal to the value specified in the value column. • '<>' – Checks if the input is not equal to the value specified in the value column. • 'Include' – Checks if the input contains the string specified in the value column.

Note: *The values displayed in the Ispec details fields are the values as recorded in the database. They do not display the values as presented on the presentation.*

For example, consider a list box with the following screen values:

Screen Values	Stored in Database as…
Asia Pacific	AP
North America	NA
South America	SA
Europe	EU

Appendix A
References

This section includes information about System Modeler, using the sample AB Suite applications, using the Logic Editor window, using the Microsoft build platform, a hierarchical list of error messages, and other reference documents that apply to developing applications in an Integrated Development Environment (IDE).

Sample AB Suite Applications

You can create new AB Suite projects by using the sample AB Suite application or the sample AB Suite Client Framework application.

Creating a Project with a Sample AB Suite Application

To create a project with a sample AB Suite application, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.
2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Samples**.
3. In the Templates pane, select **Sample Application**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to store the new project.
6. Click **OK**.

   The New Application Wizard appears.
7. From the **Server Name** list, select the SQL server. By default, this field displays the system name.
8. In the **Database Name** box, enter the model name. By default, this field displays the solution name.
9. In the **Model Name** box, enter the model name. By default, this field displays the solution name.
10. Click **Next**.

    The project creation confirmation message appears in the wizard.

11. Click **Finish**.

    The sample AB Suite application is restored.

12. Click **Close**.

The sample AB Suite application appears in the Class View window.

Creating a Project with a Sample AB Suite Client Framework Application

To create a project with a sample AB Suite Client Framework application, perform the following:

1. On the **File** menu, point to **New**, and then click **Project**.

   The **New Project** dialog box appears.

2. In the Project Types pane, expand **Templates**, expand **Agile Business Suite**, and then click **Samples**.
3. In the Templates pane, select **Sample Client Framework Application**.
4. In the **Name** box, enter the project name.
5. In the **Location** box, enter the path or browse to the location where you want to store the new project.
6. Click **OK**.

   The New Application Wizard appears.

7. From the **Server Name** list, select the SQL server. By default, this field displays the system name.
8. In the **Database Name** box, enter the model name. By default, this field displays the solution name.
9. In the **Model Name** box, enter the model name. By default, this field displays the solution name.
10. Click **Next**.

    The project creation confirmation message appears in the wizard.

11. Click **Finish**.

    The Sample AB Suite Client Framework application is restored.

12. Click **Close**.

The sample AB Suite Client Framework application appears in the Class View window.

Describing AB Suite Icons

The following table describes and illustrates AB Suite 6.1 icons:

Icon Name	Icon
Add New Item	
Attribute New (Template-based)	
Attribute Private (Template-based)	
Attribute Protected (Template-based)	
Attribute Public (Template-based)	
Attribute Unresolved (Template-based)	
Attribute New (Inherits-based)	
Attribute Private (Inherits-based)	
Attribute Protected (Inherits-based)	
Attribute Public (Inherits-based)	
Attribute Unresolved (Inherit-based)	

Icon Name	Icon
Build Preview	
Class New	
Class Private	
Class Protected	
Class Public	
Class Unresolved	
CopyEvent New	
CopyEvent Private	
CopyEvent Protected	
CopyEvent Public	
CopyIspec New	
CopyIspec Private	
CopyIspec Protected	

Icon Name	Icon
CopyIspec Public	I☰
Dependencies Tab	⇄
Diagram	品
Diagram New	✳品
Diagram Unresolved	品?
Dictionary	🟪
Dictionary Open Status	📂
Documentation Tab	📄
Event New	✳E
Event Private	E🔒
Event Protected	E★
Event Public	E
External Class	▣

Icon Name	Icon
Folder	📁
Folder Open Status	📂
Folder Unresolved	📁
Frame New	
Frame Private	
Frame Protected	
Frame Public	
Group New	
Group Private	
Group Protected	
Group Public	
Inheritance Tab	
Insertable New	
Insertable Private	

Icon Name	Icon
Insertable Protected	
Insertable Public	
Interface	
Ispec New	
Ispec Private	
Ispec Protected	
Ispec Public	
Language Tab	
List	
Logic Tab	
Logic Status Tab	
Location	
Location New	

Icon Name	Icon
Location Unresolved	◍
Members Tab	☰
Members Parameter Tab	◆
Members Variable Tab	●
Messenger	⇄
Method Overrides Tab	⬡
Method New	⬡
Method Private	⬡
Method Protected	⬡
Method Public	⬡
Method Unresolved	⬡
Model	●
Output Manager Report Wizard	☰

Icon Name	Icon
Painter Tab	
Parameter (Template-based)	
Parameter (Inherits-based)	
Parameter Unresolved (Template-based)	
Parameter Unresolved (Inherits-based)	
Presentation	
Primitive	
Profile Conditions Tab	
Profile Data Tab	
Profile Keys Tab	
Profile New	
Profile Private	
Profile Protected	

Icon Name	Icon
Profile Unresolved	
Project	
Properties Tab	🔧
Quick Navigation	
Reference	
Reference New	
Report New	
Report Private	
Report Protected	
Report Public	
Search	
Segment	
Serialization	
SQL Script New	SQL

Icon Name	Icon
SQL Script Private	SQL🔒
SQL Script Protected	SQL★
SQL Script Public	SQL
Synchronize	⇆
Teach	▤
Teach New	▤
Teach Unresolved	▤
UML Editor Tab	🖧
Unresolved Tab	☰
Value	▪
Value Checking Tab	▪
Variable (Template-based)	▪
Variable (Inherits-based)	⬢

Icon Name	Icon
Web Service	
XSD Import	

System Modeler

This section of System Modeler provides additional information about some of the user interface elements of the integrated development environment (IDE).

Searching an Element

You can search within a project for named elements that match a string, regular expression or wildcard expression. The search can be performed through the entire project, or restricted to a selected element or a selected element and its members. The search is performed from the Search Dialog Box in which you enter the search parameters.

Note: *Search can also be started from the Members list of any element. When started from the Members list, multiple items can be selected.*

The search results are displayed in the Search Results List. From this window a found item can be displayed in its default editor window.

To initiate a search, perform the following:

1. Open the Class View window by selecting **Class View** from the View menu.
2. Sort the hierarchical list into the most appropriate form to locate the element you want to use. If necessary, click the plus (+) symbol next to a node to list all the elements within the node.
3. Select the element.
4. From the **Edit** menu, select **Search**, or use the right-click context menu and select **Search**.

Note: *You can also use the Visual Studio Find and Replace functions to initiate a search. Refer to the* MSDN documentation *for using the Visual Studio Find and Replace functions.*

Search Dialog Box

Use the Search Dialog Box to enter the parameters of the search using the fields described below. When you have entered the parameters click **Search** to begin the search.

Find what – Enter the text, along with wildcards or regular expressions, for which you want to search. Alternatively, select the drop-down list to display the last items entered. Selecting one of these sets other fields, except the In: field, to the values previously used.

Wedge button – View a quick reference list of Wildcard or Regular Expression syntax. Selecting an item in this list inserts the syntax element in the Find what field.

Match case – Search only for occurrences that match the combination of uppercase and lowercase characters you enter in the Find what field.

Match whole word – Search only for whole words rather than matching the text entered in the Find what field as it occurs within words. If checked, the matches are only words delimited by white space or some punctuation (such as a full stop, comma or mathematical symbol).

The following example illustrates the usage of the Match whole word option:

Suppose you want to search ‘Attribute1 :abc’ string in the Logic Editor, in which the word to be searched includes a space before ‘:’.

If you select the Match whole word option, and then search for ‘Attribute1’, the Search Results List window displays only the Attribute1 string in the Logic Editor.

If you clear the Match whole word option, and then search for ‘Attribute1’, the Search Results List window displays both the Attribute1 string in the Logic Editor and the element Attribute1.

Therefore, if you want to search the whole name of an element, select the Match whole word option for an accurate search.

Use – Specifies that certain characters entered in the Find what field match text patterns. The exact interpretation depends on which pattern-matching syntax is selected in the edit box adjacent to the Use check box, either Wildcards or Regular Expressions. The edit box is enabled only when the Use check box is checked.

- Wildcards – Specifies that the use of certain characters in the Find what text box represent a class of character or sequence of characters. Refer to Wildcards for a complete list.
- Regular expressions – Specifies that the use of certain characters in the Find what field represent notations for patterns of text rather than the literal character. Refer to Regular Expressions for a complete list.

After selecting Wildcards, or Regular expressions, click the wedge button to the right of the selection to display a context menu, which lists some of the more common options that you can insert.

Recursive – Specifies whether to search in the sub elements.

Look In – Allows you to define the scope of the search. This can be “Selected Object” or “Solution.”

For – Lists matched elements. The checked types are the ones that are matched. If you have requested a recursive search, the search looks in all child elements of the selected element.

In – Lists properties of elements in the Model that can be searched. The checked types are ones that are searched. The properties able to be searched are; Alias, Superclass, Names, Logic, Documentation, Descriptions, Inheritance, Values.

Buttons

Search Button – Begin the search

Close Button – Close the dialog box

Stop Button – Stop a currently running search

You can stop the search at any time by opening the Search dialog and pressing the Stop button. When the search begins the Search dialog box closes and the Search Results List window opens and displays the matched results.

Wildcards

The following expressions can replace characters or digits in your search string and are the same expressions as used by Visual Studio wildcards.

Note: *You must select the Use check box in the Search dialog box and select Wildcards before using any of the following expressions as part of your search criteria.*

Expression	Syntax	Description
Any single character	?	Matches any single character.
Any single digit	#	Matches any single digit. For example, 7# matches numbers that include 7 followed by another number, such as 71, but not 17.
Characters not in set	[!]	Matches any one character that is not specified in the set. For example, 7[abc] matches with 7a, 7b or 7c, but not 71.
One or more characters	*	Matches any one or more characters. For example, new* matches any text that includes “new”, such as newfile.txt.
Set of characters	[ ]	Matches any one of the characters specified in the set. For example, 7[abc] matches with 7a, 7b or 7c, but not 71 or 7d, 7[a-c] produces the same match.

Regular Expressions

Regular expressions are a concise and flexible notation for finding and replacing patterns of text. The regular expressions used within System Modeler are a subset of the expressions used in Visual Studio, with a simplified syntax.

You can use the following regular expressions in the Search dialog box to refine and expand your search. These expressions can be used to match characters or digits in your search string within Development Environment.

Note: *You must select the Use check box in the Search dialog box before using any of the following expressions as part of your search criteria.*

Complete Character List

Character	Description	Example
.	(Period) Any single character.	b.d matches 'bed' and 'bad' but not "bead".
*	None or more of the preceding characters or expressions.	a*ck matches 'luck', 'back' and 'barrack' but not 'brace'.
+	At least one or more of the preceding characters or expressions.	a+ck matches 'back' and 'barracks' but not 'luck'.
^	The beginning of a line.	^luck matches the word 'luck' only when it appears as the first set of characters in a line of the editor.
$	The end of a line.	luck$ matches the word 'luck' only when it appears as the last set of characters possible at the end of a line in the editor.
<	The beginning of a word.	<ba matches words such as 'bad' and 'back' that begin with the letters 'ba'.
>	The end of a word.	ck> matches words such as 'luck' and 'back' that end with the letters 'ck'..
( ) or { }	Any sequence of characters between the braces.	(very)+good finds 'verygood', "veryverygood"and so on. Note that it isnot found 'vgood', 'ygood', or 'vergood' because the sequence 'very' is not in any of those strings.
|	Matches either the expression before or the one after the OR symbol (|). Mostly used in a group.	(sun|mon)day matches 'sunday' and "monday".
[ ]	Any of the characters contained in the brackets, or any of the ASCII range of characters separated by a hyphen (-).	b[aeiou]d matches bad, bed, bid, bod, and bud. r[eo]+d matches red, rod, reed, rood, reod and roed. x[0-9] matches x0, x1, x2, and so on.
[^]	Any character except this following the caret (^) character in the brackets, or any of the ASCII range of characters separated by a hyphen (-).	x[^0-9] matches xa, xb, xc, and so on but not x0, x1, x2, and so on.
:b	Any white-space character. The :b finds tabs and spaces.	Good:bday matches the phrase 'Good day' in text but not Goodday.
:z	Any unsigned decimal integer.	:z matches any integer, such as '2', '567', '99' etc. but not 'luck'. Equivalent to- [0 - 9]+.

Character	Description	Example
:i	Any LDL+ identifier.	:i matches any LDL+ identifier like ‘amount1’, ‘amount_$’ etc. but not ‘1amount’. Equivalent to- [a-zA-Z_$][a-zA-Z0-9_$]*
:q	Any quoted string.	:q matches ‘luck’ and 'luck' but not the 't of don't. Equivalent to- ((\”[^\”]*\”)|('[^']*')).
\	Removes the pattern match characteristic in the Find What text box from the special characters listed above.	100$ matches 100 at the end of a line, but 100\$ matches the character string 100$ anywhere on a line.

Search Results List Window

The Search Results List window displays all found items. For each item found, the results list displays:

- Item full name – This includes the full path so that the user can find the item to open it manually.
- Item type or class.
- Item sub-type or part of main item in which the search string was found, for example, Name, Documentation, Logic.
- Number of occurrences found for the element.
- Line number in which the match was found, for each match. This applies only to Logic and similar text.
- Line of text in which the match was found. This applies only to Logic and similar text. Only one match is shown per line.

The details listed are displayed for each part of the item. The items are displayed in a tree list, where the root of each tree is the found item. The parts of the item that matches are displayed as branches of the root.

If the system security setting determines that you do not have access to the item of the search, the list displays only the “Item full name” and “Number of occurrences found”. Additionally, you arenot able to open the item for editing.

The title bar of the Search Result window indicates the status of the search. It shows the search term, whether the search is running or complete, and the total number of matches found.

The Search Results window is a toolbar window. It can be docked or positioned as required. A new window is used for each search that is made.

Note: *If an item in the list is changed after the search has been completed so that it no longer matches the search criteria, the list does not automatically refresh to remove it.*

When the search is complete you can perform the following operations:

- Refresh the list to rerun the search from the View, Refresh menu item.
- Select an entry in the list and double-click it to open the item in its default editor. The editor displays the pane containing the match and, where appropriate, select the matched text.

Building Comment Pages

You can create a detailed report of your Agile Business Suite model using the Build Comment Pages function. This function allows you to create a hierarchical written display of your Visual Studio Agile Business Suite project.

Using this function, you can either:

- Build the report of the entire model. Or
- Build a report on individual elements or classes.

  Refer to Select a Scope for more information.

Whichever method you chose you receive a report which lists the base class, all subclasses and their members. The report identifies the name of the element, any description that you have previously provided for it, along with its member visibility and the base class to which it belongs. In addition, the report includes the security details of an element in the generated output.

The report is displayed in the Visual Studio designer window and saved as an HTML file in a folder of your choice.

From the Build Comment Pages…. dialog box, you can select the following details of an element to include them in the report:

- Property, Translations and Configuration details
  - Configurations are only included for those elements that have configuration details defined.
- Logics of Methods and Profiles
- Diagrams that show a UML class diagram of an element and its classes
- Documentation that represents the textual information included in the Documentation tab of each element

- Statistics that shows the count of each element type, such as attributes, folders, and classes, present in the model
- Security details of an element are enabled if the AccessControlled property is set to "True" for a model

A report also contains hyperlinks of properties that reference another element, such as Owner and Inherits.

When you build a comment page, it generates a report on the Agile Business Suite model.

To build a report on your Agile Business Suite model, perform the following:

1. Select the Model in Class View.
2. Select **Build Comment Pages...** from the **Build** menu.

   The **Build Comment Pages...** dialog box appears.
3. Perform one of the following in the Build Comment Pages… dialog box:
   - To build a report on the entire model, select the Entire Model option. However, by default, this option is selected. Or
   - Select the Selection option to build a report only on selected classes or to include individual elements in the report.

     Refer to Select a Scope for more information.
4. If you select the Selection option in the previous step, select the Segment classes you want to include in the report.
5. Select the type of information you want to include for each element in the Include group box.
6. Enter or browse for the location to save the HTML file of the report in the Save In: section
7. Click **OK** to build the report.

Select a Scope

A scope is a domain that contains elements that you want to include in a report. You can include elements such as Namespaces, Ispecs, and Methods as a scope. For example:

- If you select an Insertable as a scope, only the Insertable is included in the report
- If you select a Folder as a scope, the Folder and all the elements and Classes of the Folder are included in the report

The scope selection is helpful when you want to generate a report on specific elements in the model.

To select a scope, perform the following:

1. Select **Selection in the Build Comment Pages…** dialog box to enable the **Add…**.
2. Click **Add….**.

   The **Select Scope** dialog box appears.
3. In the **Select Scope** dialog box, navigate to the element that you want to include in the report and select it.
4. Click **OK**.

The selected element appears in the Build Comment Pages… dialog box as a scope.

Select Security Option

If you select the Security check box, Security Matrix is generated and is available on the home page of the generated system. Two tables are created that displays the security settings for each user or group and for each element.

Following is the description of the tables:

1. Lists all the elements specified in the domain and their security groups including hyperlinks to the actual element.
2. Lists all the Groups and Users specified in the domain and the elements of the security groups including hyperlinks to the actual element.

Notes:

- *An element appears in the list if the defined security setting of the element is different than its Owner.*
- *For Groups, the group name appears as a hyperlink. This link provides a new page that lists the users within the group.*
- *Each element is hyperlinked to its home page that displays the Security Matrix.*

The home page of an element displays the Security Matrix that includes the local defined and derived security setting. The derived security setting is displayed as a hyperlink to the element from where the security is derived. The local defined settings do not a have a hyperlink.

Using Class Diagram Editor

The Class Diagram is a graphical representation for both classes and their relationships. With the specialized editor, modifications can be made to both the appearance and class structures. A toolbox also known as “UML Designer Toolbox” provides the ability to add new diagram elements. Any changes to the appearance of the diagram are lost without a save. All structural changes are saved immediately.

The Class diagram Editor is a graphical editor that helps you to document and maintain a selection of classes. They are added with a name to any class (including the model) and can be organized using Folders and other Class Diagrams.

The changes made to the Class Diagram are immediately reflected in the class view, and the changes made in the Class View immediately affect the appearance of the Class Diagram Editor.

The Class Diagram Editor is used to modify and manipulate class diagrams. The modifications that you can make in Class Diagram Editor are:

- Adding and removing classes as diagram members.
- Specifying properties of diagram members, such as name and description.
- Adding and removing attributes and methods from diagram members.
- Adding and removing relationships between diagram members.
- Specifying properties of relationships between diagram members.
- Specifying the appearance of diagram members in terms of font type, line style, and fill color.
- Specifying the display of diagram members in terms of the inclusion of attributes, methods, kind, visibility, and initial value.
- Specifying the spatial layout of diagram members.

Refer to Manipulating Entities and Entity Properties for more information.

The modifications that you make in Class Diagram Editor are persistent and are reflected in the other model views, such as the Properties window or Class View as soon as each discrete operation is performed.

Note: *The modifications that affect the model is reflected immediately whereas to reflect the appearance changes such as font type or font color requires an explicit Save.*

Manipulating Entities

Adding Entities

You can add Classes and relationships in the Class Diagram Editor using one of the following methods:

- To add an existing class, drag the class (or classes) from other model views (such as the Class View) into the Class Diagram Editor.

Or

- To add a new class or relationship, drag the required entity from the UML Designer Toolbox into the Class Diagram Editor. You can also double-click the required entity on the UML Designer toolbox to place the new entity in the Class Diagram Editor.

Removing Entities

You can remove the classes and relationships by selecting the required entity (or entities) and pressing DELETE, or by right-clicking them and selecting **Remove**. On clicking **Remove**, you are prompted to confirm the deletion of the required entity (or entities). They are removed from the class diagram only but retained in the model.

Selecting Multiple Entities

There are two ways to select multiple entities in Class Diagram Editor:

- To select multiple entities that are adjacent, drag a boundary around the entities that you want to select.
- To select multiple entities that are not adjacent, select a single entity, press SHIFT or CTRL key, then click the other entities that you want to select.

Moving Entities

To move the entities in the Class Diagram Editor, drag them to position them as required.

Relationships between classes are automatically redrawn as their constituent classes are moved. To make any changes in an existing relationship, select the relationship in the Class Diagram Editor, and then drag the handles which are displayed as green boxes at each end of the relationship. When dragged, these handles snaps to the class closest to the current cursor position.

Refer to Class Diagram Editor Settings for more information on how to use the grid to help position entities.

Modifying Appearance

You can modify the appearance of class diagram members by selecting the required entities, right-clicking, and expanding Format, and then selecting one of the following options:

- **Text**

  This option displays the Display Options dialog box with the Text tab in the foreground.

  You can adjust the text format, such as font, size, and color and style, such as bold, italic, strikethrough, and underline as required

- **Line**

  This option displays the Display Options dialog box with the Line tab in the foreground.

  You can adjust the line style such as pattern, weight, and color as required.

- **Fill**

  This option displays the Display Options dialog box with the Fill tab in the foreground.

  You can adjust the fill style like color as required.

  Refer to Class Diagram Editor Settings for more information on how to set default colors.

Modifying Display

You can modify the display of class diagram members by selecting the required entity in the Class Diagram Editor, right-clicking and selecting Shape Display Options. This displays the UML Shape Display Options dialog box, which specifies the display of the selected class diagram member in the following categories:

- **General**

  In this category, you can specify the display of name, stereotype, method parameters, and attribute visibilities as required.

- **Attributes**

  In this category, you can specify the display of the attribute types, initial values, and multiplicities as required.

- **Suppress**

  In this category, you can specify the display of the attributes and methods as required.

To apply these display settings to subsequently added entities, select the **Apply to subsequently dropped element of the same type** check box in the UML Shape Display Options dialog box.

Entity Properties

You can access the properties of class diagram members by double-clicking the required member. Alternatively, you can access the properties by selecting the required class diagram member, right-clicking it, and selecting **Properties**. This displays the Specification dialog box of the selected class member.

The Specification dialog box includes four tabs, General, Members, Dependencies, Documentation.

General

The General tab of the Property dialog box resembles the Properties tab document window. It allows access to the following properties of the selected entity:

- Name
- Description
- Author
- Class (stereotype, multiplicity, visibility, abstraction, and persistence)

- Inheritance (superclass and subclasses)
- Derived Types

Note: *Any change made in the General Tab reflects in the other property windows.*

Members

The Members tab of the Property dialog box resembles the Members tab document window. It allows access to the members of the selected entity.

Dependencies

The Dependencies tab of the Property dialog box resembles the Dependencies tab document window. It allows access to the relationships of the selected entity to other entities.

Documentation

The Documentation tab of the Property dialog box resembles the Documentation tab document window. It allows access to notes and additional information about the selected entity.

Context Menu Options

The context menu appears when you right-click the required entity. The context menu shows the details of classes and their constituent members along with the relationships they share with other classes. You can also synchronize a class in the Class Diagram Editor with the class view using the context menu. The following context menu options are available for a class in Class Diagram Editor.

The Class diagram automatically shows relationships between classes. A quick way to include classes that have a generalization relationship is to select the class and choose the appropriate option from the context menu.

- Show base classes
    1. In the Class Diagram Editor, select the class for which you want to view the base class.
    2. On the Class Diagram context menu, select Show Base Classes.

       The base class of the selected class appears in the diagram. For any inheritance, lines appear between the subclass and its superclass. This line is a solid line, with a hollow arrowhead at the parent entity.
- Show derived classes
    1. In the class diagram, select the class for which you want to view the derived classes.
    2. On the Class Diagram context menu, select Show Derived Classes

       The derived classes of the selected class appear in the diagram. For any inheritance, lines appear between the superclass and its subclasses. This line is a solid line, with a hollow arrowhead at the parent entity.

- Synchronize Class View

  In Class Diagram Editor, you can synchronize the currently selected diagram member with the class view.

  To synchronize a diagram member, perform the following:

  1. In the Class Diagram Editor, select the diagram member to be synchronized.
  2. Select Synchronize Class View.

The element is synchronized with the class view and is selected in the Class View.

