Wonderware

Application Server User's Guide

1/14/16

Schneider Electric Software, LLC
26561 Rancho Parkway South
Lake Forest, CA 92630 U.S.A.
(949) 727-3200

http://software.schneider-electric.com

For comments or suggestions about the product documentation, send an e-mail message to ProductDocumentationComments@schneider-electric.com.

Contents

Welcome

This guide describes how to use the ArchestrA® Integrated Development Environment (IDE) to develop and manage Application Server applications.

You can view this document online or you can print it, in part or whole, by using the print feature in Adobe Acrobat Reader.

This documentation assumes you know how to use Microsoft Windows, including navigating menus, moving from application to application, and moving objects on the screen. If you need help with these tasks, see the Microsoft online help.

In some areas of the Wonderware® Application Server, you can also right-click to open a menu. The items listed on this menu change, depending on where you are in the product. All items listed on this menu are available as items on the main menus.

Documentation Conventions

This documentation uses the following conventions:

Convention	Used for
Initial Capitals	Paths and file names.
Bold	Menus, commands, dialog box names, and dialog box options.
`Monospace`	Code samples and display text.

Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Before you contact Technical Support, refer to the relevant section(s) in this documentation for a possible solution to the problem. If you need to contact technical support for help, have the following information ready:

- The type and version of the operating system you are using.
- Details of how to recreate the problem.
- The exact wording of the error messages you saw.
- Any relevant output listing from the Log Viewer or any other diagnostic applications.
- Details of what you did to try to solve the problem(s) and your results.
- If known, the Wonderware Technical Support case number assigned to your problem, if this is an ongoing problem.

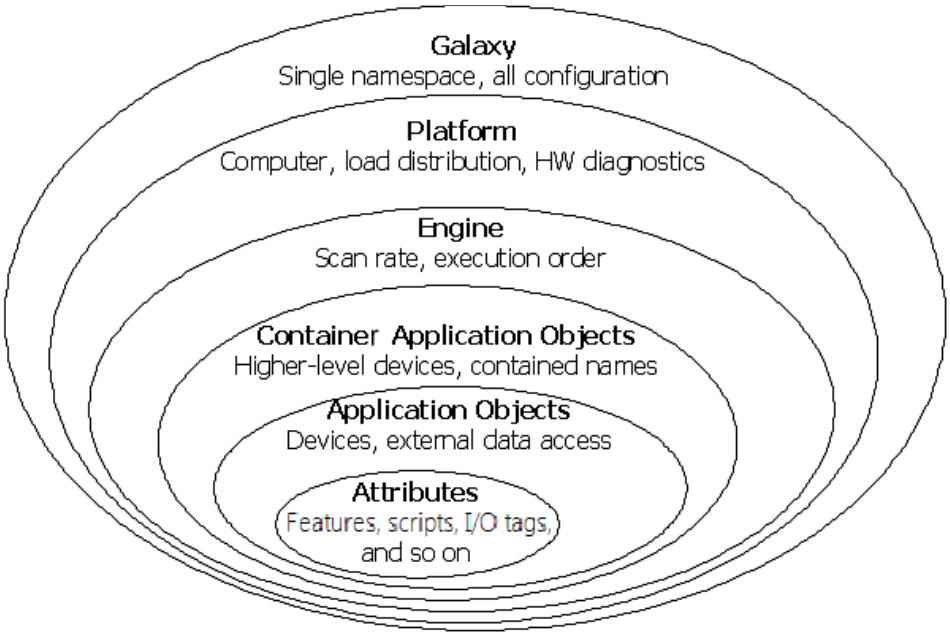
Chapter 1

Getting Started with the IDE

This section explains how to create and open a Galaxy. It also briefly describes the ArchestrA® Integrated Development Environment (IDE).

What's a Galaxy?

A Galaxy represents your entire production environment, including all computers and components that run your application. A Galaxy is a collection of platforms, engines, templates, instances, and attributes you define as the parts of your specific application. Persistent information about this collection of objects is stored in a Galaxy database.



A Galaxy database resides on a single network computer. A Galaxy database can reside on any computer on your network with the SQL Server, Bootstrap, and Galaxy Repository software installed. But, you cannot store parts of a Galaxy database on several computers.

A Galaxy Repository (GR) is the name of the single computer where the Galaxy database is located.

You can deploy Galaxy components, such as platforms and engines, on multiple computers to share the work load while applications are running. For more information, see "Deploying and Running an Application" on page 189.

A Galaxy's namespace is the set of unique object and attribute identifiers. The namespace and the values of each of its identifiers define a Wonderware® Application Server application, and can be accessed by clients of the configuration system as well as the Application Server Message Exchange in a deployed system.

A key benefit of the Application Server namespace is that it allows Application Server objects and process data to be referenced by scripts and animation links from any computer in the Galaxy without the reference needing to specify the object's location.

Galaxies also include security, which is turned off by default. Using security allows you to limit what users can do. You can add more users, security roles, and security groups later if you want. For more information, see "Working with Security" on page 353.

Starting the ArchestrA IDE

When you start the IDE, you must select an existing Galaxy or create a new Galaxy. You cannot open the IDE without opening a Galaxy.

Applying a Valid License

Before you can open the IDE, you must also have a valid license. For more information about licensing issues, see "Managing Licensing Issues" on page 416.

Managing Security Certificate Verification

The ArchestrA IDE may take 30 seconds to open on a system that is not connected to the Internet. When the IDE starts, the operating system attempts to verify the digital certificates for internal components against a Certificate Revocation List (CRL) located on a public website. If your system cannot access the public site within 30 seconds, the IDE startup process resumes and completes.

To maintain a better security profile by checking components against a CRL, keep your internet connection enabled and allow the verification to proceed.

To avoid the potential delay, in **Internet Explorer** or through **Control Panel**, open **Internet Options**, **Advanced Options**, and uncheck the **Security** option to **Check for publisher's certificate revocation**.

Creating a New Galaxy

Each time you start the ArchestrA IDE, you must connect to a Galaxy. You must either create a new Galaxy or select an existing Galaxy before opening the IDE. Also, you must have a valid Wonderware license installed before opening the IDE.

Creating a new Galaxy requires you to specify a Galaxy Repository (GR) node name and the name of the Galaxy. The Galaxy database is created and is ready for you to connect to and use.

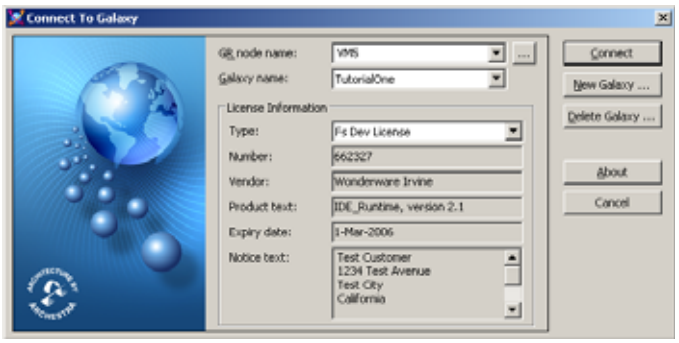
You can only create a new Galaxy on a computer with the Bootstrap and the Galaxy Repository software installed.

To be able to create a new Galaxy, TCP/IP must be enabled on the computer hosting the SQL Server database. The TCP/IP protocol setting can be verified with **SQL Server Configuration Manager**. See the *Wonderware System Platform Installation Guide* for instructions on enabling the TCP/IP protocol for SQL Server.

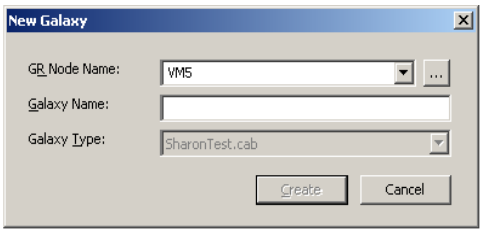
New Galaxies are created without security. To learn more about setting security for your Galaxy, see "Working with Security" on page 353.

To create a new Galaxy

1 On the **Start** menu, point to **Programs**, **Wonderware** and click **ArchestrA IDE**. The **Connect to Galaxy** dialog box appears.



2 Click **New Galaxy**.



3 Specify the properties of your new Galaxy:

- In the **GR Node Name** list, type, or select the name of a computer that has the Galaxy Repository software installed. If necessary, click the **Browse** button to locate the node where the Galaxy Repository is installed.
- In the **Galaxy Name** box, type the name of the Galaxy you want to create within that Galaxy Repository. A Galaxy name can be up to 32 alphanumeric characters, including _ (underscore), $, and #. The first character must be a letter. A Galaxy name cannot contain a blank space.
- In the **Galaxy Type** list, select the galaxy type from the available standard galaxy cab files, which are located in the BackupGalaxies folder. The following cab files are included with Application Server:
  - Default Galaxy (standard template)
  - Base_Application_Server.cab
  - Base_InTouch.cab
  - Reactor_Demo_Application_Server.cab
  - Reactor_Demo_InTouch.cab

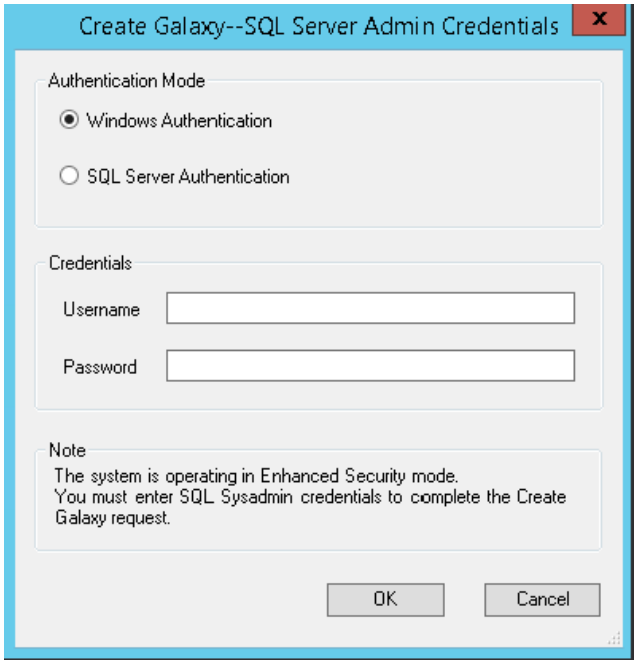
  If you have created Galaxy backups, your cab files will also appear, in addition to the cab files included with Application Server.

  The selected file is used as a template to create a new Galaxy. The system restores the selected backup Galaxy and renames it to the Galaxy name that you provided in the Galaxy Name box.

Note: You cannot use the following reserved names as Galaxy names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System. You cannot use a name that conflicts with an existing object in the backup Galaxy file.

4 Click **Create**. The **Create Galaxy** dialog box opens, showing the Galaxy database being created.

Note: If you use a backup cab file that was created in a previous version of Application Server, and if Enhanced Security mode is in effect, you will be prompted to enter SQL SysAdmin credentials before proceeding.



5 After the Galaxy database is created, click **Close**. You are ready to open the Galaxy and begin creating your application. For more information about opening an existing Galaxy, see "Connecting to an Existing Galaxy" on page 24.

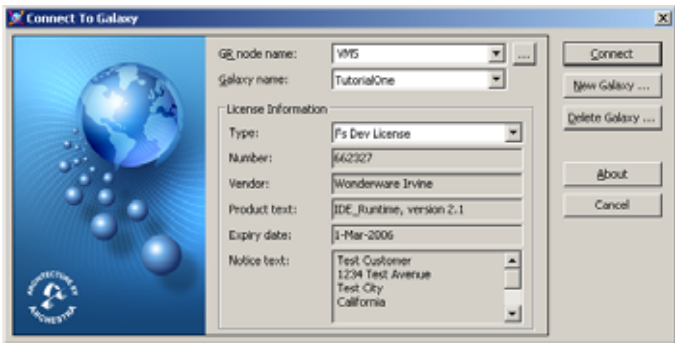
Connecting to an Existing Galaxy

Selecting an existing Galaxy lets you open a previously created Galaxy so you can work in it.

If security is enabled for an existing Galaxy, you cannot open it without logging in. If you do not have log on rights to a Galaxy, you cannot log in to that Galaxy. For more information about security, see "Working with Security" on page 353.

To connect to an existing Galaxy

1 On the **Start** menu, point to **Programs**, **Wonderware** and click **ArchestrA IDE**. The **Connect to Galaxy** dialog box appears.



2 Do the following:

- In the **GR node name** list, select the name of a computer you previously connected to. Click the **Browse** button to browse and select from the available domains and nodes on your network.
- In the **Galaxy name list**, select the name of the Galaxy on that GR node.
- Click **Connect**.

If the Galaxy you select has security enabled, the **Login** dialog box appears. Type your user name and password and click **OK**.

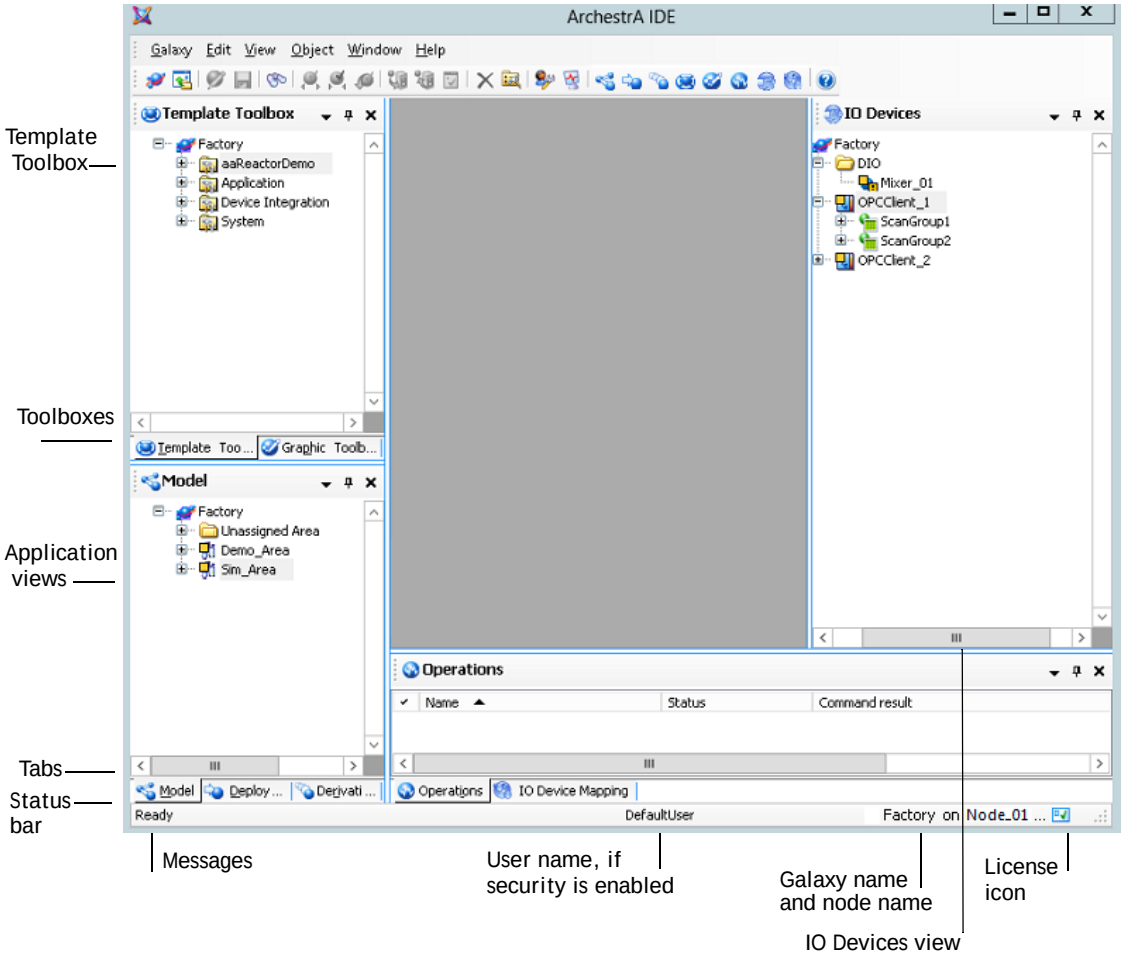
The Galaxy opens in the IDE. You are ready to start working with your Galaxy.

Getting Around the IDE

The IDE is the integrated design and development tool from which all ArchestrA objects are configured and deployed to target computers. You work from the IDE to create, configure, and maintain the objects that comprise your application and the underlying infrastructure that supports your application.

Using the ArchestrA IDE, you can import new types of objects in to the Galaxy Repository, configure new ones, and deploy them to computers in your network. Multiple users can work concurrently on different sets of objects from different ArchestrA IDEs.

After you open a Galaxy, the IDE opens and shows the different views of your Galaxy.



For a complete discussion of items such as templates and instances, see "About Templates and Instances" on page 45.

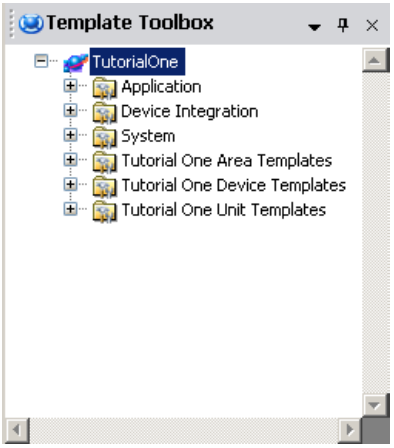
Views in the IDE include:

Template Toolbox	Expand the top level folders to see the different templates in the toolboxes.
Graphic Toolbox	Contains the global ArchestrA graphics that can be used in the Galaxy. For information, see the *ArchestrA Graphic's User Guide.*
Application views	Click the tabs at the bottom or icons at the top to open: • **Model view** - the object relationship to the automation scheme layout. The objects are organized into Areas that typically represent the physical plant layout. • **Deployment view** - the object relationship to the computers that comprise the deployed system that the objects run on. • **Derivation view** - the derivation path from base template to the instances. This view allows a user to see all object instances that were based on a given template. All templates and instances appear in this view. • **IO Devices view** - the object relationship to scan groups and DI objects. This view lets you assign objects that were configured for I/O automatic assignment in the Object Editor to scan groups, and assign scan groups to Device Integration (DI) objects. The IDE then builds the I/O references for each object. • **IO Device Mapping view** - shows the relationships of objects and attributes to scan groups and DI objects. This view lets you see, validate, and edit the I/O references of object attributes that have been automatically configured. • **Operations view** - shows the results of validating the configuration of objects.
Status bar	Shows messages, user name, Galaxy name and node, and license information. Turn off the Status bar by clicking **Status** bar on the **View** menu.

Using the Template Toolbox

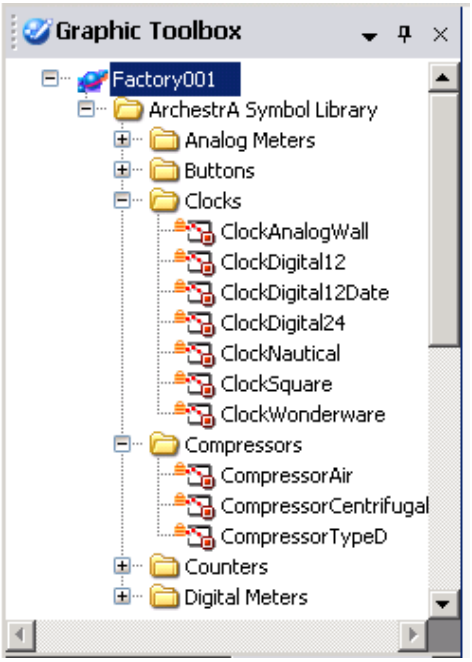
The Template Toolbox lists template toolsets, which contain object templates. The Template Toolbox shows a tree view of template categories in the Galaxy. Double-click a category to expand the toolset and show the templates within it.

A new Galaxy is automatically populated with base templates.



Using the Graphics Toolbox

The Graphic Toolbox shows a treeview of toolsets that contains ArchestrA Symbols and Clients Controls. Double-click the toolsets to open them and reveal the graphics they contain. A new Galaxy is automatically populated with a library of Graphics organized in toolsets.



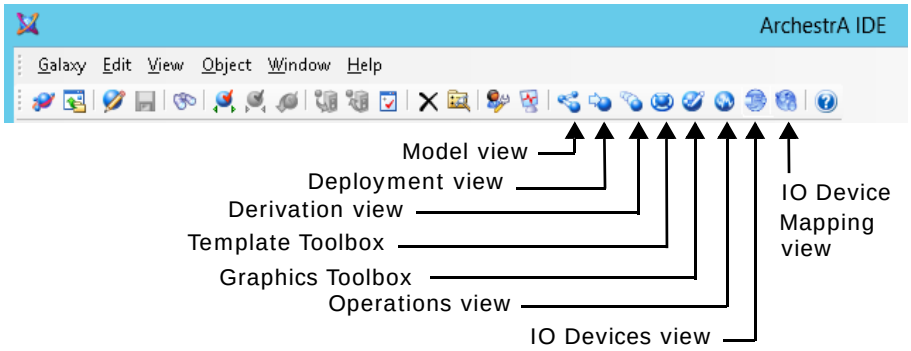
Using IDE Application Views

Base templates and non-base templates are included with Application Server. The templates are automatically imported into the IDE when you first create a Galaxy.

Base templates appear in the Template Toolbox and in the Derivation view with a $ as the first character of their name. You cannot directly modify base templates but you can use them to create your own objects or derived templates.

When you move from one view to another, the selected object in the first view is selected in the second view, if the object exists in that view. For example, templates are not shown in the Deployment view, Model view and IO Devices view.

You can open views from the **View** menu, or click the icons in the main toolbar. Keyboard shortcuts can also be used to open the views.



View	Shortcut	Icon
Model View	**Ctrl+Shift+M**	
Deployment View	**Ctrl+Shift+D**	
Derivation View	**Ctrl+Shift+R**	
Template Toolbox	**Ctrl+Shift+T**	
Graphics Toolbox	**Ctrl+Shift+P**	
Operations View	**Ctrl+Shift+O**	
IO Devices View	**Ctrl+Shift+I**	
IO Device Mapping view	**Ctrl+Shift+G**	

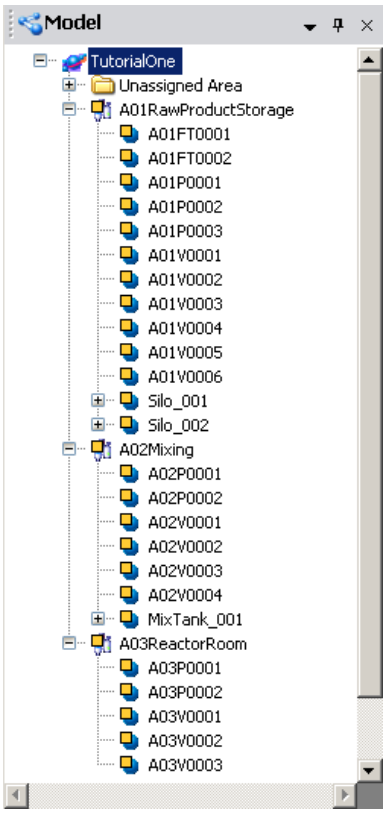
Model View

The Model view shows objects in terms of their physical or containment relationships, and allows you to organize them through a folder structure. This Model view most accurately represents an application perspective of the processes that users are emulating: for instance, specific process areas, tanks, valves, pumps and their relationships based on containment.

For more information about containment, see "Creating Contained Templates" on page 59.

Note: You must undeploy an object before reassigning it to another object.

The tree structure acts like a standard Microsoft Windows Explorer tree. Initially, it shows a simple hierarchy: <Galaxy name> and the Unassigned Area folder.



In the Model view, all objects are grouped by areas and by containment relationship. The Model view shows these relationships in the following ways:

- The top of the tree is the Galaxy.
- Top-level Areas are shown under the Galaxy.

- Within each Area, contained Areas are listed. Areas support hierarchical composition; that is, they support sub-Area construction. Areas can be nested only 10 levels (after the sub-area is 10-levels deep, you cannot add another sub-level).
- Objects that belong to an Area are listed under the Area.
- Objects contained by other objects are listed under their respective containers. Multiple levels are allowed. For more information about containment, see "Creating Contained Templates" on page 59.

Note: Contained objects belong to the same Area as the object that contains them.

Some object's hierarchical, or contained, names are truncated if you have multiple levels shown. To view the entire hierarchical name, select the object and click **Properties** on the **Galaxy** menu. The entire hierarchical name is shown in the **Properties** dialog box. For more information about hierarchical names, see "Using Contained Names" on page 65.

- Objects that currently do not belong to an Area are listed under Unassigned Area. Containment relationships between parent and child objects are shown there.

In each branch of the tree, objects are listed in alphabetical order. Default objects are shown in bold.

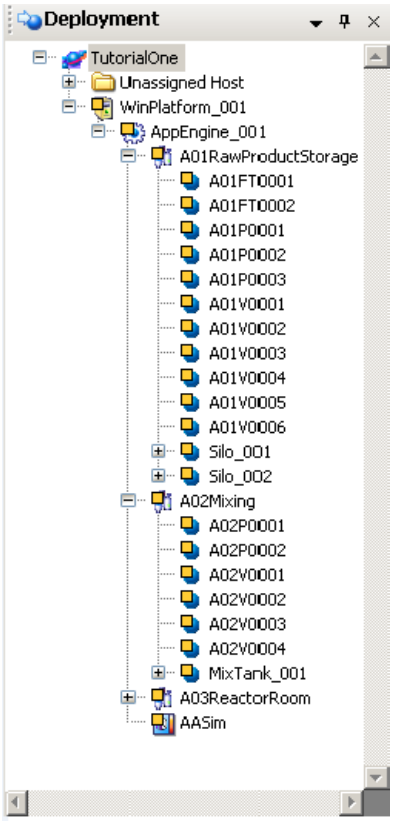
To assign an object to another, drag it onto the host object. If that object is an inappropriate assignment match, the international **Not** symbol appears. To unassign an object, drag it to the **Unassigned Host** folder.