Class Diagram Editor Settings

You can customize the Class Diagram Editor with a grid setting that provides horizontal and vertical guidelines to help align the entities in the class diagram.

To select default grid settings, perform the following:

1. Select **Options** from the **Tools** Menu.
2. Navigate to the UML Designer folder in the Options dialog box.
3. Expand Display to access the following options:
   - **Snap To Grid** – Specifies whether entities in the class diagram are snapped to grid intersection points when resized or moved. By default, this is selected.
   - **Show Grid** – Specifies whether the grid is displayed. By default, this is selected.
   - **Horizontal spacing** – Sets the distance between horizontal grid lines. By default, the value is 10 pixels.
   - **Vertical spacing** – Sets the distance between vertical grid lines. By default, the value is 10 pixels.

Class Diagram Editor Views

You can manipulate the view in the Class Diagram Editor to display the class diagram at different levels of magnification. If the class diagram is too large to fit within the current size of the window, you can manipulate the view to display a specific area of the class diagram.

To select magnification, expand **Zoom** from the **View** menu to set either of the following options:

- **Zoom 15%, Zoom 25%, Zoom 50%, Zoom 75%, Zoom 100%** – Sets the magnification level according to the selected value.
- **Zoom In, Zoom Out** – Sets the magnification level incrementally.
- **Fit Window** – Sets the class diagram to an appropriate magnification in order to fit within the current size of the window.

To select display area, perform the following:

1. Click and hold the Overview icon that appears as a 'Hand' in the bottom right corner of the scroll bars. This displays an overview of the entire class diagram.

   Note: *The Overview icon is only enabled when the scroll bars are enabled – that is, when the class diagram extends beyond the boundaries of the current view.*

2. Drag to select the area of the class diagram to display.

UML Designer Toolbox

The UML Designer toolbox lists the available entities that can be placed in a class diagram using the Class Diagram Editor. The following entities are available in this toolbox.

Class

Adding a class from the UML toolbox has the same effect as adding a class from any of the other System Modeler views. Refer to Adding System Modeler Items for more information.

Generalization

In a System Modeler context, a generalization relationship specifies the superclass of a target class. To create a generalization relationship in class diagram, perform the following:

1. Double-click Generalization on the UML toolbox.
2. Click the superclass on the Class Diagram tab.
3. Click the target class.

Alternatively, you can also drag Generalization from the UML toolbox to the superclass on the Class Diagram tab and click the target class.

Lookup Dependency

A lookup dependency relationship specifies a lookup using the predefined attributes as the key values for the target class.

To create a lookup dependency in a class diagram, perform the following:

1. Double-click Lookup Dependency on the UML toolbox
2. Click the dependent ispec attribute on the Class Diagram tab
3. Click the target class.

Alternatively, you can also drag Lookup Dependency from the UML Designer toolbox to the dependent ispec attribute on the Class Diagram tab and click the target class.

Diagram

Adding a diagram from the UML toolbox has the same effect as adding a diagram from any of the other System Modeler views. Refer to Adding System Modeler Items for more information.

Comment

A comment allows descriptive text to be embedded in a class diagram. They do not have any semantic meaning.

To create a comment, double-click **Comment** on the UML Designer toolbox to place a comment at a default position on the Class Diagram tab. Alternatively, you can also drag Comment from the UML toolbox to the desired position on the Class Diagram tab.

After the comment is added in the Class Diagram tab, select the comment to display a yellow diamond that can be dragged to associate the comment with other class diagram entities. The comment entity must have an association to a class entity otherwise the comment entity isnot saved.

Composition

A composition relationship combines the target class as an attribute of the source class.

To create a composition relationship in class diagram, perform the following:

1. Double-click Composition on the UML toolbox.
2. Click the attribute superclass on the Class Diagram tab.
3. Click the target class.

Alternatively, you can also drag Composition from the UML toolbox to the attribute superclass on the Class Diagram tab and click the target class.

Changing Model Database Server

You can change the AB Suite model database or the AB Suite Client Framework model database to a different server. To do this, you must change the previous server name contained in the .smsfm file of the AB Suite application or the AB Suite Client Framework application after importing the model to the new server.

To change the server name, perform the following:

1. Open the <AB Suite Application>.smsfm file or the <AB Suite Client Framework Application>.smsfm file using Notepad.

   Note: *The <AB Suite Application>.smsfm or the <AB Suite Client Framework Application>.smsfm is stored in the location, <Default location>\<AB Suite application>\<AB Suite application>.*

2. Specify the new server name in place of the previous server.

   For example, Change ServerName="PreviousServer" to ServerName="NewServer".

3. Save and close the <AB Suite Application>.smsfm file or the <AB Suite Client Framework Application>.smsfm file.

Using Enterprise Output Manager Reports

Enterprise Output Manager reports are specially formatted reports designed to be used with the Enterprise Output Manager print management application.

The Enterprise Output Manager Application

The Enterprise Output Manager application (EOM) is a comprehensive print management and file distribution solution for mixed platform networks.

The primary function of the Enterprise Output Manager application is to automatically route print files from any supported platform to any other supported platform. However, the main advantage when paired with NextGen Developer is to increase flexibility in designing and printing reports.

Enterprise Output Manager software is needed to both design and print Enterprise Output Manager reports. It is the responsibility of the Enterprise Output Manager administrator to set up Enterprise Output Manager for use with Developer. This includes setting up appropriate print attributes with either the supplied NextGen Data-Dependent Attribute (DDA) or user written DDAs. Refer to the Enterprise *Output Manager Configuration and Operations Guide* for more information on instructions on setting up Enterprise Output Manager for printing.

Before Creating an Enterprise Output Manager Report

There are several steps required prior to creating a Enterprise Output Manager Report in Developer:

1. Create a Windows Metafile (.WMF) using any off the shelf application you prefer. This file should contain the design and layout of your report and is used as the template.
2. Using the Enterprise Output Manager File Format Utility, add the relevant attributes and fields to the .WMF file and save as a form file (.DFF).
3. Import the form file into Developer using the Enterprise Output Manager wizard. This process creates the necessary logic for printing the basic Enterprise Output Manager report.

The basic structure of the report is automatically generated by the Enterprise Output Manager wizard, providing a skeleton for further development. While the generated report should run, it is unlikely to produce the required result.

There is no restriction on what logic you can add to your report. You can access as many structures as you like, using any logic commands. However, as there is no page layout, logic commands that rely on a page format are meaningless in Enterprise Output Manager. For example, Glb.Linecount and Glb.Pagecount are two of the commands that have no meaning in an Enterprise Output Manager report.

Creating a Form File

To define the attributes and fields in the Enterprise Output Manager Form File Utility (DFFU), perform the following:

1. Start the Form File Utility.
2. Load the Windows Metafile you previously created.
3. Select Enterprise Application Environment DFF File from the Options menu.
4. Click **Add Field** on the tool bar.
5. Draw a field on the form. The Property Inspector box is displayed.
6. Use the Property Inspector box to name the field, assign the relevant attribute, and create multiple rows or columns, as needed.

   Fields defined with a Attribute Reference name is used in the report.

   When Attribute Reference and SAME.AS Reference values are defined for a field, an attribute is created for the report in Developer. Refer to Wizard Output.
7. Repeat steps 5 and 6 until you have entered all the fields you require.
8. Save as a .DFF file.

Refer to the *Enterprise Output Manager Configuration and Operations Guide* for more information on using the Form File Utility.

Note: *The Form File Utility allows up to 60 characters for the Attribute Reference and Same.As Reference values. System Modeler allows names of up to 64 characters in length. This means that some valid System Modeler names may not be valid in the Form File Utility. This limitation also affects the use of System Modeler qualified names in the Form File Utility.*

Add Enterprise Output Manager Reports

The Enterprise Output Manager wizard enables you to add Enterprise Output Manager Reports to your application.

Before you use the wizard, ensure you have a form file (.DFF), created with the Enterprise Output Manager File Format utility. Your form file should have the relevant attributes and fields required in your report.

To start the Enterprise Output Manager wizard, perform the following:

1. Select the position in your project that you want to add the report.

   Enterprise Output Manager reports can be added anywhere a standard report can be added.
2. Right-click and select **Add**, then **Add Enterprise Output Manager Report**.

   The first page of the Enterprise Output Manager wizard is displayed, Enterprise Output Manager Form File .

3. Complete each page of the wizard as required and click **Next** to proceed to the next wizard page.

The Enterprise Output Manager wizard performs the following processes:

- Add Enterprise Output Manager Reports
- Select Keys
- Select a Persistant Object
- Set Report Options
- Wizard Output

Select the Enterprise Output Manager Form File

Use this page to select the form file required for your report.

Enter, or browse for, the location of the required .DFF file and click OK.

Select Keys

Use this page to select the attributes to be the keys to the object that is used in the report.

Fields List

Displays the attributes that are defined in the selected form file. Corresponding form file field names are also displayed for each attribute.

Select the attributes to be defined as keys and click Add to move them to the Keys List.

Keys List

Displays the attributes you have selected as keys. To remove a key, select it and click Remove.

It is important that this list displays the attributes in the same sequence as they are defined in the persistent object your report is to reference. To change the sequence of the keys, use the Up or Down buttons.

Select a Persistant Object

Use this page to select the persistent object on which to base your report. This can be a class, profile, or event set.

Keys Selected

Displays the keys selected in the previous wizard page, Select Keys . This field is displayed for reference only. To change the keys, click Back to access the previous page.

Matching Objects

Displays the classes, profiles, and event set that match the keys displayed in the Keys Selected field.

Select the object you want to use as the basis for your report. If you select <NONE>, no explicit database read loop is created in the main logic when the report is generated.

Objects are displayed in the list if the selected keys meet the following criteria:

- A class is displayed if the selected keys are a subset of the class's keys and are in the same sequence.
- A profile is displayed if the selected keys are a subset of the profile's keys and are in the same sequence. If any of the keys has a qualified name, the profile must be over the qualified class.
- Event is displayed if all the selected keys are contained in the event set.

Set Report Options

Use this page to define options for your report.

Name

Enter a name for the report. The default name is the name of the form file.

Description

Enter a description for the report. The description defaults to Enterprise Output Manager wizard generated report.

Report Frames

The names of the following default frames can be changed:

- Control
- Header
- Footer
- Details

If the field for any frame is cleared, that frame is not created.

Existing Report

If a report already exist with the name you have specified, you can select one of the following actions to deal with the Main method:

- Do not add any logic.

  No Enterprise Output Manager Wizard-created logic is added.

- Replace the existing logic.

  Any existing logic is replaced by the new Enterprise Output Manager Wizard-created logic.
- Add further logic.

  Any existing logic is kept and new Enterprise Output Manager Wizard-created logic added at the end of the Main method.

If the report you are adding does not exist, this field is not available.

When you have completed all the fields, click **Finish** to complete the wizard. Your Enterprise Output Manager report is generated and displayed in the project. You can add further logic as required.

Wizard Output

The Enterprise Output Manager wizard adds a report in the selected position in your project. The Enterprise Output Manager report contains the following objects:

Note: *You can have multiple row fields in the footer, but not in the header frame. Multiple column fields used without multiple row fields, however, are not put in the detail frame.*

Main method

This method contains the logic to loop over and select all attributes in the selected persistant object. Each attribute is printed using the Attributes read from the DFF.

If keys were selected in the wizard, a break is included in the loop logic to print the header and footer whenever the key changes.

If no persistent object is select a simple loop is added, to be modified manually.

Attribute for the selected persistent object

An attribute is created for the selected persistent object. The object is inherit from the selected persistent object. The name of this object is <SelectedObjectName_instance>. If <NONE> was selected, this attribute is not created.

Control frame

This frame is used to supply report information to Enterprise Output Manager. It is printed at the start of a report.

Details frame

This frame is the central table area of the form file.

The detail area is defined by the first field that has more than one row. The Enterprise Output Manager wizard uses the first multiple row field on the page and adds all other fields with the same number of rows to make up a table.

All the fields above the table are put in the header frame and all fields below the table are put in the footer frame. However, if your report has more than one table, you need to enhance the report structure manually.

Header Frame

This frame tells Enterprise Output Manager to start a new page, and what form file to use. It contains all the fields above the table contained in the Details frame.

The header frame is created even if there are no form file fields in it.

Footer Frame

This frame contains those fields that come after the table contained in the details frame.

The footer frame is created even if there are no form file fields in it.

Labels on the Frames

For each field in the DFF, two labels are painted on a frame. The first is the name of the DFF field. On the same line is a label for the attribute to be displayed. If the Attribute Reference for the field matches an attribute in the model, the label references this attribute. If not, an attribute is created in the frame and the label references this attribute.

If the Attribute Reference is a qualified name, this is used to find the attribute in the model. If it is not qualified and an attribute of that name is in the Selected Persistent Object, the label refers to that attribute.

Attributes in Frames

When Attribute Reference and Same.As Reference values are defined for a field in the form file, an attribute is created in the frame to which they belong in Developer if the Attribute Reference does not match an attribute in the model.

The attribute is created with the following properties:

- Name – based on the entry in the Attribute Reference field.
- Primitive – assigned ‘String’.
- Length – calculated on the size of the field and the specified font.
- Inherits – based on the entry in the SAME.AS Reference field.

All other properties for the attribute are set to their default settings.

The following rules apply to assigning attribute properties:

- If the name of the attribute is the same as an existing type, it is used to define the attribute.
- If the name of the attribute is qualified (for example, CUST.NAM where NAM is the attribute and CUST is the ispec it belongs to) and matches either an existing attribute in an ispec or an existing type, then all details except length are taken from the existing attribute. The length is calculated based on the size of the field and the specified font.
- If a valid attribute is specified in the SAME.AS Reference field, then details are taken from that attribute.
- If a qualified name does not exist, or the field name is not a valid Developer name, an attribute is not created. A message is written to the logic method generated for the frame. This isa comment in the logic and indicates the problem with creating the attribute.

Maintain Enterprise Output Manager Reports

Once an Enterprise Output Manager report is added to the project you can maintain it in the same way as any other report.

If the output of the Enterprise Output Manager wizard needs to be amended, you need to start again and make any required changes in step one of the report development cycle, though you can keep the base template and form file from the previous effort.

Automatic maintenance of an Enterprise Output Manager report is not possible as there is no direct link between the report and the form file that defines how it is printed. The only link between the report and the form file definitions are the field names themselves. However, the wizard is able to add, remove, or update any of the attribute references in the header, footer, and detail frames. That is, when the wizard encounters:

- A new field, it adds the field definition to the painted frame.
- The same field, it changes the field attributes appropriately.

The following report properties are set by the Enterprise Output Manager wizard and cannot be changed:

Report Properties	Value
Default Device	Enterprise Output Manager generated reports
Line Length	255

Report Frame Presentation Properties	Value
Display attributes:	Pitch, are not set.
Bright	True or False
Big	True or False

Report Frame Presentation Properties	Value
Under	True or False
Underscore	True or False
Reverse	True or False
Upperscore	True or False
Reset	True or False
Pitch	Pitch set to 132
Control Codes	True or False
Numeric attributes: Decimal Character Separator Blank When Zero Floating Sign	All set to Default

When you are painting a report, do not exceed the line length of 255 characters. Enterprise Output Manager only receives a maximum of 255 characters per line.

You can set the remaining options or change their defaults as required.

After a print job has been delivered to Enterprise Output Manager, it is possible to view the print output in preview mode before it is printed.

Commands in the Data Dependent Attribute

The following commands are defined in the supplied NextGen Data Dependent Attribute (DDA) and can be used to control print output by simply painting the commands, starting in column one of the data file:

```
$BARCODE$
$BITMAPS$
$DEBUG$
$DEPHDR$
$FONT$
$NEWDFF$
$NEWPA$
$NEWPAGE$
$XQTFORM$
```

Notes:

- *The printing of lines that do not conform to the syntax rules is suppressed*
- *The syntax items $BARCODE$, $BITMAP$, $DEBUG$, and $FONT$, are not automatically generated by the Enterprise Output Manager wizard. The syntax must be added after the wizard is run.*

$BARCODE$

Use this command to print barcodes. Enterprise Output Manager supports and enforces specific barcode dependent features, such as automatic checksum generation and size limitations.

```
$BARCODE$ barcodestyle "field name" field value
```

The following variations of barcode style have been defined:

- CODE39
- PostNet
- 2of5

CUST_$BARCODE$

This item is called by LINC_$BARCODE$ after the standard barcode definitions have been processed. User defined barcode definitions can be entered in the CUST_$BARCODE$ item to make upgrades to future versions of the NextGen DDA easier. Changes to the NextGen DDA require the DDA design key. Following is an example of DDA code:

```
If field_type EQ "BarcodeStyle" Begin
>Print Barcode BarcodeStyle; file_value Field Vairiable=field_name
>Set Variable num_result=$NumericResult
End Block
```

$BITMAPS$

Use this command to position graphic files, such as logos and signatures. Enterprise Output Manager scales the file to the size of the field.

```
$BITMAP$ "field name" graphicsfilename ["directory alias"]
```

By default, the directory alias is $CommandFile.

This command supports the following formats:

- BMP
- DFF
- EMF
- WMF

$DEBUG$

This command controls the level of error message display.

```
$DEBUG$ Level Max-Error
```

Level specifies the level of diagnostic detail:

0 – no debug messages.

1 – messages are inserted in the print debug file, when the print debug is turned on in the physical printer configuration.

2 – same as 1, plus messages in event log. This is the default setting.

3 – same as 2, plus variable dump included in print debug file.

4 – same as 3, plus popup message for each error.

Max-Error specifies the maximum number of error messages per print job. By default, the setting is 20.

$DEPHDR$

Lines beginning with this command are used to pass variables to Enterprise Output Manager. One use for them is for file masking. They are ignored by the NextGen DDA, but are used by file masks in Enterprise Output Manager to route print output.

```
$DEPHDR$ token [token]
```

Tokens can be separated by spaces or commas. To include spaces or commas as part of a token you must enclose the token in double quotes. Enterprise Output Manager accepts a maximum of ten tokens with a maximum total length of 170 characters. Multiple tokens on the same line are accepted. You can have multiple $DEPHDR$ lines but they should be contained in the first 4000 bytes of the print file. The Enterprise Output Manager wizard generates $DEPHDR$ lines. The lines include information such as the application name, report name, and the name if the user running the report.

$FONT$

This command changes the currently active font attributes. Font attributes are normally associated with the field in the form file. To enable dynamic font changes the font attributes for the relevant fields must be set to none. Enterprise Output Manager then uses the current active font to display the field.

```
$FONT$ fontidentifier
```

The following variations of font identifier have been defined:

- COURIERNEW-R-10
- COURIERNEW-B-10
- COURIERNEW-I-10
- COURIERNEW-BI-10

CUST_$FONT$

This item is called by LINC_$FONT$ after the standard font definitions have been processed. User defined font definitions can be entered in the CUST_$FONT$ item to make future version upgrade of the NextGen DDA easier. Changes to the DDA require the DDA design key. Following is an example of DDA code:

```
If field_type EQ "COURIERNEW_R_10" Begin
>Execute Windows Font Change Font=(Courier New, 10.0,,,,R=0,G=0,B=0)
?
>Exit
End Block
```

$NEWDFF$

For multiple paged printouts that also include different page layouts, it is possible to code a runtime control for switching to another form file:

```
$NEWDFF$ "<DFF Filename>" "<Directory Alias>" [REPLACE]
```

When REPLACE is specified, Enterprise Output Manager replaces the current Top of Form command with the new form file. If the form file contains a .WMF form, then this is used on the following pages until either a new print attribute is loaded (unconditionally replacing the Top Of Form command), or another $NEWDFF$ command with the REPLACE field is processed.

When the REPLACE field is not present the Top Of Form command is not changed. Note that when the $NEWDFF$ command is processed, the current Top Of Form command, if it exists, has been executed already for the current page.

This command can be used multiple times on a page to change the field definitions. In this case, users only need to include the REPLACE field at the start of a file. It is possible to jump to as many forms in succession as desired.

$NEWPA$

This command changes the current print attribute when it is processed:

```
$NEWPA$ <print attribute name> [NOW]
```

When NOW is not included, the print attribute change is queued and executed at the next page break.

When NOW is included, the print attribute changes immediately. The print continues at the same logical page.

Notes:

- *Not all attributes of a page can be changed immediately, for example, the page orientation.*
- *If the number of logical pages of both the old attribute and new attribute are not the same, then the result can be unpredictable.*

If the new print attribute specifies a data command file, then the file is executed.

If a Top Of Page command is specified in the new print attribute, then it is set as the new Top Of Page command. If the print attribute does not specify a Top Of Page command then the current command file is cleared. Refer to the *Enterprise Output Manager Configuration and Operations Guide* for more information on command files.

If the new print attribute specifies a DDA then the current DDA is replaced, and as a result, so could the current form file template.

$NEWPAGE$

This control statement executes a form feed to a new output page.

A $NEWPAGE$ command is written to the output file when an Advance NewPage or BeginPage Clear command is executed in the report logic.

$XQTFORM$

Use this control statement to execute a form command file when Enterprise Output Manager drivers are used.

```
$XQTFORM$ <filename.ext>  ["<directory alias>"] [BINARY]
```

The filename is required to be in 8.3 dos format. It can be a .WMF, .EMF, .DFF, or .PCL macro.

The <*directory alias*> field is optional. If it is present it must have the value of a directory alias as configured in Enterprise Output Manager. If either the directory alias or filename do not exist then there is an error message when printing the data. The Enterprise Output Manager wizard sets this to $CommandFile.

The BINARY option specifies that the file is a binary file. The BINARY option is only meaningful for .PCL macros.

When this control statement is found, the command file is executed and the resulting form is painted on the page. Typically, this command is used at the start of every page. For example, reports generated by the Enterprise Output Manager wizard have this command painted on the header frame.

Route Reports in Enterprise Output Manager

A report that has not been created using the Enterprise Output Manager wizard can also be routed to, and printed by, Enterprise Output Manager. This involves writing logic and painting the frames to output the necessary Enterprise Output Manager headers. It is not necessary to change the default device to DP in order to route output to Enterprise Output Manager, as the header information is sufficient.

Any report destined for output to Enterprise Output Manager can pass Enterprise Output Manager control information through the first line of data in the print file, starting with the $DEPHDR$ control image. This header information is automatically generated by the Enterprise Output Manager wizard, but it can be added manually. This is achieved by painting a frame with the necessary header information.

Enterprise Output Manager uses file masks to identify incoming print jobs. This process assigns print attributes to the print jobs and assigns which printers they go to. If you want to use the file mask information as defined by the Enterprise Output Manager wizard, then the painted header information should match the following:

```
$DEPHDR$ <DEPHDR-T1>
$DEPHDR$ <DEPHDR-T2>
$DEPHDR$ <DEPHDR-T3>
$DEPHDR$ "<DEPHDR-T4>"
$DEPHDR$ <DEPHDR-T5>
$DEPHDR$ <DEPHDR-T6>
$DEPHDR$ <DEPHDR-T7>
$DEPHDR$ <DEPHDR-T8>
$DEPHDR$ <DEPHDR-T9>
$DEPHDR$ "<DEPHDR-T10>"
```

Where quotation marks are placed around variables that you expect to contain spaces, and the attributes are defined as the following:

```
DEPHDR_T1: Primitive "String", Length 10 : application name
DEPHDR_T2: Primitive "String", Length 10 : report name
DEPHDR_T3: Primitive "String", Length 10 : report language
DEPHDR_T4: Primitive "String", Length 13 : generation date and time
DEPHDR_T5: Primitive "String", Length 10 : mix number
DEPHDR_T6: Primitive "String", Length 3 : number of copies
DEPHDR_T7: Primitive "String", Length 2 : save days
DEPHDR_T8: Primitive "String", Length 17 : initiated station
DEPHDR_T9: Primitive "String", Length 17 : user
DEPHDR_T10: Primitive "String", Length 50 : print banner
```

When the user defines a file mask in Enterprise Output Manager the information contained in the $DEPHDR$ headers are read into the file mask as the fields User Tag 1 through 10. The order of the $DEPHDR$ variables must match the order of User Tags as defined in the file mask. When the mask matches, the print output is assigned to a print attribute and a printer.

You are able to change the file mask if required, but it is your responsibility to ensure the information in the Enterprise Output Manager file mask matches the attributes supplied in $DEPHDR$. Refer to the *Enterprise Output Manager Configuration and Operations Guide* for more information.

Dynamic Display in Enterprise Output Manager Reports

It is possible to manipulate the Data Dependent Attribute (DDA) of an Enterprise Output Manager report to change the report display attributes at the time of printing.

In some cases, reports need to display text in variable fonts, colors, or other attributes depending on the data that is sent to Enterprise Output Manager. For example, a table of values may print positive numbers in black and negative numbers in red.

If you intend to include dynamic attributes, do not provide a default font when defining the field in Enterprise Output Manager. Invoke DDA logic to set the font depending on the contents of the field at runtime.

When Enterprise Output Manager processes incoming data, the DDA command is invoked and places the data into the appropriate field variable. Any operations that the DDA is instructed to do are performed at this point. Enterprise Output Manager then prints the data in the correct place on the page, according to the definition in the form file.

To be compatible with Developer, Enterprise Output Manager provides a generic NextGen DDA, which can be used to process any form file that you create. Each report created does not need to have its own copy of the DDA, the one default DDA can be shared. However, it can be copied and tailored by the user. For example, the DDA can be enhanced to process Output Codes, or to dynamically change fonts or colors depending on data in the output file.

The DDA key is required to design them, to tailor, or copy the default DDA. If required, each report can be associated with a specific DDA, but configuration is the responsibility of the user. This requires configuring file masks that apply a print attribute, with its associated DDA, to a given print job. Refer to the *Enterprise Output Manager Configuration and Operations Guide* for more information.

Printing Enterprise Output Manager Reports

You need to set up an Enterprise Output Manager Queue and an LPR/LPD printer.

On the Windows operating system machine where the application runs, perform the following:

1. Install TCP/IP Printing, if it is not already installed.
2. Create an LPR/LPD printer. Refer to Creating an LPR/LPD Printer.
3. Set the printer to allow 255 column lines. Refer to Setting the Printer Options.

In Enterprise Output Manager define a print queue that receives the report and prepares it for printing, perform the following:

1. Define a Communications Path. Refer to Defining a Communications Path.
2. Define a Transfer Attribute. Refer to Defining a Transfer Attribute.
3. Define a Physical Printer for receiving printouts. Refer to Defining a Physical Printer for Receiving Printouts.
4. Define a Physical Printer for printing the final report. Refer to Defining a Physical Printer to Print the Reports.
5. Import the NextGen DDA. Refer to Importing the Agile Business Suite DDA.
6. Define a Print Attribute. Refer to Defining a Print Attribute.
7. Define a File Mask. Refer to Defining a File Mask.

When you have completed these steps, you are able to direct reports to the Enterprise Output Manager queue. In Enterprise Output Manager you can preview the final report before you print it.

Notes: *Due to an issue with auto pagination, it is recommended to:*

- *Configure the printer setup for Windows to set the Enterprise Output Manager LPD device for continuous forms (US std fanfold) without the line break, through the device settings for the printer. If you do not do this, Windows automatically paginates according to the device characteristics, which is normally 66 lines per page.*
- *Set the report page length by setting the Glb.Formdepth, in logic, to a high number. For example, 9000.*

Creating an LPR/LPD Printer

1. Go to **Start** > **Settings**.
2. Double-click **Add Printer**.

   The Add Printer wizard is started.
3. Select **My Computer** and click **Next**.
4. Click **Add Port**.

   A list of printer ports is displayed.
5. Select LPR Port from the list, then click **New Port**.

   The Add LPR Compatible Port dialog is displayed.
6. Enter the name or address of the server providing lpd in the corresponding field.
7. Enter the name of the queue in the second field.

   This is the name to be used later in the Enterprise Output Manager set up. For example, you can enter Enterprise Output ManagerQueue.
8. Click **OK**, then **Close**, to return to the Add printer wizard. Click **Next**.
9. Select **Generic as the manufacturer and Generic/Text Only** as the printer, and click **Next**.
10. If asked, select to keep existing driver and click **Next**.
11. Enter a name for the printer and click **Next**.
12. Select **Not shared** and click **Next**.
13. Do not print a test page. Click **Finish**.
14. If the printer driver was not previously installed and you were not prompted at step 10, you are prompted for the path for the Windows installation files. Enter this path and click **OK**.

Setting the Printer Options

The Windows Generic/Text Only print driver defaults to 80 column lines. To setup the printer to allow 255 column lines, perform the following:

1. Point to **Start** > **Settings** > **Devices and Printers**.
2. From the **File** Menu, select **Server Properties**. Select the **Forms** tab, then select the **Create a New Form** check box.

3. Enter a description in the **Form Description for** field, such as Wide255.
4. In the Measurements field, adjust the following:
    - Under Paper Size, make the width 25.5in and the height 11.68in.
    - Leave all Printer Area Margins set to zero.
5. Click **Save Form**, then **OK** when complete.
6. In the Printers folder, right-click the **Generic/Text Only printer**, and select **Document Defaults**. Adjust the following:

    For Enterprise Output Manager reports:

    - On the **Page Setup** tab, select Continuous - No Page Break from the list in the Paper Source field.
    - On the **Advanced** tab:

        Under Paper Output, select Paper Size. In the Change 'Paper Size' Setting field at the bottom, select Wide255 (or the name you gave to the form created in step 3).

        Under Paper Output, select Paper Source. In the Change 'Paper Source' Setting field at the bottom, select Continuous - No Page Break.

    For non-Enterprise Output Manager reports:

    - On the Page Setup tab, select Continuous - Page Break from the list in the Paper Source field.
    - On the Advanced tab:

        Under Paper Output, select Paper Size. In the Change 'Paper Size' Setting field at the bottom, select Wide255 (or the name you gave to the form created in step 3).

        Under Paper Output, select Paper Source. In the Change 'Paper Source' Setting field at the bottom, select Continuous - Page Break.
7. When complete, click **OK**.

    The form created in step 3 is now be the default paper size for this printer.

Defining a Communications Path

1. Start Enterprise Output Manager.
2. Select **Configuration**, then **Communications**.
3. Click **Add/Copy Path**.
4. Enter a password if required and click **OK**.
5. Enter a name for the Communications Path and click **OK**.
6. Select LPR/LPD as the peer type, enter * as the IP Address, and ensure Max total activities is at least 1.
7. Select the Server page, ensure Activities is at least 1.
8. Click **OK** and then **Done**.

Defining a Transfer Attribute

1. Start Enterprise Output Manager if not already running.
2. Select **Configuration**, **Transfer Attribute**.
3. Click **Add/Copy Attribute**.
4. Enter a name for the Transfer Attribute. Click **OK**.
5. Select the Communications Path in the Primary Path field as you defined in Defining a Communications Path , and select **Transfer Enable**.
6. Click **OK** and then **Done**.

Defining a Physical Printer for Receiving Printouts

1. Start Enterprise Output Manager if not already running.
2. Click **LPR/LPD** in the **Add Printer** field.
3. Enter a name for the printer. Click **OK**.
4. Select the Transfer Attribute that you defined previously in the Access Printer via Transfer Attribute field. Click **OK**.
5. Select the new printer in the list and check Printer Available in the properties area. Click **Modify**.
6. Click **Done**.

Defining a Physical Printer to Print the Reports

1. Start Enterprise Output Manager if not already running.
2. Click **Win Pr** in the **Add Printer** field.
3. Enter a name for the printer and click **OK**.
4. Select the printer to be used in the Windows Printer Name field. This is most likely to be the printer you normally use when printing from the Windows environment. Click **OK**.
5. Select the new printer in the list and select Printer Available in the properties area. Click **Modify**.
6. Click **Done**.

Importing the Agile Business Suite DDA

1. Start Enterprise Output Manager if not already running.
2. Select **Tools** > **Configuration Control** > **Import Configuration**.
3. Enter, or browse for, the path to the directory where the file DDATTR.CFG is located.
4. Select Data Dependent Attribute in the Entity Type list.
5. Select LINC from the Entity Names list and click **OK**.
6. If requested, click **Yes** to confirm the change.
7. Click **Done**.

Defining a Print Attribute

1. Start Enterprise Output Manager if not already running.
2. Select **Configuration**, then **Print Attribute**.
3. Click **Add/Copy** Attribute.
4. Enter a name for the Print Attribute and click **OK**.
5. On the **Advanced** page, in the **Data Dependent** Attribute field, select **LINC**.
6. On the Hardware page, in the Printer Driver field, select **Windows Driver**.
7. Click **OK** and then **Done**.

Defining a File Mask

1. Start Enterprise Output Manager if not already running.
2. Select **Configuration**, then **File Mask**.
3. Click **Add Mask**.
4. Enter a name for the mask and click **OK**.
5. Double-click the new mask in the list of masks. The File Mask properties are displayed.
6. Select the Masked Fields page.
7. In the Mask Field Name Style list, select **LPR/LPD**.
8. If there is anything in the Masked Field Summary, select each line and delete them.
9. In the Field list, select **Host Queue**.
10. In the Operator list, select **EQ**.
11. Enter the queue name that you defined in Creating an LPR/LPD Printer.
12. Click **Add**.
13. From the list on the Print Jobs page, select the Print Attribute that you defined in Defining a Print Attribute.
14. In the Printer list, select the printer to be used for report printout.
15. Click **Add**.
16. Click **OK** and then **Done**.

Logic Editor

The Logic Editor in the Agile Business Suites Developer is used to create, edit, and save logic for methods within System Modeler. It can also be used to validate methods. These methods can include:

- Overridden Built-in scripts.
- Frame methods for reports.
- Segment methods.
- Insertable class methods.
- User-defined methods.
- Built-in SQL script methods – SQL commands cannot be validated, and consequently any SQL errors may not be detected until deployment.

To open Logic Editor for a method, perform either of the following:

- Double-click the method in the **Class View** or **Members** tab.
- Right-click the method in the **Class View** or **Members** tab and click **Open**.
- Right-click the method in the **Class View** or **Members** tab and click **Open With** and then select **Logic**.

Entering Logic

Use Logic Editor to enter logic for methods. Refer to Logic Commands for more information on the usage and syntax of logic.

Note: *SQL script methods are written in SQL. You should consult a SQL reference for details on the scripting of these commands.*

If you add attributes by dragging them from Class View to logic editor, full path of attribute is shown, for example Test.Cust.CustId. This is the designed behavior of Visual Studio. However, while validation the logic returns an error. So, you should always enter attributes in logic editor manually instead of dragging-and-dropping them from Class View.

Logic is not saved and/or validated unless the **Save** and **Validate** commands are used accordingly. Refer to Validating Logic for more information.

Getting help on logic commands

To get help on a specific logic command, move the cursor over the logic command text in the Logic Editor document window and press **F1**.

Logic Editor Status Information

Logic Editor status information is displayed on the Microsoft Visual Studio status bar. This information includes:

- The current cursor position (line number and column) in the Logic Editor document window.

  The current character position in the Logic Editor document window is not indicated.
- Help and warning messages.
- An overstrike (OVR) or insertion (INS) mode indicator for the Logic Editor document window.

  In overstrike mode, the cursor in the Logic Editor document window does not change its display to a block, unlike the Microsoft Visual Studio Code and Text Editor.

Using Member Lists

Specify the display of member lists using the Tools, Options, Text Editor, LDL+, General, **Auto list members** check box.

Using member lists

When a class name is appended with a period (.), a list of context-sensitive member items is displayed in the Logic Editor document window.

You can use the **List Members** command from the **Intellisense** submenu of the **Edit** menu to force display of a member list.

Navigate through a member list using the scroll bar, or the UP/DOWN ARROW keys. Shortcut to a particular item by typing the first few letters of its name.

Press **ENTER**, **TAB**, or double-click a member list item to insert its name into the logic.

Press **ESC** or click outside the list box to close the member list.

The following items are displayed in the member list, depending upon the current cursor position:

Cursor position	Member list items
Start of a line	Logic commands, attributes, and methods in scope
After a period (.)	For class names, attributes in the identified class
At any other position	Context determined unqualified variables or command options

Editing Logic

The following are some of the editing commands available from the **Edit** menu:

Command	Usage Notes
Navigate To	Navigates to the selected element in Class View.
Format Selection (Advanced submenu)	Formats the selection according to the **Command Style** defined in the **Options** menu. You can define the format in **Tools** > **Options** > **Text Editor** > **LDL+** > **Formatting** > **Command Style**. This option can be applied to all entered logic automatically in the **LDL+** > **Formatting**. • Automatically format on enter • Automatically format on paste Refer to Setting LDL+ Editor Options for more information on defining command format.
Format Document (Advanced Submenu)	Formats the document according to the **Command Style** defined in the **Options** menu. You can define the format in **Tools** > **Options** > **Text Editor** > **LDL+** > **Formatting** > **Command Style**. Refer to Setting LDL+ Editor Options for more information on defining command format.
Parameter Info (Intellisense submenu)	Displays the parameter information for the selected method. You can enable/disable this option in **Tools** > **Options** > **Text Editor** > **LDL+** > **Parameter information**.
Comment/Uncomment Selection	Inserts/removes a comment character from the selection.

Refer to the *Microsoft Visual Studio Online Help* for more information on these standard Microsoft Visual Studio editing commands.

Performing Quick Actions

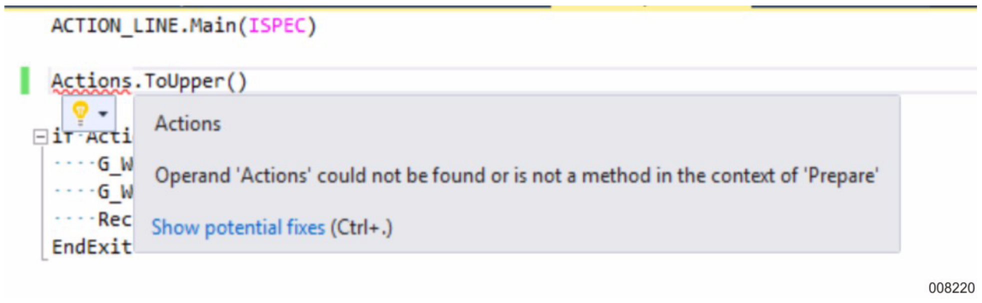
Quick Actions enables you to resolve errors in logic by selecting an appropriate solution from a list of suggested fixes. A Quick Action is denoted by the icon 💡▾. When you see a red squiggle, you can hover over it to display the quick action icon. Quick Actions provide suggested fixes to:

- Correct a misspelled element's name. E.g. Change 'Actions' to 'Action_Line'
- Create a variable of an appropriate type. E.g. Create Variable 'Prepare.Actions'
- Create a variable of user defined type. E.g. Create Variable 'Prepare.Actions' from a type

Note: *Ensure to select Quick Actions option in the* ***Tools*** *>* ***Options*** *menu. Refer to Setting LDL+ Editor Options for more information.*

To correct logic using Quick Actions, perform the following:

1. Right click the error indicated by red squiggle in editor and select **Quick Actions…**.



```
ACTION_LINE.Main(ISPEC)

Actions.ToUpper()
if Acti
    G_W
    G_W
    Rec
EndExit
```

Actions

Operand 'Actions' could not be found or is not a method in the context of 'Prepare'

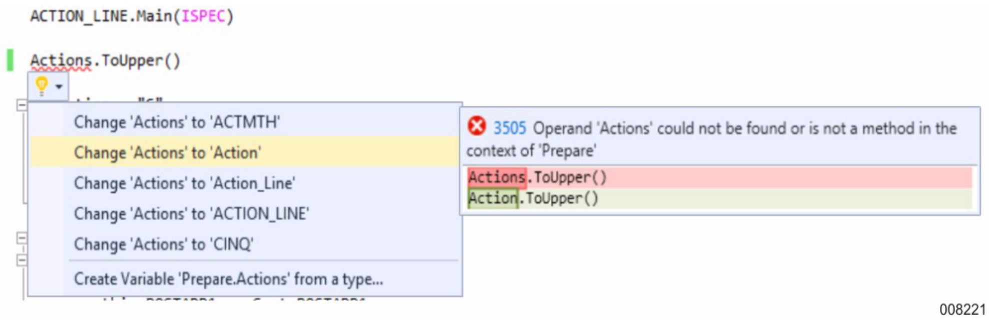
Show potential fixes (Ctrl+.)

008220

2. Click **Show potential fixes** link.

   A list of suggested fixes appears.

3. Select the appropriate fix from the list.



```
ACTION_LINE.Main(ISPEC)

Actions.ToUpper()
```

Change 'Actions' to 'ACTMTH'
Change 'Actions' to 'Action'
Change 'Actions' to 'Action_Line'
Change 'Actions' to 'ACTION_LINE'
Change 'Actions' to 'CINQ'
Create Variable 'Prepare.Actions' from a type...

3505 Operand 'Actions' could not be found or is not a method in the context of 'Prepare'

```
Actions.ToUpper()
Action.ToUpper()
```

008221

Notes:

- *You can use the shortcut key Ctrl + .(period) anywhere on a line of logic with errors to directly view the list of potential fixes.*
- *Quick action icon appears only when there is a valid selection.*
- *Potential fixes link appears only when there is a valid fix.*

Using Reverse Auto-Completion

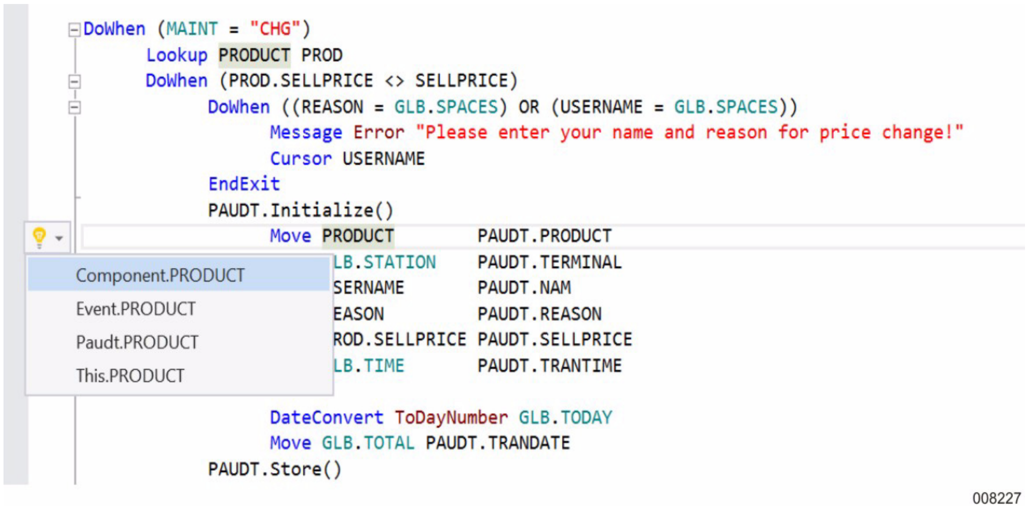
Reverse Auto-Completion feature of Agile Business Suite displays a list of qualified paths to all accessible instances of a named attribute. You can scroll through the displayed list and select a fully qualified name to replace the attribute name with. This is useful when you know the name of an attribute but do not know the qualification path to that attribute.

To invoke the auto-completion dialog, perform the following:

1. Type the attribute name in the context in which it is used.
2. Press **Ctrl+.** (period sign) at the cursor that appears immediately after the attribute name.

3. Select the appropriate choice from the list of possible qualification paths and press Enter to replace the partial path with the complete qualified path.

Note: *If the trailing part of the path is known, this can also be included with the attribute name. To do this, position the cursor immediately after the owner name and press Ctrl+. key combination.*
For example, you can use this feature to enter the correct qualification path for an attribute PRODUCT in Logic Editor. Type in the attribute name PRODUCT and press the Ctrl+. key combination, a list of qualification paths is displayed.



```
DoWhen (MAINT = "CHG")
    Lookup PRODUCT PROD
    DoWhen (PROD.SELLPRICE <> SELLPRICE)
        DoWhen ((REASON = GLB.SPACES) OR (USERNAME = GLB.SPACES))
            Message Error "Please enter your name and reason for price change!"
            Cursor USERNAME
        EndExit
        PAUDT.Initialize()
            Move PRODUCT       PAUDT.PRODUCT
                 LB.STATION    PAUDT.TERMINAL
                 SERNAME       PAUDT.NAM
                 EASON         PAUDT.REASON
                 ROD.SELLPRICE PAUDT.SELLPRICE
                 LB.TIME       PAUDT.TRANTIME

            DateConvert ToDayNumber GLB.TODAY
            Move GLB.TOTAL PAUDT.TRANDATE
        PAUDT.Store()
```

Accessible attributes with the same name may exist in different namespaces such as various Ispecs or Groups. Hence, the paths in the list will be qualified by 'Owner', 'Super', 'This' or 'Component' to provide an unambiguous and complete path for the attribute. The auto-completion list in this example includes the following:

- Component.PRODUCT
- Event.PRODUCT
- Paudt.PRODUCT
- This.PRODUCT

Working with Peek Definition

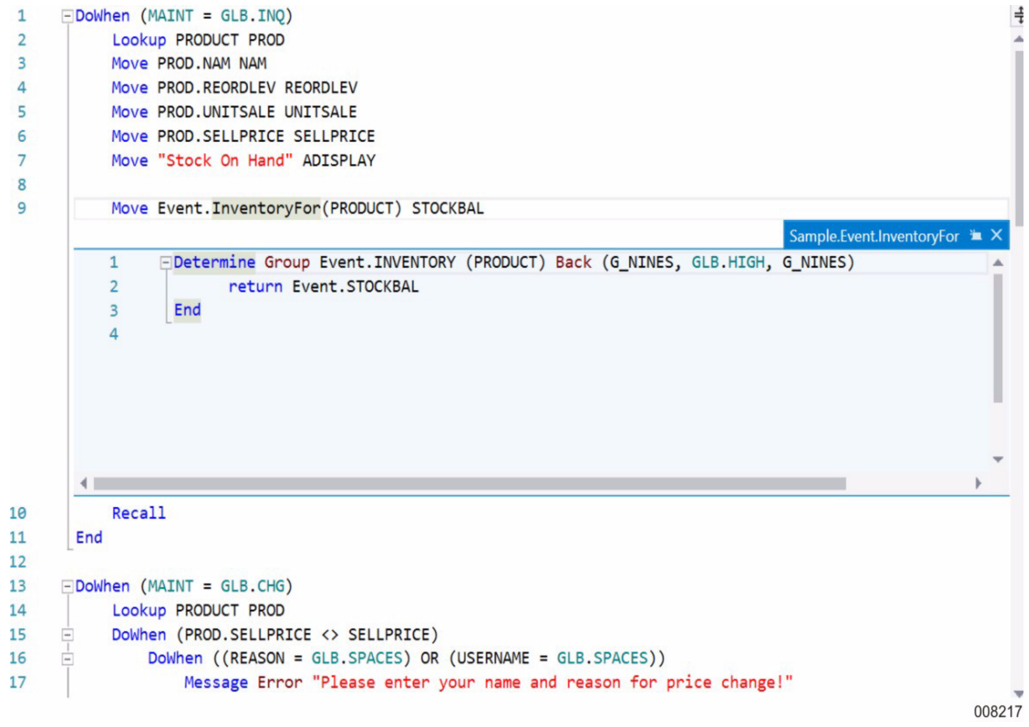
Peek Definition enables you to view the definition of an Insertable, Method or Profile and edit the definition inline, without switching away from the original logic. Using this command, you can quickly view one method without loosing your place in the original method.