Deployment View

The Deployment view shows instances only in terms of their assignment relationships. This view allows you to organize those objects through a folder structure.

This view shows which objects instances reside on which computers. In the ArchestrA environment, the physical location of object instances is not required to approximate the real-world environment it models. The Deployment view does not need to reflect your physical plant environment.

The tree structure acts like a standard Windows Explorer tree. It is initially divided into two hierarchical levels: <Galaxy Name> and the Unassigned Host folder.



In the Deployment view, objects appear in a tree according to their distribution relationships in a multi-node system in the following ways:

- The top of the tree is the Galaxy.
- WinPlatforms are shown under the Galaxy.
- Under each WinPlatform, assigned AppEngines are listed.
- Under each AppEngine, assigned Areas and DIObjects, such as DINetwork Objects, are listed.
- Under each Area, assigned ApplicationObjects are listed.
- Under each ApplicationObject, contained ApplicationObjects are listed. Multiple levels are allowed.
- Under each DINetwork Object, assigned DIDevice Objects are listed.
- Unassigned objects are grouped together in the **Unassigned Host** folder. Area and containment relationships are maintained in this view.

Important: DINetwork objects have specific configuration limits such as whether more than one object can be deployed to a single WinPlatform. The IDE does not check for these limits. For more information about configuration limits, see the online help for the DINetwork object.

Under each branch of the tree, objects are listed in alphabetical order. Default objects are shown in bold.

For objects shown in any view, you see the following symbol in the corner of the object's icon:

	Not deployed
	Deployed
	Deployed, but pending configuration changes exist that have not been deployed.
	Deployed, but software modifications exist that have not been deployed.
	Applies only to redundant AppEngines. An AppEngine is undeployed, but its redundant pair is deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, but its redundant pair is not deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, its redundant pair is not deployed, and pending configuration changes exist that have not been deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, its redundant pair is not deployed, and software modifications exist that have not been deployed.
[No icon]	Good. No additional icon is displayed for deployed objects running on-scan or off-scan with a good status.
	Warning
	Error. The object in an Error state and cannot be deployed.
	The process of asynchronously transferring InTouchViewApp files to the target node is not yet complete. This icon is normally visible for only a few moments at the end of an InTouchViewApp deployment operation, unless the object is deployed on a slow network. If the icon remains for more than a few moments, check if the InTouchViewApp is OnScan, as the files will not transfer if the InTouchViewApp is OffScan. This icon completely replaces the original while it is shown.

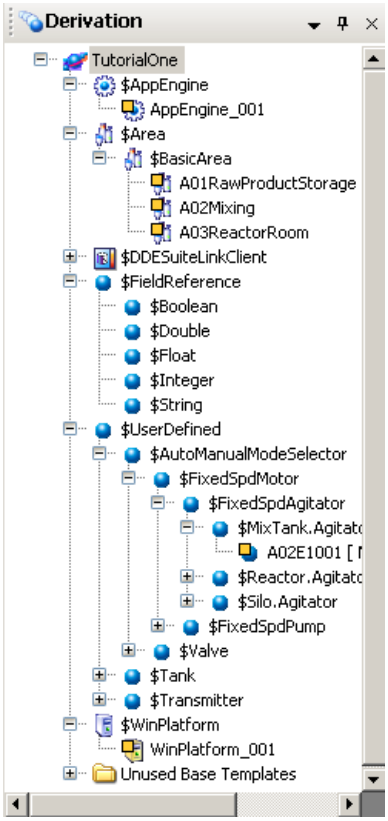
To assign an object to another, drag it onto the host object. If that object is an inappropriate assignment match, the international **Not** symbol appears. To unassign an object, drag it to the **Unassigned Host** folder.

Note: You must undeploy an object that is currently deployed before reassigning it from one object to another.

Derivation View

The Derivation view shows objects and templates in terms of their parent/child relationship. An object derived from another object appears in a hierarchy level under it.

The tree structure acts like a standard Windows Explorer tree, and initially is divided into three hierarchical levels: <Galaxy Name>, <Used Base Templates>, and the Unused Base Templates folder.



In the Derivation view, objects appear according to their parent-child relationship in the following ways:

- The top of the tree is the Galaxy.
- Base templates with associated child objects, either derived templates or instances, are shown under the Galaxy.

- Under each base template, derived templates and instances created from the base template are listed. Multiple levels are allowed. Instances created from derived templates are listed under their parents.
- Templates with no associated derived templates or instances are grouped together in the **Unused Base Templates** folder.

Objects with names that start with a "$" are templates, either base or derived. Under each branch of the tree, child objects are listed in alphabetical order.

As in other views, dragging one object onto another in the **Derivation view** associates the two objects based on the predefined rules of the object types. For example, you can drag ApplicationObjects onto other ApplicationObjects but you cannot drag ApplicationObjects to an Engine.

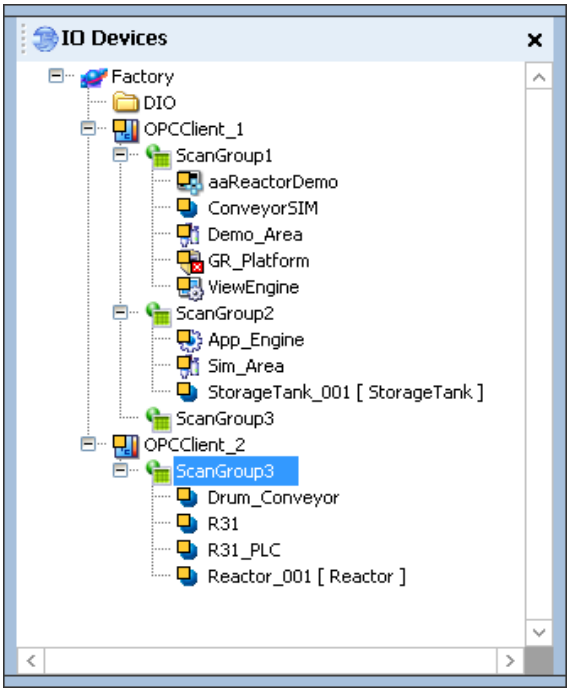
IO Devices View

The **IO Devices** view is used for I/O auto assignment and shows the relationship of system objects and application objects to scan groups and Device Integration (DI) objects. Before using this view, system and application objects must be configured to use I/O auto assignment by adding one or more attributes to the object and activating the I/O feature. See "Editing Objects" on page 69 and "Using I/O Auto Assignment" on page 156 for additional information.

You will also use the Object Editor to add scan groups to DI objects (OPCClient, DDESuiteLinkClient, or RedundantDIObject) before using the **IO Devices** view.

Once you have configured system and application objects for I/O auto assignment, and added scan groups to at least one DI object, you will assign the system and application objects to scan groups in the **IO Devices** view. This links (auto-binds) the system and application objects to PLCs, through the DI objects.

To open **IO Devices** view, click **IO Devices** on the **View** menu. You can also open the **IO Devices** view by pressing **Ctrl+ Shift+I**, or click the **IO Devices** icon in the main toolbar of the IDE.



System and application objects appear as child objects to scan groups, and scan groups appear as child objects to DI objects.

Note: Only instances, and not templates, appear in the **IO Devices** view.

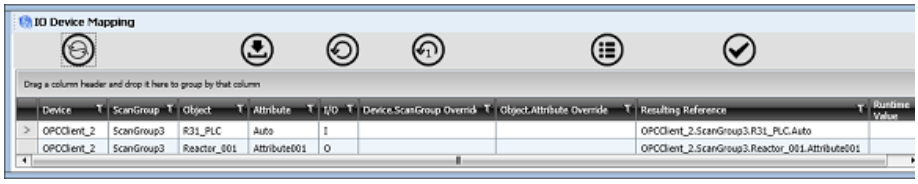
Device linkage, which is established by assigning a system or application object to a scan group, ensures that all of the object's attributes that have been prepared for automatic I/O assignment will match up with a corresponding input source or output destination in the scan group. The I/O references are then referred to as auto-bound.

Important: Objects must be undeployed before they can be assigned to a scan group.

System and application objects that are not yet assigned to a scan group are placed under the folder "Unassigned IO Devices." To assign objects to a scan group, and thus allow I/O automatic assignment to occur, simply select objects in the Unassigned area, and drag and drop them to the applicable scan group. Once an object has been assigned to a scan group, the I/O reference for each I/O attribute is automatically configured.

IO Device Mapping View

The **IO Device Mapping** view shows I/O references for attributes that have been configured for I/O auto assignment, and that have been assigned to a scan group through the **IO Devices** view. To open this view, click **IO Device Mapping** in the **View** menu. The **IO Device Mapping** view will also open if you select a DI object, scan group, or object assigned to a scan group in the **IO Devices** view. Other ways to open this view are by pressing **Ctrl+Shift+G**, or by clicking the **IO Device Mapping** icon in the main toolbar.



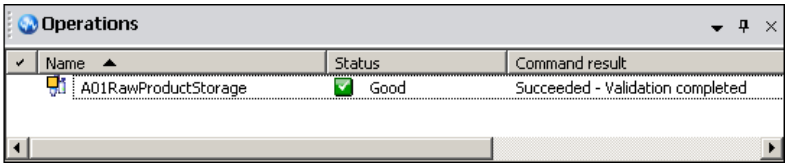
The **IO Device Mapping** view shows I/O auto assigned attributes with resolved I/O references for objects that have been assigned to scan groups. The **IO Device Mapping** view lets you enter override values to change the I/O reference. You can also validate the I/O references.

You select the references that are displayed in the **IO Device Mapping** view by selecting objects in the **IO Devices** view. Manually configured references are not displayed in the **IO Device Mapping** view, nor are attributes associated with objects that are not assigned to a scan group. Only I/O references for the object or group of objects currently selected in the **IO Devices** view are displayed. You can select DI objects, scan groups, and application and system objects in the **IO Devices** view. If only the top level item in a device hierarchy is selected, all subordinate items are automatically selected. If, however, you select a subordinate item along with the top level item, only I/O references for the selected items are displayed. You can select multiple items at different hierarchical levels.

Note: Selecting a subordinate object will exclude non-selected objects within the device hierarchy, even if the parent object is selected.

Operations View

The **Operations** view shows the results of validating the configuration of objects. You may need to open it before you see it. On the **View** menu, click **Operations**.



During the validation of an object, its icon and name appear with the status of the operation.

Important: You can validate both templates and instances if they are checked in.

The status of the object (**Status** column) is shown with an icon and a descriptive word or phrase.

When validation is complete, the **Command Result** column shows a "Succeeded" or "Failed" message and additional information about the validation results. For more information about validating objects, see "Validating Objects" on page 105.

Note: You can validate all objects in the Galaxy by running the Validate operation on the Galaxy object. In that case, Command Result messages are shown after all objects in the Galaxy are validated.

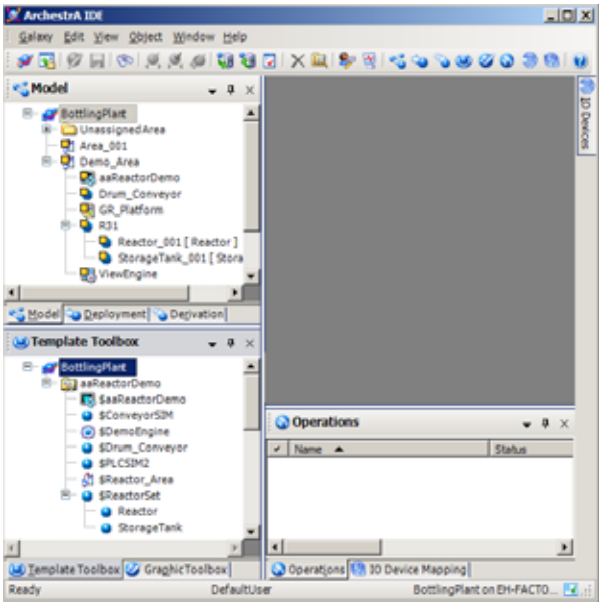
The **Operations view**, like the **Template Toolbox** and **Applications views**, is also updated as the status and conditions of objects in the Galaxy change.

Customizing Your Workspace

You can customize your workspace by docking and floating the IDE's views. You can also hide some of the views.

Docking Views

To dock a view, drag the view to the location you want it. For example, drag the title bar of the Template Toolbox and dock it under the Application views. You can also undock it by dropping it anywhere on your desktop. Drag it back to dock it again.



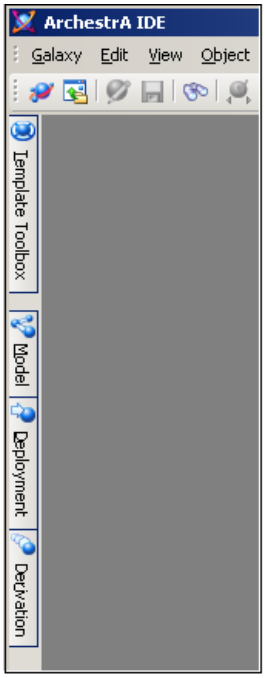
Floating Views

You can also float a view. With your cursor over the view you want to float, click the arrow. On the menu that appears, click **Floating**.

You can also float just one view. To float the **Derivation** view, click the arrow. On the menu that appears, click **Floating**. The view floats on your desktop. You can move it to another location or dock it.

Hiding Views

To hide a view, click the **Pin** icon. The views "hide" as tabs on the side of the window. The Operations view hides at the bottom of the window.



When you move the mouse over the tab, the view expands into the workspace.

Resetting the Workspace

You can easily reset the IDE workspace and restore views back to their default locations.

To return the views to the default docking

- On the **View** menu, click **Reset Layout**.

Synchronizing the Views

You can specify that a selected object stay selected as you move through the views. In any of the views, select the object you want to synchronize. On the **View** menu, click **Synchronize Views**. Now as you move from one view to another, that object stays selected.

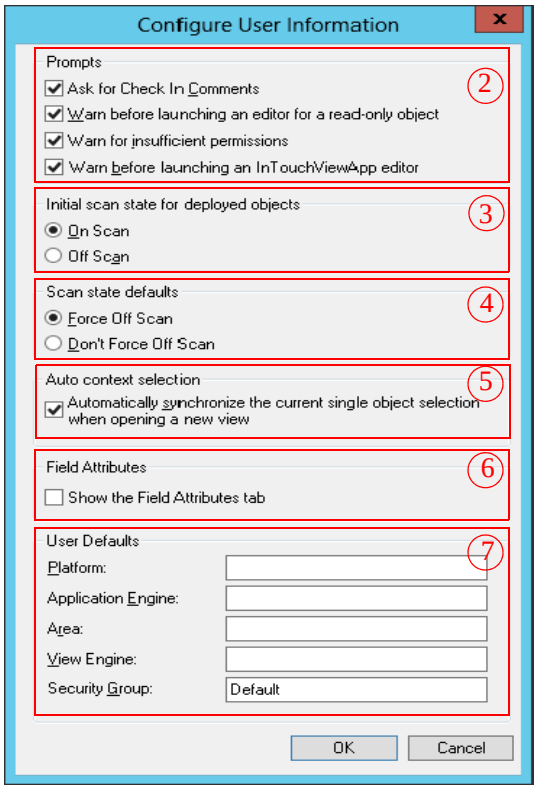
Configuring User Information

You can configure options for a Galaxy, including prompts for check-in comments and user defaults. These user options apply to the currently open Galaxy and do not change options for other Galaxies.

If you specify a security group, that security group must already exist. For more information about security and security groups, see "Configuring Security" on page 359.

To configure user information

1 On the **Edit** menu, click **User Information**. The **Configure User Information** dialog box appears.



2 In the **Prompts** area, do one or more of the following:

- Select the **Ask for 'Check In' Comments** check box if you want to be prompted to type comments when checking in templates and objects.

- Select the **Warn before launching an editor for a read-only object** check box if you want to see a prompt that informs you if you are opening an instance or template as read-only. The prompt warns you if you open an instance or template while someone else is working on it. If someone else is working in the instance or template, you cannot make changes.
- Select the **Warn for insufficient permissions** check box if you want to see a prompt that tells you if you have permission to create or modify InTouchView applications. These permissions authorize or prevent you from creating and modifying InTouchView Applications.
- Select the **Warn before launching an InTouchViewApp editor** check box if you want to see a prompt each time you attempt to edit an InTouchView application instance. You can edit the associated template or cancel the operation. If you do not select this check box, the request to edit the InTouchViewApp instance is automatically redirected to the associated template.

3 Select the **Initial scan state for deployed objects**. You can select **On Scan** or **Off Scan**. You can change this setting on an individual basis in the **Deploy** dialog box when you deploy. For more information, see "Deploying and Running an Application" on page 189.

4 Select the **Scan state defaults** when undeploying or redeploying instances. **Force Off Scan** will attempt to take the target object offscan when an already deployed object is redeployed. **Don't Force Off Scan** does not force the target to go off scan when you deploy.

Note: Redeployment of objects that are currently deployed on-scan will be cancelled unless this option is selected.

5 Make your **Auto context selection**.

6 In the **Field Attributes** section, select the **Show Field Attributes** check box to display the **Field Attributes** tab in the User Defined Object editor.

By default, the **Field Attributes** tab is displayed only if field attributes are already defined for the object. You can override the default by selecting this option. For more information about working with legacy field attributes, see "About the Field Attributes, UDAs, and Extensions Pages" on page 85

7 In the **User defaults** area, provide the following object names of Framework objects you select to be defaults with respect to assignment relationships.

- Type the **Platform** name.
- Type the **Application Engine** name.
- Type the **Area** name.

- Type the **View Engine** name.
- Type your **Security Group** name, if any.

8 When you are done, click **OK**.

Logging on and Logging off

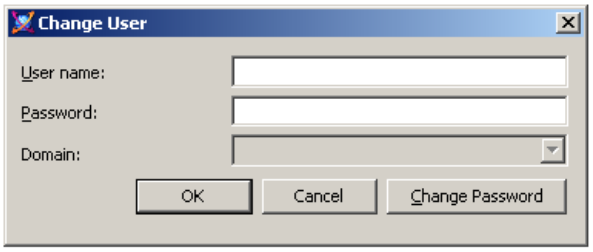
Some Galaxies have security associated with them. If you try to open a Galaxy with security, you need to log on to the Galaxy to open it.

Note: If you do not have logon rights, you cannot open a Galaxy.

For information about setting up security in the ArchestrA environment, see "Working with Security" on page 353.

To log on to a Galaxy

1 When you open a security enabled Galaxy, the **Change User** dialog box appears.



2 Type your **user name** and **password**. If OS authentication security is enabled for the Galaxy, you must select the **Domain** on which your user account is located. If the list is unavailable, the selected Galaxy is on your local computer.

You can change your password after you type your user name and password. Click **Change Password**. Type the new password and then retype it.

3 Click **OK**. Your logon data is validated by the Galaxy Repository being accessed. Depending on operating system security, the IDE opens. If the GR does not recognize your user name or password, you are prompted to enter them.

Changing Users

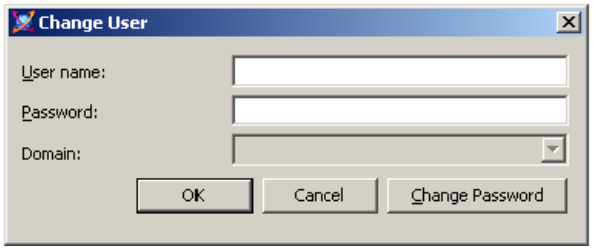
You can change users in a Galaxy at any time. Any security restrictions associated with a user change when the user logs on or logs off from the Galaxy. For more information about users and security, see "Configuring Security" on page 359.

If the Galaxy has not been configured to enable security, you see a message. All users in an open security environment are treated as the DefaultUser by the Galaxy. This means all users have full access to everything.

To change users

1. On the **Galaxy** menu, click **Change User**.

   If security is enabled on the Galaxy, the **Change User** dialog box appears.



2. Enter your logon information and click **OK**.
   - If needed, click **Change Password** to change the password for the new user.
   - Type the new password and then retype it.
3. Click **OK**.

Chapter 2

Getting Started with Objects

Before you start modeling your application using the Application Server, you must understand templates and object instances.

Templates are elements in Application Server that contain common configuration parameters for object instances that you use multiple times in your application.

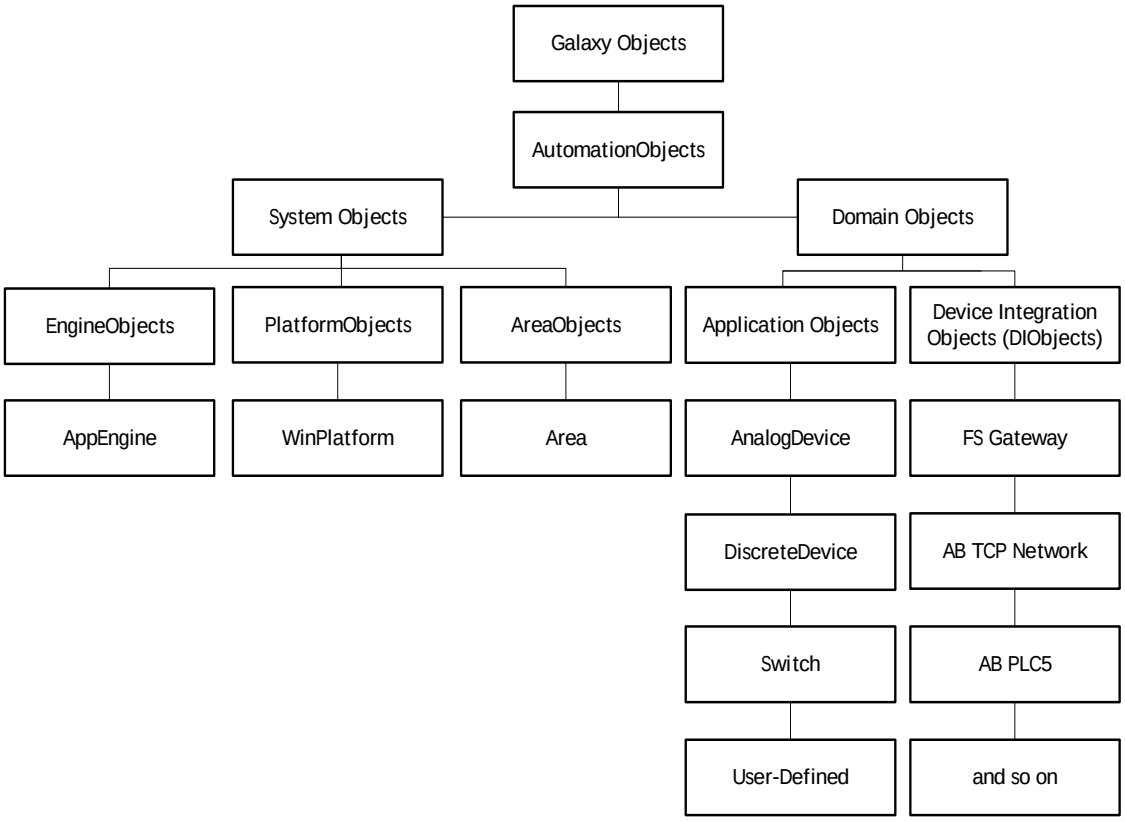
For example, you might create a template for valves. You configure the template with all the unique attributes for valves. You use that template to make object instances of valves. You can further configure and customize each object instance to represent a specific valve.

Object instances are the specific devices in your environment, such as diaphragm valves or very complex devices, like a reactor. You create an instance from a template and then customize the specific instance as needed.

Instances are deployed to the run-time environment. Templates exist in the development environment and cannot be deployed.

Creating templates and instances is very similar to object-oriented programming. For example, templates and instances have a parent/child relationship that involves inheriting attributes. There are differences, however, between object-oriented programming and creating templates and instances in Application Server.

Collectively, templates and instances are called objects. The following graphic shows the different kinds of objects and how they are organized.



If you are new to this kind of programming, the next section explains the basic concepts you need to know before you start. If you are familiar with object-oriented programming, the concepts in the next section may be familiar to you, but notice the important differences between object-oriented programming and Application Server.

About Templates and Instances

Understanding templates and instances is critical to working with Application Server.

Instances

Instances are the run-time objects created from templates in Application Server. Instances are the specific things in your environment like processes, valves, conveyer belts, holding tanks, and sensors. Instances can get information from sensors on the real-world device or from application logic in Application Server. Instances exist during run time.

In your environment, you may have a few instances or several thousand. Many of these instances may be similar or identical, such as valves or holding tanks. Creating a new valve object from scratch when you have several thousand identical valves is time-consuming. That's where templates come in.

Templates

Templates are high-level definitions of the devices in your environment. Templates are like a cookie cutter from which you can make many identical cookies.

You define a template for an object, like a valve, one time and then use that template when you need to define another instance of that item. Template names have a dollar sign ($) as the first character of their name.

A template can specify application logic, alarms, security, and historical data for an object.

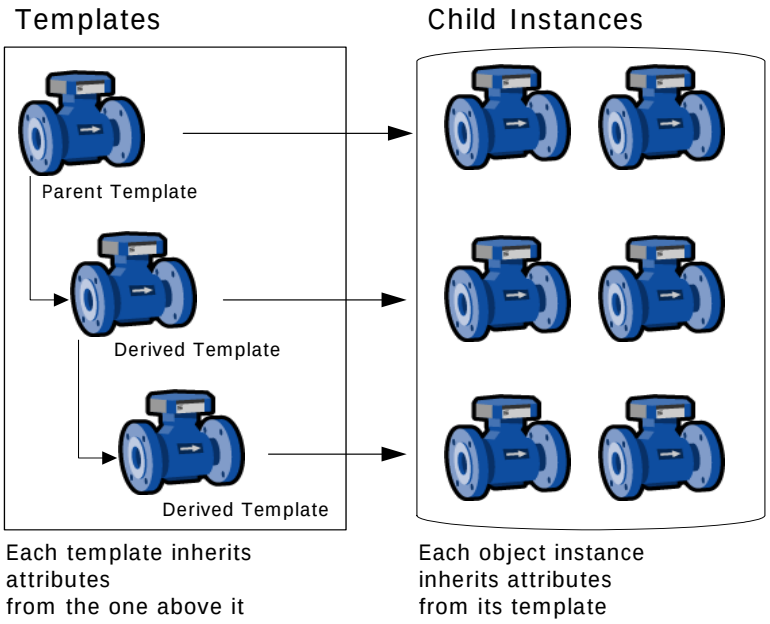
A template can also define an area of your environment. You can extend and customize a template by adding attributes, scripts, or features to meet the specific needs of your environment. Objects inherit attributes, scripts and features from their parents.