Opening a Peek Definition Window

To open a peek definition window, perform the following:

1. Right-click the element for which you want to view definition.
2. Select **Peek Definition**.

Peek definition window appears below the selected element. This window does not hide any of the other code in the logic editor. The code following the selected element appears below the peek definition window.



```
DoWhen (MAINT = GLB.INQ)
    Lookup PRODUCT PROD
    Move PROD.NAM NAM
    Move PROD.REORDLEV REORDLEV
    Move PROD.UNITSALE UNITSALE
    Move PROD.SELLPRICE SELLPRICE
    Move "Stock On Hand" ADISPLAY

    Move Event.InventoryFor(PRODUCT) STOCKBAL
```

```
Determine Group Event.INVENTORY (PRODUCT) Back (G_NINES, GLB.HIGH, G_NINES)
        return Event.STOCKBAL
End
```

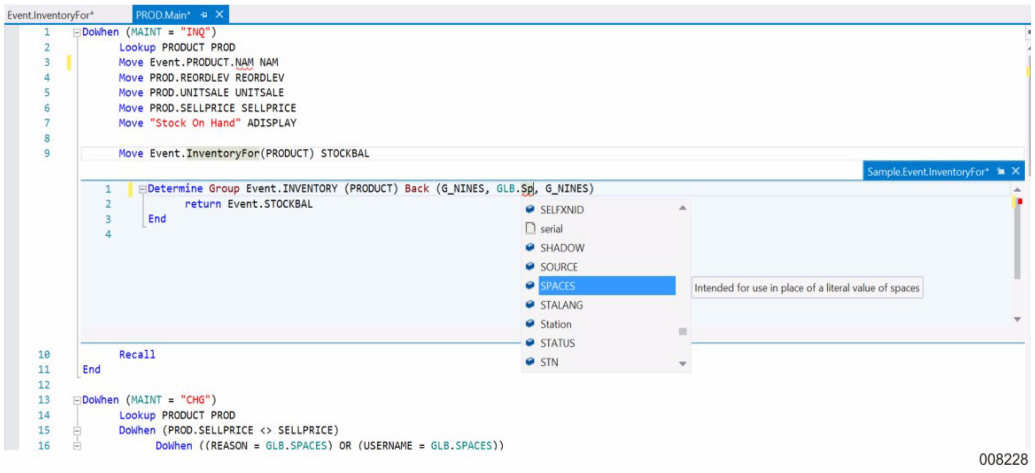
```
    Recall
End

DoWhen (MAINT = GLB.CHG)
    Lookup PRODUCT PROD
    DoWhen (PROD.SELLPRICE <> SELLPRICE)
        DoWhen ((REASON = GLB.SPACES) OR (USERNAME = GLB.SPACES))
            Message Error "Please enter your name and reason for price change!"
```

Notes:

- *Press Esc key or click Close icon in the definition window to close the definition window.*
- *Click Promote to Document icon to promote the peek definition to a standard document window.*

Editing the Definition

You can edit and save the logic in the peek definition window. When you start editing the definition, the logic that you are editing automatically opens in a separate tab in the code editor. All the edit, undo, and save changes made in the Peek Definition window reflect in the code in this tab.



```
DoWhen (MAINT = "INQ")
       Lookup PRODUCT PROD
       Move Event.PRODUCT.NAM NAM
       Move PROD.REORDLEV REORDLEV
       Move PROD.UNITSALE UNITSALE
       Move PROD.SELLPRICE SELLPRICE
       Move "Stock On Hand" ADISPLAY

       Move Event.InventoryFor(PRODUCT) STOCKBAL
```

```
Determine Group Event.INVENTORY (PRODUCT) Back (G_NINES, GLB.Sp, G_NINES)
       return Event.STOCKBAL
End
```

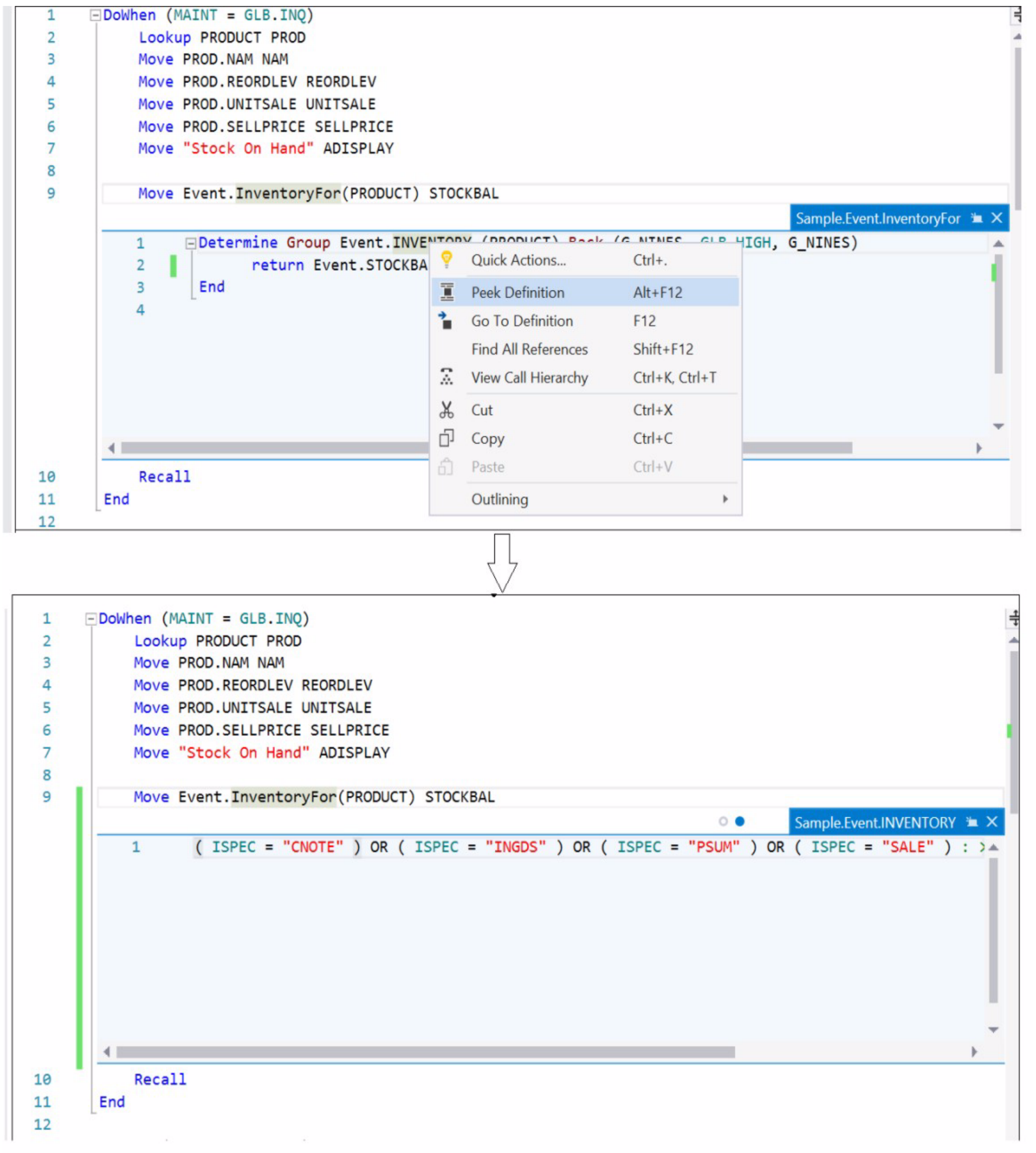
Intended for use in place of a literal value of spaces

```
       Recall
End

DoWhen (MAINT = "CHG")
       Lookup PRODUCT PROD
       DoWhen (PROD.SELLPRICE <> SELLPRICE)
              DoWhen ((REASON = GLB.SPACES) OR (USERNAME = GLB.SPACES))
```

008228

Opening a Peek Definition window from within a Peek Definition window

You can open another peek definition window from the peek definition window. When you open a definition window within the peek definition window, a set of breadcrumb dots icon appear. You can navigate between peek definition windows using these breadcrumb dots. The tooltip on each breadcrumb dot displays the file name and path of the definition file.



```
DoWhen (MAINT = GLB.INQ)
    Lookup PRODUCT PROD
    Move PROD.NAM NAM
    Move PROD.REORDLEV REORDLEV
    Move PROD.UNITSALE UNITSALE
    Move PROD.SELLPRICE SELLPRICE
    Move "Stock On Hand" ADISPLAY

    Move Event.InventoryFor(PRODUCT) STOCKBAL
```

```
Determine Group Event.INVENTORY (PRODUCT) Pack (G_NINES, GLB.HIGH, G_NINES)
        return Event.STOCKBA
End
```

```
    Recall
End
```

```
DoWhen (MAINT = GLB.INQ)
    Lookup PRODUCT PROD
    Move PROD.NAM NAM
    Move PROD.REORDLEV REORDLEV
    Move PROD.UNITSALE UNITSALE
    Move PROD.SELLPRICE SELLPRICE
    Move "Stock On Hand" ADISPLAY

    Move Event.InventoryFor(PRODUCT) STOCKBAL
```

```
( ISPEC = "CNOTE" ) OR ( ISPEC = "INGDS" ) OR ( ISPEC = "PSUM" ) OR ( ISPEC = "SALE" ) :
```

```
    Recall
End
```

008219

Note: *You can navigate between different peek definition windows using the set of breadcrumbs appearing above the definition window.*

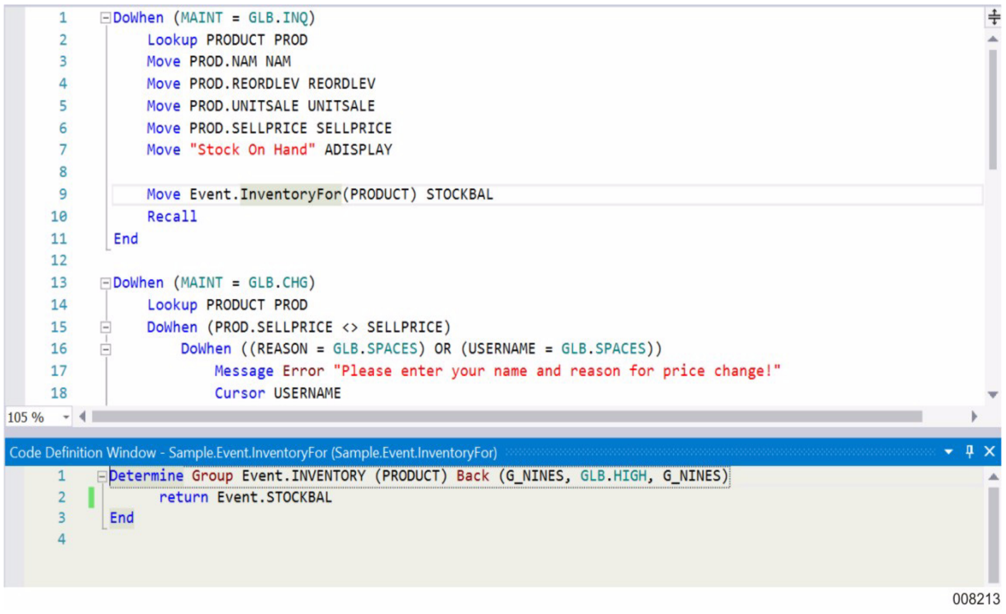
Working with Code Definition Window

Code definition window displays the definition of the currently selected Insertable, Method or Profile. You can only view definition for element in this window and cannot edit it.

To view definition for element, perform the following:

1. Select the element in editor.
2. On the **View** menu, select **Code Definition Window**.

   Code Definition Window appears below the logic editor window.



```
DoWhen (MAINT = GLB.INQ)
    Lookup PRODUCT PROD
    Move PROD.NAM NAM
    Move PROD.REORDLEV REORDLEV
    Move PROD.UNITSALE UNITSALE
    Move PROD.SELLPRICE SELLPRICE
    Move "Stock On Hand" ADISPLAY

    Move Event.InventoryFor(PRODUCT) STOCKBAL
    Recall
End

DoWhen (MAINT = GLB.CHG)
    Lookup PRODUCT PROD
    DoWhen (PROD.SELLPRICE <> SELLPRICE)
        DoWhen ((REASON = GLB.SPACES) OR (USERNAME = GLB.SPACES))
            Message Error "Please enter your name and reason for price change!"
            Cursor USERNAME
```

105 %
Code Definition Window - Sample.Event.InventoryFor (Sample.Event.InventoryFor)

```
Determine Group Event.INVENTORY (PRODUCT) Back (G_NINES, GLB.HIGH, G_NINES)
        return Event.STOCKBAL
End
```

008213

Validating Logic

Validate logic using the Validate command.

To validate logic in Logic Editor, perform the following:

1. Open the Logic Editor for the method you want to validate.
2. From the **Build** menu, select **Validate**.

To validate logic in Class View, perform the following:

1. In the Class View, select the methods you want to validate.

   You can also select one or more classes if you want to validate all the methods they own.

2. From the **Build** menu, select **Validate**.

Note: *You can also use the shortcut key Ctrl+F7 to validate logic in the class view and in the logic editor.*

The validation process inspects both the syntax and semantics of the logic.

Syntax checking

Checks that each non-empty line of logic contains a valid logic command, and that each logic command contains an appropriate number of parameters.

Semantic validation

Validates each logic command, in the context of checking that any attributes and/or classes referred to by the logic command exist and have valid attributes. For example, whether arguments of an Add logic command are number-primitives.

Any insertable classes are expanded before the validation process begins. Insertable classes are not required to be semantically complete. Refer to Validating Insertable Classes in Isolation for more information on validating insertable classes in isolation.

Validation errors are displayed in the Error list.

Validate

Logic is validated using the Validate command.

One of the following procedures can be used to validate logic:

- To validate a single method, initiate Logic Editor for the desired method, then from the Build menu, select **Validate**.
- To validate an entire entity, select the desired entity in the Class view, Members list, or Logic Status list; then from the Build menu, select Validate. Alternatively, right-click the desired entity in the Class view, Members list, or Logic Status list; then select Validate <element Name>.

Logic is first saved and validated. Errors, if any, are displayed in the Error list. The number of errors found is displayed on the Microsoft Visual Studio status bar.

Customize the behavior of the Validate command using the **Options** command from the **Tools** menu. Refer to Setting Validation Options for more information.

The Validate command can also be executed using a command line interface. The following command line syntax is used for validating the elements in a model.

```
Validate.exe ((/m ModelName & /s ServerName) | /p ProjectName) /f FileName [/t][/r]
```

Where:

Parameter	Description
[]	Denotes an optional argument.
/m or /M ModelName	Identifies the name of the model database to be used. ModelName is a required parameter.
/s or /S ServerName	Identifies the database instance with the model database. ServerName is a required parameter.
/p or /P ProjectFile	Identifies the name of the project file to be validated. This field must include full path and the project file name and file extension. ProjectFile is a required parameter.
/f or /F FileName	Identifies the name and path of the file holding a list of fully qualified names to be validated. FileName is a required parameter.
/t or /T	Prefixes all output lines with a date and time stamp. This is an optional parameter.
/r or /R	Reset validation status for all logics to force revalidation. This is an optional parameter.

For help on these parameters, enter **Validate /?** at the command prompt.

If you wish to capture output to a file for logging purposes, use the following command as an example:

```
Validate  /m meta  /s serverA  /f c:\temp\input.txt  >> c:\temp\validate.txt
```

One or more of the following output messages may be produced:

Message Type	Message
Error	Fatal Error: <exception message>
Error	Model <*ModelName*> not found on Server <*ServerName*>
Error	Model or Project name missing
Error	Project file <*ProjectName*> not found, check name and path.
Error	Cannot open file <*FileName*>, error = <*ErrorCode*>
Error	No parameter
Error	Invalid parameter
Error	Model name missing
Error	Server name missing
Error	File name missing

Message Type	**Message**
Informational	>>> START validate of <*FileNameEntry*>
Informational	Element: <ElementName>, Validate completed UNSUCCESSFULLY
Informational	Element: <*ElementName*>, Validate completed SUCCESSFULLY
Informational	Element: <*ElementName*> not found in Model
Informational	Element: <*ElementName*>, is external and cannot be validated.
Informational	Element: <*ElementName*>, Validate FAILED
Informational	Element: <*ElementName*>, Found no elements to validate
Informational	<*TotalElementCount*> Element(s) validated <*TotalErrorCount*> in error
Informational	>>> END validate of <*FileNameEntry*>

With Informational messages, the processing continues by reading next record from the file specified by the "/f" parameter.

Any Error messages prevent completion of the operation.

Note: *You can cancel validation using the Cancel button, which is part of the Build/ Compile button group.*

Logic Validation Errors

You can use Logic Editor to resolve logic validation errors. To get help on a specific error item, click the error item in the Error list and press **F1**.

Validation errors are displayed in the Validation Output and standard Error window provided by the Visual Studio IDE. The errors are displayed in the Error list, which includes both direct errors and those indirectly resulting from Model changes. When the logic is validated and errors are detected, the Error list is displayed. The **Errors** command from the **Other Windows** submenu of the **View** menu can also be used to display or close the Error list.

You can navigate to the corresponding line of logic from the Output window and Error list window. You can view the Help on an error item by right-clicking an error item from the Error Window and selecting Show Help.

Model Changes

Logic is sensitive to changes to the application model and consequently logic validation errors may arise as a result of changes to the application model external to Logic Editor.

For example, the logic may include an automatic entry to an ispec class, and this logic validated without error. However if the receiving ispec class is subsequently removed from the application model, the logic then needs re-validating.

Microsoft Build Engine

Microsoft Build Engine (MSBuild) allows you to build an AB Suite application without using the Visual Studio environment. It also provides you the flexibility to use the powerful functionality of Team Foundation Server (TFS), which enables you to build an AB Suite project on the TFS server while running a build definition on a TFS client.

An AB Suite application is directly built from a Command line by using the command line arguments or by reading the settings from a customized MSBuildSettings.xml file.

Refer to the MSDN documentation for more information on MSBuild.

Using Command Line Arguments for Building

The MSBuild builds applications by using a series of command line arguments. To build an AB Suite application from a command line, perform the following without using Visual Studio environment:

1. Point to the bottom-left corner of the screen to enable the Start icon, click **Start**, and then click **Run**.
2. Type **cmd** in the Run dialog box and press **Enter** to open a Command Prompt.

   Note: *You must run the command prompt as Administrator to build the System Modeler with MS Build.*

3. In the command prompt, change the working directory to:
   ```
   C:\Windows\Microsoft.NET\Framework\vX.X.XXXXX
   ```
   Note: *vX.X.XXXXX is the latest.Net Framework version.*
4. Type "MSBuild.exe /?". This displays the complete usage of the utility.

Using Visual Studio environment:

1. Point to the bottom-left corner of the screen to enable the Start icon, click **Start**, type and select Developer Command Prompt for VS2015 on the desktop.

   Note: *If the Developer Command Prompt is not present on the desktop, pin the application to the desktop. Refer to the Windows 8 documentation for more information.*

2. Type "MSBuild.exe /?". This displays the complete usage of the utility.

Building Applications Using Command Line

The syntax of MSBuild command line to build a project includes several switches to specify the various features of the building process:

```
MSBuild.exe [ProjectFile] [Switches]
```

The syntax of MSBuild command line arguments for an AB Suite application is as follows:

Without using MSBuild Settings File

```
MSBuild.exe <ProjectPath> </p:c> </p:pf> [/p:d]  [/p:s] [/p:el] [/p:ef] [/p:u] [/
p:bp] [/p:ac][/p:ap][/p:cc][/p:fc][/p:fp] [/p:fa] [/p:fob] [/p:od] [/p:bd] [/p:sdp]
[/p:edp] [/p:rca] [/target] [/flp:logfile] [/nologo / noconsolelogger]
```

Using MSBuild Settings File

```
MSBuild.exe <ProjectPath> </p: MSBuildSettingsFile> [/p:c] [/p:pf] [/p:d]  [/p:s]
[p:el] [/p:ef] [/p:u] [/p:bp] [/p:ac][/p:ap][/p:cc][/p:fc][/p:fp] [/p:fa] [/p:fob] [/
p:od] [/p:bd] [/p:sdp] [/p:edp] [/p:rca] [/target] [/flp:logfile] [/nologo /
noconsolelogger]
```

Examples:

- Msbuild "C:\MSBuildSample.smproj" /Property: MSBuildSettingsFile="c:\ MSBuildSettings.xml"
- Msbuild "C:\MSBuildSample.smproj" /target:clean /Property: MSBuildSettingsFile="c:\ MSBuildSettings.xml"
- Msbuild "C:\MSBuildSample.smproj" / Property:Platform=Windows;Configuration=Release;UserCode=Cust;Password=Cus t
- Msbuild "C:\MSBuildSample.smproj" / Property:Platform=Windows;Configuration=Release; UserCode=Cust;Password=Cust;ElementList= EAESAMPLE.Reports; EAESAMPLE.ORDER_ENTRY.COMP_ORD

The following tables lists the command line arguments used to build an AB Suite application, which are listed as Common, Windows, and MCP switches:

Common Switches

Switch	Short Form	Description
ProjectPath		Specifies the complete path of an AB Suite project file that needs to be built. This input is mandatory for MSBuild.exe.
Target	/t	Specifies the target to be built. For example: /target: rebuild
Property	/p	Specifies the properties to build or rebuild an application. For example: /p:c=release;pf= windows
LogFile	/flp	Specifies the name of the file where logging information is stored. A log file is not created if a file name is not specified. For example: /flp=MyLogFileFullName

Switch	Short Form	Description
NoConsoleLogg er	/nologo	Stops displaying logging message at the command prompt. By default, logging messages are displayed. For example: /noconsolelogger /nologo

Target Switches

Switch Value	Short Form	Description
Clean		Specifies removing the generated files for the selected model. The cleaned elements are rebuilt next time an application is built. For example: /target:Clean ***Notes:*** • For MS Build, the **/target: Clean** command must not be used with **/el** command. • For builder.exe, the **/target: Clean** command must not be used with **/re** and **/rf** commands.
Rebuild	b	The elements associated with the build are rebuilt even if they have not changed since the last build. For example: /target:Rebuild

Property Switches

Switch	Short Form	Description
Database	d	Specifies the name of a model containing the application you wish to build. For example: /Property:d=MyDatbaseName ***Note:*** *This parameter is optional if the "sf" option is used and if you specify the "<Database>" option in the builder setting file.*
Configuration	c	Specifies the name of a configuration that you wish to build. For example: /Property:c=Release ***Note:*** *This parameter is optional if the "sf" option is used and if you specify the "<Configuration>" option in the builder setting file.*

Switch	Short Form	Description
Platform	pf	Specifies the name of a platform on which the application is built. For example: /Property:pf=MCP ***Note:*** *This parameter is optional if the "sf" option is used and if you specify the "<Platform>" option in the builder setting file.*
MSBuildSettingsFile	sf	Specifies the path to the MSBuilder Settings File. For example: /Property:sf= MSBuildSettingsFileFullName ***Note:*** *MSBuilder Settings File is an XML file and contains all the build details to build an AB Suite application.*
SQLServer	s	Specifies the SQL Server that contains the model stated previously. The server is the name of the SQL Server that contains the model and "(Local)" is the default local SQL Server instance. For example: /Property:s=MyDatabaseInstance
ElementsList	el	Specifies a deployable element, fully qualified name of a single report, fully qualified name of a single ispec in the model that you wish to build. For example: /Property:el=segment1;segment2;
ElementsFile	ef	Specifies a file name containing a list of elements you wish to build. For example: /Property:ef=MyElementsFileFullName
UserCode	u	Specifies the user code to connect to the host machine. For example: /Property:u=MyUserCode
Password	p	Specifies the password used to connect to the host machine. For example: /Propert:p=MyPassword
FolderOnlyBuild	fob	Builds the specified folder and does not build the sub folders. If a deployable element argument is not specified, this argument is ignored. This means, the whole model is built using the "d" <model name> switch. For example: /Property:fob=True

Switch	Short Form	Description
OverwriteDatabase	od	If set to true, overwrites the existing database. For example: /Property:od=True
BackupDatabase	bd	If set to true, creates a backup of the runtime database. For example: /Property:bd=True
BuildPreview	bp	If set to true, the build displays a summary of the elements that require regeneration. For example: /Property:bp=True

Windows Switches for Property

Switch	Short Form	Description
StartDeployPhase	sdp	Specifies the initiation of the deploy phase for a build. For example: /Property: sdp =generate
EndDeployPhase	edp	Specifies the termination of the deploy phase for a build. For example: /Property: edp =generate
EnableRCA	rca	If set to true, enables to analyze the online changes for reports; that is, not ignoring the dependencies on report build. For example: /Property: rca =True

MCP Switches for Property

Switch	Short Form	Description
AccessCode	ac	Specifies the Access Code for an MCP host. For example: /Propert:ac=MyAccessCode
AccessPassword	ap	Specifies the Access Password for an MCP host. For example: /Property:ac=MyAccessPassword
ChargeCode	cc	Specifies the Charge Code for an MCP host. For example: /Propertycc=MyChargeCode

Switch	Short Form	Description
EnableRCA	rca	If set to true, • Enables to analyze the online changes for reports; that is, not ignoring the dependencies on report build. For example: /Property: rca =True
FTPUserCode	fu	Specifies the FTP User Code for an MCP host. For example: /Property:fc=MyFTPUserCode
FTPPassword	fp	Specifies the FTP Password for an MCP host. For example: /Property:fp=MyFTPPassword
FTPAccount	fa	Specifies the FTP Charge Code for an MCP host. For example: /Property:fa=MyFTPAccount

The following MSBuild switches are supported but do not have any effect on AB Suite build and there is no change in the MSBuildSettings.xml file:

- /maxcpucount [:n]
- /toolsversion:<version>
- verbosity:<level>
- /consoleloggerparameters
- /fileLogger[n]
- /fileloggerparameters[n]:<parameters>
- /distributedFileLogger
- /logger:<logger>
- /validate
- /validate:<schema>
- /ignoreprojectextenstions : <extensions>
- /nodeReuse:<parameters>
- /preprocess[:file]
- /detailedsummary
- @<file>
- /noautoresponse

- /version
- /maxcpucount [:n]

Note: *Refer to* MSDN documentation *for more information on the above mentioned MSBuild options.*

MSBuild Settings File

The MSBuildSettings.xml file consists of all the build details to build and clean the AB Suite application. You can modify the value of any item with customized values and MSBuild.exe reads the values to build or clean AB Suite application.