Application Server comes with predefined templates, called base templates. You cannot modify base templates. All templates you create are derived from base templates.

You can also nest templates, or contain them. Contained templates consist of nested object templates that represent complex devices consisting of smaller, simpler devices, including valves. A reactor is a good candidate for containment.

Templates only exist in the development environment.

Using the Diaphragm valve template, you can quickly create a Diaphragm valve instance when you need another Diaphragm valve in your application.

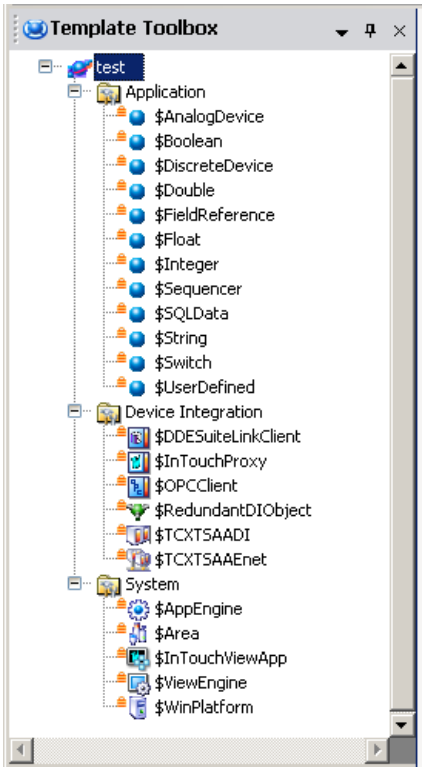


Propagation

If you need to change something about all diaphragm valves, you can change the template for the Diaphragm valve and all diaphragm valves in your application inherit the changes, assuming the attributes are locked in the parent template. This makes it easy to maintain and update your application.

About Base Templates

When you first open the IDE, you see the base templates in the **Template ToolBox**.



You cannot modify base templates. You use base templates to create derived templates, which are copies of the base templates. You modify your derived templates and create instances of them for your applications.

The template classes are as follows:

- Application templates

  Use these templates to represent real devices in your Galaxy. These devices represent real objects in your environment. For example, use the DiscreteDevice base template to create a derived template for valves.

- Device integration templates

  Use these templates to create instances that communicate with external devices. For example, use the DIObject base template to create a derived template for a PLC device.

- System templates

  Use these templates to define system instances, like other computers.

Application Templates

These base templates let you easily create devices in your Galaxy. They contain the properties you need to set for each kind of device. For example, a DiscreteDevice device contains all the settings you need to specify for an on/off device. Of course, you can extend and customize *any* device by using attributes, scripts, and features.

Device Integration Templates

These base templates represent the communication with external devices. External devices run on the application engine.

For example:

- DINetwork object – Refers to the object that represents the network interface port to the device through the Data Access Server. The object provides diagnostics, and configuration for that specific card.
- DIDevice object – Refers to the object that represents the actual external device (such as a PLC or RTU), which is associated to the DINetwork Object.

System Templates

These objects represent the parts of a Galaxy and not the domain they are monitoring/controlling. These base templates let you create more system level grouping and computers, such as areas you add objects to or another host AppEngine.

WinPlatform Object

The WinPlatform platform object is a key base object because you need a platform to host the objects you are modeling. This object:

- Calculates various statistics for the node it is deployed to. These statistics are published in attributes.
- Monitors various statistics related to the node it is deployed to. These monitored attributes can be alarmed and historized.
- Start and stop engines, based on the engines startup type which are deployed to it.
- Monitor the running state of engines deployed to it. If the platform detects an engine failed, it can, optionally based on the value of the engine's restart attribute, restart the engine.

AppEngine Object

The AppEngine object must have a Platform on which to run. This object:

- Hosts ApplicationObjects, device integration objects and areas.
- Contains the logic to set up and initialize objects when they are deployed.
- Contains the logic to remove objects from the engine when they are undeployed.
- Determines the scan time which all objects within that particular engine run.

Area Object

All application objects belong to an area. Areas can contain sub-Areas. Areas provide a key organizational role in grouping alarm information and providing that information to those who use alarm/event clients to monitor their areas of responsibility.

The values of three Area object alarm attributes can be saved to the historian:

- Active alarm counter
- Unacknowledged alarm counter
- Disabled (or silenced) alarm counter

ViewEngine Object

The ViewEngine object must have a Platform on which to run. This object:

- Hosts InTouchViewApp objects.
- Contains the logic to set up and initialize objects when they are deployed.
- Contains the logic to remove objects when they are undeployed.
- Determines the scan time which all objects within that particular engine run.

InTouchViewApp Object

The InTouchViewApp object must have a ViewEngine on which to run. This object:

- Manages the synchronization and delivery of files required by the associated InTouch® application.
- Provides run-time access to tags on the associated InTouch application.
- Starts WindowMaker for the associated InTouch application when edited.

About Derived Templates

All templates you create within the IDE are derived templates.

When creating your Galaxy application, plan ahead and create derived templates for devices of a certain type so you can use the templates to create the individual instances.

A new derived template is an exact copy of its parent template with the possible exceptions of locking and security and modified attribute values. You can lock attributes to prevent them from being modified in child templates.

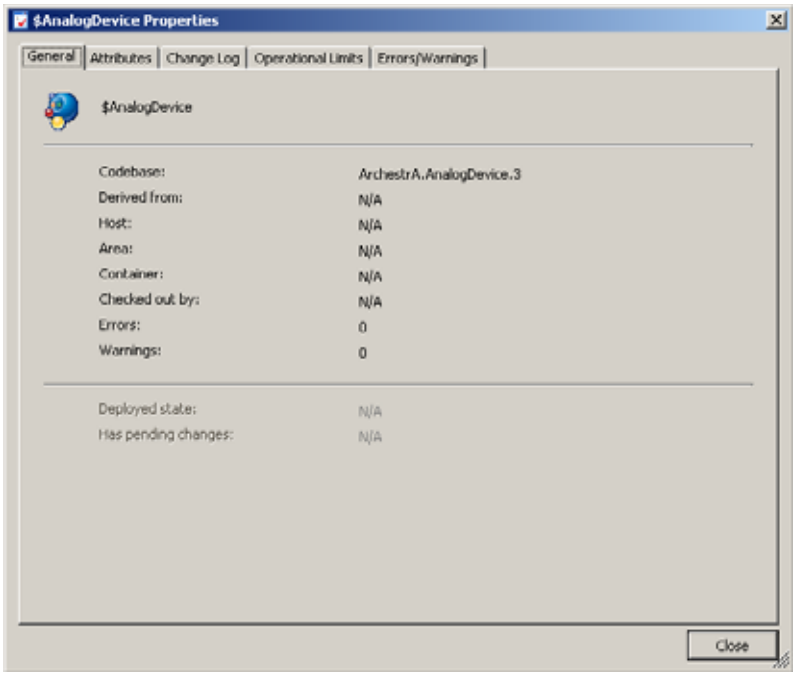
After you create a new derived template, you can customize it. For more information about customizing and extending templates, see “Creating Derived Templates” on page 56.

Every template has a set of attributes and default values. When you create an instance, attributes are inherited by the instance. In the instance, you can reconfigure many of the attributes inherited from the parent template if they are not locked on the parent template.

For more information about customizing instances, see “Working with Objects” on page 53.

Viewing Object Properties

You can view the properties of an object by right-clicking and clicking **Properties**. Object properties vary, depending on the type of selected object and whether it is a base template, a derived template, or an instance.



The **Properties** window contains the tabs that allow you to examine different object properties. Only tabs applicable to the object are displayed.

Tab Name	Displayed for Object Type:
General	All objects including GalaxyObject
Attributes	All objects except GalaxyObject
References	Instances only
Cross References	Instances only
Change Log	All objects including GalaxyObject
Operational Limits	All objects including GalaxyObject
Errors/Warnings	All objects except GalaxyObject

The **General** tab contains basic information for the object, such as the codebase (major version), what template it is derived from, and any current errors and warnings.

The **Attributes** tab lists the object's attributes and values. In addition to attributes that have been added to the object, attributes for Errors and InAlarm are also listed, as are attributes for CodeBase (major version of the object), MinorVersion ("dot" version), and ConfigVersion. The ConfigVersion will increment each time the object has been checked in after an edit.

The **References** tab lists I/O references for the object.

The **Cross References** tab lists any attributes that are cross referenced for the object.

The **Change Log** tab is a historical list of changes, including deploy/undeploy operations, that have been made to the object.

The **Operational Limits** tab lists prohibited actions and the reason for the prohibition. For example, Check In is not allowed for objects that are already checked in.

The **Errors/Warnings** tab lists any errors or warnings for the object.

For more information about specifying the properties of objects, see "Working with Objects" on page 53.

Chapter 3

Working with Objects

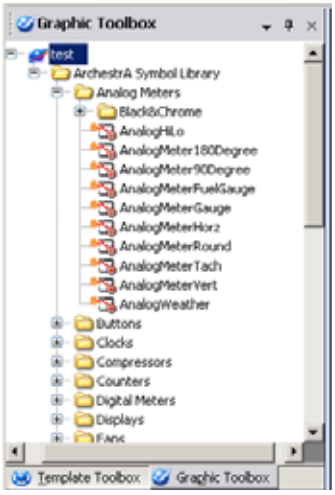
You can work with objects using the Application Server development environment. Both templates and instances are collectively referred to as objects. For more information about what templates and instances are, see “About Templates and Instances” on page 45.

Managing Toolsets

Toolsets are high-level folders shown in the Application Toolbox. You can create a toolset to store the templates and graphics used in your application. You can also create toolsets within toolsets.

Use the Template Toolbox to view and organize object templates.

Use the Graphic Toolbox to view and organize ArchestrA Symbols.



You can move content between their respective toolsets. You can also show or hide toolsets to make the workspace less cluttered.

Creating Toolsets

When you create your own toolset, it must have a unique name. Toolset names are not case sensitive, so `Valves` is the same name as `valves`. A toolset name can be up to a maximum of 64 alphanumeric and special characters, including spaces, except $.

To create a new toolset in the Toolbox

1. On the **Galaxy** menu, point to **New** and click either **Template Toolset** or **Graphic Toolset**.
2. Type a name for the new toolset.

   A new toolset appears and is in focus. Now, you can drag templates into the new Template toolset, or you can drag graphics into the Graphics toolset.

Creating Child Toolsets

Toolsets can be created within existing toolsets. Nested toolsets help in further organizing templates and graphics. You can create a maximum of ten levels of toolset folders.

To create a child toolset

1. Select the parent toolset.
2. On the **Galaxy** menu, point to **New** and click either **Template Toolset** or **Graphic Toolset**.
3. Type a name for the new toolset.

   A new toolset appears beneath the parent toolset and is in focus.
4. Drag templates into the new child Template toolset, or you can drag graphics into the Graphics toolset.

Deleting Toolsets

You can delete toolsets you no longer want or need. Before you start, make sure you move or delete all content from the toolset.

The toolset you want to delete must be empty, or it cannot deleted.

To delete a toolset

1. Select the toolset you want to delete.
2. On the **Edit** menu, click **Delete**.
3. Click **Yes** to delete the toolset.

Managing Galaxy Style Libraries

A Galaxy Style Library includes a set of Element Styles. A Galaxy Style Library can be exported from the IDE and imported into other Galaxies.

Note: Only one Galaxy Style library can be loaded and used in a Galaxy at a time.

Two types of Element Styles are included in a Galaxy Style Library:

- Pre-Defined Element Styles. These Element Styles have been configured with default values, but you can modify them to your requirements.
- User-Defined Element Styles. These Element Styles do not contain preset values and can be configured to meet the needs of your applications.

You can create your own Galaxy Style Library by overriding the configuration of Element Styles and exporting the library. For information about setting Element Style overrides, see Chapter 7, "Working with Element Styles," in the *Creating and Managing ArchestrA Graphics User's Guide.*

If you want to use the original default visual properties for your graphic elements, you can reset Pre-Defined and User-Defined Element Styles to their defaults.

Importing a Galaxy Style Library

To import a Galaxy Style or Element Style Library

1. On the **Galaxy** menu, click **Import** and click **Galaxy Style Library**.
2. Browse for the library file. Galaxy Style and Element Style Library files are XML files.
3. Select the file and click **Open**. The selected Galaxy Style or Element Style Library is loaded in the IDE.

If you imported a Galaxy Style Library, it is now the active Galaxy Style Library for the entire Galaxy.

Note: If a Galaxy Style Library is imported to the IDE while WindowViewer running, application graphics are refreshed with new Element Styles and appear automatically in WindowViewer without requiring a restart.

Exporting a Galaxy Style Library

You can create your own Galaxy Style Library by overriding the configuration of Element Styles and exporting the library. For information about setting Element Style overrides, see Chapter 7, "Working with Element Styles," in the *Creating and Managing ArchestrA Graphics User's Guide.*

To export a Galaxy Style library

1. On the **Galaxy** menu, click **Export** and click **Galaxy Style Library**. The **Export Galaxy Style Library** dialog box appears.
2. Browse to a path and type a name for the exported library file.
3. Click **Save**. The file is saved with the specified name and a **.xml** file extension.

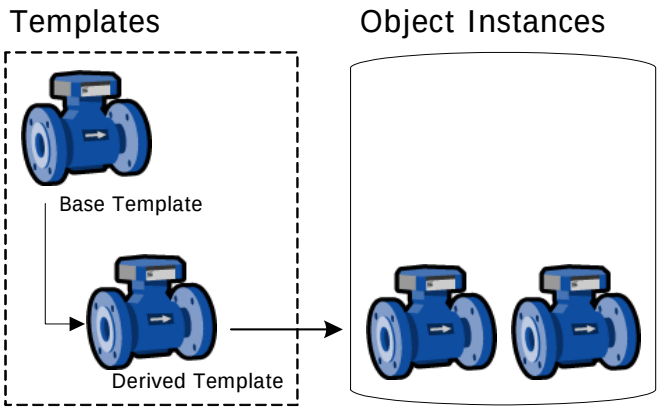
Resetting a Pre-Defined or User-Defined Galaxy Style Library

If you changed the values of Pre-Defined or User-Defined Galaxy Element Styles, you can restore the default Element Styles by clicking the **Reset to Default** button.

Creating Derived Templates

All templates you create are derived templates. A derived template inherits attributes and behaviors from the parent template. You cannot change the attributes in a base template.

After you create a derived template, you can customize and modify its attributes. If you change locked attributes in the parent template, the changes propagate to the derived template.



After you create derived templates, you can customize them, and you can create instances of the templates. You can change and modify unlocked attributes in the instances, making adjustments to meet the needs of the specific object you are modeling.

For example, your plant processes can use several models of a pump made by a single vendor. Each model has unique characteristics that map to different attribute values of the DiscreteDevice base template.

To derive a template from another template

1. Select the base template to use as the parent template in the **Template Toolbox** or **Derivation** views pane.
2. On the **Galaxy** menu, click **New** and click **Derived Template**. A derived template is created in the same toolset as its parent and placed in name edit mode. The default name is the same as the parent template followed by a numeric sequence.
3. Rename the derived template, if needed. Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. Template names cannot contain spaces.

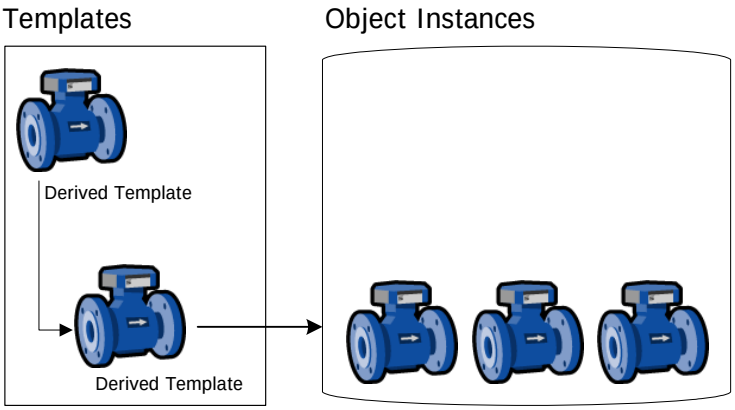
Note: The following reserved names cannot be used as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

4. You are ready to customize your new template. For more information, see “Editing Objects” on page 69.

Deriving Templates from Another Derived Template

You can create derived templates from other derived templates. The child template inherits attributes from all parent templates. Any changed attributes in the immediate parent overrides attributes changes in grandparent levels.

If you change locked attributes in the parent template, the locked attributes propagate to the derived template.



A derived template is an exact copy of its parent with the exceptions of locking, security, and the unlocked attributes that have been edited. If you create a new derived template from an existing container template, the new derived template has the same contained templates.

A good practice is to create a hierarchy of derived templates until you reach logical endpoints. Then create instances from each unique derived template.

To create a derived template from a derived template

1. In the **Template Toolbox** or **Derivation view**, select the derived template you want to use as the parent template.
2. On the **Galaxy** menu, click **New** and click **Derived Template**. A derived template is created in the same toolset as its parent. You can edit the name of the new derived template. The default name is the same as the parent template followed by a numeric sequence.

   Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. Template names cannot contain spaces.

Note: The following reserved names cannot be used as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

3. You can create another derived template by repeating the previous steps, or you can customize your new derived template. For more information about customizing your template, see “Editing Objects” on page 69.

Creating Contained Templates

Containment is the relationship in which one object includes another. Containment relationships organize objects in a hierarchy. You can build objects that represent complex devices consisting of smaller, simpler devices.

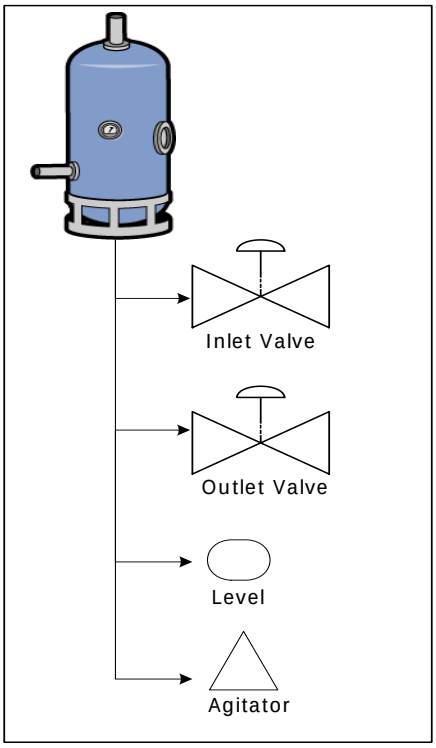
In scripts, these objects can be referred to by the name that derives from the containment relationship. This name is called a hierarchical name.

An object can have three kinds of names if it is contained by another object. The three names include:

Name	Description
Tag Name	The unique name of the individual object. For example, `Valve1`.
Contained Name	The name of the object within the context of its container object. For example, the object whose Tag name is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name “Outlet”.
Hierarchical Name	Hierarchical names that are fully-qualified names of a contained object include the name of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: “Reactor1” contains Tank1 (also known within Reactor1 by its contained name “SurgeTank”). “Tank1” contains Valve1 (also known within Tank1 by its contained name “Outlet”). Valve1 could be referred to as: “Valve1” “Tank1.Outlet” “Reactor1.SurgeTank.Outlet”.

Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

Higher level objects contain lower level objects. This allows you to more closely model complex plant equipment, like tank systems. You can nest templates to 10 levels.



Note: Objects can only contain objects like themselves. For example, ApplicationObjects can only be contained by other ApplicationObjects. Areas can only contain other Areas.

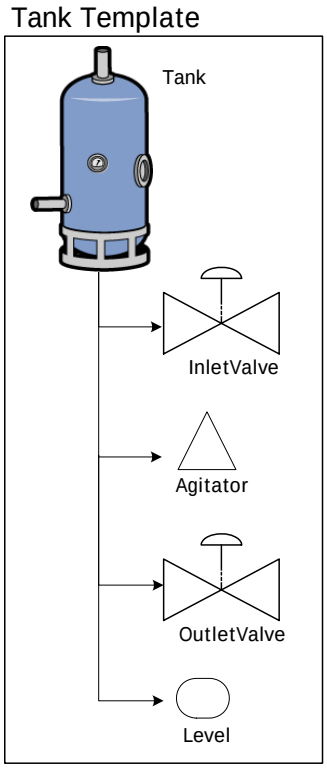
ApplicationObject Containment

ApplicationObjects can be contained by other ApplicationObjects. This provides context for the contained object and a naming hierarchy that provides a powerful tool for referencing objects.

Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

An example of a containment hierarchy is a tank that contains the following objects:

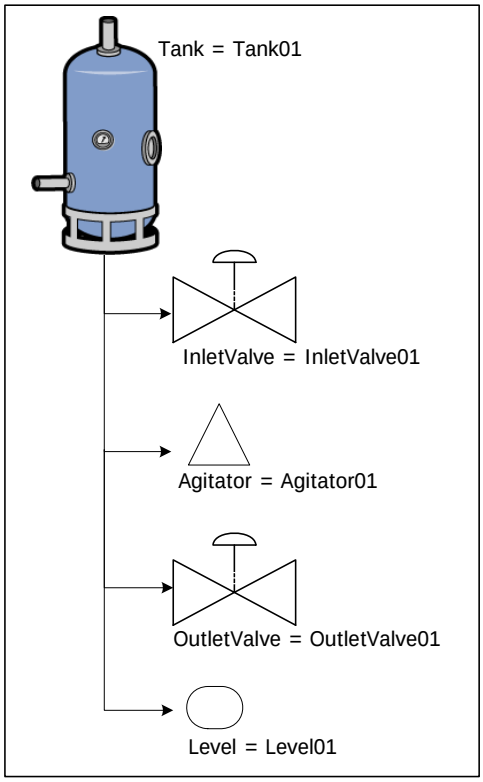
- Inlet Valve
- Agitator
- Outlet Valve
- Level



To enable referencing and flexibility within scripting, these objects can be referenced in several different ways. Each object has a unique tag name, such as:

- Inlet Valve = InletValve01
- Agitator = Agitator01

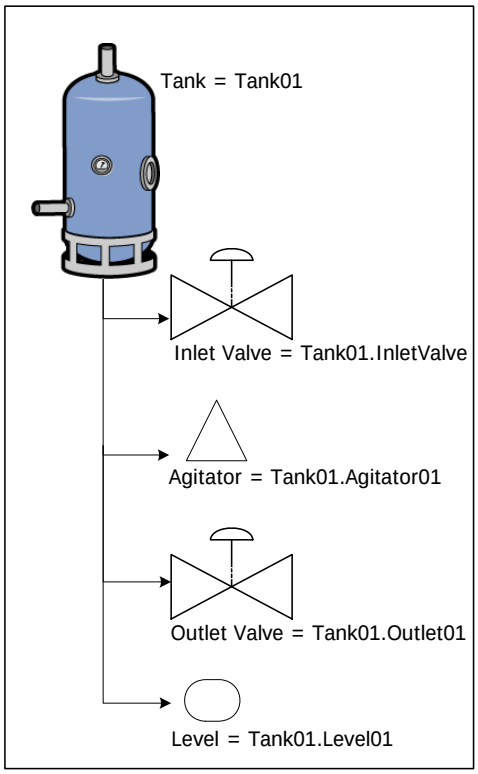
- Outlet Valve = OutletValve01
- Level = Level01



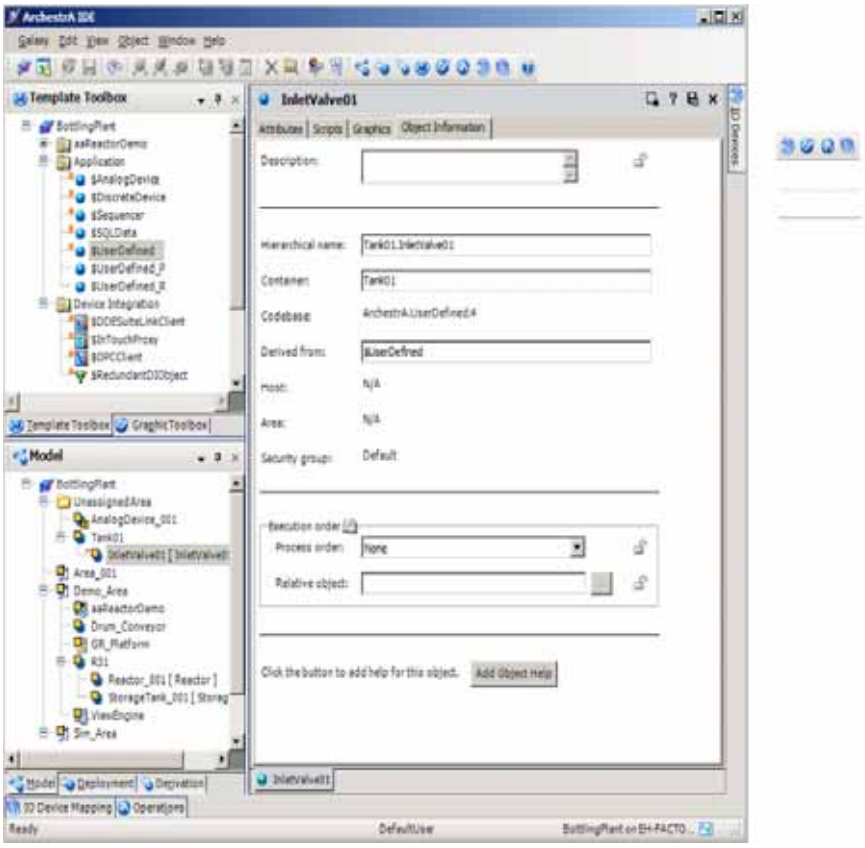
Within the context of each hierarchy, the contained names are unique, in that the names only refer to this tank system and the contained objects.