A sample MSBuildSettings.xml file is provided in the bin directory of AB Suite for you to use. The following is the content of the sample MSBuildSettings.xml file:

```
<?xml version="1.0" ?>
<configuration xmlns="http://tempuri.org/config.xsd" version="1.0">
    <Builder>
    <Database></Database>
    <SQLServer></SQLServer>
    <ImportProject>false</ImportProject>
    <BuildPreview>false</BuildPreview>
        <FolderOnlyBuild>false</FolderOnlyBuild>
        <Configuration>Release</Configuration>
        <Platform>Windows</Platform>
        <ElementList>
            <Element></Element>
        </ElementList>
        <BackupDatabase>true</BackupDatabase>
        <EnableRCA>false</EnableRCA>
        <OverwriteDatabase>false</OverwriteDatabase>
        <Windows>
            <UserCode>Windows UserCode</UserCode>
            <Password>Windows password</Password>
            <Domain>Windows domain</Domain>
            <StartDeployPhase>Generate</StartDeployPhase>
            <EndDeployPhase>Generate</EndDeployPhase>
        </Windows>
        <MCP>
            <UserCode>MCP UserCode</UserCode>
            <Password>MCP Password</Password>
            <AccessCode>MCP AccessCode</AccessCode>
            <AccessPassword>MCP Access Password</AccessPassword>
            <ChargeCode>MCP Charge Code</ChargeCode>
            <FTPUserCode>FTP UserCode</FTPUserCode>
            <FTPPassword>FTP Password</FTPPassword>
            <FTPAccount>FTP Account</FTPAccount>
            <DelayDate>YYYYMMDD</DelayDate>
            <DelayTime>HHMM</DelayTime>
            <NumberOfParallelGenerates>2</NumberOfParallelGenerates>
            <NumberOfReorgTasks>1</NumberOfReorgTasks>
            <NumberOfParallelCompiles>1</NumberOfParallelCompiles>
        </MCP>
    </Builder>
</configuration>
```

Meaning of the XML Tags in MSBuildSettings.xml

The following table lists the XML tags in the MSBuildSettings.xml file for each platform, which are listed as common, Windows, and MCP tags:

Common XML tags

XML Tags	Description
Database	Specifies the name of the model containing the application you wish to build.
SqlServer	Specifies the SQL Server that contains the model stated previously. The server is the name of the SQL Server that contains the model and. "(Local)" is the default local SQL Server instance.
ImportProject	If set to true, the project is imported.
Configuration	Specifies the name of the configuration you wish to build.
UserCode	Specifies the user code to connect to the host machine.
Password	Specifies the password used to connect to the host machine.
Platform	Specifies the name of the platform on which the application is built.
Elementlist	Specifies the elements that are built apart from the model.
Backupdatabase	If set to true, creates a backup of the runtime database.
OverwriteDatabase	If set to true, overwrites the existing database.
BuildPreview	If set to true, displays a summary of the elements that require regeneration.
FolderOnlyBuild	Builds the specified folder and does not build the subfolders. If a deployable Element argument is not specified, this argument is ignored. This means, the whole model is built using the "d" <model name> switch.

Windows XML tags

XML Tags	Description
Domain	Specifies the domain name.

MCP XML tags

XML Tags	Description
AccessCode	Specifies the MCP AccessCode set on the target host under which the application is generated.
AccessPassword	Specifies the Password of the Accesscode used in the AccessCode tag.

XML Tags	Description
ChargeCode	Specifies the MCP ChargeCode set on the target host.
FTPUserCode	Specifies a valid usercode for FTP connection to the host. This usercode must be defined as a valid user in the FTP Server software on the specified target host. You must first log into the Builder Server by using MCP login details before generating a System or Report,. The information required to log in is determined by the target host. In addition, an FTP session must be established for all hosts and the FTP login details must be specified.
FTPPassword	Specifies the password of the FTP username specified in the FTPUserCode
FTPAccount	Specifies the MCP chargecode for the FTP Server software on the specified target host.
DelayDate	Specifies the date to delay the host part of the build.
DelayTime	Specifies the time to delay the host part of the build.
NumberOfParallelGenerates	Specifies the number of Build threads used.
NumberOfReorgTasks	Specifies the maximum number of reorganization tasks that can occur simultaneously. By default, the value is 1 and can range from 1 to 9999.
NumberOfParallelCompiles	Specifies the number of compile threads (multithreading) running concurrently. By default, the value is 1 and can range from 1 to 9999.

Building Applications by Using TFS

MSBuild allows you to build AB Suite applications on TFS. To build an application from TFS, you must create a build definition.

Note: *By opening an AB Suite solution, default values are assigned to the required fields.*

To create a Build Definition without a solution file being open, perform the following:

Connecting to TFS

You must connect to TFS before creating a build definition. To connect to TFS, perform the following:

1. On the **Team** menu, click **Manage Connections**.

   The **Team Explorer** window appears.
2. Click **Manage Connections** > **Connect to Team Project**.

   The **Connect to Team Foundation Server** dialog box appears
3. Select **Servers...** to select a server or to add a new server.

4. Select **Team Project Collections** where the AB Suite Team Project is located.
5. Select **Team Projects** where the AB Suite version files are located.
6. Click **Connect**.

Creating a Build Definition

You must create and configure a build definition to build an application. To create and configure build definitions, perform the following. The following example uses TFS 2015 to build the system for Windows runtime:

1. Select **New Build Definition** on the Team Explorer window from the Build context menu.

   The **New Build Definition** dialog box appears.

   Note: *Create a build agent before creating a New Build definition. Refer to the Microsoft documentation for more information on configuring and managing your build systems.*
2. Click **Add build step...**.

   The **ADD BUILD STEPS** dialog box appears.
3. In the **ADD BUILD STEPS** dialog box, select **Build**.
4. Scroll through the displayed list and locate to **Visual Studio Build**.
5. Click **Add** to add the visual studio build.
6. Click **Close** to close the **ADD BUILD STEPS** dialog box

   A build definition is created.

Note: *To rename the build definition, right-click the build definition and click* ***Rename...****. Enter the Name and click* ***OK****.*

To set project, MSBuild Arguments, configuration, and platform parameters under the **Build** tab, perform the following:

1. In the **Solution** field, browse the AB Suite project (System Modeler project (.smproj)) file that needs to be build.
2. Enter the build arguments in the **MS Build arguments** field in the following format:
```
/t:buildmode /p:s=<build_sqlserver>;d=<model_database>;u=<host_usercode>;
p=<host_password>;sdp=generate;edp=install;ip=true
```
   Description for various place holders in the build argument format is given in the following table:

Switch/ Property	Name	Description
/t:buildmode	Target	Build mode: rebuild, build, or clean.

Switch/ Property	Name	Description
s	SQLServer	SQL Server that contains the AB Suite model to build. **Local** is the default local SQL Server instance.
d	Database	Name of a database containing the application you want to build.
u	UserCode	User code to connect to the host machine.
p	Password	Password used to connect to the host machine.
sdp	Start Deployment Phase	Generate MSI package is compiled and created
edp	End Deployment Phase	Install Installs the package in the Package Installation Directory
ip	ImportProject	The project will be imported and a new model database (specified by the property d) will be created if the database does not exist when this property is set to true.

3. Enter **Release** in the **Configuration** field.
4. Enter **Windows** in the **Platform** field.
5. Expand the **Advanced** node and select **MSBuild x86** in **MSBuild Architecture**.

On the **Options** tab, perform the following:

1. Select the **Multiconfiguration** checkbox if you want to continue running other configurations in case of any failure in the current configuration settings.

   Or

   Clear the **Multiconfiguration** checkbox if you do not want to continue running other configurations in case of any failure in the current configuration settings.

On the **Repository** tab, perform the following:

1. Select **Team Foundation Version Control** from the list as **Repository type**.
2. **Repository name** field displays the repository name.

On the **Triggers** tab, perform the following:

1. Select **Continuous integration (CI)** to build each check in.
2. Select **Scheduled** to schedule the build every week on a following day.

On the **General** tab, perform the following:

1. Select the **Default Queue** from the displayed list.
2. Enter the build description in the **Description** field.
3. Specify the maximum time (in minutes) for which the build can be executed on an agent before being cancelled by server in the **Build job timeout in minutes** field.

Using Build Definitions

You can add the build definitions to a queue and trigger a build for an application. To set a queue for a build definition, perform the following:

1. Select your application from the **Team Explorer** window, right-click and select **Queue New Build**.

   The **Queue Build** dialog box appears.
2. Click **Queue**.

   The build begins and a message is displayed upon completion.

To view the build summary, perform the following:

1. Select **My Builds** on the **Team Explorer** window.

   Builds created are listed under **My Builds**.
2. Double-click the build for which you want to view the build summary.

   The build summary page displaying build details appears.
3. Click **Download all logs as zip** to download all build logs.

ReEIDor Utility

ReEIDor.exe is a standalone command line utility used for reformatting and refactoring LDL+ logic on the models migrated from EAE 3r3. It performs the following functions:

- Minimize Insertable substitution strings (Refactor LDL+ logic to reduce insertable substrings)
- Pretty Print (Reformat LDL+ logic using specified rules)
- Change insertable attribute to class for the correct usage of insertable

The utility logs its activities in a log file that is created in the working directory (by default, the utility creates a ReEIDor.log in C:\ProgramData\Unisys\ABSuite\6.1\ReEIdor\ReEIdor.log unless a log file name is provided using the –L parameter of the ReEIDor utility).

Note: *Backup the model before executing this utility so that you can revert to the original model, if the changes made by the utility are not acceptable.*

Minimize Insertable Substitution Strings

To minimize the Insertable substitution strings use the –I command line option of the ReEIDor utility. It walks through the specified model, and analyzes each insertable and the methods that inserts the insertable, and updates the insertable substitution strings based on some of the following rules:

1. Remove all the substitution strings that are exactly same in every Insert statement.

   Example

   ```
   // Before Refactor Insert statements
   Insert IGLG ( _DA1 = ABC.DA1 & _DA2 = ABC.DA2)
   Insert IGLG ( _DA1 = ABC.DA1 & _DA2 = XYZ.DA2)
   // Before Refactor Insertable IGLG logic
   Move "X" _DA1
   Move "Y" _DA2

   // After Refactor Insert statements
   Insert IGLG ( _DA2 = ABC.DA2)
   Insert IGLG ( _DA2 = XYZ.DA2)
   // After Refactor Insertable IGLG logic
   Move "X" ABC.DA1
   Move "Y" _DA2
   ```

2. Remove common qualifiers from qualified names in substitution strings.

   Example

   ```
   //// Before Refactor Insert statements
   Insert IGLG ( _DA1 = GRPA.GRPB.DA1 & _DA2 = GRPA.GRPB.DA2 & _DA3 = GRPA.GRPB.DA3)
   Insert IGLG ( _DA1 = GRPX.GRPY.DA1 & _DA2 = GRPX.GRPY.DA2 & _DA3 = GRPX.GRPY.DA3)
   // Before Refactor Insertable IGLG logic
   Move 1 _DA1
   Move 2 _DA2
   Move 3 _DA3
   // After Refactor Insert statements
   Insert IGLG ( _PARAM1 = GRPA.GRPB)
   Insert IGLG ( _PARAM1 = GRPX.GRPY)
   ```

```
// After Refactor Insertable IGLG logic
Move 1 _PARAM1.DA1
Move 2 _PARAM1.DA2
Move 3 _PARAM1.DA3
```

Command line options for minimizing Insertable substitution strings

The command line for refactoring or reformatting insertable substitution strings is:

```
ReEldor -Insertables(I) [-SQLServer(SS) <SQLServer Instance name>] -Model(M) <model
name> [-Segment(S) <segment name>] [-Element(E) <element name>] [-NoUpdates] [-
NoValidate] [-LogFile(L) <file name>]
```

Where:

Syntax	Description
Insertables(I)	Is a necessary parameter to enable insertable substitution string refactoring.
SQLServer Instance name	Named instance of the SQL Server to connect to. If not specified, the default "local" instance is used.
Model(M)	Name of the AB Suite Model to refactor. This is a necessary parameter and can be used in the short form as –M instead of –Model.
Segment(S)	Name of the Segment to refactor. This is optional parameter and defaults to the Model name, if not specified.
Element(E)	Name of the Element/Folder to refactor. This is optional and defaults to ALL elements in the Segment.
NoUpdates	Only shows all the changes that could have been made. This does not update the Model. By default, the behavior is to update the Model.
NoValidate	Does not validate any logic after the changes have been applied. By default, the behavior is to validate.
LogFile(L)	Name of log file. By default, it is ReEldor.log.

Pretty Print

Pretty Print utility can be initiated using the –PP or –PrettyPrint parameter of ReElDor utility. It walks through the specified model, retrieves the logic for each method from the model, applies the formatting options (as specified in the command line options for Pretty Print), and updates the corresponding method with the newly reformatted logic.

Note: *The utility uses internal structures stored in the model to perform its activities. These internal structures are created during the validation phase (validation either through import, developer, or build). So before initiating this tool, ensure that the model has been validated.*

Command Line Options for Pretty Print

The command line for applying Pretty Print is:

```
ReEldor –PrettyPrint(PP) [-SQLServer(SS) <SQLServer Instance name>] -Model(M) <model
name> [-Segment(S) <segment name>] [-Element(E) <element name>] [-NoUpdates] [-
NoValidate] [-LogFile(L) <file name>] [-EndOfLineCommentColumn(EOLCC) <int>] [-
CommandAbbreviated(CA) | -CommandMixedCase(CMC) | -CommandUpperCase(CUC)] [-
DoWhen(DW) | -If] [-Operand1Col(O1C) <int>] [-Operand2Col(O2C) <int>] [-
OperandColumnAbsolute(OCA)]) [-TabSize(TS) <int>]  [-IndentCase(IC)]  [-
ReplaceTabsWithSpaces(RTWS)] [-MaximumLineLength(ML) <int> [-
ContinuationIndentCount(CI) <int>]]
```

Where:

Syntax	Description
Model (M)	Name of AB Suite Model to refactor. This is a necessary parameter and can be used as a short form of –M instead of –Model.
SQL Server (SS)	Specifies the name of the SQLServer instance that contains the AB Suite model to refactor. This parameter is optional.
Segment (S)	Name of Segment to refactor. This is optional parameter and defaults to the Model name, if not specified.
Element (E)	Name of Element/Folder to refactor. This is optional and defaults to ALL elements in the Segment.
NoUpdates	Only shows all the changes that could have been made. This does not update the Model. By default, the behavior is to update the model.
NoValidate	Does not validate any logic after the changes have been applied. By default, the behavior is to validate.
LogFile (L)	Name of the log file. By default, it is ReEldor.log.
EndOfLineCommentColumn (EOLCC)	The start column for end of line comments
CommandAbbreviated (CA)	Displays commands in abbreviated form.
CommandMixedCase (CMC)	Displays commands in mixed case. It is the default format.
CommandUpperCase (CUC)	Displays commands in upper case.
DoWhen (DW)	Uses DoWhen when DoWhen/If command is present in the logic. It is the default value.
If	Uses If when DoWhen/If is present in the logic.
Operand1Col (O1C)	Starts the Operand one of commands in this column.
Operand2Col (O2C)	Starts the Operand two of commands in this column.
OperandColumnAbsolute (OCA)	When specified with Operand1Col and Operand2Col the column positions are absolute (refer to example 2 of applying Pretty Print for details)

Syntax	Description
TabSize (TS)	The number of spaces for each tab. By default, the value is 4.
IndentCase (IC)	Indents case from the BeginCase. By default, the value is False.
ReplaceTabsWithSpaces (RTWS)	Expands tabs with spaces. By default, the value is False.
MaximumLineLength (ML)	Used to specify the maximum length of lines. By default, the value is 80.

Change Insert Attribute to Class

Insert command uses insertable class or an attribute that has the same name as the inherited insertable class. Change Insert Attribute to Class (CIAC) command identifies the incorrect usage of insert command. It creates a new attribute with the same name as the insertable class and replaces the existing attribute with the new attribute in the logic.

Command Line Options for CIAC

The command line for CIAC is:

```
ReEldor -CIAC [-SQLServer(SS) <SQLServer Instance name>] -Model(M) <model name> [-
Segment(S) <segment name>] [-Element(E) <element name>] [-NoUpdates] [-NoValidate] [-
LogFile(L) <file name>]
```

Where:

Syntax	Description
CIAC	Identifies the INSERT statement and replaces the insertable attribute that has the same name as the insertable class.
SQLServer Instance name	Named instance of the SQL Server to connect to. If not specified, the default “local” instance is used.
Model(M)	Name of the AB Suite Model to refactor. This is a necessary parameter and can be used in the short form as –M instead of –Model.
Segment(S)	Name of the Segment to refactor. This is optional parameter and defaults to the Model name, if not specified.
Element(E)	Name of the Element/Folder to refactor. This is optional and defaults to ALL elements in the Segment.
NoUpdates	Only shows all the changes that could have been made. This does not update the Model. By default, the behavior is to update the Model.
NoValidate	Does not validate any logic after the changes have been applied. By default, the behavior is to validate.
LogFile(L)	Name of log file. By default, it is ReEldor.log.

Examples for using ReEIDor Utility

This section provides some examples on using the ReEIDor utility for minimizing the insertable string substitution and Pretty Print. Consider the following model AModel with Segment ASegment for the examples that follow.

```
AModel
    ASegment
        Insertable IGLG
            Main
            Move "X" _DA1:Assign value X to incoming parameter
            Move "Y" _DA2:Assign value Y to incoming parameter
        Insertable IGLG2
            Main
            Move "X" _DA1:Assign value X to incoming parameter
            Move "Y" _DA2:Assign value Y to incoming parameter
        Report R1
        Main
Insert IGLG (_DA1 = ABC.DA1 & _DA2 = ABC.DA2)
Move SYSOP.KDVSAPRO WG_WORK.WG_KDVSAPRO :KDVSAPRO
Move GLB.REPNAME GD_REPNAME :GD_REPNAME
Move GLB.ZEROS GG_HEADER_SRV_UIT.GG_PAR :GG_PAR
Insert IGLG (_DA1 = ABC.DA1 & _DA2 = XYZ.DA2)
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND \
        (GD_REP_ASP = GC_NUM))
    Insert IGLG (_DA1 = GRPA.GRPB.DA1 & _DA2 = GRPA.GRPB.DA2)
    Move GLB.ZEROS GG_FAR
        End
        Report R2
        Main
        Insert IGLG2 (_DA1 = ABC.DA1 & _DA2 = ABC.DA2)
```

Examples on Refactoring Insertable Substitution Strings

Examples on Applying Pretty Print

Examples on CIAC

Examples on Refactoring Insertable Substitution Strings

Example 1

This example illustrates how the ReEIDor utility can be used to refactor the insertable substitution strings on a segment named ASegment and model named AModel, without updating the model and storing the logic. The summary of the action performed by executing the command is stored in a log file named AInsSub.log Use the following command line to do this:

```
C:\Program Files\Unisys\NGEN\bin>reeldor -I -NoUpdates -M AModel -S ASegment -L
AInsSub.log
```

After its execution, the following summary is displayed in the command line:

```
*****Logic not updated due to setting of NoUpdate*****
Found 2 Elements in Segment ASegment
Refactor Insertable ASegment.IGLG
Refactor Insertable ASegment.IGLG2
Total Refactor Time: 0.4 seconds
Number of updated IGLGs: 1
Number of updated Methods: 4
Total Elapsed Time: 0.4 seconds
```

The logic in R1.Main and IGLG.Main/IGLG1.Main is left untouched.

Example 2

This example illustrates how the ReElDor utility can be used to refactor the insertable substitution strings on a segment named ASegment and model named AModel when NoUpdates is not specified in the command line.

The activity log is stored in the log file named AInsSubUpdate.log. Use the following command line to do this:

```
C:\Program Files\Unisys\NGEN\bin>reeldor -I -M AModel -S ASegment -L
AInsSubUpdate.log
```

After its execution, the following summary is displayed in the command lines:

```
Found 2 Elements in Segment ASegment
Refactor Insertable ASegment.IGLG
Refactor Insertable ASegment.IGLG2
Total Refactor Time: 0.5 seconds
Validation Time: 0.3 seconds
Number of updated IGLGs: 1
Number of updated Methods: 4
Total Elapsed Time: 0.8 seconds
```

The resulting refactored logic is:

```
IGLG.Main:

Move "X" _PARAM1.DA1 :Assign value X to incoming parameter
Move "Y" _PARAM2.DA2 :Assign value Y to incoming parameter

IGLG2.Main:

Move "X" ABC.DA1 :Assign value X to incoming parameter
Move "Y" ABC.DA2 :Assign value Y to incoming parameter

R1.Main:

Insert IGLG (_PARAM1 = ABC\
 & _PARAM2 = ABC)
Move SYSOP.KDVSAPRO WG_WORK.WG_KDVSAPRO :KDVSAPRO
Move GLB.REPNAME GD_REPNAME :GD_REPNAME
Move GLB.ZEROS GG_HEADER_SRV_UIT.GG_PAR :GG_PAR
Insert IGLG (_PARAM1 = ABC\
 & _PARAM2 = XYZ)
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND \
        (GD_REP_ASP = GC_NUM))
Insert IGLG (_PARAM1 = GRPA.GRPB\
 & _PARAM2 = GRPA.GRPB)
    Move GLB.ZEROS GG_FAR
End

R2.Main:

Insert IGLG2
```

Examples on Applying Pretty Print

Example 1

This example illustrates how ReElDor utility can be used to reformat the LDL+ logic such that all the end of line comments (if they exist) start from column 70 and start Operand1 at column 10 and Operand2 at column 30. Use the following command line to do this:

```
C:\Program Files\Unisys\NGEN\bin>reeldor –PP –EOLCC 70 –O1C 10 –O2C 30 -M AModel -S
ASegment -L APretty1.log
```

Where,

- EOLCC 70 is used to force all end of line comments to start from column 70

- O1C 10 is used to start operand1 from column 10

- 02C 30 is used to start operand 2 from column 30

After its execution, the following summary is displayed in the command line:

```
Found 4 Elements in Segment ASegment
Pretty R1.Main
Pretty IGLG.Main
Pretty IGLG2.Main
Pretty R2.Main
Total Refactor Time: 0.6 seconds
Validation Time: 0.2 seconds
Total Lines of Logic: 21
Number of Pretty Logics: 4
Total Elapsed Time: 0.9 seconds
```

The reformatted logic appears as follows:

```
IGLG.Main:

Move   "X"                  _PARAM1.DA1                           :Assign value X to
incoming parameter
Move   "Y"                  _PARAM101.DA2                         :Assign value Y to
incoming parameter

IGLG2.Main:

Move   "X"                  ABC.DA1                               :Assign value X to
incoming parameter
Move   "Y"                  ABC.DA2                               :Assign value Y to
incoming parameter

R1.Main:
Insert IGLG (_PARAM1 = ABC\
 & _PARAM101 = ABC)
Move     SYSOP.KDVSAPRO      WG_WORK.WG_KDVSAPRO      :KDVSAPRO
Move     GLB.REPNAME         GD_REPNAME                            :GD_REPNAME
Move     GLB.ZEROS           GG_HEADER_SRV_UIT.GG_PAR :GG_PAR
Insert IGLG (_PARAM1 = ABC\
 & _PARAM101 = XYZ)
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND (GD_REP_ASP = GC_NUM))
Insert IGLG (_PARAM1 = GRPA.GRPB\
     & _PARAM101 = GRPA.GRPB)
    Move GLB.ZEROS           GG_FAR
End
```

Example 2

This example illustrates how to use the ReElDor utility on some indented logic in R1.Main as shown below:

```
R1.Main logic before refactoring:
Move SYSOP.KDVSAPRO WG_WORK.WG_KDVSAPRO :KDVSAPRO
Abort GA_SDK
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND (GD_REP_ASP = GC_NUM))
   Move GLB.ZEROS GG_FAR
   Add 1 GG_FAR
End
```

The above LDL+ logic needs to be reformatted such that Operand1 starts at column 10 and Operand2 at column 30. We can do this using either of the following two methods:

Method 1 – Using the OperandColumnAbsolute (OCA) Option

```
C:\Program Files\Unisys\NGEN\bin>reeldor -PP -O1C 10 -O2C 30 -OCA -M AModel -S
ASegment -L APretty1.log
```

When the –OperandColumnAbsolute (OCA) option is used for the indented logic it results in the Operand1 and Operand2 columns being placed at the absolute respective positions 10 and 30 respectively.

The resulting reformatted logic of R1.Main after refactoring:

```
Move     SYSOP.KDVSAPRO     WG_WORK.WG_KDVSAPRO :KDVSAPRO
Abort    GA_SDK
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND (GD_REP_ASP = GC_NUM))
   Move  GLB.ZEROS          GG_FAR
      Add   1                  GG_FAR
End
```

Here the indented logic within the DoWhen block the Operand1 and Operand2 column positions are at absolute positions, 10 and 30 respectively.

Method 2 – Without using the OperandColumnAbsolute (OCA) Option

```
C:\Program Files\Unisys\NGEN\bin>reeldor -PP -O1C 10 -O2C 30 -M AModel -S ASegment -L
APretty1.log
```

When the –OperandColumnAbsolute (OCA) option is not used for the indented logic it results in the Operand1 and Operand2 columns to be positioned relative to the start column of the command.

The resulting reformatted logic is:

R1.Main logic after refactoring:

```
Move     SYSOP.KDVSAPRO     WG_WORK.WG_KDVSAPRO :KDVSAPRO
Abort    GA_SDK
DoWhen ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND (GD_REP_ASP = GC_NUM))
Move     GLB.ZEROS           GG_FAR
      Add      1                   GG_FAR
End
```

Here the indented logic within the DoWhen block the Operand1 and Operand2 column positions are relative to the start position of the command.

Example 3

This example illustrates how to use the ReEIDor utility to reformat LDL+ logic using the CA option to force the usage of command abbreviations. This results in the conversion commands to their abbreviated form such as MOVE to MV, DoWhen to DW, etc.

The command line to be used is:

```
C:\Program Files\Unisys\NGEN\bin>reeldor -PP -CA -M AModel -S ASegment -L
APretty2.log
```

After its execution, the following summary is displayed in the command line:

```
Found 4 Elements in Segment ASegment
Pretty R1.Main
Pretty IGLG.Main
Pretty IGLG2.Main
Pretty R2.Main
Total Refactor Time: 0.5 seconds
Validation Time: 0.2 seconds
Total Lines of Logic: 21
Number of Pretty Logics: 4
Total Elapsed Time: 0.8 seconds
```

The resulting reformatted logic is:

```
IGLG.Main:

MV "X" _PARAM1.DA1                           :Assign value X to incoming parameter
MV "Y" _PARAM101.DA2                         :Assign value Y to incoming parameter

IGLG2.Main:

MV "X" ABC.DA1                               :Assign value X to incoming parameter
MV "Y" ABC.DA2                               :Assign value Y to incoming parameter

R1.Main:

INS IGLG (_PARAM1 = ABC\
 & _PARAM101 = ABC)
MV SYSOP.KDVSAPRO WG_WORK.WG_KDVSAPRO        :KDVSAPRO
MV GLB.REPNAME GD_REPNAME                    :GD_REPNAME
MV GLB.ZEROS GG_HEADER_SRV_UIT.GG_PAR   :GG_PAR
INS IGLG (_PARAM1 = ABC\
 & _PARAM101 = XYZ)
DW ((GG_HEADER_SRV_UIT.GG_FAR = GD_REP_FAR) AND (GD_REP_ASP = GC_NUM))

    INS IGLG (_PARAM1 = GRPA.GRPB\
     & _PARAM101 = GRPA.GRPB)
    MV GLB.ZEROS GG_FAR
END

R2.Main:

INS IGLG2
```

Examples on CIAC

This example illustrates how to use the ReEIDor utility to identify the INSERT statement and replaces the insertable attribute that has the same name as the insertable class.

Consider the following model structure:

```
AModel
ASegment
    Insertable IGLG
        Main
            Move "X" _DA1:Assign value X to incoming parameter
    Move "Y" _DA2:Assign value Y to incoming parameter
Report R1
    InsAttr (inherited from IGLG)
Main
    Insert InsAttr
```

In this example, when you validate R1.Main method, a validation error appears:

```
error 2214: The operand InsAttr cannot be inserted. INSERT statement can only insert
an Insertable class.
```

The command line to be used is:

```
C:\Program Files\Unisys\NGEN\bin>ReEldor -CIAC -M AModel -S ASegment -L CIAC.log
```

After its execution, the following summary is displayed in the command line:

```
Found 1 Elements in Segment ASegment
Validate R1.Main
******* Error in Validation of R1.Main ***********
======== Fix Start ========
Line:1: replace "InsAttr" with "IGLG" in logic "Insert InsAttr" of Method R1.Main
======== Fix end ========

Total Refactor Time: 10.7 seconds
Validate R1.Main
Validation Time: 0.4 seconds
Total Elapsed Time: 11.2 seconds
```

Then, the model structure changes to:

```
AModel
ASegment
    Insertable IGLG
        Main
            Move "X" _DA1:Assign value X to incoming parameter
    Move "Y" _DA2:Assign value Y to incoming parameter
Report R1
    InsAttr (inherited from IGLG)
    IGLG (inherited from InsAttr)
Main
Insert IGLG
```

The resulting refactored logic is:

```
R1.Main
Insert IGLG
```

Access Layer API—Logging Information

The Access Layer API log file records all actions that are executed when you connect to the AB Suite Client Framework application and perform other transactions. It also shows the details of messages being sent to and received from the host application. You can refer to this log file to resolve issues with the client applications. It can be a useful debug tool for Unisys support.

The log details from the Access Layer API log operations are logged in to the AccessLayer.Connector.log file. By default, this log file is created in the Temp folder at the following location:

C:\Temp\AccessLayer.Connector.log

AccessLayer.Connector.log

You can access the AccessLayer.Connector.log file each time you perform a task on the Access Layer API.

<TechnologyFolderName>_Config.rtxml Configuration File

You can configure the tags in the <TechnologyFolderName>_Config.rtxml file to capture the details of Client Framework Access Layer API operations. The <TechnologyFolderName>_Config.rtxml file is located in the Access Layer API Deploy folder.

You can configure the following tags in the <TechnologyFolderName>_Config.rtxml file to record the operations performed using the Client Framework Access Layer API:

- <LogLevel>
- <LogFolder>

<LogLevel>

You must modify the <LogLevel> tag to indicate the level of logging performed. Following are the different types of log levels:

- Info – Used to capture lightweight log information, such as connection details, send request confirmation, and receive response confirmation.
- Debug – Used to capture detailed information.
- Error – Used to capture log exceptions.

<LogFolder>

You must modify the <LogFolder> tag to create the log file in a location other than the default location.

For example, if you want to save the log file in C:\LogFile, you must modify the <LogFolder> tag as follows:

```
<LogFolder> C:\LogFile </LogFolder>
```

Error Messages

--- A --- Error Messages

A Flag to the key '%1' is not valid - 2114

Cause: The destination attribute of the Flag command is a Key. Flag to a key is not allowed.

Resolution: Do not flag to a 'key' attribute

Array '%1' needs to have %2 dimensions - 2121

Cause: The number of dimensions specified in the multiplicity of the Array (%1) is not appropriate in this context.

Resolution: Ensure that the correct number of dimensions are specified.

Array Missing Selectors - 2147

Cause: The Array attribute is not fully selected.

Resolution: The Array attribute must be fully selected.

--- B --- Error Messages

There are currently no error messages listed in this category.

--- C --- Error Messages

Case command cannot be used after an Otherwise command - 2002

Cause: The structure of the BeginCase … Case … Otherwise … End block does not allow a Case node after Otherwise. If used, otherwise must be the last 'node' in the BeginCase block before the end.

Resolution: Move the Case construct to before the Otherwise command.

Case value is not compatible with Begin Case - 2108

Cause: A value specified on a Case command cannot be compared to the attribute or expression specified in the BeginCase due to type incompatibility.

Resolution: Only use values which are compatible with the attribute or expression type used on the BeginCase command.

Command cannot be used in a Copy From Main method - 2138

Cause: Incorrect usage of the command in main method of a Copy From ispec/event.

Resolution: Remove the command from the main method of a Copy From ispec/event.

Command name can only be used in a loop - 2006

Cause: The 'Break' command has been used in a context, which is not in a Loop. The 'Break' command may only be used inside a loop such as Determine, LookUp, ForEach, or Loop

Resolution: Remove the break command. If necessary use a JumpTo <label> to achieve the desired result.

Command name was not expected here - 2004

Cause: An 'Else', 'Continue', 'Case', or 'End' node was encountered in an unexpected context. For example an 'Else' command was not within a DoWhen-End block, a 'Continue' was not within a block, a 'Case' was not within a 'BeginCase'-'End' block or an 'End' did not have a corresponding block start command.

Resolution: Correct the structure of the logic by removing the offending command or containing it within an appropriate block.

Command name was not found - 2003

Cause: The command <name> is not a valid command. This may be simply due to a typographical error.

Resolution: Replace <name> with a valid command name.

Command cannot be used in a Copy From Edit method - 2130

Cause: The 'Flag' and 'Recall' commands may not be used in the Edit method of a Copy From Ispec or Event.

Resolution: Remove the offending command(s).

Command cannot be used in a System close GLG - 2129

Cause: The 'Roc', 'Run', and 'Wake' commands cannot be used in the SystemClose Global Logic (method).

Resolution: Remove the offending command(s).

Command cannot be used in a System GLG - 2131

Cause: The 'Recall' command cannot be used in a System Global Logic (method)

Resolution: Remove the offending command(s)

Compare routine is invalid. - 1001

Cause: In the Match command, the compare routine must be one of 'CompareAscending' or 'CompareDescending'

Resolution: Use either 'CompareAscending' or 'CompareDescending'

Conditional Profile attributes must be persistent members - %1 - 2133

Cause: Attributes use in a profile constraint (condition) must be persistent members of the class which owns the profile.

Resolution: Use only attributes which are persistent members of the profile owner.

Copy From Error - 2138

Cause: Command cannot be used in a Copy From Main method.

Resolution: Delete the command

CRITICAL POINT cannot be used inside a database access loop (DT, LU, ForEach) - 2166

Cause: Critical point being used within a database access loop (e.g. Determine loop / Lookup / ForEach loop)

Resolution: Do not use critical point within a database access loop.

CRITICAL POINT can only be used inside a Report Main method - 2165

Cause: Critical point being used inside the Main method of the Report.

Resolution: Do not use critical point within the main method of the report.

--- D --- Error Messages

Date flag is invalid. - 1006

Cause: In the simple form of the DateConvert command, the format specifier must be one of 'ToDayNumber, 'ToDate', or 'ToAlpha'

Resolution: Use only one of the approved format specifiers

Date format is invalid. - 1005

Cause: The format specifier in a DateConvert command is not one of those in the list shown under the DateConvert command.

Resolution: Use only one of the approved format specifiers

▶ Date format is invalid. - 1004

Cause: The format specifier in a MoveDate command is not one of those in the list shown under the DateConvert command.

Resolution: Use only one of the approved format specifiers

▶ Database name is invalid. - 1002

Cause: The database name used in an AccessExt command must be either 'DB1' or 'DB2'.

Resolution: Use one of the allowed database names.

--- E --- Error Messages

There are currently no error messages listed in this category.

--- F --- Error Messages

▶ Forward definition of the label %1 was not found - 2001

Cause: The label %1 used in a JumpTo command was not found during a forward search of the logic. Labels must be defined after later in the logic than any JumpTo command which refers to them.

Resolution: Ensure that the appropriate Label command exists within the correct scope.

--- G --- Error Messages

There are currently no error messages listed in this category.

--- H --- Error Messages

There are currently no error messages listed in this category.

--- I --- Error Messages

▶ Incompatible Parameter - 2144

Cause: The parameter is incompatible with the method definition

Resolution: Check the definitions of the parameter and the define method parameter.

▶ Index %1 of array '%2' should be type '%3' - 2122

Cause: An attribute used as an array index does not evaluate to a number.

Resolution: Correct the offending term.

▶ Insertable attribute methods cannot be called - 2145

Cause: It is invalid to invoke a method on an Insertable Class instance.

Resolution: Only invoke Insertable class methods from with the Insertable Class Main method.

▶ Internal semantic error - 2999

Cause: An undefined internal semantic error has occurred.

Resolution: Report this error to ACUS support.

▶ Internal Syntax Error - 1999

Cause: An internal syntax error has occurred

Resolution: Report this error to ACUS support.

▶ Invalid arithmetic expression - 2126

Cause: A term was used in an arithmetic expression which does not evaluate to a number.

Resolution: Correct the offending term.

▶ Invalid conditional expression - 2127

Cause: Terms or expressions used in a conditional expression are the wrong type to produce a Boolean result (true or false).

Resolution: Use terms or expressions which produce a true or false result when evaluated

▶ Invalid logical expression - 2128

Cause: Terms or expressions used in a conditional expression are the wrong type to produce a Boolean result (true or false).

Resolution: Use terms or expressions which produce a true or false result when evaluated

▸ Invalid Insert - 2143

Cause: This command cannot be used in an Insertable class.

Resolution: Remove the command.

▸ Invalid Set operation - 2111

Cause: The types use in the IN operator are incompatible - that is MYBool IN (12,13,14), assuming MYBool is a boolean attribute.

Resolution: Ensure types are valid.

▸ IO parameters - %1 cannot be a literal - %2 - 2160

Cause: A literal is being used as a parameter that has direction defined as InputOutput/Output in the model

Resolution: Use an attribute instead of a literal.

▸ Iterator %1 should inherit from the dataset - 2132

Cause: The iterator variable specified (%1) is not the correct type. It must inherit from (be the same type as) one of the classes in the inheritance structure being iterated over.

Resolution: Set the type of the iterator variable to inherit from an appropriate class.

--- J --- Error Messages

▸ Jump to label %1 is not allowed - 2005

Cause: You may not jump into a Loop or BeginCase-End block.

Resolution: JumpTo a label either before or after the block containing label %1

--- K --- Error Messages

▸ Key List Error - 1012

Cause: The syntax of the keys is incorrect.

Resolution: Refer to the LDL+ syntax documentation.

--- L --- Error Messages

▶ Label %1 has already been defined at line %2 - 2007

Cause: The label %1 has already been used within the method.

Resolution: Use a different label name to avoid duplication.

--- M --- Error Messages

▶ Mapper can only be used with a Report Frame - 2106

Cause: This occurs in an Extract command where Mapper option can only be used with a Frame.

Resolution: Correct the cause.

▶ Match Extract File Incompatible - 2141

Cause: The Match comparison parameter is %1.

Resolution: Use %2.

▶ Match File Error - 2140

Cause: The Match comparison parameter %1 must inherit from GLB.File.

Resolution: Correct the cause.

▶ Match File the same - 2142

Cause: The same extract file cannot be used in a Match command.

Resolution: Change the extract file.

▶ Method '%1' cannot be used here - 2146

Cause: The call to the method specified by %1 is incorrect in the current context.

Resolution: Don’t call the specified method in the current context.

▶ Method '%1' needs to have %2 parameters - 2124

Cause: An incorrect number of parameters has been supplied on a call to the Method '%1'

Resolution: Supply the correct number of parameters in the method call.

Method call error - 2146

Cause: Method cannot be used here

Resolution: Correct the cause.

--- N --- Error Messages

Number of operands is incorrect. - 1009

Cause: A command which has mandatory arguments does not have sufficient arguments specified

Resolution: Supply the missing arguments according to the particular command.

Number of keys does not match the profile or ispec definition - 2112

Cause: A LookUp or Determine command does not have the correct number of keys specified.

Resolution: Specify the correct number of keys.

Numeric literals can have a maximum of %1 digits - 2163

Cause: The length of numeric literal is more than 28 digits.

Resolution: Ensure that the numeric literal has a maximum of 28 digits

--- O --- Error Messages

Only one comparison function is allowed when using match with extract files - 2139

Cause: Using more than one comparison function while matching extract files without specifying keys is not allowed.

Resolution: Use only one comparison function while doing a match on the extract files without specifying keys.

Operand name could not be found. Check that it has the correct Visibility - 2117

Cause: The named operand could not be found within the current scope.

Resolution: Check that the operand does exist within the specified scope – it may need qualification. Also check that the named entity has the correct visibility – public, protected, etc.

▶ Operand name should be a length of at least number - 2110

Cause: In the DateConvert command, the named operand is not long enough to contain the converted date in the format specified.

Resolution: Ensure that the operand is at least long enough to contain the date in the specified format.

▶ Operand name should be a member of name - 2119

Cause: In the Determine Total command the attribute to be summed must be owned by the class <name>.

Resolution: Use an attribute which is owned by class.

▶ Operand name should be a method - 2123

Cause: The named operand should be a method.

Resolution: Only use a method where this operand has been used.

▶ Operand name should be a shadow report - 2107

Cause: Only a Shadow report (OutputStream) can be used in the place where operand <name> has been used.

Resolution: Use only a Shadow Report (OutputStream) where <name> has been used in this command or make name to be of the correct type (ShadowReport/OutputStream).

▶ Operand name should be an array - 2120

Cause: An array attribute was expected, but <name> is not an array. – that is DString := SDNumber[1,2,1] where SDNumber is not an array.

Resolution: Ensure entry is an array attribute.

▶ Operand name should be an extract file - 2116

Cause: Where <name> has been used in the indicated command an array attribute is expected (multiplicity > 1).

Resolution: Replace <name> with an attribute which is an array.

▶ Operand name should be persistent - 2102

Cause: The indicated operand <name> must be persistent.

Resolution: Ensure that the operand has 'IsPersistent' = true.

Operand name should be type 'Number' - 2115

Cause: The 'position' and 'length' operands of the complex Move command must be numeric.

Resolution: Change the type of the <named> operand to Number.

Operand name should be writeable - 2101

Cause: The destination/target operand named must be capable of being written to.

Resolution: Ensure that the named operand is not defined as 'read only'.

Operand %1 cannot be void (Check the that it has a return value) - 2105

Cause: A method was expected to return a value - ie MyNumber := AMethod(56) if AMethod does not have a return value, then this error occurs.

Resolution: Ensure method has a return value.

Operand %1 must be an Object (Check that is has Multiplicity 0) - 2135

Cause: The operand must be an object/instance. That is, it must have multiplicity > 0.

Resolution: Correct the multiplicity. If necessary create an attribute which inherits from the class and ensure that it has multiplicity > 0.

Operand needs to be an identifier. - 1007

Cause: The indicated operand cannot be a literal or expression – it must be a variable, attribute, or parameter.

Resolution: Correct the statement by replacing the indicated operand with an variable, attribute, or parameter reference.

Operand %1 needs to be an instance of class in same inheritance heirarchy as %2 - 2169

Cause: The operand specified by %1 does not belong to the inheritance tree that %2 operand belongs to.

Resolution: Ensure that operand %1 is an instance of a class in the same inheritance hierarchy as %2.

Operand %1 needs to be of the same type as %2 - 2168

Cause: Operands specified by %1 and %2 are of different/incompatible types.

Resolution: Ensure that the operands specified by %1 and %2 are of similar types.

--- P --- Error Messages

▶ Parameter %1 is incompatible with the method definition - 2144

Cause: The supplied parameter is incompatible with the type defined for that parameter in the method.

Resolution: Ensure the type of the supplied parameter is the same as the parameter in the method definition.

▶ Parameter %1 of method '%2' should be type '%3' - 2125

Cause: The argument %1 passed in the call to method %2 is of the wrong type and cannot be cast to the correct type. It should be of the type specified as %3.

Resolution: Pass the correct type in the specified parameter position.

▶ Parameter List Error - 1014

Cause: The syntax of the parameters is incorrect.

Resolution: Refer to the LDL+ syntax documentation.

▶ Position operand cannot have decimals - 2162

Cause: A decimal literal/attribute has been supplied for the position option of Detach command

Resolution: Position can only be a non-decimal signed/unsigned number.

--- Q --- Error Messages

There are currently no error messages listed in this category.

--- R --- Error Messages

▶ Restart cannot be used here - 2164

Cause: The restart command (without an extract file name as a parameter) is invalid when used outside a Determine Actual loop. It is invalid inside a Determine Actual Extract loop when specified with an extract file that is same as the extract file for the Determine Actual loop.

Resolution: Use the Restart command in accordance with the above rules. Use Restart (without an extract file name as a parameter) within the Determine Actual loop. Use Restart outside Determine Actual Extract loop when specified with an extract file that is same as the extract file for the Determine Actual loop.

Right parenthesis is missing. - 1010

Cause: A function/method call parameter list is missing a right parenthesis.

Resolution: Correct by adding a right parenthesis in the appropriate place.

Right square bracket is missing. - 1011

Cause: An array reference is missing a right square bracket ']' from the selector list.

Resolution: Correct by adding a right square bracket in the appropriate place.

Routine name is invalid. - 1003

Cause: In the OnChange command, the statistic routine name specified is not one of those allowed from the list.

Resolution: Use one of the allowed statistic routine names or correct the spelling.

--- S --- Error Messages

Screen Field name should be an enterable screen field - 2109

Cause: The operand used on the Cursor command must be an enterable field in the presentation/User Interface.

Resolution: Correct by ensuring that the operand has a direction of I(nput) or IO.

Source key '%1' cannot be used in the key list - 2113

Cause: The key specified is owned by the class being iterated over and is Persistent. Persistent attributes of the class being iterated over cannot be used to supply key values.

Resolution: Use a non-persistent attribute of the class being iterated over in the key list or any attribute of a different class which does not share inheritance with the subject class.

Symbol is readonly - 2161

Cause: The supplied parameter has the IsConstant property set to true.

Resolution: Change the property to false or supply another parameter that has the property set to false.

Syntax Errror - 1013

Cause: The syntax of the LDL+ command is invalid.

Resolution: Refer to the LDL+ syntax documentation.