So if the tank is named `Tank01`, the contained names are:

- `Tank01.Inlet`
- `Tank01.Agitator`
- `Tank01.Outlet`
- `Tank01.Level`



This naming convention adds context to the instances contained by `Tank01`.



Additionally, you can use containment references in scripts such as:

- `Me.Outlet`: Allows a script running within the parent object to generically reference its child outlet instance.
- `MyContainer.Inlet`: Allows a script running in any of the children instances to reference another child instance named Inlet that belongs to the same parent.

Using Contained Names

The contained name of a contained object only has to be unique in the context of its container.

An object can have three kinds of names, depending if it is contained by another object. The three names include:

Name	Description
Tag name	The unique name of the individual object. For example, `Valve1`.
Contained name	The name of the object within the context of its container object. For example, the object whose tag name is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name “Outlet”.
Hierarchical name	Hierarchical names that are fully-qualified names of a contained object include the name of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: “Reactor1” contains Tank1 (also known within Reactor1 by its contained name “SurgeTank”). “Tank1” contains Valve1 (also known within Tank1 by its contained name “Outlet”). Valve1 could be referred to as: “Valve1” “Tank1.Outlet” “Reactor1.SurgeTank.Outlet”.

For example, an instance of a $Tank is named Tank01. An instance of $Valve called Valve01 is contained within the instance of $Tank.

Change the contained name of Valve01 to InletValve. Now Valve01 can also be referred to by its hierarchical name Reactor1.InletValve. The name of the contained object can be changed, though, within the scope of the hierarchy.

Contained names can be up to 32 alphanumeric or special characters. The second character cannot be $ and the name must include at least one letter. You cannot use spaces.

Containment Examples

You can have a Tank object that contains two DiscreteDevice objects that represent its inlet and outlet valves.

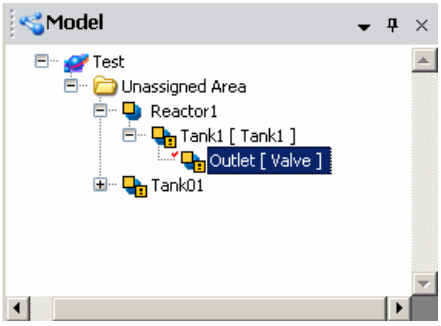
Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

To implement containment

1. Create the following instances: `Tank1` from $UserDefined and `Valve` from $DiscreteDevice. `Valve` has only one name, `Valve`.
2. In the **Model** or **Deployment view**, drag `Valve` on to `Tank`.

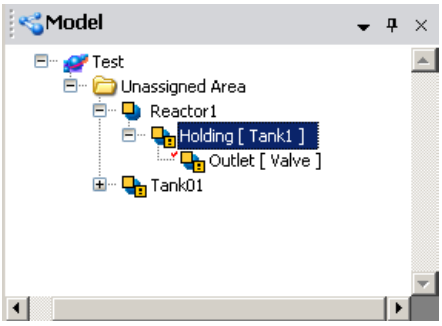
Note: If `Tank1` already contains an object with a contained name of `Valve`, the Galaxy generates a unique contained name for the newly contained object, such as `Valve_001`.

3. Change the contained name of `Valve` within `Tank1` to `Outlet`. `Valve` can now be referred to by its tagname, `Valve`, as well as its hierarchical name, `Tank1.Outlet`.
4. Create an instance called `Reactor1` from $UserDefined.
5. In the **Model** or **Deployment view**, drag `Tank1` onto `Reactor1`.



6. Change the contained name of `Tank1` to `Holding`. `Tank1` now has two names, `Tank1` and `Reactor1.Tank`. Also, `Valve1` has a three-part hierarchical name: `Reactor1.Tank.Outlet`.

For the three objects in this example (Reactor1 containing Tank1 containing Valve1), the following naming hierarchy exists:

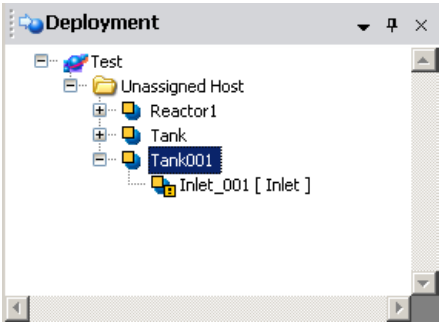


To implement template-level containment

Note: Contained Templates do not have tagnames. When an instance hierarchy is created from a template and its contained children, unique tagnames will be created for the instances based on their contained names.

1. Create the following derived templates: $Tank from $UserDefined and $Valve from $DiscreteDevice.
2. Derive $Inlet from $Valve.
3. In the **Template Toolbox**, drag $Inlet on to $Tank. If $Tank already contains a template named Inlet, the Galaxy generates a unique tagname for the new contained template, such as Inlet_001.

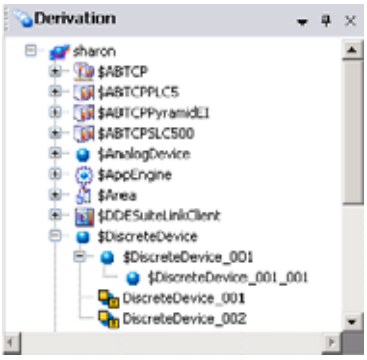
   The contained template now has a hierarchical name $Tank.Inlet.
4. Create an instance (Tank001) of $Tank.
5. The **Model** and **Deployment views** show an instance Tank001 that contains an instance called Inlet.



Viewing Containment Relationships

Containment relationships appear for templates in the **Template Toolbox**. For instances, the relationship appears in both the **Model** and **Deployment views**.

In the **Derivation** view, if a template contains other templates, you can expand it to show the containment under that template.



The **Derivation view** shows templates and instances with regard to containment in the following ways:

- Non-contained instances show their tagnames.
- Contained instances show their tagnames and hierarchical names.
- Non-contained templates show their template name.
- Contained templates show their hierarchical name.

Renaming Contained Objects

Before you rename a contained name of an object, make sure that the object is not checked out to another user or currently deployed.

The new contained name must comply with naming restrictions. Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. You cannot use spaces.

Contained names also cannot be the same contained name as an existing contained object within the same level of hierarchy in the containment relationship.

WARNING! Be careful renaming contained objects. References from other objects to the object being renamed are not automatically updated with the new name. You must update the references. Objects with broken references receive bad quality data at run-time.

To rename an object's contained name

1. Select the object in an Application view.
2. On the **Edit** menu, click **Rename Contained Name**.
3. Type a new contained name.

   All IDEs connected to the Galaxy show the object's new contained name.

Editing Objects

With the Object Editor, you define attributes specific to an object. When you open the Object Editor in non-ReadOnly mode, the object is checked out. No one else can edit an object while you are working with it. If someone else is already working on the object, you can open it to view but you cannot make changes.

The Object Editor contains pages that can be used to modify objects. To view a page in the editor, click its tab. Four pages – **Attributes**, **Scripts**, **Graphics**, and **Object Information** – are common to all objects, while other pages are unique to certain object types. If you import a Galaxy or objects with field attributes, the **Field Attributes** page will be present for those objects; the Field Attributes page can also be enabled through a Galaxy user configuration option. See "Configuring User Information" on page 39 for more information.

When editing an object, you may see attribute text boxes showing a --- (dash dash dash). The --- is a placeholder reference that does not cause the associated object to be placed in a warning configuration status when it is validated. You may also see attribute text boxes showing a ---.--- (dash dash dash dot dash dash dash). You need to provide a valid reference in the text box. The ---.--- placeholder causes the associated object to be placed in a warning configuration status when the associated object is validated.

Important: Edit the I/O auto assignment placeholder in the Object Editor ONLY if you do NOT want to use I/O auto assignment for the object. In most cases, I/O auto assignment is the preferred method.

When you are finished editing an object and save it, the configuration data for the object is validated. If errors or warnings are identified during validation, a message appears. You can cancel the save or save the object configuration as it is.

- If you cancel, the Object Editor remains open so you can correct the problems.
- If you save the configuration as it is, the object is placed into a bad or warning state. The object's status is marked in the Galaxy database as Good, Warning or Error. Error means the object is undeployable.

To edit an object in the Object Editor

1. Select the object.
2. On the **Galaxy** menu, click **Open**. A red check mark appears next to the object's icon indicating it is checked out and the Object Editor opens.

Note: If you are adding I/O attributes to an application object or system object, such as an area object, the preferred method of adding and editing I/O references is through the **IO Devices** view and **IO Device Mapping** view. Editing I/O references set for automatic assignment in the Object Editor disables automatic assignment. For more information about I/O auto assignment, see "Using I/O Auto Assignment" on page 156.

3. Make your changes. For more information about locking attributes, see "About the Attributes Page" on page 77. For more information about setting security, see "Setting Object Security" on page 75.
4. When you finish configuring the object, click **Save** or **Close** on the **Galaxy** menu.
   - **Save** keeps the editor open and saves all configuration changes to the Galaxy database.
   - **Close** closes the editor.
   - To keep the object checked out, select the **Keep Checked Out** check box before closing.

Getting Help

Tooltips are available in the Object Editor. Point to any editor option and a tooltip appears, showing the attribute name. This name is used when referring to the attribute in scripts, for example.

Each object also includes documentation about usage, configuration, run-time behavior, and attributes. For help with configuring the object, click the question mark on the toolbar to open the help for that object.

Help File Structure

The header part of the Help file contains the following information:

- Tag name — The object's name.
- Contained Name — The object's contained name. For more information, see "Creating Contained Templates" on page 59.
- Description — A short summary of what the object is for.
- Code Base — The code version of the object.

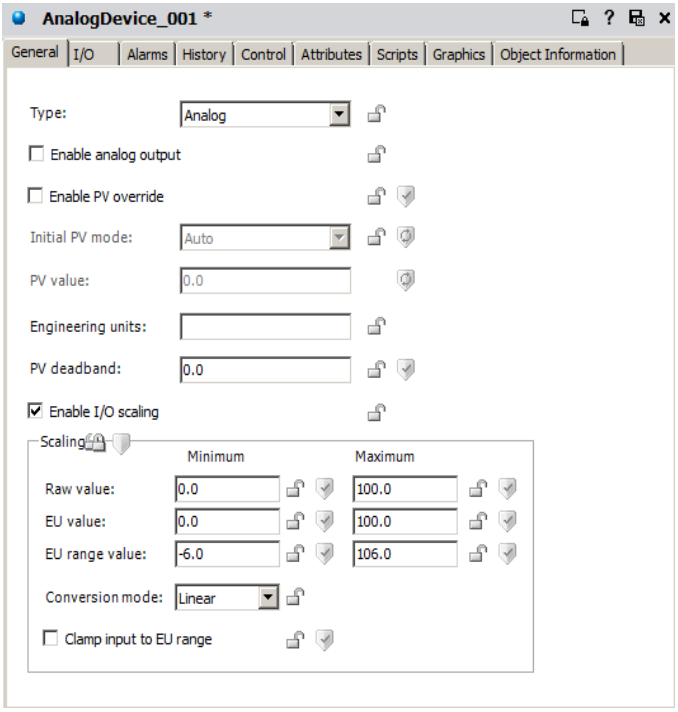
- Derived From — The immediate parent template for the object.
- Object Version — The configuration version of the object.
- Process Order — The run-time execution order within the host engine's scan (none, before, after) relative to the **Relative Object** element.
- Relative Object — The object that runs before or after in the **Process Order**.

The rest of the help file shows general information about the object.

About the General Editor Layout

When you open the attributes for an instance or a template, you see the Object Editor. The Object Editor is where you configure the object's attributes and add scripts or associated graphics to the object. The Object Editor has several pages related to the type of object you select. If you are working with an instance, you see different pages than if you are working with a template.

For example, the following illustration shows an analog device template.



When you open the Object Editor, the object is automatically checked out so no other user can work on it. When you close the Object Editor, the object is checked in to the Galaxy database, if it was automatically checked out when the editor was opened.

- To keep the object checked out, click **Keep Checked Out** before closing.

- To save configuration changes you made, close the editor, and check the object back in About the Object Information Page, click the **Close** icon.

After the object is checked in, other users can edit it.

Locking and Unlocking Template Attributes

When you create derived templates, you can lock or unlock some or all of the attributes and enabled features. Locking an attribute prevents it from being changed in derived templates or instances. You can only lock attributes in templates.

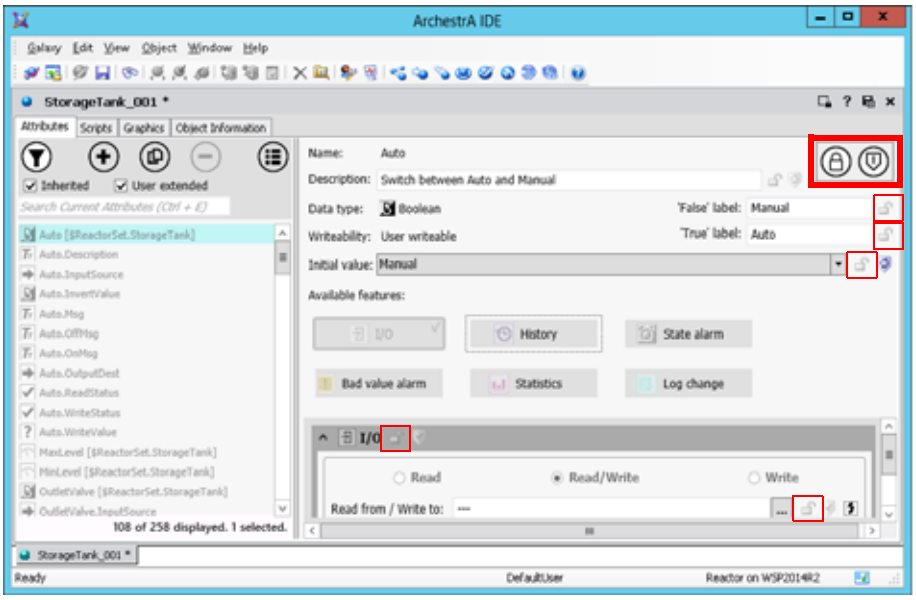
Locking an attribute in a template specifies that its value or setting is inherited by all derived objects, both templates and instances. Locking an attribute also makes the attribute act as a constant during run time.

In the **Attributes** page, you must enable locking before an attribute or any of its features can be locked in a template. To enable locking, click the **Show / Hide Lock** icon to the right of the attribute name. When locking is enabled, lock symbols will appear next to values that can be locked. Lock symbols are not visible in the template or its derived instances unless locking is enabled.

- Attributes that are locked in a parent template are referred to as “locked in parent.” This parent can be at any parent level above the selected object.
- Attributes that are locked in a template are referred to as “locked in me.”

Caution: When using I/O auto assignment, do not lock the “Read from” field (input source) or “Write to” field (output destination). If you lock these fields in the parent template, they cannot be updated with the resolved reference when the object is deployed and the runtime value will be “---Auto---”.

See “Group Locking/Security” on page 76 for information about locking or unlocking all of a feature’s attributes with a single click.



Lock controls and status are shown with an icon. If the option is enabled, click the lock control to switch it between locked and unlocked. These icons mean:

Icon	Name	Description
	Locked (in me)	The associated attribute is locked (in me) and enabled. Only templates can have this kind of lock. The attribute value is read/write. Derived templates and instances do not have a unique copy of this attribute. Child objects share the locked attribute of the parent. Changing the value of a locked attribute in the parent template updates the value of that attribute in all derived templates and instances.
	Locked (in parent)	The associated attribute is locked in the parent object and cannot be unlocked or modified by the child object. Both templates and instances can have these. The attribute is read-only. The templates and objects do not have a unique copy of this attribute, but instead use the attribute value in the parent where the attribute is locked.
	Unlocked	The associated attribute is unlocked and enabled. Both templates and instances can have this kind of lock. The attribute is read/write. The object has its own copy of the attribute value and the value is not shared by derived objects.

Icon	Name	Description
	Indeterminate	Refers to a specified group of options. An indeterminate state indicates different lock states for individual options in the group.
	Undefined	The associated attribute doesn't exist. This indicates that another attribute be enabled before the associated attribute is created and before its lock status can be determined.

Note: Locking an attribute during configuration makes its value a constant. You cannot write to locked attributes during run time.

To lock an attribute example

1. Create a derived template from the $DiscreteDevice base template. Name the derived template `$Valve`.
2.  Open the `$Valve` template and on the **States** page, lock one of the attributes by clicking the **Lock** icon.
3. Save `$Valve`.
4. Create a derived template from `$Valve`. Name it `$BigValve`.
5.  Create an instance from `$Valve` named `Valve1`.

   In the editor of `$Valve`, the attribute lock icon shows the attribute is locked in `me`.

    You cannot change the attribute value in `$BigValve` and `Valve1`. The editor options for the attribute are disabled and the lock icon, if shown, indicates a lock in the parent. Also, the attribute lock  icon in children derived from `$Valve` is now locked and disabled.

   If you change the attribute value in `$Valve`, the change propagates to `$BigValve` and `Valve1` after you save the changes.

To unlock an attribute example

1. Using the objects from the previous example, in the `$Valve` template's editor, unlock the locked attribute.
2.  Save `$Valve`.

   In the editor for `$Valve`, the attribute lock icon shows it is unlocked.

    The lock type for this attribute of `$BigValve` now indicates locked  in `me`. The lock type for this attribute of the `Valve1` instance shows unlocked but the locking icon is unavailable.

Setting Object Security

Operators interact with objects through the individual attributes of those objects. Each attribute on the Object Editor that can be modified by operator's at run time and can have an associated security control, which is used to modify its run-time security classification.

In the **Attributes** page, security icons must be enabled before you can set security for an attribute or any of its features in a template. To enable security, click the **Show / Hide Security** icon to the right of the description field. When security is enabled, security symbols will appear next to values for which security is configurable. Security symbols are not visible in the template or its derived instances unless enabled in the parent template.

If an attribute's security classification is configurable, click the security control to select one of seven possible states:

Security Icon	Description
Free Access	Lets you change this value without restriction even if you have no defined permissions on the object. Anyone can write to these attributes to perform safety or time critical tasks that can be hampered by an untimely logon request, such as halting a failing process.
Operate	Lets you work with Operate permissions to do certain normal day-to-day tasks. These include writing to attributes like Setpoint or Command for a Discrete Device object. This level of security requires you to have Operate permission for the security group for the object.
Secured Write	Requires you to authenticate using your user name and password each time you want to write to the attribute. You also need to have Operate permissions for the object.
Verified Write	Requires you to have Operate permissions to log on again and a second, different user to also log on as the verifier before writing to the attribute. The verifier needs to have Can Verify Writes permission for the object.
Tune	Allows end users with Tune Operational permissions to tune the attribute in the run-time environment. Examples of tuning are attributes that adjust alarm setpoints and PID sensitivity.

Security Icon	Description
Configure	Allows end users with Configure Operational permissions to configure the attribute's value. Requires that the user first put the object off scan. Writing to these attributes is considered a significant configuration change. For example, a PLC register that defines a Discrete Device input.
Read Only	Only allows users to read this attribute's value in the run-time environment. This attribute is never written to at run time, regardless of the user's permissions.

If an attribute's security is shown in gray, its security classification is locked in its parent object and cannot be changed or it requires the enabling of a group attribute.

Group Locking/Security

The lock and security controls associated with option groups and features quickly set those conditions for all options in the group.

The group control typically reflects the setting for all options in the group or feature set. But, if at least one option in the group has a lock or security control that is different from the other options, the group control shows an indeterminate icon.

In addition to the undefined controls, the group controls for locking and security are the same as those for individual options.

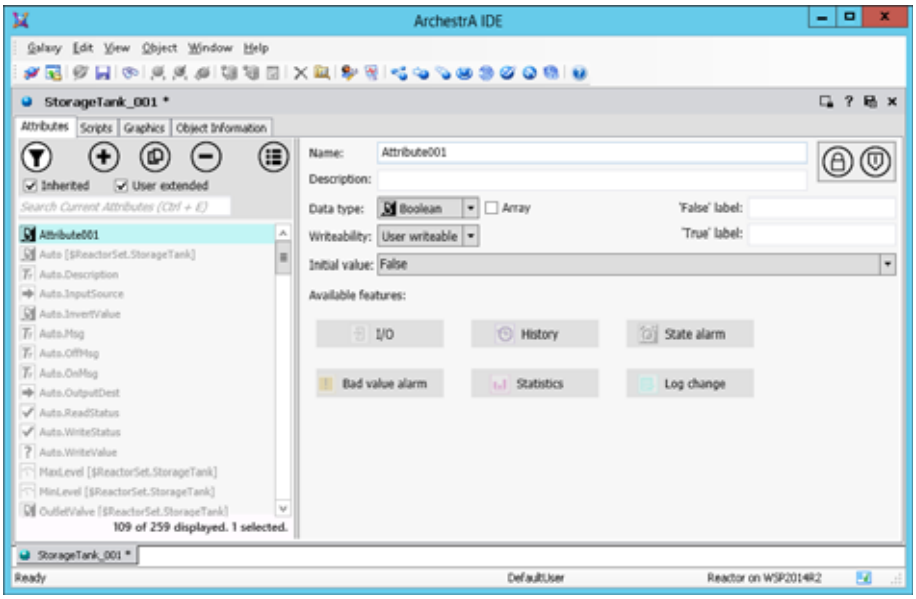
You can lock or unlock all of the attributes associated with a feature by selecting the lock icon next to the feature name, after you activate the feature. This will lock or unlock all of the attributes for the feature, unless an attribute was locked at a higher level. For example, if you locked an attribute in a template, and then created another derived template, the attribute that was locked in the original template cannot be unlocked (locked in parent).

If an attribute is locked in the template, you can change the value in that template, but not in the derived children. If you change the value in the parent template, the change propagates to all child objects. For more information, see Chapter 5, "Enhancing Objects."

About the Attributes Page

If the object you are editing does not have any attributes defined, the **Attributes** page will be empty, other than buttons for adding, filtering, duplicating, deleting, and viewing attributes and attribute features.

You can activate various features, such as I/O, History, State Alarm and Statistics. See “Enhancing Objects” on page 125 for more information about attributes and features. When you add an attribute to an object, information about the attribute is shown.



- To add an attribute, click the **Add** button.
- To filter attributes, click the **Filter** button. You can then select filtering criteria by checking source type, enabled feature type, writeability type, lock status, data type, visibility type (hidden or not hidden), and diagnostic type (configuration, runtime, or both).
- To duplicate an attribute, select it and click the **Duplicate** button.
- To remove an attribute, select it and click the **Delete** button.
- To reveal details about an attribute, click the **Show Details** button. The name, description, and icons representing activated features for the object’s attributes will be shown.

When you add an attribute to an object, the Attributes page divides into three sections. The left side of the page lists attributes, the top right shows information about the currently selected attribute, and the bottom of the right side contains fields for configuring features.

You can search attributes by entering the characters you are trying to find in the *Search Current Attributes* text box. This will display attributes that contain the characters you enter.

Note: The search function finds attributes that contain the search term *anywhere* within the attribute name. For example, if you want to locate all attributes associated with Pump125, you could enter “125” in the text box. All attributes with 125 anywhere in their names will be shown.

You can also search for attributes by typing the character you are searching for when the cursor is pointed at the attribute list. The first attribute that contains the character, anywhere within its name, will be highlighted.

Note: The search begins at the attribute that is currently selected. Selection will move to the next attribute that contains the character entered.

- On the left side of the page, all attribute names for the object are listed. Pressing the **Show Details** button reveals additional information about the listed attributes. Check boxes are provided for **Inherited** and **User extended**.
    - **Inherited**: Checking this displays attributes that were added in the parent template used to create the object.
    - **User extended**: Checking this displays details about any features you added to the attribute, for example, for I/O.
- At the top of the right side, information for the current attribute is listed. Additional fields may be listed, depending on the data type of the attribute. However, all data types include the following properties:

Property	Description
Name	Attribute name (for example, Attribute001).
Description	User defined; a description of the attribute.

Property	Description
Data type	Can be set to Boolean, Integer, Float, Double, String, Time, ElapsedTime, or InternationalizedString. The data type determines which features can be added to an attribute. For example, when configuring a Boolean data type, only two alarm features are available: **State Alarm** and **Bad Value Alarm**. When configuring an Integer data type, there are four available alarm features: **Limit Alarms**, **ROC Alarms**, **Deviation Alarms** and **Bad Value Alarms**. To create an array, check the **Array** option and specify the array's length in the **# of elements** box. You can create an array for each data type except **InternationalizedString**.
Writeability	Can be set to Calculated, Calculated retentive, Object writeable, or User writeable. See the *Object Viewer User's Guide* for additional information about writeability categories. • **Calculated:** Permits only scripts within the same object to write to the attribute. Calculated attributes are not saved across restarts. If you select Calculated for an attribute, only scripts running on the same object can write to the attribute. • **Calculated Retentive**: Permits only scripts within the same object to write to the attribute. Calculated Retentive attributes are saved across engine restarts. • **Object Writeable**: Permits other objects to write to this attribute in addition to being set by scripts within this object. Object Writeable attributes are saved across restarts. This category is not user writeable. • **User Writeable**: Permits other users to write to this attribute in addition to being set by scripts and objects throughout the system. User Writeable attributes are saved across restarts. They can be locked at configuration time. If locked, they are read only.

Property	Description
Initial value	Specifies the initial value for the attribute when the object is deployed. Enter value data for each data type. In the case of a non-arrayed **Boolean**, select **True** or **False** in the list box. For an arrayed Boolean, select the appropriate checkbox.

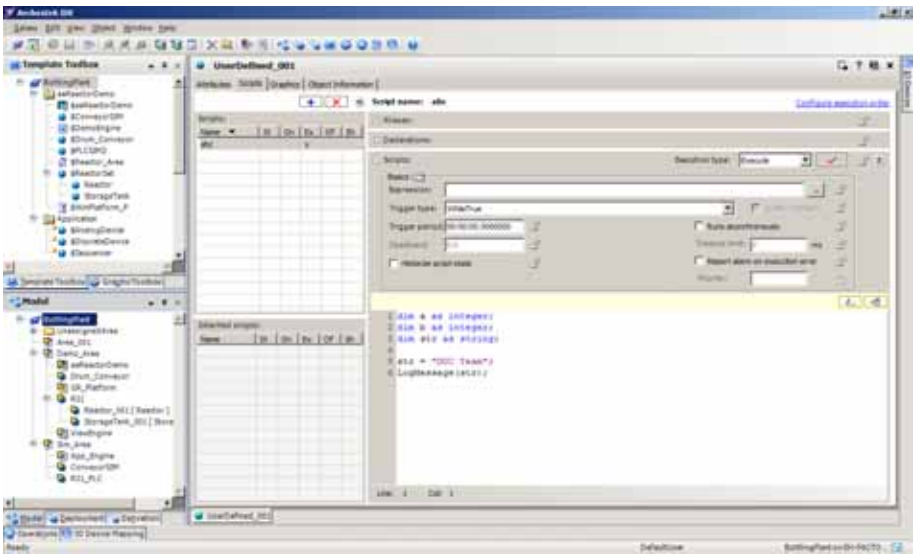
- At the bottom of the right side, fields for configuring activated features are displayed. Available features include:

Feature	Description
	The I/O feature is used to activate the attribute for I/O. You can select **Read** (input), **Read / Write** (input and output), or **Write** (output). When you select the I/O feature for application objects and system objects, such as areas, the "Read from / Write to attribute text box contains the placeholder reference `<IODevice>.[HierarchicalName].[AttributeName].InputSource` or `<IODevice>.[HierarchicalName].[AttributeName].OutputDest`. You should not edit these placeholders in the Object Editor, but instead use the **IO Devices view** to assign the object to a scan group. This will allow the placeholder to automatically resolve to the correct I/O reference. See "Using I/O Auto Assignment" on page 156 for additional information.
	The History feature enables historization to the Wonderware Historian. **Note:** The AppEngine hosting the object must be configured for historization.
	The State Alarm feature is available for Boolean data types. You can set alarm priorities, the alarm state, an alarm message, and time deadband.
	The Limit Alarms feature is available for integer, float, and double date types. With it, you can set value alarms (Hi, HiHi, Lo, LoLo), and the limits, priority, and messages to apply to each alarm limit.

Feature	Description
	The Rate of Change (ROC) Alarm feature is available for integer, float, and double date types. With it, you can set alarms if changes within a time period exceed specified values, and the limits, priority, messages, and time period to apply to each ROC alarm.
	The Deviation Alarm feature is available for integer, float, and double date types. With it, you can set deviation alarms (major and minor), and the tolerance, priority, messages, target value, deadband, and settling period.
	The Bad Value Alarm feature adds an alarm if the value returned from the attribute is determined to be bad quality.
	The Statistics feature monitors statistics associated with the object.
	The Log Change feature will cause the attribute to generate an event each time the attribute value changes.

About the Scripts Page

The **Scripts** page has five areas. To learn more about using scripts, see “Writing and Editing Scripts” on page 173.



The main areas of the **Scripts** page include:

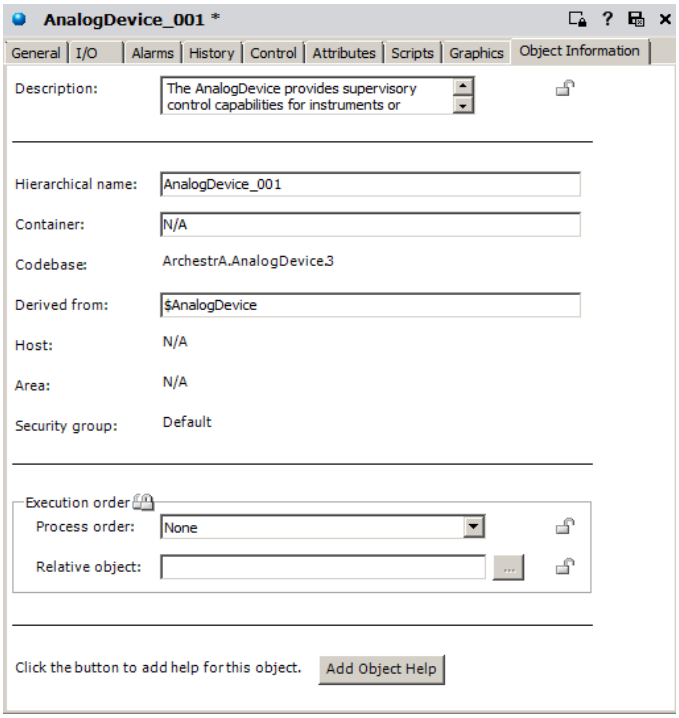
- **Scripts** list: Shows all scripts currently associated with the object. The columns indicate which kind of trigger the script uses: Startup, On Scan, Execute, Off Scan and Shutdown. Click the **Add** button to add a new script.
- **Inherited scripts name** list: Shows all scripts associated with the object's parent. The columns indicate which kind of trigger the script uses: Startup, On Scan, Execute, Off Scan and Shutdown.
- **Aliases** area: Lets you create and modify aliases that apply to the script you are working on. Aliases are logically descriptive names for typically long ArchestrA reference strings that you can use in the script to make the script more readable.
- **Declarations** area: Provides a place to add variable declaration statements, such as `DIM MyArray[1] as FLOAT;`. These declared variables live from the start to the shutdown of the object and can be used to hold values that persist from one execution of the script to the next. They apply only to the script in which they are declared.
- **Basics** area: Provides a location in which you set the expression, triggering conditions, and other settings that run the script in the run-time environment. See "Writing and Editing Scripts" on page 173 for descriptions of triggers and when they are executed. This area includes:

  Configure Execution Order: Sets the execution order of multiple scripts (inherited and local) associated with this object.

  Historize Script State: Select to send the state of the script to the Wonderware Historian.
- **Script Creation** box: Shows the script you are writing.

About the Object Information Page

The **Object Information** page is common to all object configuration editors.



This page includes the following fields:

- **Description**: A short summary of the object's purpose.
- **Hierarchical Name**: The fully qualified name of a contained object, including the container object's TagName.
- **Container**: The name of the other object that contains this object, if applicable.
- **Code Base**: The code version of the object.
- **Derived From**: The immediate parent template of the object, either a base or derived template.
- **Host**: Another object to which the object is assigned (for example, a WinPlatform hosts an AppEngine). An object's host determines where an object will run when it is deployed.
- **Area**: An object that represents a logical grouping to which this object belongs. An object's area mostly affects the way in which its alarms are reported to alarm clients.
- **Security Group**: The security group the object is associated with. For more information, see "Working with Security" on page 353.

- **Execution Order**: If you want this object to run before or after another object within its engine's scan, select from the **Process order** list. Click the **Browse** button to specify the **Relative object** in the **Attribute Browser**. For more information about the **Attribute Browser,** see “Referencing Objects Using the Galaxy Browser” on page 93.
- **Add Object Help**: Opens a copy of the HTML help page for the template this object is derived from. You can edit this information. This allows you to create Help about the object you are currently configuring for downstream users. This Help appears when you select an object in a view and then click **Object Help** on the **Help** menu.

Customizing Help

Do not use Microsoft Word as an editor to create downstream object HTML help pages. Use an HTML or plain text editor instead.

If clicking **Add Object Help** opens Word on your computer, change the program associated with editing HTM files. Open the Windows Explorer's **Folder Options** dialog box and go to the **File Types** page to make this change. For more information about associating programs with files, see your Windows help.

Finding the Help Folders

The path to each object’s Help folder is unique. It depends on the path you selected when you installed the Galaxy Repository. The path to an object’s Help is:

```
\<Installation Path>\Framework\FileRepository\
<YourGalaxyName>\Objects\<TheObjectID>\Help\1033
```

The default is:

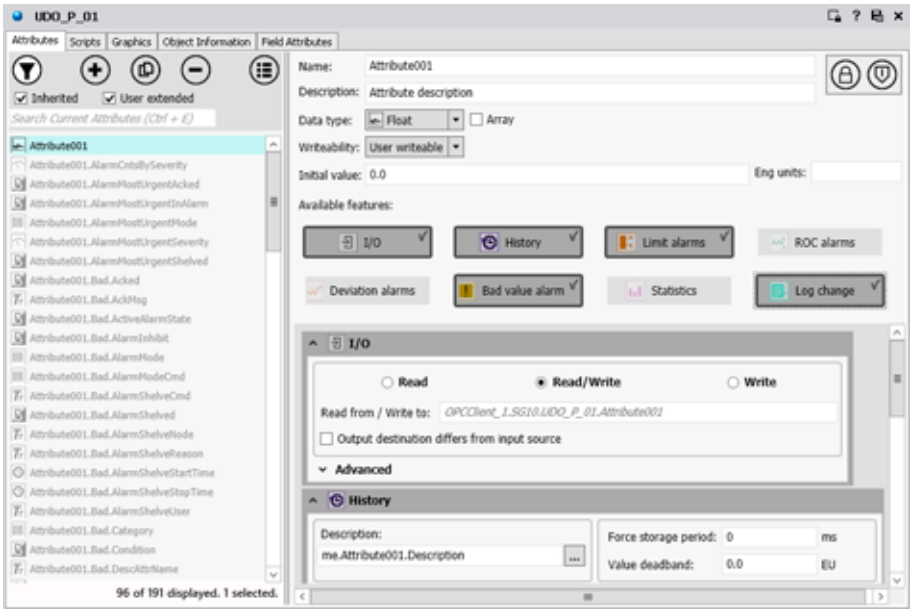
`<Installation Path>` is `\<Program Files\ArchestrA\`

To add images to the Help file, place the images in the proper folder on the Galaxy Repository computer and use a relative path to those images in the HTML file.

For the previous example, place images in the `\1033` folder or create an images folder under it.

About the Field Attributes, UDAs, and Extensions Pages

Starting with Wonderware Application Server 2014 R2, the functionality of Field Attributes, UDAs and attribute extensions has been combined into a single unit of functionality simply called an Attribute. The functions and capabilities of the **Field Attributes**, **UDAs**, and **Extensions** pages in the **Object Editor** have been combined and made available in the **Attributes** page.



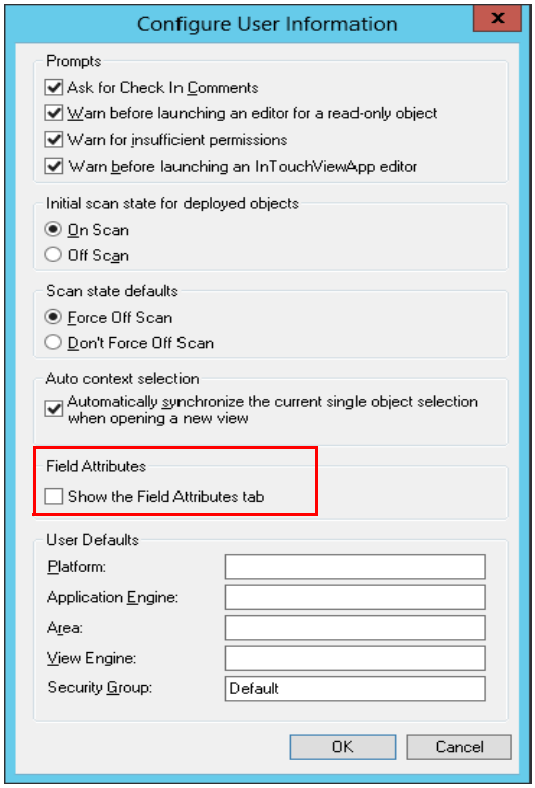
The **UDAs** and **Extensions** pages are no longer present in Application Server 2014 R2 and subsequent versions. By default, the **Field Attribute**s page does not normally appear in Application Server 2014 R2 and later versions, but will under the circumstances detailed in the next section.

Viewing the Field Attributes Page

While the **Field Attributes** page is not displayed by default for Galaxies created with Application Server 2014 R2 and later versions, it will be displayed when you open an UserDefined Object in the Object Editor and any of the following conditions apply:

- The derived template or instance created from the $UserDefined base template contains Field Attributes.
- You import UserDefined Objects created in a previous version of Application Server that contain Field Attributes.
- You have an existing Galaxy with Field Attributes already defined and migrate it to Wonderware System Platform 2014 R2 or later.

- You select the **Show Field Attributes** option in the **Configure User Information** dialog. See “Configuring User Information” on page 34 for more information.



Determining Whether to Convert Field Attributes

Converting Field Attributes to Attributes provides the following advantages:

- It is easier and faster to work with Attributes, rather than Field Attributes.
- All of your Attributes will appear on the same Object Editor page if you convert Field Attributes.
- Attributes can provide performance advantages over Field Attributes during run time (the actual performance improvement can vary between systems).

You may not want to convert Field Attributes if:

- You have a large number of objects that use Field Attributes.
- You do not want to modify and test scripts that reference Field Attributes.
- The objects are working well using Field Attributes, and there is no active development on these objects. Additional effort will be required for converting, testing, and redeploying objects.

Optimizing Performance if You Use Field Attributes

If you choose to continue using Field Attributes, use the following guidelines to maximize performance:

- No more than 64 Field Attributes (maximum of 32 analog and 32 discrete) should be used for an individual object. If this limit is exceeded, the performance of object editing and check-in operations slows down significantly.
- Do not extend an existing Field Attribute by activating features on the **Attributes** page.

  For example, to add historization to an existing Field Attribute, select the Enable history option from the Field Attributes page. DO NOT use the Attributes page to add the History feature.
- Converting from Field Attributes to Attributes will greatly enhance performance, particularly if many Attributes are used.

Converting Field Attributes to Attributes

You can choose which objects with Field Attributes to convert to Attributes, but only first-level derived objects can be selected. That is, you can only select templates and instances that are directly derived from the $UserDefined base template. Objects derived from first-level templates or individual Field Attributes cannot be individually selected for conversion.

When you start the conversion process, you are presented with a list of all first-level templates and instances that contain Field Attributes. First-level templates without Field Attributes, but that have derived objects with Field Attributes, are also listed. From this list, you will select which objects to convert.

The conversion process is hierarchical and cascades to all child templates and instances of the selected objects. All Field Attributes contained in a template's hierarchy of derived objects are converted.

Note: Objects will be unavailable for selection (grayed out) if the object or any of its derived objects are deployed, checked out, or protected. See "Protecting Objects on Export" on page 111 for more information about protecting objects.

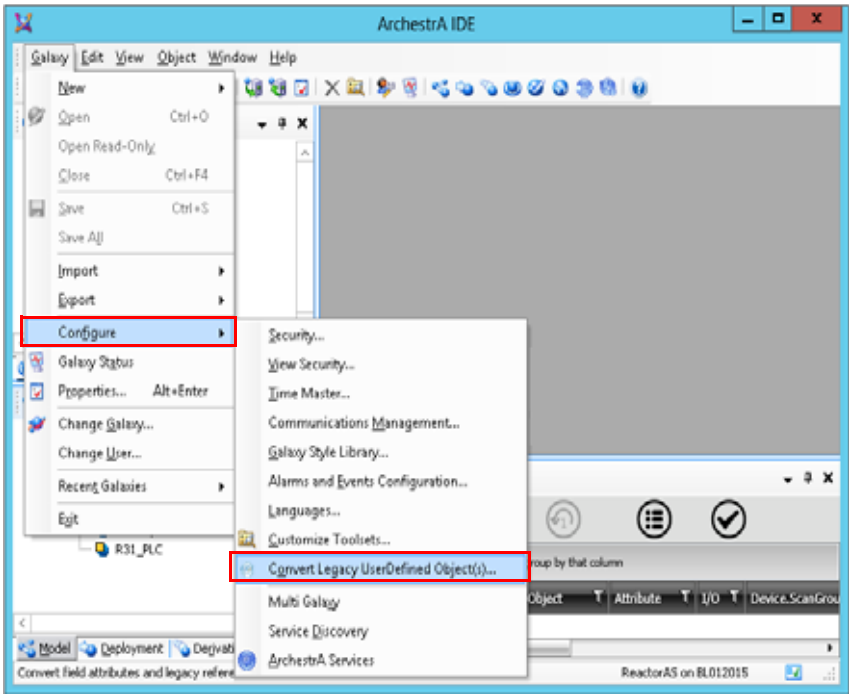
To convert field attributes to attributes

Before starting the conversion process, see "Special Considerations when Converting Field Attributes" on page 91 for a discussion of potential differences in behavior between converted Attributes and the original Field Attributes.

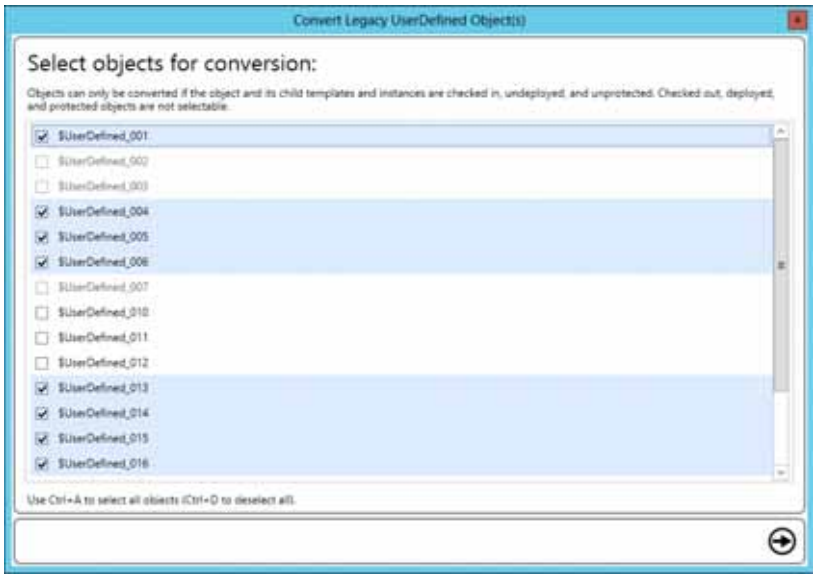
1. Back up all UserDefined Objects (UDOs) before starting Field Attribute conversion.

2 Ensure that all UDOs that you want to convert are checked in and not deployed. You will not be able to convert Field Attributes for objects that are deployed, checked out, or protected.

3 From the Galaxy menu, select **Configure**, then **Convert Legacy UserDefined Object(s)...**



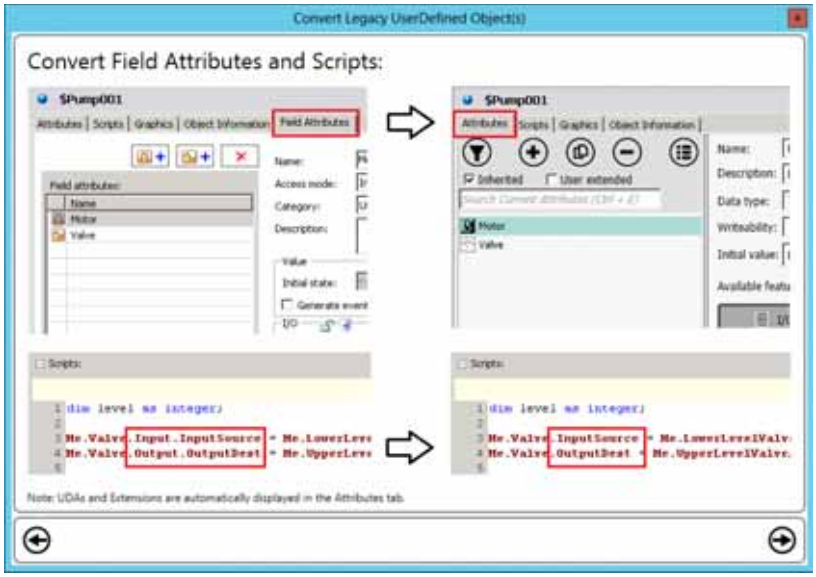
4 When you select **Convert Legacy UserDefined Object(s)...** from the menu, a list of all first-level objects with Field Attributes is displayed. First-level templates that do not themselves include Field Attributes, but have child objects with Field Attributes, are also listed.



5 Select the objects you wish to convert and click **next**.

Note: Objects will be grayed out and unavailable for selection if the object or any of its derived objects are deployed, checked out, or protected.

6 The next screen is informational, and shows that Field Attributes will be consolidated onto the Attributes page. Click **next** to proceed.



7 The screen that follows is informational as well, and lists the limitations that apply to script conversion. These are:

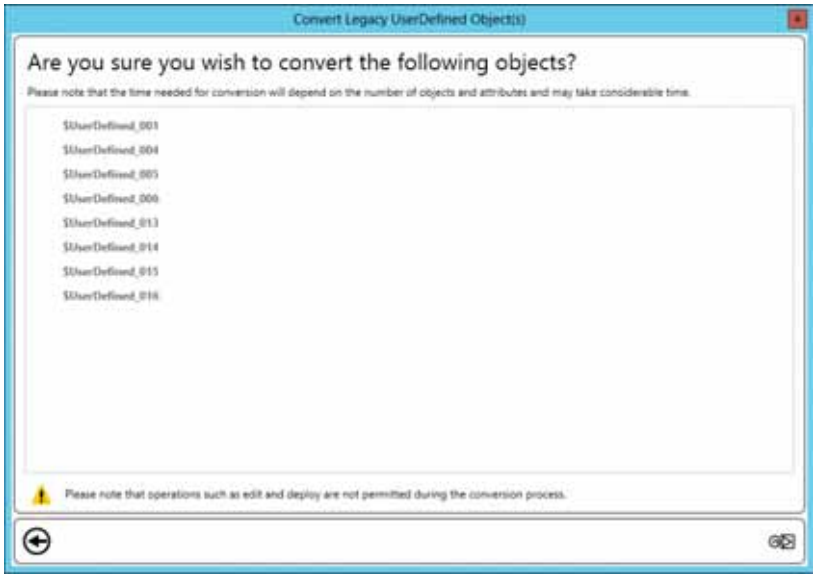
- Only field references that begin with “me.” are converted.
- Field references outside the selected objects are not converted.
- Field references in scripts that use indirect functions or string manipulations are not converted.
- Field references within ArchestrA graphics are not converted.

8 Click **next** to proceed.

9 To begin converting Field Attributes for the selected first-level objects and their child objects, click the **Convert** button.

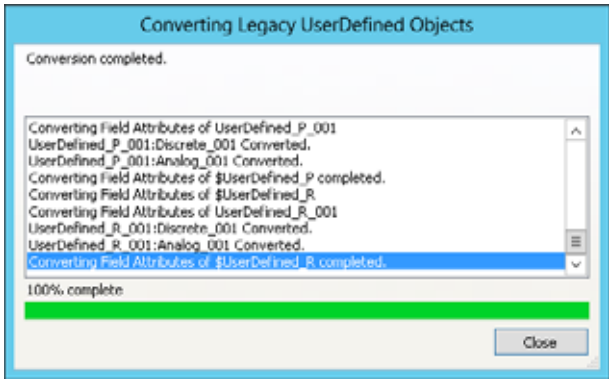
WARNING! Once you start the conversion process, it cannot be cancelled. The time needed to convert Field Attributes depends on the number of objects with Field Attributes, the number of Field Attributes, and the depth and complexity of the object hierarchy. Field Attribute conversion requires at least as much time as Galaxy migration.

- Click **back** if you need to make any changes.
- To cancel, simply close the dialog.



WARNING! While the conversion process is active, you cannot deploy or edit objects.

10 Each object is listed as its Field Attributes are converted. The status of the conversion process is also reported to the SMC Logger.



Important: In the event that Field Attribute conversion fails for an individual template, the template will remain checked out after the conversion process stops. Undo the checkout for the template, and then you can resume attribute conversion for any remaining objects. To find which template did not convert, check the SMC Logger.

11 When all objects containing Field Attributes have been converted, click **Close**.

Special Considerations when Converting Field Attributes

When you convert Field Attributes to Attributes, there may be some differences between the original Field Attribute and the converted Attribute.

- If the Field Attribute “Bad.Condition” was extended (for example, by adding historization), it will send alarm notifications after conversion, even if alarming was not enabled.
- Prior to conversion, the Field Attribute description is `Analog_001.Desc`. After conversion, this changes to `Analog_001.Description`.
- Prior to conversion the History Extension description points to `Analog_001.Desc`. After conversion, the History Feature points to `Analog_001.Hist.DescAttrName`.
- If a Field Attribute has “Generate event upon change” enabled, then the **Log change** feature (`LogDataChangeEvent`) will be added to the converted attribute.
- When the Field Attribute I/O option, “Output destination differs from input source,” is selected and left unlocked, but **Output destination** is locked, the converted Attribute will have the following locking properties:
  - “Write to” will be locked. This is the same as the original Field Attribute property.
  - “Output destination differs from source” will be locked. This differs from the original Field Attribute property.

Note: Default names for Field Attributes begin with either “Discrete” or “Analog.” The conversion process preserves this part of the Field Attribute name, even though this division does not apply to the converted Attributes.

Limitations and Exclusions when Converting Field Attributes

Objects that are protected, checked out, or deployed will not be converted. If an object derived from a first-level template you want to convert is protected, checked out, or deployed, the first-level template will not be selectable for conversion. The entire hierarchy of the first-level object must be available for conversion (conversion works on an all or none basis).