--- T --- Error Messages

The AutoLookup dependency on target %1 is invalid. - 1762

Cause: The target %1 is not set to a key attribute.

Resolution: Ensure that a key attribute for the target is set to %1.

The Array attribute %1 must be fully selected - 2147

Cause: The array element is not fully specified.

Resolution: Specify all of the correct indices for the array.

The Key definition is incorrect - 2137

Cause: No key/s is specified to sort the file.

Resolution: Specify at least one key to define the sort order.

The Match comparison parameter %1 must inherit from GLB.File - 2140

Cause: Invalid first parameter being passed to the Match command.

Resolution: First parameter to the Match command needs to be inherited from GLB.File

The Match comparison parameter %1 should be using %2 - 2141

Cause: The same extract file is being used in the parameters to the Compare functions.

Resolution: The comparison parameter should be the one specified by parameter %2

The name %1 is in error (Check that arrays have indexes and functions have parameters) - 2148

Cause: Arrays are not fully selected or functions are called without parameters.

Resolution: Verify that if arrays are being referenced they are fully selected, and parameters are specified for a function that has non-default parameters.

The same extract file cannot be used in a Match command - 2142

Cause: The same extract file is being used in the parameters to the Compare functions.

Resolution: Use two different extract files for the comparison.

This command cannot be used in an Insertable class - 2143

Cause: Invalid usage of the command in an Insertable Report or GLG

Resolution: Remove the command from the class.

Too Many Match Files - 2139

Cause: Only one comparison function is allowed when using match with extract files.

Resolution: Remove additional comparison functions.

Type of the %1 is invalid. Valid type(s) are - '%2' - 2100

Cause: The specified operand is of the wrong type in that context in the current command.

Resolution: Replace with an operand of one of the specified types.

Type of the %1 is not compatible with the return type of the method - 2104

Cause: The specified operand cannot be converted to the return type of the method.

Resolution: Replace with an attribute, variable, or expression of the correct type.

Type of the %1 is not compatible with the type of the %2 - 2103

Cause: The source operand cannot be converted to the type of the destination operand.

Resolution: Ensure that the source and destination operands are of compatible types.

--- U --- Error Messages

Under Segment/Report, only Primitives or Groups can be persistent. - 1760

Cause: When you are using Segment/Report, only Primitives or Groups must be set to persistent.

Resolution: Ensure that IsPersistent is set to No.

Unrecognized line. - 1000

Cause: The line does not start with a recognizable command.

Resolution: Correct the offending line.

--- V --- Error Messages

Value logic can only reference 'This' - 2134

Cause: The only attribute which may be named as the left operand in a value logic/ constraint expression is 'this'.

Resolution: Replace the offending operand with 'this'.

--- W --- Error Messages

There are currently no error messages listed in this category.

--- X --- Error Messages

There are currently no error messages listed in this category.

--- Y --- Error Messages

There are currently no error messages listed in this category.

--- Z --- Error Messages

There are currently no error messages listed in this category.

Glossary

This glossary defines terminology used within Agile Business Suite which is unique to the application. Note that it does not include terms which are associated with object oriented concepts, or terms generally used within Visual Studio, such information should be obtained from external OO documentation or the *Visual Studio Online Help* respectively.

Terminology Changes

The table below lists the comparative terms between the current release of the Agile Business Suite product and the previous Enterprise Application Environment product. Where terms have not changed, they are found in the rest of this glossary, along with new terminology.

Enterprise Application Environment Term	Agile Business Suite Term
Activity	Folder
Application	Project
Big Buffer Ispec	Ispec Stereotyped Class
Branching	Branching
Breakpoint	Breakpoint
Business Rules	Documentation
Business segment	Segment Stereotyped Class
Callable Global Logic	Method
Copy from ispec	Copy Ispec Stereotyped Class
Copy From Event	CopyEvent Stereotyped Class
Data Dictionary	Dictionary
Data Item, Setup Data Item (SD)	Attribute
DEPCON	Enterprise Output Manager
Developer Test	Debugger
Dictionary data item	Class
Edit logic	Method Edit
Enterprise Application Remote Access Server	Remote Access Server
Event	Event Stereotyped Class
Fireup Ispec	Fireup Ispec
Frame	Frame Stereotyped Class
Generate	Build

Enterprise Application Environment Term	Agile Business Suite Term
Generate Client/Generate Server	Builder
Generate set	Configuration
Global Data Dictionary	Dictionary
Global Setup Data Block (GSD Block)	Folder
Global Setup Data Item	Attribute
Graph	UML Diagram
Group Global Setup Data Item	Group Stereotyped Class
Group Setup Data Item	Group Stereotyped Class
Insertable Global Logic (GLG)	Insertable Stereotyped Class
Ispec character screen, graphical screen, Report frame layout	Presentation
Ispec Component	Ispec Stereotyped Class
Ispec Ordinate	IsKey
Keyword	Group Stereotyped Class
LCIF	Model Export File
LDL (LINC Definition Language)	LDL+ (Logic Definition Language)
LINC cycle	Business segment cycle
Logic, Performable GLG, Callable GLG	Method
Main logic	Method Main
Memo Component	Ispec Stereotyped Class
Model	Model
Option	Property
OVI Ispec	External class
Performable Global Logic	Method
Pre LINC logic	Prepare method
Pre Screen logic	Construct method
Pre screen, or Preamble	Construct method
Profile	Profile
Profile Ordinate	Key
Relation	Dependency
Repository	Model
Same.as	Inherits

Enterprise Application Environment Term	Agile Business Suite Term
Setup Data Array	Attribute (with multiplicity 1)
SQL Script	SQL Script Stereotyped Class
System Data Item	Attribute (Builtin)
Usage - Output	IsPersistent

A

AB Suite Application/Model

Uses Winform or Component Enabler Interfaces based on screens designed in System Modeler Painter.

AB Suite Client Framework Application/Model

Uses a synchronized Client Framework interface, where the Client Applications are designed using a technology of choice.

Action Line

A Field that appears on most screens in mainframe Enterprise Application and Agile Business Suite Systems, enabling fast-track navigation to required functions. Also referred to as the Action field.

Administration tool

The Runtime Administration Tool is a snap-in to the Microsoft Management Console. It configures and manages the runtime environment and deployed Agile Business Suite applications.

Administrator

Person responsible for administration of the Developer environment.

Previous term was – Administrator.

Aggregation

More specifically composition. This models the whole/part relation. Objects are often made up of other components, each of which may be an object in its own right.

Animation

The function by which Debugger continues to execute logic without user intervention, highlighting each LDL+ statement as it is executed

Archive

Copy files to a directory as protection against accidental loss, deletion, or damage.

Asynchronous

A process whose caller continues immediately without waiting for the process to complete.

Attribute

An object that belongs to a Class. Data items are persistent attributes of an <<Ispec>> Class. Setup data items are transient attributes of an <<Ispec>> Class or <<Frame>> Class (or variables in a business segment method). Global setup data items are attributes of a <<Segment>> Class.

The Multiplicity property specifies the number of instances. Arrays using a comma separated list of integers with the number of values specified denoting the dimension.

Previous term was - Data Item, Setup Data Item (SD), Global Setup Data Item (GSD), Arrays.

Mapping between Agile Business Suite Attributes and Enterprise Application Environment Usage Types

AB Suite <<ispec>> Class Attribute Properties		**EAE Usage Equivalents**
Direction	*Persistence*	
InOut	No	Input
InOut	Yes	Input-Output
In	No	Input, but cleared
In	Yes	Input-Output, but cleared
Out	No	Inquiry
None	Yes	Out
None	No	Setup Data Item

Attribute Direction

The Direction property enables the attribute to be entered or displayed on the screen.

Automatic entry capable

Automatic entries are used to pass ispec information between deployed applications. This information passing can occur between applications on the same or on different hosts. Transactions that initiate inter-application communication are fully recovered in the event of a system failure (except where the two-phase commit process abandons a transaction).

B

Background Run

For OS 2200 Systems, a background run that controls certain functions; for example recovery, Report handling, and setting up of Common Banks. There is one Background Run for each Runtime.

Banner

A system-generated page that prints at the start of a Report and provides control information about that Report.

Base Year

The year upon which the DATE.CONVERT; command bases relative day numbers. Base year is defined using the Business Segment dialog box in Developer. It may be accessed through the System Data Item GLB.BASE. Refer to relative day number.

Also, refer to R.

Branching

A feature of Source Control that allows you to develop and save a set of revisions for an element separate from a different set of revisions for the same element. These sets of revisions are referred to as ‘branches’.

Branch Label

An optional, user-defined short name that can be assigned to a branch of an element in the Version Control Bank.

Branch Search Order

A list of branch labels that Version Control can search in sequence to find an element when a Get Latest Revision or Check Out is performed.

Branching

A feature of Developer Version Control that allows you to develop and save a set of revisions for an Object separate from a different set of revisions for the same Object. These sets of revisions are referred to as 'branches'.

Breakpoint

A feature of the Developer Debugger that stops logic execution in response to a certain state of the system. There are four types of breakpoints:

Type	Description
Simple (line) breakpoint	Logic execution stops before a specific line of logic.
Conditional breakpoint	Logic execution stops before a line of logic if certain conditions are met.
Reference breakpoint	Logic execution stops before a line of logic in which a specified Data Item is referenced (read or written).

Type	Description
Change breakpoint	Logic execution stops after a line of logic in which a specified Data Item is updated.

Build

Generating, compiling and optionally deploying an application.

Previous term was – Generate.

Builder

Generation is the process of generating a complete set of source files and then compiling and linking those files to create a set of executables. Through Builder, a Developer workstation can generate a System to a target host Runtime environment.

Builtin Attributes

These are inherited due to Stereotype specification. A Data Item that is automatically available in System Modeler. It is used for accessing or setting parameter or control-type data in logic

Previous term was - System Data Item.

Built-In Method

System Modeler provides several built-in methods for some classes, for the user to invoke.

Bundle

Bundles are used to control the generation of Component Enabler components and interface applications. They are a group of Input and Input-Output Ispecs usually designed for a specific task. You can define more than one Bundle for each system, each one suited to the needs of a specific set of users.

Business Model

Repository containing details of Segments (Specifications) held in Developer.

C

Change Analysis

A method for tracking changes to elements which can be generated, to eliminate the need to regenerate elements that have not changed since the last time the generate process was run.

CHG

Object Changes (CHGs) are used to distribute new features, fixes, customer requests and newly validated versions of support software on the ClearPath OS 2200 platform. Also, refer to IC.

CL

Connection Library. For the MCP Platform CLs provide, in addition to normal library functionality, per client state, multiple interfaces each with its own set of procedures and data and full control over connections (linking and delinking).

Class

Definition of a type of object, including its interfaces and member attributes and methods. Classes have been extended in Agile Business Suite to include the description of subsets of the class's instances (as represented by profiles and SQL scripts in EAE 3.3). Business segments, ispecs, reports and insertable global logics become special kinds of classes in Agile Business Suite. To emphasize this point they are sometimes referred to as 'segment class', 'ispec class' etc., but this is synonymous with 'segment', 'ispec' etc. Functionality attributed to classes in this document also applies to each of the specialized EAE classes.

Class diagram

Class diagrams use standard UML notation to represent the static relationships between model entities.

Class View

A Visual Studio window which displays the fine grain structure of the Developer model, which includes class inheritance, attributes and methods. These structures are editable by the user.

Classifier

Where it is defined, a Type or Variable takes its definition from its Classifier. The Classifier may be another Type or Variable, or it may be a Class (a Variable with a Class as its Classifier is an object). A Type with a Classifier effectively acts as an alias for its Classifier. The term "classifier" is taken from UML. The Same. As is now only a term that could be used to describe a specific variation of the classifier notation. An attribute or variable that classifies to another element of same kind is effectively a Same.As relationship

Previous term was – Same.As.

Client

Windows program, typically running on a workstation that works cooperatively with one or more programs or services running on a server computer. In some cases the client program may reside on the server computer, but uses the network interfaces to communicate with the server program or service.

Client Listener

A service that listens for and accepts connection requests from the Administration Client and Deployment Client.

Client/Server

A distributed architecture in which client workstations communicate with servers through a network. For instance, a client typically provides initial processing, data gathering functionality, and the user interface. It then communicates the data and requests to a server for further processing.

CLR

Common Language Runtime. Code that you develop with a language compiler that targets the Windows runtime is called managed code; it benefits from features such as cross-language integration, cross-language exception handling, enhanced security, versioning and deployment support, a simplified model for component interaction, and debugging and profiling services. It is the IL virtual machine plus the Windows classes. Garbage Collection is an important major feature of CLR.

Clustered Index

Microsoft SQL Server database index in which the logical or indexed order of the key values is the same as the physical stored order of the corresponding rows that exist in a table. The terms cluster index and clustered index are used interchangeably.

CODES file

Output control codes file used when defining printers

CODESASSN file

A CODESASSN file associates a report destination (as defined in Glb.Stn) with a model of printer, enabling the correct printer control codes to be used when you run a report to output device TP (as defined in Glb.Device), or when you print report output from ROC using the output device TP.

COM and COM+

COM (Component Object Model) is Microsoft's component software architecture developed primarily for Windows. It is the foundation upon which OLE and ActiveX are based, and provides a means to re-use code without requiring re-compilation. In COM, a component is a platform-specific binary file that compliant applications and other components can utilize. Programs incorporating a component's services never have access to its internal data structure, but instead include pointers to its standardized interface. Thus, it is possible for components to interact with each other regardless of how they work or what language they are written in.

COM+ is an enhanced version of COM that provides better security and improved performance. DCOM (Distributed Component Object Model) is an extension of COM that allows applications and components to communicate with each other over a network.

Compile

To create object files from source files.

Component

A component is a cohesive unit of the business application, which is distributed and deployed as a unit. The outermost classes in a model are generated as components.

Component Enabler

The product with which developers can build their own GUI interfaces, or Views, to systems. These Component Enabler applications use the Enterprise Application Remote Access Server to communicate with systems on the host. Component Enabler allows applications to use current Web technology and the Enterprise NT world.

COMS

Communications Management System. On MCP COMS provides an extremely flexible and dynamic Message Control System. Of special interest to EAE/Agile Business Suite is its transaction code routing and synchronized recovery for DMSII databases.

COMSTP Program

A pre-compiled program that contains the necessary logic for routing of user transactions in an MCP-based System.

COMUS

For OS 2200-based Systems, a product used to build Runtime before installation. Refer to SOLAR.

Conditions

Conditions determine which records are selected by a profile.

Configuration

A set of properties describing how a project or model element is built and deployed.

Previous term was – Generate Set.

Construct method

The logic used to construct an ispec screen prior to display. Technology that has moved on from screens to windows and programmatic invocation via component interfaces. The pre screen logic has become the _Construct method in Agile Business Suite.

Previous Term was - Pre Screen logic

Copy Event Stereotyped Class

Copy events (copy event stereotyped classes) are events that participate in the copy cycle. They behave similarly to copyIspecs, with the additional characteristics of events.

Only attributes that have graphical presentations are able to be copied. Copied attributes are those with their corresponding graphical object's IsCopied presentation property of set to true.

Copy Ispec Stereotyped Classe

Copy ispecs (copy ispec stereotyped classes) are ispecs that participate in the copy cycle.

Only attributes that have graphical presentations are able to be copied. Copied attributes are those with their corresponding graphical object's IsCopied presentation property of set to true.

Critical Point

A user-specified recovery point within a Report. In the event of failure, recovery restores the environment to the last successful Critical Point, then resume execution at the location of the Critical Point.

D

Data Dependent Attribute

The Data Dependent Attribute defines appropriate print attributes for Enterprise Output Manager reports. Agile Business Suite comes with a simple DDA.

Database ID

The identifier for an SQL Server database.

Debugger

A workstation testing environment for Developer. Includes a logic debugger.

Debug Mode

Option setting to determine what is built and run when a debug session is initiated.

Debug Settings

A set of breakpoint, watch, and debug options that can be saved to a file and loaded into a Debugger session as needed.

Dependency

A relationship between two elements where one element depends on the other for something. The term “dependency” is taken from UML. Dependencies may be created implicitly by referring to an object in logic; or they may be created explicitly to represent an intention to refer to an object in logic (an “abstract” dependency). Where a real dependency is created, the abstract dependency becomes a real one.

Previous term was – Relation.

The fireup ispec is now defined using a Dependency. All dependencies consist of a client and supplier element and the type of dependency. The fireup ispec dependency is valid between a class of stereotype <<segment>> and a supplier that is either <<ispec>> or <<event>>. The supplier Ispec that is displayed when the System is initiated.

Deploy

To implement a set of executables and a database on a host machine. The result of deployment is the creation of an application on the host.

Deployment Client

A client GUI for completing the deployment function.

Deployment Folder

A Folder Model Element with Configuration Property IsDeployable set to 'yes'. Used as a means for grouping elements to deploy within Logical Modeler.

Descriptive properties

These properties are either used for design or as a default. They do not affect the runtime behavior of the class, but are either descriptive such as Description property or influence the semantics of other entities such as MemberPersistence property.

Dictionary

A Dictionary is a form of Folder. It is however limited by the fact that it can only contain Objects. These are mainly considered Classes, a Dictionary can be described as a Class Dictionary. In the same way that Folders can be added to any Namespace, Dictionaries can also be added.

Global dictionaries (defined directly under a model) can be used to share dictionary definitions across multiple applications. Local dictionaries (added to a segment) can be used to define data types for use only in that segment. Each dictionary may contain primitive and non-primitive classes.

Differences Report

An XML report of the differences found when two files, objects, revisions, or two versions of a specification are compared. This report can be viewed and printed from an Internet browser, or manipulated as an XML file.

Direct Report

A Report that uses the Report Output Control System (ROC), and which sends output directly to an output device and not to the ROC database. Also, refer to Report Output Control System. Contrast with Standard Report.

DMSII

The native database server for MCP machines. DMSII is also known as Enterprise Database Server.

Document window

The document window hosts the various designers that make up the Developer specific interfaces. A document window is displayed for each open element. The available designers for the System Modeler specific interfaces change according to the kind of element that is opened, and its properties.

Documentation

Information entered as text which supports an Element.

Previous term was – Business Rules.

Dynamic Object

An object on a painted form that must be bound to an attribute for it to function.

E

Element

Collective term for the individual parts of the Business Model.

Enterprise Output Manager

The Unisys Enterprise Output Manager software application is a comprehensive print-management and file distribution solution for mixed-platform networks. Coupling Unisys Agile Business Suite with Enterprise Output Manager increases the designs possibilities and flexibility of your Reports.

Previous term was – DEPCON.

Event

An activity performed by an organization, for example a sale, purchase, or payment.

A store of data about an activity performed. An Event consists of a screen layout and associated logic (Pre-Screen logic, Pre-LINC logic, and Main logic). Together with Components, Events form the fundamental building blocks of a System. Also, refer to Ispec.

Event Stereotyped Class

A class that automatically includes the behavior and characteristics of an event. This includes the characteristics of an ispec plus all its persistent attributes automatically become "same as" an equivalent attribute in the event structure in the business segment.

Previous term was – Event.

Exclusive Use

When applied to a Model, prevents other clients from signing on to the model until removed, (for example during load, extract or build).

Exporter

The Exporter utility allows you to export of an entire Model, or selected elements into an Export File, or export all the changes that have occurred since a user specified date and time.

Extended Language Message System (ELMS)

An OS 2200-based facility that provides translatable versions of messages used by software.

External Class

A placeholder for something outside Developer that is referenced in an application. An external class may represent a .Net or COM component, or it may represent an application providing transactions (the functionality of an OVI ispec).

Previous term was – OVI Ispec

Extract File

A flat text file created or read by a Report.

F

File Equation

For MCP, options that can cause a task to use different files, or to use files in a different way, than it otherwise would.

Filegroup

A discrete part of an SQL Server database. File groups in the same database can be stored on different physical devices.

Filter Definition

Filter Definitions enable you to create and modify filters to display selected information for the Members Page.

Fireup Ispec

The Ispec that is displayed on your terminal when you sign on to an Enterprise Application System.

FLSS

For systems targeting OS 2200, you have the option to generate as Fast-Load Self-Contained Subsystems. FLSS deployments contain one or more specially linked user-defined subsystem object modules, made up of a number of Ispecs and Separately-Compiled Global Logics, which are statically linked and contain no unresolved references. They are generally more stable than an SCSS deployment, which relies on dynamic linking.

Folder

An activity is now defined using a Folder. A folder allows members of a namespace to be grouped together who represent a definable A group of objects which together perform a business function.

Previous term was – Activity, Global Setup Data Block.

Form File Utility (FFU)

The Form File Utility (FFU) allows users to graphically place fields on an electronic form and to supply the characteristics of those fields. The output from the FFU is a .DFF file, which is a combination of a Windows Metafile (WMF) or Windows Enhanced Metafile (EMF) and a collection of fields. The .DFF file is then used by the Enterprise Output Manager DDA feature to place data in the fields when composing the printed page.

Formdepth formula

A formdepth formula defines an output sequence to be sent to your output device for the output control code specified. It is an extension of the sequence of hexadecimal values in the usual output control code.

Frame Stereotyped Class

A class that implements the functionality of a frame. It may have an interface and inherits a "Main" method that is executed on a printframe LDL command. The Main method may be overridden. The Class functions as a report frame by providing a main method to override and restricting properties to those applicable to reports. When an attribute of this class is invoked, the main method is executed before printing the presentation.

Previous term was – Frame.

Framework Method

System Modeler provides several methods for some classes. These methods are automatically called at the appropriate stage in the Developer processing cycles. Overriding the method invokes the user logic.

G

Generalized Interface (GLI)

The GLI program (GLI.exe) accepts input formatted for GLI from stdin and sends it to the target system. Responses are returned to the GLI program and echoed to stdout. Some error conditions may be noted in stderr.

GLI input format uses commands and keywords in a freeform style. Although GLI input files are much larger than fixed format NOF files, your applications may find it easier to read and write the GLI format.

Generate Threads

A number of parallel generate tasks that can be initiated through Builder.

Group member

Attributes with public visibility and no persistence. By virtue of being public, they define the string buffer representation of the owner attribute. Assigning a string to the owner is actually assigning it to these members. The sequence number determines the order of assignment.

Previous term was – Group Global Setup Data Item, Group Setup Data Item.

Group Stereotyped Class

Groups (group stereotyped classes) are a way of representing structures in your application. They can contain only attributes as members. Member attributes can be primitives or instances of other groups. Attributes cannot be persistent.

H

Host

Computer on which an application is running.

HUB

A proprietary interface that allows updating and inquiry between two runtime applications. HUB also supports two phase commit, which ensures consistency in any distributed update transaction.

HUB Background Run

For OS 2200 Systems, a background run that processes external Automatic Entries. There is one Background Run for each Runtime. Also, refer to HUB.

HUB Listener

This service listens for and accepts HUB connection requests from other Enterprise Application/Agile Business Suite user systems.

I

IC

Interim Corrections (ICs) are used to distribute new features, fixes, customer requests and newly validated versions of support software. Also, refer to UCF.

ICP

Refer to Initial Control Program (for OS 2200-based Systems) and Ispec Control Program (for UNIX-based Systems).

Importer

The Import utility is provided to migrate model specifications from 3.x into the new System Modeler constructs. The Import LCIF component is for migration purposes, however the Import utility also allows you to import an existing Developer model from another Developer database.

Inheritance

Classes can inherit members from other classes. Classes can be inherited by normal inheritance within the model, or using the Insert command.

Ispec Control Program (ICP)

The program that controls the generated Ispec subprograms in OS 2200-based Systems. Contrast with COMSTP Program (for MCP-based Systems) and Ispec Control Program (for UNIX-based Systems).

IsPersistent

Specifies whether the element is persistent.

Inner class

A class that is contained within another class. It has access to the members of its containing class. This is a concept that is taken from Java.

Insertable Stereotyped Class

A class that implements the functionality of an insertable global logic. It has a Main method and optionally has an interface. The "insert;" LDL command causes the logic to be inserted in place.

Previous term was – Insertable Global Logic (GLG).