The following Field Attribute features are not converted to Attribute features. You will have to be enter the information for them manually.

Field Attribute Feature	Equivalent Attribute Feature
Enable Deadband	Limit Alarms
Description	Description
Engineering Units	Eng units

Updating Scripts to Reference Converted Attributes

The conversion process applies certain changes to scripts that contain I/O references. In the conversion process, the Field Reference I/O namespace is truncated to match the Attribute namespace. These reference changes are as follows:

- `Me.Analog_001.Input.InputSource` converts to `Me.Analog_001.InputSource`
- `Me.Analog_001.Output.OutputDest` converts to `Me.Analog_001.OutputDest`
- `Me.Analog_001.Input.Value` converts to `Me.Analog_001.Value`

Note: Only Field Attributes that begin with “me.” are converted. No other changes are made to scripts. Scripts within graphics are not converted.

While Field Attribute conversion updates I/O references within scripts, other attribute references are not updated. For example, while a Field Attribute description is truncated to “Desc,” the description is not truncated for Attributes. Therefore, you will have to do some manual updating.

Referencing Objects Using the Galaxy Browser

Use the Galaxy Browser to browse for:

- Attributes of objects. You can quickly find an object attribute or attribute property and add a reference to it when you are configuring an object.
- ArchestrA graphics.
- Graphic element properties.

The Galaxy Browser shows attributes, graphics, or attributes and elements, depending on what you are doing at the time you access the browser.

Browsing for Attributes

You use the Galaxy Browser to browse for:

- Attributes of objects, either instances or own relative references.
- Attributes of templates.

You can open the Galaxy Browser to browse for attributes from:

- Within an AutomationObject Editor. For example, from a script, from an attribute of type MxReference, or from a custom alarm message attribute field).
- Within an ArchestrA Graphic Editor. For example, to use in scripts, animation links and references, properties and custom properties.
- Within InTouch WindowMaker. For example, to use in a reference expression from an animation link or a script.
- Within an OPC UA server associated with an ArchestrA UA Client service.

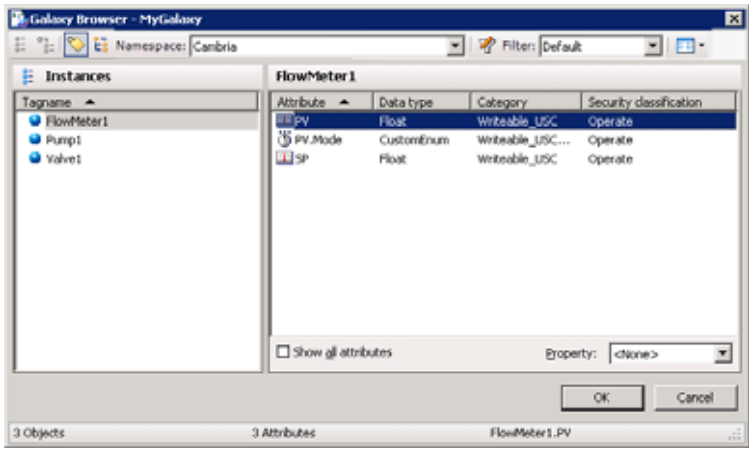
The Galaxy Browser shows objects in the left pane and the attributes associated with the current selection on the right pane. Only attributes that can be referenced at run time are shown. You can browse “Me.” references for alarm messages only.

When you open the Galaxy Browser, the last browsed location for attributes is shown. The Galaxy Browser shows the object list based on the last used state (tag name or Hierarchical name). If the last used state of the browser was Tagname and the selected editor reference is a Hierarchical name, the browser opens in Tagname mode.

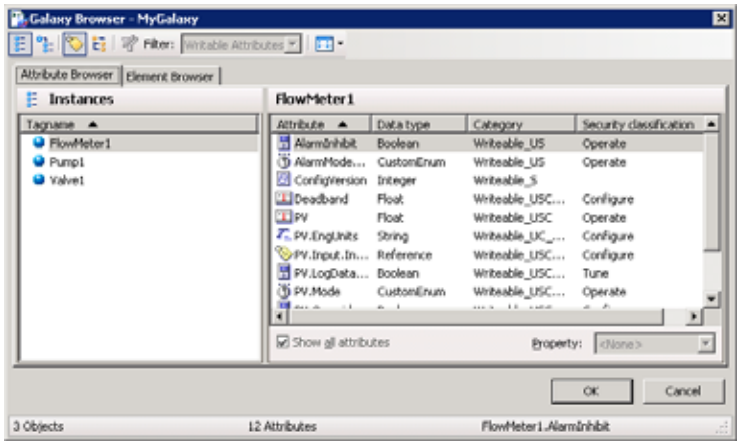
The status bar displays the attribute property name, and the it displays the graphic element attribute name and description.

To browse for attributes

1 In any area on a page, click the **Browse** button, if available. The Galaxy Browser opens.



If you are browsing for attributes to use with an ArchestrA Symbol, such as for an animation or script, the Galaxy Browser shows the attributes in an **Attribute Browser** tab.



2 By default, the browser shows only those attributes that are frequently accessed. If you are viewing the attributes of an object for the first time, the right pane can be blank. Select the **Show all attributes** check box to show all of the object's attributes.

3 To filter the list of tagnames, click the **Filter** button. For information about configuring a filter, see "Creating a Filter for the Galaxy Browser" on page 100. To switch the content of the left pane between a **Tagname** and **Hierarchical Name** list of objects, click the **Show Tag name** or **Show Hierarchical name** buttons.

4 You do not have to explicitly make a selection in the **Property** list. If you only select an attribute (leaving **Property** set to <none>), the property of the attribute defaults to **Value**.

If the option in the Object Editor is already configured with an object reference, the **Attribute Browser** shows it or expands to the nearest matching object/attribute/property currently configured in the Galaxy.

If you selected text in the script editor, that text is used as the initial reference string and the browser finds the nearest attribute reference to the selected text.

5 When you are done selecting the attribute/property, click **OK** to place the reference into the Object Editor and close the **Galaxy Browser**.

- The fully-qualified reference string appears in the editor option.
- If you are working in the script editor, the selected reference appears in the script at the current cursor position and replaces text that was selected.

Browsing with an ArchestrA OPC UA Client

You can use an instance of the ArchestrA OPC UA Client Service hosted by your Galaxy to browse a namespace in an OPC UA server.

You must prefix the item reference or attribute name with the Scope name/Service name defined in the OPC UA Client Service editor.

To access an OPC UA server item reference, use the following syntax:

```
<namespace>:<UAItemReference>
```

Namespace is the OPC UA client service scope name defined in the client service editor. For example “TestServer”.

UAItemReference is the identifier of the item (the complete path of the OPCUA Server item, followed by the NodeId) in the OPC UA Server.

For example:

```
TestServer:PLC01.PLCObject1
```

For more information about browsing with an ArchestrA UA Client and the syntax variations, see the *ArchestrA OPC UA Client Service Guide*.

Viewing Attribute Details in the Galaxy Browser

When you view attributes in the Galaxy Browser, you see two areas. The objects shown in the left area include all of the logged in user's checked-out objects plus the checked-in versions of all other objects.

Important: The Galaxy Browser shows only the Primary AppEngine and its attributes of a redundant pair. Any Backup AppEngine is not shown. For information about using redundancy, see "About Redundancy" on page 429.

The right area shows the attributes of the object selected in the left pane. Depending on the attribute selected, you can see these properties:

<none>	Automatically defaults to the Value property of the selected attribute.
Category	Determines when and where the attribute's data exists (for example, configuration or run time), which users can write to it, and whether the attribute is lockable or unlockable.
Dimension1 (only for arrays)	Returns the dimension of the attribute if it is an array.
Locked	Determines whether the attribute is currently locked. Valid values are: Unlocked LockedInMe LockedInParent.
Quality	The quality of the attribute as defined in the OPC Draft 3.0 quality definition. ArchestrA stores and transports OPC quality as a 16-bit value. OPC quality is stored for an attribute as a current quality, and it can be historized and sent to clients.
SecurityClassification	Determines which permissions a user has with respect to the attribute when using an ArchestrA application in the run-time environment. Relevant only for attributes that can be written to by users in the run-time environment. If an attribute has no security, this column is blank. For more on security classifications, see "Working with Security" on page 353.

Data Type	The data type of the attribute: • Integer • Boolean • Float • Double • String • Internationalized String • Time • ElapsedTime • ReferenceType • CategorizedStatusType • DataTypeEnum • SecurityClassificationEnum • DataQualityType • CustomEnum • CustomStruct For information about each of these, see the help for the object.
Value	The primary value of the attribute. Sometimes, a list of numbers is included in the **Property** list. Those numbers map to single bits in an integer attribute's Value property. Valid bit field specifiers are: .00 (least significant bit) .01 .02 .03 .04 .05 .06 .07 .08 .09 .10 .11 .12 .13 .14 .15 .16 .17 .18 .19 .20 .21 .22 .23 .24 .25 .26 .27 .28 .29 .30 .31 (most significant bit)

Important: Bit field specifiers are not allowed for integer arrays. Although bit field access is only supported in integers, they appear to be allowed for data types besides integer because they do not cause a warning during configuration. They cause errors in the run-time environment.

Browsing for Graphics

You use the Galaxy Browser to browse for graphics from:

- The ArchestrA Symbol Editor, when editing a graphic from the Galaxy, being either a graphic from an object (Template or Instance) or a graphic from the Graphic Toolbox.
- InTouch WindowMaker when editing a managed InTouch application hosted by the Galaxy.

The graphic currently being edited (or browsed from) does not appear in the list of graphics.

Within the same user session, the Galaxy Browser remembers the last browsed location for graphics and presents it whenever called as the starting location so that context is kept. Initial default location for graphic browsing is the Graphic Toolbox with the root selected (Galaxy node).

You can use the Galaxy Browser to browse graphics from:

- The Graphic Toolbox. The browser shows the Graphics Toolbox toolset organization on the tree in the left pane and a list of the graphics contained in the currently selected node (Galaxy node or toolset node) in the right pane.

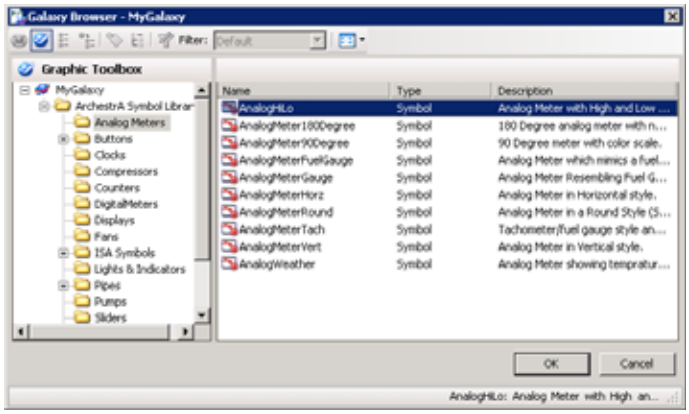
- AutomationObject templates. The browser shows the Template Toolbox in the left pane and the right pane shows the graphics associated with the currently selected template in the right pane.

- AutomationObject instances. The browser shows a flat list of existing instances of objects in the left pane. The right pane shows the graphics associated with the currently selected instance. You create or apply filters to reduce the scope of the instances shown in the left pane.

- Relative references. This is possible only when you edit a graphic belonging to an object and browse for graphics from that specific object.

To browse for graphics from the Symbol Editor

1 Click the **Embed Graphic** button in the Symbol Editor. The Galaxy Browser opens, showing the location of the graphics on the left pane and the graphics associated with the current selection on the right pane.



2 Select a graphic from the list and then click **OK**.

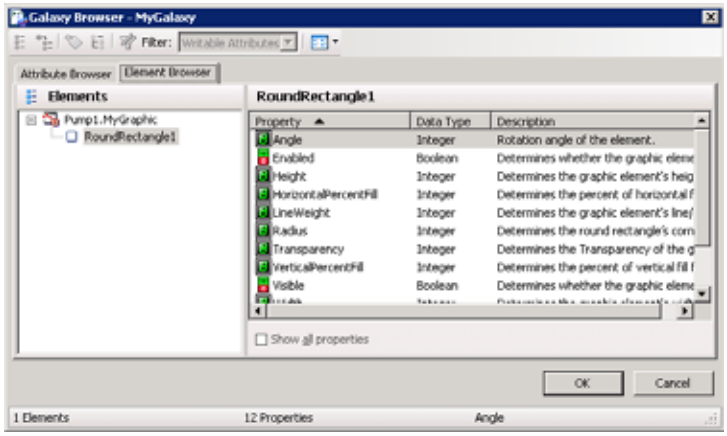
3 Click in the canvas to place the graphic.

Browsing for Element Properties

If you are working on a graphic, you can create references to properties of other graphics. For example, you can reference another graphic's properties from an animation link or script. You can browse the properties of all elements on the canvas or custom properties.

To browse for element properties

1 In any area on a page, click the **Browse** button, if available. The Galaxy Browser opens.



2 Click the **Element Browser** tab.

3 By default, the browser shows only those properties that are frequently accessed. If you are viewing the properties of an element for the first time, the right pane can be blank. Select the **Show all properties** check box to show all of the object's attributes.

4 Select the property and then click **OK**.

- The fully-qualified reference string appears in the option.
- If you are working in the script editor, the selected reference appears in the script at the current cursor position and replaces text that was selected.

For more information, see Chapter 7, "Working with Element Styles," in the *Creating and Managing ArchestrA Graphics User's Guide.*

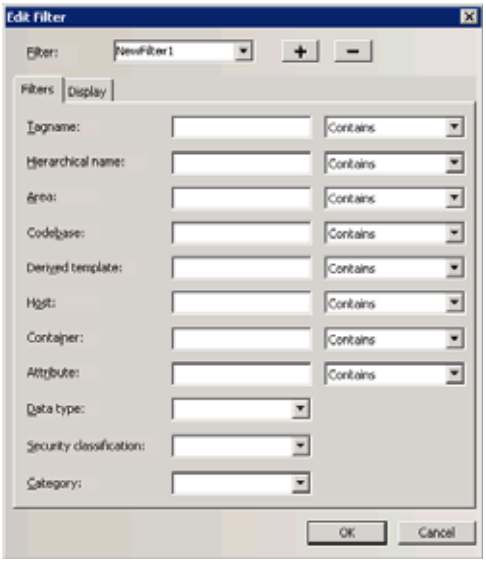
Creating a Filter for the Galaxy Browser

You can create one or more filters that limit the list based on the object name or common attributes. You can also configure the columns you want to show for the list.

The Default filter provides an unfiltered list of objects and attributes. It cannot be edited.

To create a filter

1 Click the **Filter** icon. The **Edit Filter** dialog box appears.

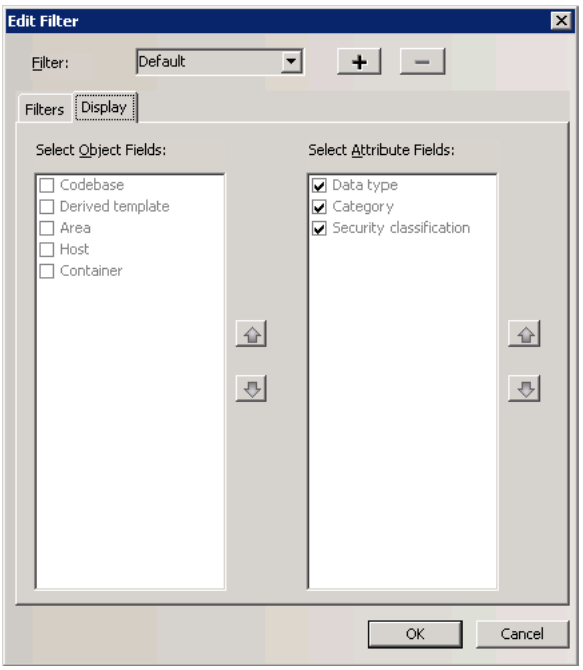


2 Click the **Plus** button and type a name for your new filter.

3 Click the **Filter** tab.

4 Configure the filter details.

5 Click the **Display** tab.



6 Configure the columns to show in the right pane of the **Galaxy Browser**. Use the up and down arrows to set the order for the columns.

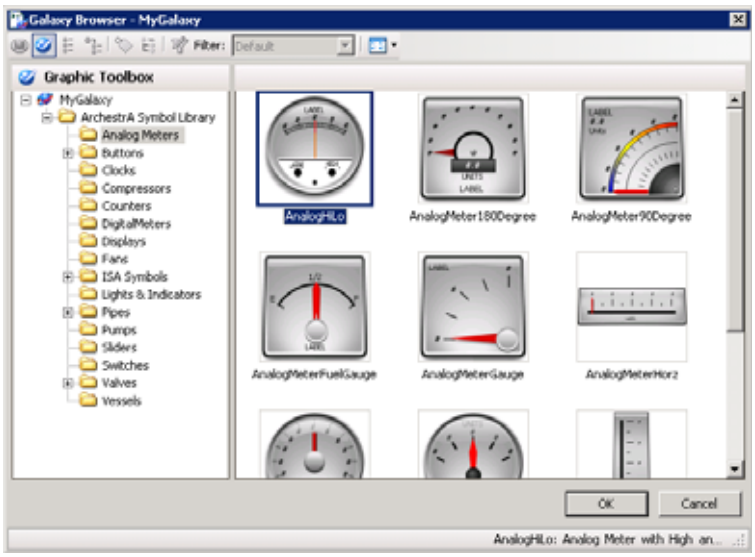
7 Click **OK**. The new filter appears in the **Filter** list.

Changing How Information is Shown in the Galaxy Browser

You can change how information shown in right pane of the Galaxy Browser. You can view:

- A list of only the attribute or graphic names.
- A list of details for attributes or graphics.
- Named graphic icons.
- Thumbnails for graphics.

The following figure shows graphic thumbnails.



Chapter 4

Managing Objects

After you create several objects, like templates, you need to manage them. For example, you need to check objects in and out, you need to validate objects, and you may need to rename objects. You can also export and import objects, allowing you to reuse objects created in one Galaxy in another Galaxy.

Checking Objects Out

To make changes to an object, you must check out the object. Then, you can modify the object and save private versions of it before checking the object in for other users to use. You can select more than one object for checking out at the same time.

The Galaxy marks the objects as checked out to you and it updates the object's **Change Log** that you can view in the **Properties** dialog box. A check mark is shown next to an object's icon in the IDE. No one else can check out the object until you check it back in or until you perform an Undo **Checkout operation**. However, others can open the object for read-only viewing.

To find objects that have been checked out, use the **Find** dialog box. See "Finding Objects" on page 347 for additional information.

To check objects out

1. In the **Template Toolbox** or **Application views**, select the objects you want to work with.
2. On the **Object** menu, click **Check Out**. Or, open the Object Editor. An object is automatically checked out to you when you open its editor.

The Object Editor opens. You are ready to make your changes.

Checking In Objects

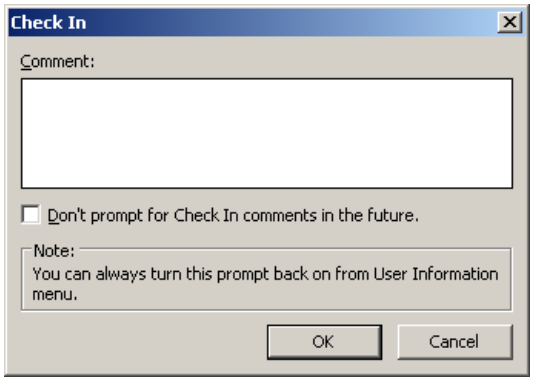
After you finish making your changes, you check the object back into the Galaxy. When you check the object back in, a dialog box prompts you to enter comments about changes you made.

You can turn this dialog box off if you do not want to enter information about your changes. For more information, see “Customizing Your Workspace” on page 37.

Note: If the object was automatically checked out when the editor was started and you close the editor without making any changes to the object’s configuration, an undo-checkout is automatically performed.

To check in an object to the Galaxy database

1. In an **Application view** or the **Template Toolbox**, select the object after you are finished making your changes.
2. On the **Object** menu, click **Check In**. The **Check In** dialog box appears.



3. Type any comments you want and click **OK**.

Validating Objects

Objects need to be validated before they can be deployed. An object validates it's configuration either when you are configuring it, or typically when you save that configuration to the Galaxy database.

Validating an object's configuration includes:

- Checking allowable attribute value ranges.
- Compiling its scripts.
- Verifying its attribute references.
- Validating its extensions.
- Validating other configuration parameters that are unique to the object.

Important: Script validation on a template does not resolve references used in the script. For example, references to attributes that do not exist will not be discovered.

Typically, each option on the Object Editor that requires a string or numeric input has an allowable range of inputs. If you type an input outside the allowable range and then try to change the Object Editor page, close the Object Editor or save the object's configuration, a message appears about the input error, showing the allowable range.

To open the Validation area

◆ On the **View** menu, click **Operations**. The **Validation** area opens.

Validating Scripts and Other External Components

Some objects depend on external components to run, such as script function libraries and references to other objects' attributes. The status of these external components can change, perhaps disabling some capability of the object.

For example, an object refers to a value of an attribute of another object, which is subsequently deleted. This will result in the remaining object going to a Warning status.

Normally, the system will update the validation status of an object when the missing script function or object/attribute is later added to the system. But there are a few cases where the status of an object needs to be manually validated by the user.

For example:

- When importing scripts and script libraries, there are cases when the script will import before the associated library and validate incorrectly, and
- When graphics associated with an object are imported along with a graphic they embed, the containing graphic may be imported first and validated incorrectly.

In each of these situations, the object may incorrectly have a status of either Bad or Warning. In this case, you may want to manually validate the object to update its status, especially if the status is preventing the object from being deployed. For more information, see "Validating Manually" on page 106.

Two kinds of indicators are shown in the object icons:

- Deployment status for instances only.
- Configuration status for templates and instances.

Validating Manually

After you check an object in, you can verify that an object's configuration is valid and update its status by manually validating it. You can use the **Template Toolbox**, the **Application views** or the **Find** dialog box to find objects that need to be validated.

To validate all objects in the Galaxy, validate the Galaxy object.

Note: For a large Galaxy this is potentially a time consuming operation, and should be used only when necessary.

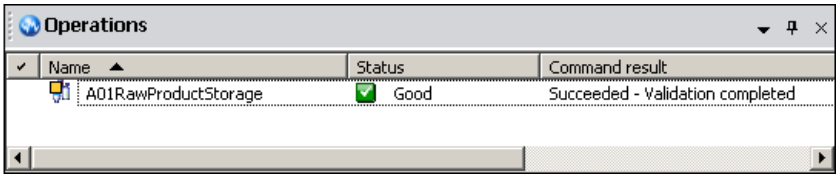
You can select more than one object for validation.

If an object is being edited, validation may not be performed. Also, if validation is in process on an object, other operations you start on the object will fail.

Note: You cannot cancel validation operations.

To manually validate one or more objects

1. In the **Template Toolbox**, the **Application views** or the **Find** dialog box, select the objects you want to manually validate.
2. On the **Object** menu, click **Validate**. The **Operations** view in the IDE opens.



3 Continue using the IDE to perform other operations, if needed, while validation is going on, including work on other objects in the Galaxy. If you are validating a Galaxy, then you must leave the Galaxy alone until the validation process is complete.

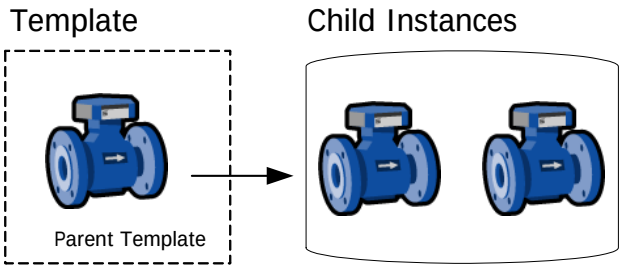
4 When the validation process is complete, you see the results of the validation in the **Operations** View. If the validation failed on an object, you see a message. Correct the problem and validate again.

Note: You can also see the errors or warnings that led to an object having a status that is not Good by looking at the Error/Warnings tab within the object's Properties.

Creating Instances

After you create templates, you can create instances. Creating instances makes a specific object from a template, with all the characteristics and attributes of the template.

After you have created an instance of an object, it can be deployed.



You can also customize an instance, if needed. For example, you can have a valve template. When you create an instance of that valve, you can specify the inputs and outputs for that specific valve on your factory floor.

To create an instance

1 Select the template you want to use for the instance. For example, to create a valve instance, select a valve template.

2 On the **Galaxy** menu, click **New** and then click **Instance**. An instance is created.

3 Rename the instance. Select the instance. On the **Edit** menu, click **Rename**. Type the new name. Instance names can be up to 32 alphanumeric characters. You cannot use $ as the first character. The name must include at least one letter. Instance names cannot include spaces.

4 To move the new instance in the **Deployment** view or **Model** view, drag the instance to the new location.

You are ready to configure the instance, if needed. For more information, see “Editing Objects” on page 69.

Renaming Objects

You can rename an object. Although you can change an object’s containment relationship with another object, you cannot directly rename an object’s hierarchical name. You can rename its tagname and contained name, and its hierarchical name will change automatically. See “Renaming Contained Objects” on page 68 for more information about renaming contained objects.

Object names must be unique within each namespace, not within the Galaxy.

- Template names can be up to 32 alphanumeric characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. You cannot use spaces in an object name.
- Instance names can be up to 32 alphanumeric characters. You cannot use $ as the first character. The name must include at least one letter. You cannot use spaces.

Note: You cannot use the following reserved names for objects: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

An object can have three kinds of names if it is contained by another object. The three names include:

Name	Description
Tagname	The unique name of the individual object. For example, `Valve1`.
Contained name	The name of the object within the context of its container object. For example, the object whose Tagname is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name “Outlet”.
Hierarchical Name	Hierarchical names are fully-qualified names of contained objects that include the name of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: “Reactor1” contains Tank1 (also known within Reactor1 by its contained name “SurgeTank”). “Tank1” contains Valve1 (also known within Tank1 by its contained name “Outlet”). Valve1 could be referred to as: “Valve1” “Tank1.Outlet” “Reactor1.SurgeTank.Outlet”.

When you rename an object, references from other objects to the object being renamed can be broken. Objects deployed with broken references receive bad quality data during run time.

Note: Some objects may refer to themselves or to parent/host objects up in the parent/child hierarchy. References that go up or down the hierarchy to refer to other objects are called relative references. Objects with relative referencing are updated automatically if you rename them.

After renaming, all IDEs connected to the Galaxy show the new object name.

To rename an object's tagname

1. Select the object you want to rename.
2. On the **Edit** menu, click **Rename**.
3. Type the new name for the object.
4. When you are done, press **Enter**.

Deleting Objects

You can delete both templates and instances with the following exceptions. You cannot delete:

- Deployed instances
- Containers for other objects
- Objects checked out by other users
- Templates that have children (derived templates or instances)

Note: Make sure you correctly select the objects you want to delete. After you delete an object, you cannot undelete it. You must recreate it.

To delete an object from the Galaxy

1. In the **Template Toolbox** or **Application views** area, select the object to delete. Select multiple objects by using **Shift+click** or **Ctrl+click**.
2. On the **Edit** menu, click **Delete**. When the message appears, confirm you want the object deleted and click **Yes**.

Exporting Objects

You can export some or all of your Galaxy objects. When you export, you are exporting the objects' associated templates, configuration state, and containment state of those objects. The information is saved in an .aaPKG file.

After the Galaxy objects are exported, you can import them into the same or another Galaxy.

If your objects have scripts associated with them, you need to export the script library separately. For more information about exporting script libraries, see "Exporting Script Function Libraries" on page 115. For more information about scripts and script libraries, see the *Application Server Scripting Guide.*

Before you start, make sure all objects you want to export are checked in. If an object selected for export is checked out, the checked in version of that object is exported instead. This can lead to old versions of objects being exported.

Exporting an entire Galaxy is different than backing up the database. Unlike the case with backups, change logs for the objects are not exported. When you export objects, only the related security information for the specific object is exported.

To export an object

1. In the **Template Toolbox** or **Application Views**, select one or more objects to export.
2. On the **Galaxy** menu, click **Export** and then click **Automation Object(s)**. The **Export Automation Object(s)** dialog box appears.

   To export all of the objects in the Galaxy, on the **Galaxy** menu, click **Export** and then click **All AutomationObjects**.
3. In the **Export** dialog box, browse to a path and type a name for the exported file.
4. Click **Save**. The file is saved with the specified name and an .aaPKG extension.
5. When the export is complete, click **Close**. Now you can import the .aaPKG file into another existing Galaxy.

Protecting Objects on Export

Protect symbols and derived templates by flagging the objects as protected in the galaxy database. Protected symbol and template behavior is similar to that of a base template. This option is available through a specialized form of the **Export Object(s)** operation.

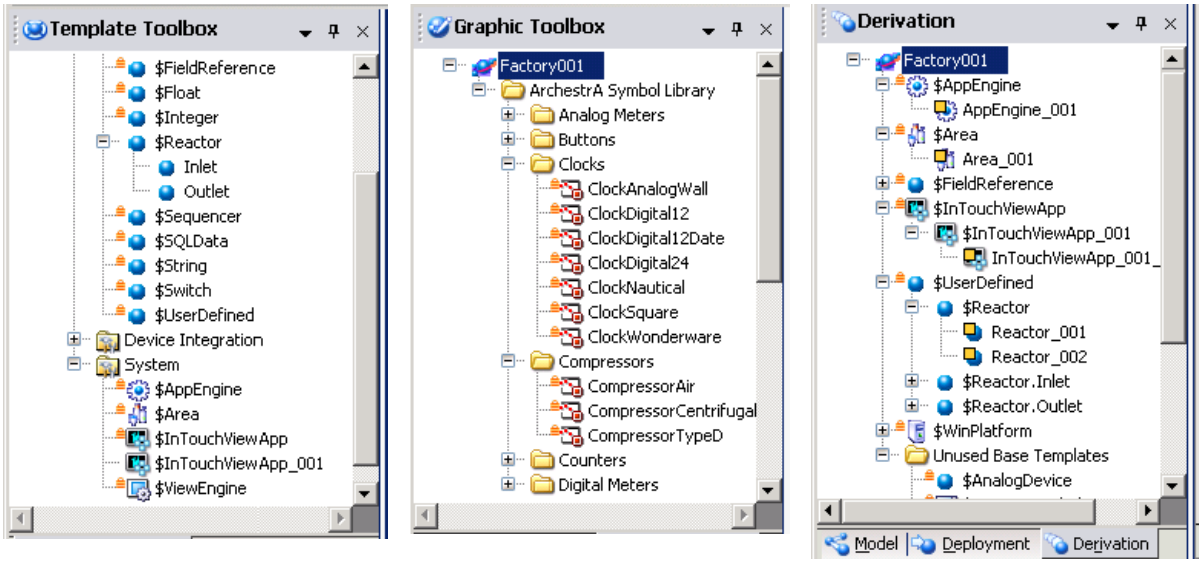
System Integrators and other system designers can use this functionality to protect objects designed in a master galaxy that are intended for use in production galaxies or galaxies on run-time nodes.

About Protecting Objects on Export

Protecting an object on export does not change the object in the galaxy from which it was exported. Protection is effective only on import of protected objects. Specific behavior is described as follows:

Element or Function	Description
Base Templates	Are protected by default and cannot be checked out, edited or renamed.
Protected templates and symbols	Cannot be checked out, edited, or renamed, but can be deleted. A template derived from a protected template is unprotected. Protected templates and symbols are marked with a lock icon.
Ancestor objects	Are protected in the exported .aaPKG file when a child object is protected.
Protection effective	Protection is effective on import of protected objects. A protected object retains its protected status even when exported using the standard **Export Object(s)** workflow.
Instances	Cannot be protected. Only templates and symbols can be protected.
Select both template and instance for export	The option to export objects as protected is disabled. Only templates and symbols can be protected.
Export workflow	The export workflow is the same as for all objects except that you can select the option to export **As Protected Object(s)**.
Graphics	Symbols and client controls directly or indirectly embedded in a protected graphic are also protected. Protected symbols can be opened in the Graphic Editor as read-only.
Scripts	If the Execution type of a script is locked, the script and its attributes will not be visible in the **Object Editor** or **Properties** window. This also applies to all child objects.

Protected objects are marked in the **Template Toolbox, Graphic Toolbox**, or Application views (**Model**, **Deployment**, or **Derivation View**) with a gold-colored padlock icon.



Exporting Objects as Protected

The export objects workflow is the same as for all objects with the exception of specifying the export of selected objects as protected.

To protect objects on export

1. In the **Template Toolbox, Graphic Toolbox**, or in the **Application Views** (**Model**, **Deployment**, or **Derivation**), select one or more templates or symbols to export.
2. On the **Galaxy** menu or context menu, click **Export** and then click **As Protected Object(s)**. The **Export Automation Object(s)** dialog box appears.
3. In the **Export Automation Object(s)** dialog box, browse to a path and type a name for the exported file.
4. Click **Save**. The file is saved with the specified name and an .aaPKG extension.
5. When the export is complete, click **Close**. Now you can import the .aaPKG file into another existing Galaxy.

Exporting Objects with I/O Auto Assignment

Objects configured for I/O auto assignment can be exported the same way as other objects. See “Exporting Objects” on page 110 for additional information.

Application objects configured for I/O auto assignment are linked to DI objects and scan groups. Therefore, when you import these objects into a different Galaxy, the objects will look for DI object and scan group names that match the linkages that were made in their originating Galaxy. If a matching DI object and scan group are not found, all I/O auto assignment information is discarded, and the application objects are placed under the “Unassigned IO Device” folder in the **IO Devices** view.

- When exporting application objects, some I/O mapping assignments may be unavailable when importing the objects into a different Galaxy. Application objects that do not find a DI object and associated scan group that match their I/O references will lose their I/O mapping assignments and will have to be remapped.
- When exporting an application object with overrides in the I/O mapping table, overrides are preserved if a DI object and scan group that match its assignment from the original Galaxy are present in the new Galaxy.

Note: The I/O mapping references are permanently deleted if matching DI objects and scan groups are not found. The I/O references will not reconstitute, even if you later add the scan groups and DI objects to which the application objects were formerly mapped.

- When exporting DI objects, there are no special considerations related to I/O assignments. I/O references for application objects will not change when the DI objects are imported into another Galaxy.

Note: When importing a DI object into a different Galaxy, an existing DI object with the same name as the imported DI object may be overwritten. This can break device linkages to application objects previously configured within the Galaxy.

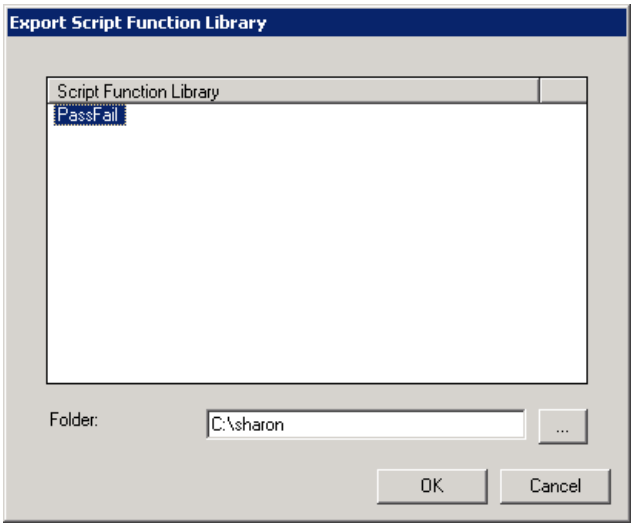
Exporting Script Function Libraries

If you want to export objects that use scripts, the scripts are exported with the object.

Some scripts include functions that depend on external files called script function libraries. In this case, you must export the script function libraries separately.

To export a script function library

1 On the **Galaxy** menu, click **Export** and click **Script Function Library**. The **Export Script Function Library** dialog box appears.



2 In the **Script Function Library** list, select the library or libraries you want to export. If needed, browse to folder where you keep your script libraries.

3 Click **OK**. The selected script library is exported. Each script is named with the name of the script and an .aaSLIB file name extension.

4 When the export is complete, click **Close**. Now you can import the .aaSLIB file into another existing Galaxy.

Importing Objects

You can reuse objects from another Galaxy in your Galaxy. This saves you a lot of time if the objects are already set up in another Galaxy.

Importing instances previously exported from a Galaxy retains previous associations, when possible, such as assignment, containment, derivation, and area.

You can import objects from exported .aaPKG files or from an .aaPDF file. An .aaPDF file contains the configuration data and implementation code for one or more base templates. It is created by a developer using the ArchestrA Object Toolkit.

You cannot have two objects with the same name or more than one copy of the same version of an object in the same Galaxy. When you import an object, you can choose how you want naming and version conflicts handled.

You should perform an "Upload Runtime Changes" before importing a new version base template if instances of the template are deployed. This saves changes made at Runtime to the Galaxy database. For more information, see "Uploading Run-time Configuration" on page 203.

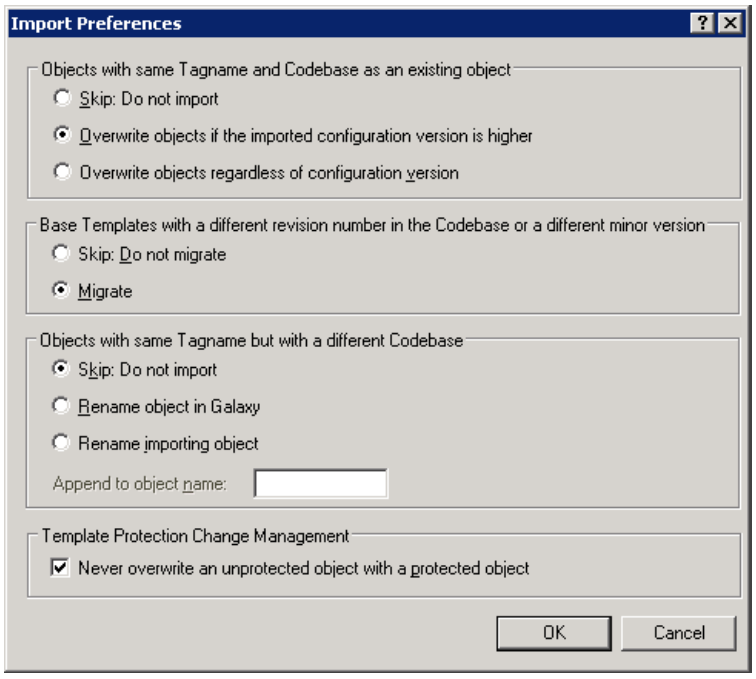
Objects that were created in a Galaxy running a newer version of Wonderware Application Server 2014 R2 cannot be imported into a Galaxy running an older version. For example, you cannot import an object created in Application Server 2014 R2 into a Galaxy running Application Server 2012.

Note: The Application Server version is not the same as the Codebase version. Objects with newer Codebases can be imported.

To import objects

1 On the **Galaxy** menu, click **Import** and click **Automation Object(s)**. The **Import AutomationObject(s)** dialog box appears.

2 Browse for the file with either an .aaPKG or an .aaPDF extension. You can select more than one file. Click **Open**. The **Import Preferences** dialog box appears.



3 In the **Objects with same Tagname and Codebase as an existing object** area, select one of the following:

Skip: Do not import leaves the existing object unchanged.

Overwrite existing objects if the imported configuration version is higher (default) replaces the existing object with the object being imported if the imported object has a newer configuration.

Always overwrite even if imported configuration version is same or lower replaces the existing object regardless of whether the existing object has an older configuration or the same configuration.

4 In the **Base Templates with newer or older Codebases (or minor version updates)** area, select one of the following:

Don't migrate older objects or import minor updates. Skip objects requiring migration does not migrate objects with an older codebase when a newer codebase exists in the Galaxy.

Migrate objects that support/require migration. Import minor version updates migrates an older codebase when the replacement object is available.

For more information about migrating, see “After You Import” on page 122.

5 In the **Objects with same Tagname but with a different Codebase** area, select one of the following:

Skip: Do not import leaves the existing object unchanged.

Rename object in Galaxy imports an object with a matching tagname but a different codebase from the existing one. The existing object is not overwritten but is renamed.

Rename importing object imports an object with a matching tagname but a different codebase from the existing one. The existing object is not overwritten. The imported object is renamed.

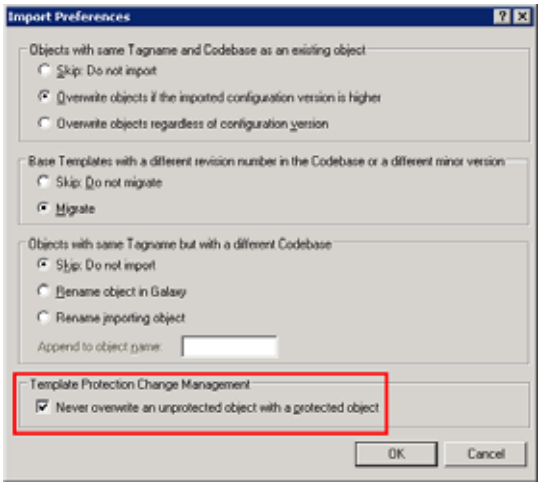
6 In the **Template Protection Change Management** area, click the checkbox to prohibit overwriting an unprotected object with a protected object (default). Clear the checkbox to allow overwrites.

7 Click **OK**. The import process starts.

8 When the import process is complete, you can start using the objects you imported.

Importing Protected Objects

The import objects procedure is the same for all objects, protected or unprotected. The following describes import preferences and results specific to importing protected objects:

Import Preferences	Result for Protected Objects
Objects with same Tagname and Codebase as an existing object: • **Overwrite existing objects if the imported configuration version is higher** (default) • **Always overwrite even if imported configuration version is same or lower**	Overwrite preferences are governed by the **Template Protection Change Management** option, "Never overwrite an unprotected object with a protected object" (default).  • Overwrites can occur if both the packaged and galaxy objects have the same protection status. • A protected object can overwrite an unprotected object only if specifically permitted by clearing the "Never overwrite ..." option. • When overwriting a protected parent object, protection is not cascaded to child objects. Unprotected child objects remain unprotected.

Import Preferences	Result for Protected Objects
Base Templates with newer or older Codebases (or minor version updates): • **Don't migrate older objects or import minor updates. Skip objects requiring migration** • **Migrate objects that support / require migration. Import minor version updates**	Import and migration of base templates is unchanged by protected object import functionality.
Objects with same Tagname but with a different Codebase: • **Rename object in Galaxy** • **Rename importing object**	Object renaming on import can occur only if the object to be renamed is not protected. • If the object in the Galaxy is protected, and the importing object is not, and **Rename importing object** is selected, the importing object is renamed to have the suffix "_new". • If the object in the Galaxy is unprotected, and the importing object is protected, and **Rename object in Galaxy** is selected, and the **Never overwrite an unprotected object with a protected object option** is not selected, the object in the Galaxy is renamed with the suffix "_old".

Importing Objects with I/O Auto Assignment

Objects configured for I/O auto assignments can be imported the same way as other objects. See “Importing Objects” on page 116 for additional information.

When importing objects that utilize I/O auto assignment, be aware of the following:

- If you are importing DI objects, there are no special considerations related to I/O auto assignment. I/O device mapping is not affected. An imported DI object will overwrite an existing DI object with the same name.
- If you are importing application objects, different outcomes can result, depending on the DI objects and scan groups that already exist in the Galaxy.
  - I/O device mapping is recreated, if possible, for each application object that is successfully imported and for which a DI object and scan group are found that match the existing I/O assignment. This includes all user-configured I/O attribute overrides.
  - Objects for which a DI object and scan group with matching names cannot be found are moved to the unassigned area. All I/O information for those objects is discarded. This includes default I/O auto assignment information as well as any custom configured I/O attribute overrides. You will need to assign these objects to a DI object and scan group in the new Galaxy and recreate all user-configured overrides. None of this information is preserved for application objects that do not find a matching DI object and scan group.

Importing Script Function Libraries

You can enhance an object’s functionality by attaching a script to it. Some scripts include functions that depend on external files called script function libraries. Scripts are included in the object import operation, but you must import the script function libraries separately.

If you import an object whose script references a script function library that is not resident in the Galaxy, the imported object is set to Bad state and cannot be deployed. To correct this, import the script function library and validate the object. For more information about scripts, see the *Application Server Scripting Guide*. For more information about validating scripts, see “Validating Objects” on page 105.

If you import a script function library that is a different version than the current library, a message is displayed to notify you that there are dependent objects. The message indicates how many objects will need to be redeployed if you continue with the import. If you continue with the import, the dependent objects are marked for redeployment.

Script function libraries that are COM libraries developed using Visual Studio 6 or earlier are not automatically deployed. To place this COM library on the target platform, you can either:

- Install it directly on the target platform and register it.
- Import it to the Galaxy, and then export it as an aaSLIB. Modify the aaSLIB xml to designate that the library is to be registered as a COM object. Reimport the aaSLIB so that it is automatically deployed and registered.

Importing Client Controls

If you import a client control that is a different version than the current object, a message is displayed to notify you that there are dependent objects. The message indicates how many objects will need to be redeployed if you continue with the import. If you continue with the import, the dependent objects are marked for redeployment.

When the import process is complete, restart the IDE to apply the changes made by the import process. After the IDE has restarted, you can start using the objects you imported.

After You Import

Imported templates are listed in the proper toolset in the Template Toolset as defined in the object. Imported instances are shown in the Application views.

The following post-import rules apply:

- If a toolset does not exist, it is created.
- If the object belongs to a security group that does not exist, it is associated with the Default security group.
- If the object belongs to an area that does not exist, it is associated with the Unassigned Area.
- If the host to which the object is assigned does not exist, it is assigned to the Unassigned Host.

- If you selected **Migrate** from the **Import Preferences** dialog box, the migrated objects are marked with “software upgrade required” if they are deployed. These objects will be upgraded when the objects are redeployed.
- If you import a new version of an existing instance, the new version is marked as requiring deployment if the existing object is already deployed.

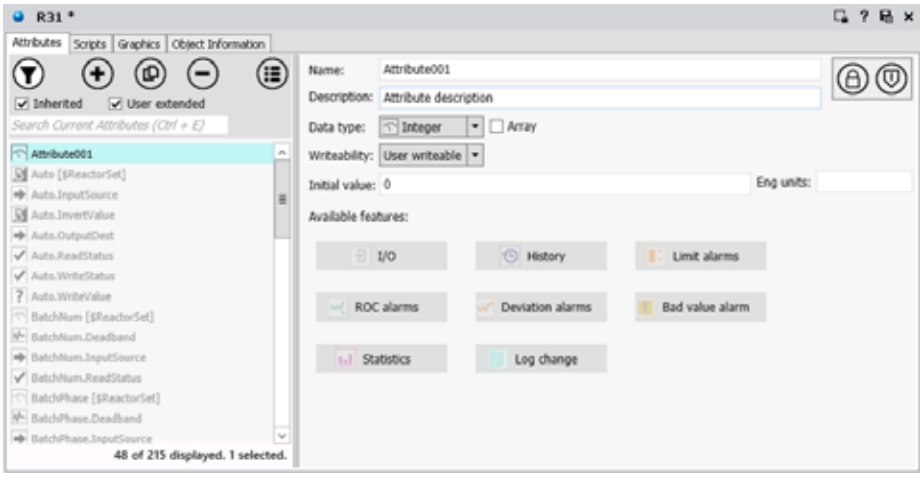
Chapter 5

Enhancing Objects

You can enhance and extend an object by using Features, Attributes, scripts, and graphics. You can add scripts on the **Scripts** page of the **Object Editor**. You can add graphics on the **Graphics** page of the **Object Editor**, For an overview of the **Attributes** page, see "About the Attributes Page" on page 77.

Creating and Working with Attributes

You can add attributes to a derived template or to an instance. When you add an attribute to a template, the attribute, its data type, and writeability are automatically locked in the child instances.



If attribute parameters such as initial values and security classifications are locked in the template, they cannot be changed in child instances. If these parameters are unlocked in the template, the initial value and security are editable and lockable in derived templates. When unlocked in either the base or derived template, the value is editable in instances.

After you add an attribute to an instance, it appears in the **Attribute Browser** list for use with the scripting and attribute configuration functions. For more information about using the **Attribute Browser**, see "Referencing Objects Using the Galaxy Browser" on page 93.

In the **Attributes** page, you can define the following initial information and parameters for the attribute:

- Add a new attribute to an object.
- Name the attribute and provide a description.
- Configure its data type.

  For a Boolean data type, you can specify different text strings for the **'False' label** and **'True' label**. For example, if a Boolean attribute is associated with the status of a motor, you can specify the states as "Stopped" and "Running". Text boxes appear for you to enter these strings when you select a Boolean data type.

  These labels are also shown in the **Value** and **Limit** columns of the Alarm and Event database and InTouch AlarmView control.
- Specify the attribute writeability.
- Set initial values if the attribute is user writeable.
- Enable and set locks and security on the new attribute.
- Set whether the new attribute is an array and how many elements are in the array.

You can then add Features to the attribute. For more information, see "Adding Features to Attributes" on page 128.

Attribute Naming Conventions

Attribute names can have up to 32 alphanumeric characters, including periods. Attribute names must include at least one letter.

- An attribute name that starts with an underscore (_) as the first character of the name is a hidden attribute.

  Hidden attributes do not appear in the **Attribute Browser**, the **Properties>Attributes** dialog box, or the Object Viewer unless you select to view **Include Hidden**.

- Using the word “quality” as an attribute name is not supported. “Quality” is used by InTouch HMI in a set of dotfields to show the reliability of data values assigned to an I/O tag. An attribute named “Quality” cannot be accessed through FS Gateway or InTouch due to a naming conflict.

Important: After creating an attribute, it is available for adding Features or additional attributes. If you extend an attribute you have added to an object or another attribute, and then delete or rename that attribute, all associated Features and additional attributes added to the object are lost.

To create and associate a attribute with an object

1. On the **Attributes** page of the Object Editor, click the **Add** button. An attribute is added to the **Attributes** list.
2. Type the new attribute name and add an optional description.
3. In the **Data type** list, select the **Data type** for the new attribute. The available Features and basic options change depending on your selection in the **Data type** list.
4. Set the remaining parameters as needed.

Note: For detailed information about each item on the **Attributes** page, see "About the Attributes Page" on page 77.

5. Lock the values, if needed. The lock is available only when you are working with a template. If you are working with an instance, it shows the lock status for the value in the parent object.
6. Set any security you need. For more information about setting security, see "Setting Object Security" on page 75.
7. Save and close the Object Editor when you are done.

Attributes and Scripting

When using attributes in scripting, keep the following guidelines in mind.