Integrity Management

An optional feature of System Modeler Source Control which ensures that the contents of the Developer Repository are in synch with the contents of the Version Control Bank.

Interrogation Point

IsKey

The ordinate of an Ispec is now defined using the IsKey property of an Attribute. The key of a class Data Item of a Component or Profile that acts as the unique identifier to distinguish one record from another. Several attributes can define the key. The access path to individual Component or Profile records.

Previous term was – Ispec Ordinate.

Ispec Stereotyped Class

A collective term for Components and Events. In Enterprise Application Environment, an Ispec models an entity or activity in the real world. An Ispec also specifies the user interface, the processing rules, and the database structure to be used to represent it in the deployed user system.

A contraction of the term Interface Specification.

A class that automatically includes the behavior and characteristics of an ispec. This varies according to whether the class is functioning as an input ispec (has an interface but no persistent attributes), output ispec (no interface but has persistent attributes), or IO (has both an interface and persistent attributes). The characteristics it inherits changes as interfaces or persistent attributes are added or removed. These include inheriting the various ispec processing cycles and framework methods (Construct, Prepare, Edit and Main), and automatically adding a variable to the <<Segment>> Class for each persistent <<Ispec>> Class, and a public method for each <<Ispec>> Class with an interface

Previous term was – Ispec.

Mapping between Agile business Suite <<ispec>> Class types and Enterprise Application Environment Usage Equivalents

AB Suite <<ispec>> Class Attribute Properties			EAE Usage Equivalents	
Presentation	**Persistence Members**	**Has Keys**	**Usage**	**Ispec Type**
Yes	No	No	Input	memo
Yes	Yes	One	Input-Output	standard
Yes	Yes	Multiple	Input-Output	Memo with automaint profile
Yes	Yes	No (but has a default profile)	Input-Output	Memo with automaint profile
Yes	Yes	No	Input-Output	Memo
None	Yes	One	Output	Standard

Mapping between Agile business Suite <<ispec>> Class types and Enterprise Application Environment Usage Equivalents including ClearPath MCP

AB Suite <<ispec>> Class Attribute Properties				EAE Usage Equivalents	
Presentation	**Persistence Members**	**Has Keys**	**Ispec Type MCP Config**	**Usage**	**Ispec Type**
Yes	No	No	NA	Input	memo
Yes	Yes	One	standard	Input-Output	standard
Yes	Yes	Multiple	standard	Input-Output	Memo with automaint profile
Yes	Yes	No (but has a default profile)	standard	Input-Output	Memo with automaint profile
Yes	Yes	No	standard	Input-Output	Memo
None	Yes	One	standard	Output	Standard
Yes	Yes	None or more	Table	Input-Output	Table
None	Yes	None or more	Table	Output	Table
Yes	Yes	None or more	Direct	Input-Output	Direct

AB Suite <<ispec>> Class Attribute Properties				EAE Usage Equivalents	
Yes	Yes	None or more	Direct	Output	Direct

J

No entries under “J”

K

Key

The profile ordinate is now defined using Profile Keys. The key defines the attributes by which a Profile accesses individual records. A Profile can have several Profile Keys.

Previous term was – Profile Ordinate.

Keyword

A class with attributes as members that have a presentation. The usage in an Ispec is represented as an Attribute that is derived from the class.

Kind

The Kind property of an entity determines the kinds of entities it can own. For example, a method cannot own another method. Can change depending on property values, Class-Attribute-Parameter

Kinds include: Attribute, Class, Class diagram, Folder, Method, Parameter, Profile, Teach.

L

Labelling

An optional feature of System Modeler Source Control that enables you to assign a short name to a branch, a version, or group of related versions.

Version labels can be used to implement Release Management.

Language

A natural language in which text components of the system can be displayed. The default language is usually English, but can be any language you chose. A system can have up to 14 languages concurrently installed. Users, or the software itself, can chose to display screens, Reports, or prompts in any one of the installed languages.

LDL+ (Logic Definition Language)

An acronym, to describe the language of the logic used.

Previous term was – LINC Definition Language.

Literal

A literal is a value used directly by a system, without requiring any named storage area. A literal may also be used in many logic commands in place of Data Items, and to define the characteristics of Setup Data Items. A literal must consist of at least one character and must be enclosed in parentheses. Refer to Working with Literals for more information on using Literals.

Lock

System Modeler has three forms of locking: An automatic soft lock that is set up whenever you attempt to modify an Object. This lock is released when the transaction ends or you cancel the operation. A manual lock that emulates the behavior of the host resource locking. If Source Control is installed, a Version lock that is set when an object is checked out of the Version Control Bank.

Logic Editor

The Logic Editor is used to create, edit, save, and compile logic for methods within Developer.

Logical Delete

Mark a record for deletion without physically removing it from the Repository.

Logical Printer

A logical printer is a printer name specified in a Report. You can use the Printer Setup command from the File menu to map the logical printer name to a Windows printer name.

Logical Reorganization

Transforms the data from the existing physical database schema into the format of the newly developed logical schema in memory each time the data is accessed. File formats are not altered. Allows a new database schema to be run against the existing old database schema prior to a physical reorganization.

M

Method

Logic is called by name explicitly or implicitly as part of a processing cycle, and has data passed to it by way of parameters. Performable GLGs, frames and ispec logic become methods. Functionality attributed to methods in this document also applies to each of the logic types, with the exception that startup and closedown in a business segment, frame 0 in a report and the existing ispec logic have predefined interfaces with no parameters.

Previous term was – Logic, Performable LG, Callable GLG.

Method Edit

A framework method for a <<Copy From>> class that contains logic that is executed during the first processing pass for a copy from ispec.

Previous term was – Edit Logic.

Method Main

A framework method:for <<ispec>>, and <<event>> classes which is automatically executed in the processing cycle, for <<SQL Script>> classes which is executed for each loop iteration, for a <<Frame>> class, which is executed on a "print.frame" command.

Previous term was – Main Logic.

Members

Elements to which you can add other elements, and all the elements contained within the selected element.

Merging

A feature of System Modeler and Source Control which allows you to combine two versions of an Object.

Model

A structured description of a software system (as opposed to, say, a text file representation.

Model Export File

A flat file of a system created by the Extract Utility for the purpose of loading a model to another machine, or offer a convenient way to export elements from your model.

MSI (Microsoft installer) file

A deployment package consisting of a structured storage file containing the components, web pages and database schema for the application.

MSMQ

Microsoft Message Queuing. A message-based protocol that guarantees resiliency and delivery. Messages are stored in a central location, so if either end of the transaction is not there when the message comes in, the message is queued until the other end is there. This is especially suitable for distributed applications where you may have PDA-like units that are not always connected to the system.

Multiple Check Out

The multiple check out facility of System Modeler Source Control allows a revision of an element to be checked out of a Version Bank by more than one user at the same time. This operation is controlled by the Reserve Elements on Checkout option.

Multiple Language Facilities

A collective term for the facilities that enable a system to be translated and used in different languages.

Multiplicity

The Multiplicity property specifies the number of instances. Each instance allows the assignment of a value to the Object, typically referred to as an Attribute or Variable.

N

Name

Long names of Objects are up to 64 characters and allow double-byte characters, such as Kanji, to be used for an element name. They also allow you to give elements more meaningful names.

Namespace

All elements, with the exception of the model itself, belong to an owning element. Named Elements are identified within their owner by their name, and for this reason their owner is called a Namespace.

NOF

Non-Formatted Input/Output. A message-oriented interface between a user system and, usually, an external system and terminal. The interface to Graphical Interface Workbench uses NOF.

Nonclustered Index

Microsoft SQL Server database index in which the logical order of the index does not match the physical, stored order of the rows on disk. The data is stored in one place, the index in another, with pointers to the storage location of the data.

O

Object Browser

The Object Browser enables you to view all the elements in your model. However it does not enable any updates to be performed. This browser has three panes.

The Objects pane displays the container elements as a tree view.

The Members pane displays certain contained members of a container element selected in the Objects pane.

The Description pane displays details about an element selected in either of the other panes.

Offline input (OFF)

The Offline Input client program (OFF.exe) is used to load entire files of data into a target application. The input data is piped into the Offline client's stdin, and application responses are displayed to stderr. The input records must be in the correct format required by the target application, and must be error free.

OLTP

Online Transaction Processing (OLTP) is a generic method for transferring transactions to other systems.

OLTP Buffer Definition File

An OLTP Buffer Definition file holds the data format for communicating with external OLTP Servers.

OLTP View Ispec

On the host, an OLTP View Ispec (OVI) is used to store an OLTP View Description file. This file holds the data format for communicating with external OLTP Servers.

Overrides

Overrides are methods of a class that override either the standard methods of a class (framework methods) or the methods of a superclass.

P

Pack

A definition of the location where part of a user system or an environment database is stored. This may be either a tablespace or a directory path.

Painter

A facility used to define an Ispec screen layout, Report Frame layout or Teach Screen.

Parameter

A variable that forms part of a method interface, to pass data into and/or out of a method.

Persistent

Lasts longer than the current processing cycle. For example, persistent attributes in an <<ispec>> class are stored in the database until they are deleted (this is "indefinite" persistence), and global work lasts longer than the current transaction, and critical point SDs are stored in the database so that report processing can be recovered.

Physical Delete

Removal of a record from a database or repository.

Physical Reorganization

Copies and transforms the physical data from the existing database into a form which matches the new database schema.

Prepare method

The customers' logic executed before an ispec's automatic look ups and main logic. Its role is to validate input in preparation for the rest of the processing. This becomes the "_Prepare" method.

Previous term was – Pre LINC logic.

Presentation

Format for an interface into or out of a class. This can be a user interface (screen, form) for an ispec, or a report frame out of a report."

Previous term was – Ispec character screen, graphical screen, Report frame layout.

Primitive

A primitive object is described by what it needs to be able to do. Its contents (single, double or mixed byte text, signed or unsigned numeric, or various kinds of date), are indicated by its Type property.

Profile

Defines a set of criteria for selecting database records for a persistent class, and possibly constraints that limit to a subset of records (a conditional profile).

Previous term was – Profile.

Project

A set of classes which would be built and deployed together. In the component world this might become a component of a larger application.

Previous term was – Application.

Property

One of the characteristics of an element in the Developer model that can be defined in the properties pane in System Modeler. These are generally called "options" in previous releases of Developer.

Previous term was – Option.

Properties window

The Properties window displays the properties of a selected element and enables you to modify those properties, unless they are read-only. It also enables you to select multiple elements and displays the common properties.

Protocol Adapter

Protocol adapters are the services that enable established protocols such as RATL, SOAP, NOF, and HUB to communicate with a generated system.

Q

Query

A set of user-defined search criteria for extracting specific information from a database. Version Control permits you to define queries to interrogate the Developer Repository or a Version Control Bank. Version Control Administrators can define queries to search the Audit Store.

Query Menu

For MCP a user-maintained list of queries that can be run as needed.

R

Recovery

Process which restores database files from a backup and performs roll-forward recovery from a current transaction log file. Failed Reports must be restarted manually, as they are not automatically restarted by the recovery process.

Recursively

Performs an action on a selected element and all its individual children.

Relative Day Number

A date expressed as the number of days since January 1 of the base year. The relative day number of January 1 of the base year is zero.

Release Management

A feature of System Modeler Source Control. By assigning the same label to revisions of a number of elements, you can ‘get’ all Objects from the Version Bank with that label for the purpose of building a release.

Remote Access Server

The Remote Access Server (RAS) resides on the target host server and provides the basis for communication between user Views and systems. Can be used with Graphical Interface Workbench and Component Enabler Viewer applications.

Remote Access to LINC (RATL)

The RATL protocol is a simple wrapping of a standardized and extended set of NOF messages that provide all the functionality needed to support GUI forms.

Reorganization

Process by which the physical database for a system is updated to match the logical definition held in the Model. Refer to Logical Reorganization and Physical Reorganization for more information.

Report

Part of a System, generated and used to produce output or to carry out specialized batch processing of a Database. Consists of Report Frames and Report Main method, and a number of options that define the operations and output of the Report. The Class functions as a report by allowing only report Properties to be defined and disallowing a Presentation.

As the final part of its processing cycle, a <<Report>> class deletes itself.

Report Group

A set of Reports in a System Configuration that are generated as a group. All the Reports assigned to the group are generated by selecting the Report Group for generation.

Report Output Control (ROC)

Runtime for Windows Operating Systems uses Report Output Control (ROC) to print text only output from reports on TCP printers, including PostScript-compatible printers. In addition, you can print formatted output on TCP printers that use control characters in reports.

Report Output Control (ROC) manages output for standard reports. Control information is stored in the ROC database, while output is stored in COBOL text files. Users can view reports using the ROC client interface through Presentation Client. Procedures for browsing reports are given in the ROC help file.

Reserve

Locks and element to ensure integrity of the model, and to disallows two persons from modifying the same element at the same time. Refer to Lock.

Revision

A set of changes to an element which are stored in a Version Control Bank. A revision is assigned a unique number by Source Control. You can also assign an optional version label.

Runtime

A collective term for the software programs required to operate, control, and audit a system and its Database.

S

Schema

Database structure.

Segment Stereotyped Class

A class that automatically includes the behavior and characteristics of a business segment. This includes inheriting a processing cycle, built-in attributes like the glb.* ones.

Segments (segment-stereotyped classes) generally function as the top-level definition of an application. They are sometimes referred to as a “component”, and own all the model entities that the application consists of.

Previous term was – Business segment.

Segment Cycle

The processing cycle for processing NOF and OLTP messages in EAE 3.3; and the business segment’s Process method in Agile Business Suite.

Previous term was – LINC Cycle.

Semantic properties

These properties define the class behaviour and force the inclusion of framework members. These properties define the type and may be optional such as the AutomaticEntryCapable property. Some examples of semantic properties are Length, Primitive, PresentationType, and Stereotype. Once a semantic property has been specified it cannot be modified by an inheriting object (length).

Server Library

For MCP Server libraries are simple way of sharing stateless code using a one-way linkage mechanism (client > server).

Session Language

The session language is the language currently being used by Agile Business Suite and defines the language that is used to display items such as values and captions.

Sleeping Report

A Report that, by the use of the SLEEP Logic command, stops executing for a predetermined number of seconds or until reactivated by your system.

SOAP

Simple Object Access Protocol. A standard for encoding XML messages for transmission over HTTP. It is a lightweight protocol for the exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

Solution

A collection of projects (possibly using different implementation technologies) in Visual Studio that would be built and work together. These may all be Developer projects, or built with other tools, or a mixture of the two.

Previous term was – Application (refer to Project).

Solution Explorer

A Visual Studio window which displays a flat view of the versionable elements for each Developer project/model and provides Source Control operations on these elements.

Source Control

Visual Studio allows integration of a variety of version control tools with its source control services, providing the tool supports the SCC interface used by Visual Studio. Developer incorporate the Borland Star Team version control tool. You are not restricted to using this particular tool for System Modeler or other Visual Studio projects that you may work with.

Source Control Bank

The Source Control Bank is a file storage facility that holds controlled copies of elements, but it does not replace the Model. The Source Control Bank supports source control services, such as locking and maintaining the history of an element, while the Model functions as a work area where copies of the controlled elements are modified and tested before being moved back into the Source Control Bank.

SMTest Case

An SMTest case is an XML file, which contains a series of one or more test steps. These test steps are performed to cover a procedure or transaction, such as connection, addition, deletion, or updating data through the application.

SMOrderedTest Case

An SMOrderedTest case is a collection test steps and other SMOrderedTests grouped together. The SMOrderedTest acts as a container for SMTest cases. You can organize SMTest cases into SMOrderedTest for better management of the SMTest cases. A SMOrderedTest is also saved as an XML file.

SQL Script Stereotyped Class

A class that implements the functionality of a SQL Script. It inherits a Construct method (called “preamble” in the existing Developer) which is often called to open the cursor in preparation for iterating through the results of a select statement a Main method which is executed for each iteration of the determine statement, and a Destruct method (called “post amble” in existing Developer) that is called to release any resources used by the script. These can be overridden using SQL for a specific SQL Script Stereotyped class

Previous term was – SQL Script.

SQL Server

A relational database management system that is optionally used as the database software for systems based on the Windows operating system.

Standard Report

A specific type of Report that uses the Report Output Control (ROC) System, and whose output is written to the database. Subsequent use of the output is determined by the user. Refer to Report Output Control System. Contrast with Direct Report.

Static Object

An object on a painted form that cannot be, or is not required to be, bound to an attribute.

Stereotype

A concept taken from UML for representing a behavioral distinction. A stereotype is a subclass of an existing element with the same attributes and relationships as that element but with a different intent and possibly additional constraints. A tag indicating how an element in UML is to be interpreted. Stereotypes are used to distinguish the EAE specializations and patterns from generic model elements.

Subclass

A class that inherits public and protected members for another Class (its "superclass"). Inherited public or protected members are added to those defined for the subclass. Methods and profiles may be overridden (logic redefined) in the subclass to give it different functionality.

Subsystem

On MCP-based runtime hosts, Ispecs are grouped into subsystems. For each subsystem, there is a corresponding COMSTP program with the file name system/COMS_LINC_TP and the program name the same as the corresponding subsystem.

This is not to be confused with Agile Business Suite SADD Subsystems.

Superclass

A class that has its public and protected members inherited by another class (its "subclass").

Synchronous

A process whose caller waits for the process to complete, behaving as though it is being executed by the same thread.

System

The set of application programs and database generated from a Specification.

System Modeler

System Modeler in Agile Business Suite Developer models an application at the logical level using a combination of common Visual Studio and Developer specific windows and views. It merges the best of the mainstream object and component concepts, including COM and UML, to provide a powerful development environment.

T

Target Host

The name of the specific host on which a generated System is deployed.

Task

MCP: A task is a process. It may be either synchronous (dependent, parent waits while process executes) or asynchronous (independent, process executes in parallel to parent). For dependent, the system creates references to the objects stored by the parent while for independent the system creates copies of these critical objects when the process is initiated. The most crucial difference is, that an independent process can continue to exist after its parent has terminated. A dependent process must terminate before its parent does.

Tasks are typically used to provide a parallel processing environment, which can benefit the performance of an application.

Every process makes use of certain objects originally declared by another process. These include the task variable, the procedure the process is executing, and any objects passed as actual parameters to the process. These objects are referred to as the critical objects of the process.

Teach Screen

A screen in a host system that displays user-written help information about an Ispec. An Ispec may have more than one Teach screen associated with it. Text on graphical screens has clear (that is, see-through) backgrounds so that it inherits the color of the screen background.

Terminal Printer Spooler (TPS)

An MCP-based Enterprise Application Utility that enables users to manipulate the output of a Report to terminal printers.

Test Result

A test result is an XML file, which is saved when you play back a test case in ATT.

Tool Tip

When the mouse hovers over a dynamic object, information about the attribute behind the control is be displayed as a Tool Tip.

Transfer Attribute

Transfer attributes tell the Enterprise Output Manager application how, where, and when to transfer a file.

Translations

Translations specify how captions and strings are translated into other languages.

U

Unified Modeling Language (UML)

UML is a commonly accepted graphical language, which uses a range of different kinds of diagrams to describe systems, including software systems, but which has also been applied to other systems like organizations.

UML diagram

Class diagrams use standard UML (Unified Modeling Language) notation to represent the static relationships between model entities. You should consult a UML reference for details on UML notation.

Universal Description, Discovery and Integration (UDDI)

UDDI is a specification for maintaining standardized directories of information about web services, recording their capabilities, location and requirements in a universally recognized format.

Unreserve

Removes an existing lock on a selected element and allows the element to be edited. Refer to Reserve.

USER

The USER protocol enables users to create a custom subprogram that can be called from logic to communicate with an external system. A sample file, USER.cpp, is supplied with the Runtime installation.

The User Interface facility enables an application to initiate a transaction to another type of system, and to receive a response from that system.

V

Value Logic

Logic associated with a Type that restricts the values that item may take. For example, a numeric item may be restricted to the range 1 through 12.

Vanilla Class

A logical class element contained in the Model. This kind of class has no stereotype to enforce any pre-defined attributes or behavior, nor does it automatically participate in the Business Segment Cycle'.

Variable

Memory that stores data and is identified by name. An attribute is a variable that belongs to an object; a parameter is a variable that passes data into or out of a method, or a variable may be local to a method.

Version Control

An optional feature of System Modeler that provides source control for versionable elements. Version Control stores each version of an element so that you can access and use earlier versions if required. Refer to Source Control.

Version Control Bank

The file storage facility for Source Control. The optional Version Control Bank contains 'controlled' versions of elements but does not replace the Model database as the store for these elements. The Version Control Bank resides within any complying Version Control Tool.

Version Label

An optional, user-defined name that can be assigned to a revision of an element in a Version Control Bank.

Visibility

The Visibility property of an entity determines its scope, which is its accessibility to other entities, in relation to inheritance. For example, a method cannot refer to an attribute that is not within scope.

Visibility can be Private, Protected or Public.

W

Watch

A feature of the Debugger that monitors, or ‘watches’, the state of an attribute. When logic execution is stopped during debugging, the value of the watch item is displayed in the Watch Window.

Web Service

A generic means of invoking an operation remotely. Web services are being standardized by a number of bodies. They use common protocols to pass the request (including HTTP and SMTP), XML to encode the request, WSDL to describe their interface, and UDDI to publish their interface descriptions. Web services pass messages asynchronously. They do not operate within a session.

WinForm

A container application used to present information to the user and to accept input from the user during a runtime session.

WSDL

Web Service Description Language. A machine-parsable description of a Web Service, a WSDL file is an XML file that describes the protocol used to access a Web Service on a given host. Given a properly formed WSDL file, most Web Service frameworks are able to generate wrapper classes so that clients can talk to Web Services without needing to know about the underlying protocols.

X

XML

Extensible Markup Language (XML) is a flexible, universal file format for creating data for common information formats and sharing both the format and the data on the Web, intranets, and elsewhere. An XML file contains data in identified fields that are used by XSL style sheets to present the data in the XML file.

XSL

Extensible Style sheet Language (XSL) is used for creating style sheets that describe how data sent over the Web using XML is to be presented to a user. XSL gives developers the tools to describe exactly which data fields in an XML file to display, and exactly how and where to display them.

Y

No entries under "Y"

Z

No entries under "Z"

Appendix B
Related Product Information

The following publications contain information relevant to the definition and operation of a system. These publications are reference sources for users who have completed Agile Business Suite training courses. See your local Unisys representative for information on available training courses.

These documents are published by Unisys Corporation and are available on the Internet at http://www.support.unisys.com. Order hardcopy documents online through the Unisys Book Store at http://unisysbookstore.cgxsolutions.com/Home.aspx.

In addition to these documents, you may require documentation specific to your host to describe the operation of related software.

Unisys Agile Business Suite Installation and Configuration Guide

This document describes the installation of Agile Business Suite in standalone or multiuser environments and the administration of its working environment, including database administration, security and other issues.

Unisys Agile Business Suite Developer User Guide

This document provides information on using Developer System Modeler to design, develop, build and test systems on workstations.

Unisys Agile Business Suite Programming Reference Manual

This document contains reference material for developers, such as logic commands and System Data items used in creating systems.

Unisys Agile Business Suite Runtime for Windows® Operating System Administration Guide

This document describes the generation and operation of systems and Reports and the general administration of systems on the Windows host.

Unisys Agile Business Suite Runtime for ClearPath MCP Administration Guide

This document describes the generation and operation of systems and Reports and the general administration of systems on the ClearPath MCP host.

Unisys Agile Business Suite Component Enabler User Guide

This document describes how to configure your system for a Component Enabler application, how to configure the Component Enabler Generator and clients, and how to generate Component Enabler applications for use on client workstations or using the Web.

Unisys Agile Business Suite Component Enabler Class Reference Summary

This reference card is supplied in Acrobat PDF format to provide a quick reference by task to the common methods you would use to log on to a system using Component Enabler and retrieve data. It also lists the common response and error codes.

Unisys Agile Business Suite Client Framework Programming Reference Manual

This document contains reference material about the usage of AB Suite Access Layer APIs and customization of the client applications by using various client development tools.

Index

3826 5823-008