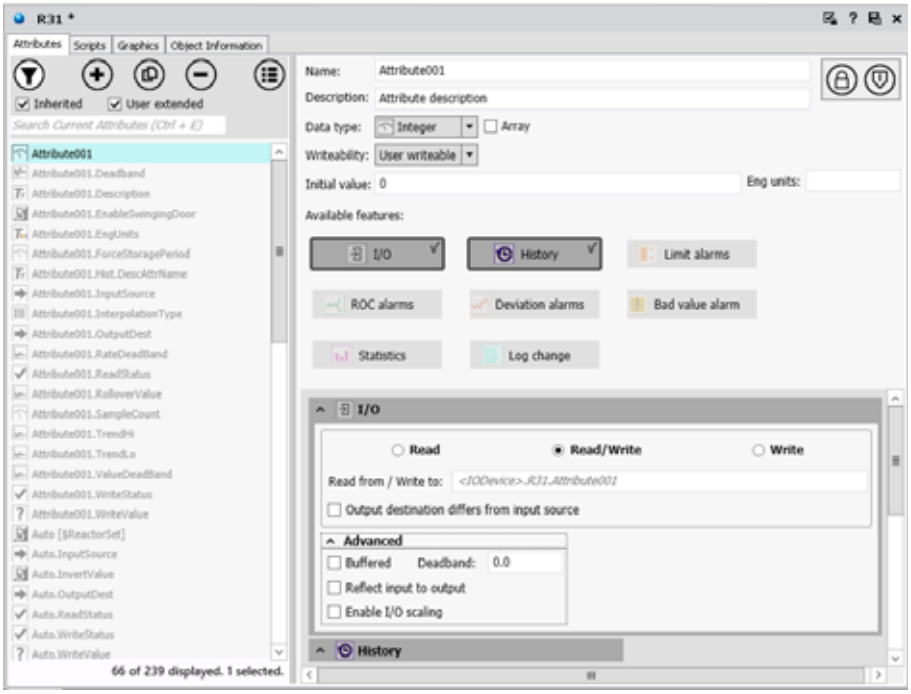
- If you use **Calculated** and **Calculated retentive** attributes as counters, they must be manually initialized. For example, if you use `me.Attribute=me.Attribute+1` as a counter in a script, you must also initialize the attribute with something like `me.Attribute=1` or `me.Attribute=<some attribute value>`.
- **Calculated** attributes can be initialized in scripts with **Execution type** triggers of On Scan and Execute, but not initialized in Startup scripts.

- You must initialize **Calculated retentive** attributes in Startup scripts and you can initialize these attributes in On Scan and Execute scripts. A **Calculated retentive** attribute retains the attribute's current value after a computer restart, redundancy-related failover, or similar situation in which valid checkpoint data is present.

  Your Startup script should contain a statement testing the Boolean value of the StartingFromCheckpoint attribute on the object's AppEngine. If the value is TRUE, do not initialize the attribute. If the value is FALSE, initialize the attribute. For more information about StartingFromCheckpoint, see the help for the AppEngine object, available from the AppEngine object editor.

Adding Features to Attributes

The **Attributes** page allows you to configure an existing attribute for I/O, alarms, and history functionality not embedded in the original object.



About Features Inheritance

You can add Features to attributes that are in either derived templates or in instances. You cannot add Features to attributes in Base templates. The following parent-child object characteristics also apply to Features added to objects:

- If you add a Feature to an attribute in a derived template that has objects derived from it, all child objects inherit the Feature.
- You cannot add a Feature to attributes on derived objects that duplicate parent object Features in name and type.
- You cannot add a Feature with the same name as an existing Feature.
- Renaming a Feature on an attribute in the template to which it was originally added renames all other objects derived from the template. This change happens when the template is checked in.
- You can check in a template with an attribute configured with a new Feature with the same name as an existing Feature on an attribute in a derived object. The template definition of the Feature overrides the Feature in the derived object.
- If you remove a Feature on an attribute from a template, that Feature is removed from any child object. You see the change when you check in the template.

To create and associate a Feature with an object

1. On the **Attributes** page, select an attribute from the **Attributes List**. The available Features dynamically change to allowed Feature rules for the selected attribute type.
2. Click the button for the Feature you want to apply to the selected attribute. The associated parameters for each kind of Feature become available. For detailed information about each item on the **Attributes** page, see "About the Attributes Page" on page 77.
3. Select the parameters for the Feature you have added. Available Features, which vary depending on the attribute's Data Type and Writeability, are the following:
   - **I/O**: For information about adding the I/O Feature, see "Using the I/O Feature" on page 130.
   - **History**: For information about adding the History Feature, see "Using the History Feature" on page 141.
   - **Limit alarms**: For information about adding a Limit alarm Feature, see "Using the Limit Alarms Feature" on page 146.
   - **ROC alarms**: For information about adding a Rate of Change (ROC) alarm Feature, see "Using the ROC Alarms Feature" on page 148.
   - **Deviation alarms**: For information about adding a Deviation alarm Feature, see "Using the Deviation Alarms Feature" on page 150
   - **State alarm**: For information about adding a State alarm, see "Using the State Alarm Feature" on page 152.

- **Bad Value alarm**: For information about adding a Bad Value alarm, see "Using the Bad Value Alarms Feature" on page 153.
- **Statistics**: For information about adding a Statistics Feature, see "Using the Statistics Feature" on page 155.
- **Log change**: For information about adding a Log Change Feature, see "Using the Log Change Feature" on page 156.

4 Lock the values, if needed. The lock symbol is available only when you are working with a template. If you are working with an instance, it shows the lock condition of the value in the parent object.

5 Set any security for the attribute. For more information about setting security, see "Setting Object Security" on page 75.

6 Save and close the Object Editor to include the new Features in the configured object.

Using the I/O Feature

The I/O Feature allows you to configure all aspects of data input and output for an attribute.

You can configure I/O type and you can specify input sources and output destinations. The I/O types you can specify are:

- Read (Input): See "Configuring I/O as Read-only" on page 133.
- Read/Write (InputOutput): See "Configuring I/O as Read/Write" on page 135.
- Write (Output): See "Configuring I/O as Write-only" on page 137.

You can also configure advanced properties, described in the following table. The attribute's data type and I/O type determine what Advanced I/O properties are available.

I/O Feature Advanced Property	Description
Buffered	Enable buffered data to propagate to data subscribers the entire subset of values accumulated within a single scan cycle. Buffering data ensures that if a given attribute changes its value several times during a single scan cycle, there is no folding of data, which occurs when an accumulation of values of multiple data changes within a single scan are overwritten and only the latest value is stored.
Deadband	Specify the minimum amount by which a value must vary in order for the attribute to register a change, for example, by triggering an alarm, historizing an alarm or event, or triggering a script. The Deadband property is not available for string data types.
Reflect input to output	Enable to propagate an input value to an output destination. Enabling automatically disables the **Output destination differs from input source** option. Typically, set this option when you want to read an input from one source, manipulate its value in a script, and send the manipulated value to a different destination address, all during a single scan of an object. For more information, see "Using Read/Write I/O in Scripts" on page 137. Available only when the I/O type is Read/Write. You must set separate **Read from** and **Write to** properties.
Output every scan	Available only when **Reflect input to output** is selected. Write to the specified output destination occurs even when there has been no state or data change since the previous scan. The timestamp is not updated if there has been no state or data change since the previous scan.

I/O Feature Advanced Property	Description
Enable I/O scaling	Enables scaling between the raw value and the Engineering Units (EU) value. Scaling is the process of taking raw data from a device and presenting it as an appropriate value for your application. Wonderware Application Server supports two types of data scaling: linear and square root, described in this table. Available only for integer, float, and double data types.
Maximum	Available only when I/O scaling is enabled. Raw: The raw input maximum to be used in the scaling equation. EU value: The maximum value in engineering units to be used in the scaling equation. Extended EU range: The highest value allowed for the attribute before it is clamped, if clamping is enabled, or set to NaN. This value must be greater than or equal to the specified maximum EU value.
Minimum	Available only when I/O scaling is enabled. Raw: The raw input minimum to be used in the scaling equation. EU value: The minimum value in engineering units to be used in the scaling equation. Extended EU range: The lowest value allowed for the attribute before it is clamped, if clamping is enabled, or set to NaN. This value must be less than or equal to the specified minimum EU value.

I/O Feature Advanced Property	Description
Conversion mode	Available only when I/O scaling is enabled. Select Linear or Square Root conversion mode from the list. Linear: Typically converts directly from one value scale and type (raw) to another value scale and type (EU). *ScaledValue = ((RawValue-RawMin)/(RawMax-RawMin) * (EngUnitsMax-EngUnitsMin)) + EngUnitsMin* Square Root: Provides more precise scaling, typically used when the raw value is too large to be usefully scaled to an EU. *ScaledValue = sqrt((RawValue-RawMin)/(RawMax-RawMin))* *If RawValue < RawMin, then ScaledValue = (sqrt(RawMin) * (EngUnitsMax-EngUnitsMin)) + EngUnitsMin*
Clamp input to EU range	Available only when I/O scaling is enabled. If enabled, the scaling calculation result is clamped at either the maximum EU range value or the minimum EU range value, and the attribute quality is set to Uncertain. If not enabled, and the attribute value exceeds the EU range, then the value will continue to scale out of range, and the quality is set to Bad.

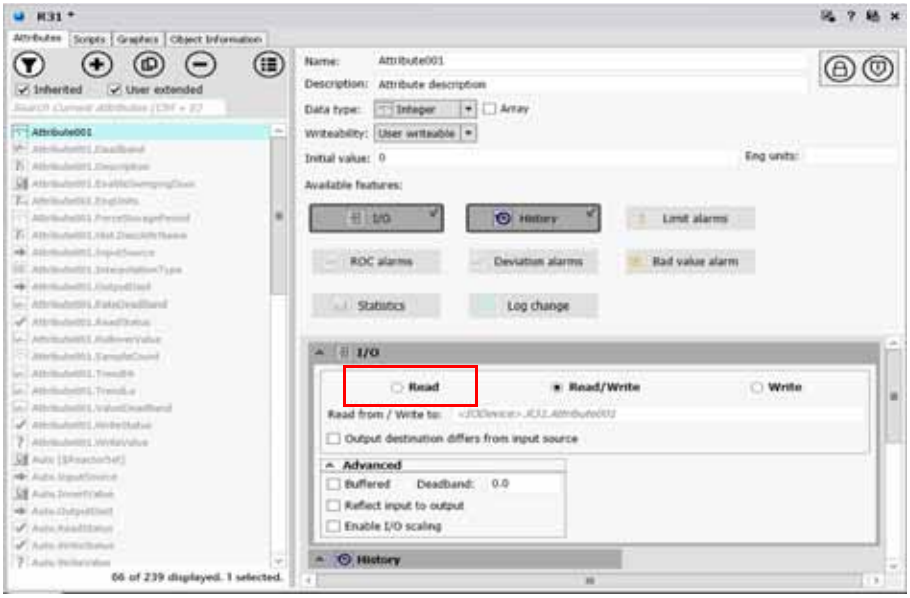
Configuring I/O as Read-only

Select **Read** in the I/O parameters area. Define an input source by using I/O auto-assignment, or by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

- Use I/O auto-assignment to prepare the input source for automatic assignment to a configured Device Integration (DI) object or other data source. For information about using I/O auto-assignment, see "Using I/O Auto Assignment" on page 156.

Important: If the InputSource attribute is locked in the parent template, the attribute cannot be updated with the resolved reference when the object is deployed, and the run-time value will be "---Auto---".

- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see "Referencing Objects Using the Galaxy Browser" on page 93.



You can add multiple Read (input) I/O Features to an object. However, you cannot add a Read/Write (InputOutput) I/O Feature to an attribute that already has either a Read or a Write (output) I/O Feature. Arrays are not supported.

Note: Lockable attributes can be configured with a Read I/O Feature, but they only function correctly during run time if the configured attribute is unlocked.

If the data types of the attribute and its I/O source attributes are the same, they are set to equal values according to the object's execution rate. If the two attributes are different data types, coercion rules are applied.

If coercion fails or the input value is out of the attribute's range, quality for the configured attribute is set to Bad. Otherwise, the configured attribute quality matches the source attribute. When the object is Off Scan, quality is always Bad and user sets are accepted.

Attributes configured with a Read I/O Feature are not protected by their security classification. The only enforced security specifies if an IDE user can edit or add features to the object. A Read I/O Feature can be added to a template or instance. If added to a template, the existence of the Read I/O Feature is automatically locked in derived objects.

Configuring I/O as Read/Write

Select **Read/Write** in the I/O parameters area. Define an input source by using I/O auto-assignment, or by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

- Use I/O auto-assignment to prepare the input source for automatic assignment to a configured Device Integration (DI) object or other data source. For information about using I/O auto-assignment, see "Using I/O Auto Assignment" on page 156.

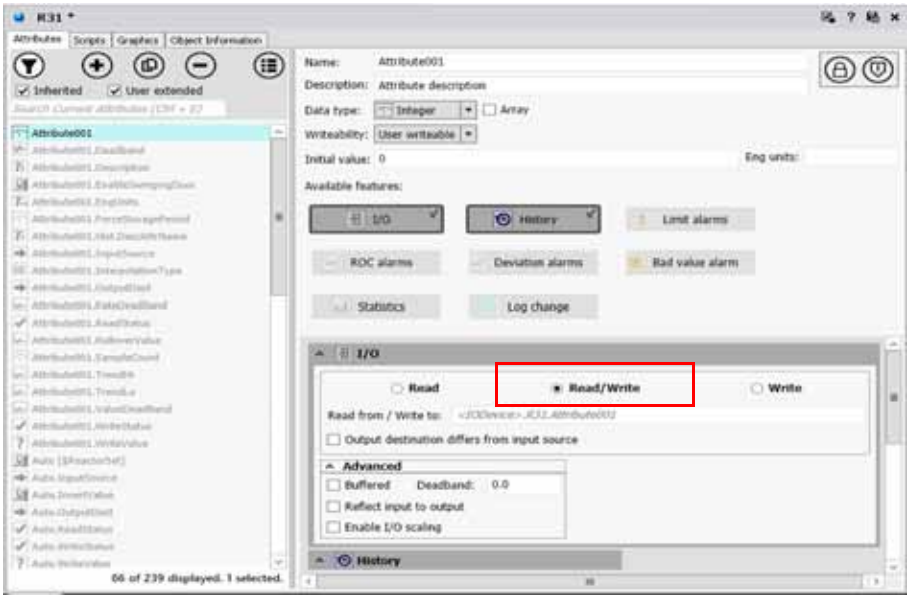
  If InputSource or OutputDest attributes, or both, are locked in the parent template, the attributes cannot be updated with the resolved reference when the object is deployed, and the run-time value will be "---Auto---".

- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see "Referencing Objects Using the Galaxy Browser" on page 93.

If the output destination and the input source are not the same, click **Output destination differs from input source**. Enter a **Destination** attribute by using I/O auto-assignment, by typing in the reference string, or by using the **Attribute Browser** to search for the reference string in an object.

Important: If you clear the **Output destination differs from input source** check box, the **Write to** text box automatically shows “---”. In the run-time environment, “---” is the same reference as the **Read from** value entered during configuration time. During run time, you can change the **Read from** reference. During configuration, do not lock the **Write to** parameter if you clear the **Output destination differs from input source** check box.

A Read/Write I/O Feature allows an attribute in a template or an instance to be configured so that its value is both read from and written to an external reference. The Read/Write I/O Feature monitors the value/quality of an input and sends outputs on state change.



The **Write to** (output) destination can be the same or different from the **Read from** (source). The references are always to another acceptable attribute type in the Galaxy.

You can add multiple Read/Write I/O Features to an object. However, you cannot add a Read/Write I/O Feature to an attribute that already has a Read or Write I/O Feature.

Note: You can add a Read/Write I/O Feature to lockable attributes, but they only function correctly during run time if the configured attribute is unlocked.

When Objects Are On Scan

When an object is On Scan, the value and quality of the attribute configured with a Read/Write I/O Feature mirrors the quality of the externally referenced attribute during a successful read. The data quality of the attribute is set to Bad when reads fail. Reads can fail because of communication errors or datatype conversion failures.

While the object is On Scan, the data can change quality. If an external set (for example, from a user) to an attribute changes either the value or quality, then a write of the attribute's value to the destination occurs during the next execute phase. The quality must be Good or Uncertain for a write to occur. For writes to occur because of a quality change, the quality change must be a transition from Bad or Initializing to Good or Uncertain.

The attribute called WriteValue is publicly exposed and plays an important role in driving outputs. When the object is Off Scan, quality is always Bad and user sets are accepted.

Using Read/Write I/O in Scripts

Two common types of scripts can be written on an attribute configured with a Read/Write I/O Feature: One can look at the input side and one can look at the output side.

The input side script uses the current value coming from the input source location and performs logic or calculations on it. This script refers directly to the attribute in its expressions. For example, if the attribute is "me.attribute1", the script refers directly to "me.attribute1" for data change conditions and for expressions within the script.

The output side script can modify output or validate a new requested output value. This script refers to the "WriteValue" attribute configured on the attribute: "me.attribute1.WriteValue".

To validate a new requested value to the attribute1, for example, a data change condition expression is written on "me.attribute1.WriteValue". In addition, if the script wants to do clamping or validation, it can manipulate the "me.attribute1.WriteValue" directly to clamp the output value. For example:

```
If (me.attribute1.WriteValue > 100.0 ) then

  Me.attribute1.WriteValue = 100.0;

Endif;
```

The data change expression for this script is "me.attribute1.WriteValue" because this value changes when a new value is about to be written to the field.

The script can intercept this value just before output and manipulate it. To prevent WriteValue from being written out, its data quality can be set to Bad with the SetBad() function.

For more information, see "Working with Outputs" on page 139.

Configuring I/O as Write-only

Select **Write** in the I/O parameters area. Define a **Write to** (output) destination by using I/O auto-assignment, or by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

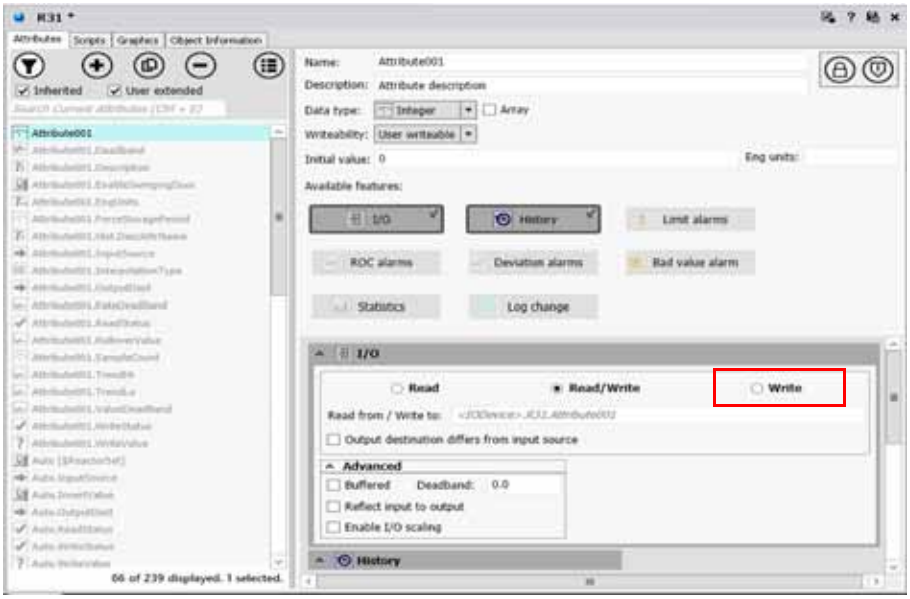
- Use I/O auto-assignment to prepare the output destination for automatic assignment to a configured Device Integration (DI) object or other data source. For information about using I/O auto-assignment, see "Using I/O Auto Assignment" on page 156.

If the OutputDest attribute is locked in the parent template, the attribute cannot be updated with the resolved reference when the object is deployed, and the run-time value will be "---Auto---".

- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see "Referencing Objects Using the Galaxy Browser" on page 93.

Select the **Output Every Scan** check box if you want the attribute to write to the **Destination** attribute every scan period of the object. Otherwise, the write executes only when the value is modified or when quality changes from Bad or Initializing to Good or Uncertain.

Writeable and Calculated attributes can be configured with a Write I/O Feature. Arrays are not supported.



A Write I/O Feature can be added to a derived template or to an instance. If added to a template, the existence of the Write I/O Feature is automatically locked in derived objects. The output **Destination** attribute is separately lockable in templates.

If the data types of the configured and destination attributes are the same and only when the quality of the extended attribute is good, the two attributes are set to equal values according to the configured object's execution rate. If the two attributes are different data types, coercion rules are applied. If coercion fails, the extended attribute is placed into a configuration error and type mismatch state.

An attribute that is enhanced with a Write I/O Feature has the following characteristics:

- A value can be output only when quality is Good or Uncertain. The quality is not output, only the value is output, because quality is not output on sets.
- When the quality changes from Bad or Initializing to Good or Uncertain, the value is output, even if the value is not modified.
- When the quality changes from Good to Uncertain, with no value modification, the value is not output.
- When the object goes Off Scan, no output is done.
- When the extended object is Off Scan, quality is always Good and user sets are accepted.

Working with Outputs

The following information applies to the functionality of Read/Write and Write Features as well as to the output function of the Field-Reference, Switch, and Analog-Device objects.

If a single set request is made to a destination attribute during a single scan cycle, that value is sent to the destination. During a single scan cycle, though, more than one set request to the same destination is possible. In that case, folding occurs and the last value is sent to the destination.

During a single scan cycle, only the last value requested during a scan cycle is sent to its destination when the object executes. Its status is marked as Pending as it waits for write confirmation from the destination object. All other set requests during that scan cycle are marked as successfully completed.

If one or more new sets are requested during the next scan cycle, then the second scan cycle's value is determined as described in the preceding paragraphs. It is then sent to the destination when the object executes again and the value sent to the destination during the previous scan cycle is marked with successful completion status even if write confirmation is not received.

Within a single scan cycle, data is folded and only the last set requested is sent to the destination, unless the **Buffered** attribute is enabled. For example, an {11,24,35,35,22,36,40} sequence of set requests results in a value of 40 being sent to the destination object. All other values result in successful completion status.

The exception to this behavior is when you enable the **Buffered** attribute. For more information, see I/O Feature advanced properties information in "Using the I/O Feature" on page 130. You will find more extensive information about buffered data in "Working with Buffered Data" on page 321.

Boolean data types are an exception to folding behavior when the **Buffered** attribute is not enabled. Boolean data types are used in User sets from InTouch or FS Gateway. This allows an unknown user input rate (for example, repeated button pushes) with a consistent object scan rate for outputs, and creates reproducible results.

In this case, a combination of folding as described plus maintenance of a queue of one element deep better meets the expectation of users. To begin with, the first value set after the object is deployed (the default True or False) is always written to its destination.

Subsequently, the following occurs during a single scan cycle: A two-tiered caching scheme of a Value to be Sent and a Next Value to be Sent is implemented. The Value to be Sent is based on data change as compared to the last value sent to the destination object. The Next Value to be Sent is based on data change as compared to the Value to be Sent value.

When the first data change occurs, the new value is cached in the Value to be Sent queue. Folding occurs if the same value is requested again. If another value change occurs, this second value is cached in the Next Value to be Sent queue. Again, folding occurs if the same value is requested again.

The Value to be Sent value is sent during the next scan cycle, and the Next Value to be Sent value is sent during the following scan cycle.

Note: In the case of Boolean data types used in Supervisory sets (sets between ApplicationObjects and ArchestrA) or a mixture of Supervisory and User sets during a single scan cycle, the behavior is the same as the other data types.

For Boolean data types and User sets, the following examples apply:

Previous Scan Cycle Value Sent	Scan Cycle Set Requests	Value to be Sent	Next Value to be Sent
0	1,0,0,1,1	1	none
1	1,0,0,1,1	0	1
0	1,1,0,0	1	0
1	1,1,0,0	0	none

When the same attribute is extended with an Input extension and an Output extension, writes to the Output extension's **Destination** occur every scan regardless of whether the extended attribute has changed.

This behavior occurs even when the **Output Every Scan** check box is cleared, which may add more network traffic. The behavior does not apply to an Input extension.

Quality of Read, Read/Write, and Write I/O

When the object is On Scan, the value and quality of an attribute configured with a Read I/O Feature mirrors the quality of the externally referenced attribute in the case of successful reads. The data quality of the configured attribute is set to Bad when reads fail because of communication errors or datatype conversion failures.

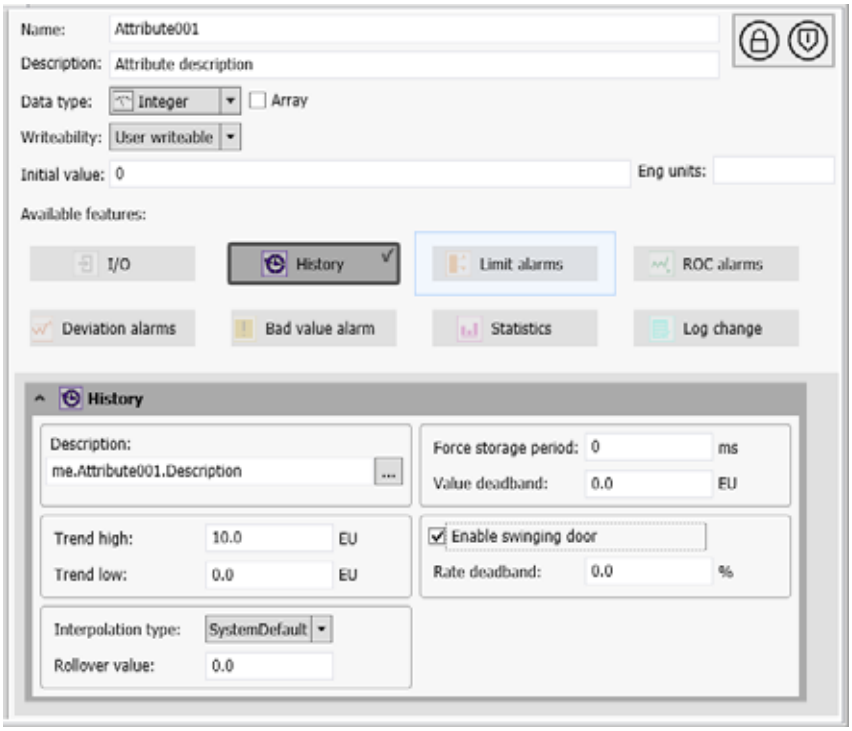
While the extended object is On Scan, it behaves as follows: If an external set (for example, from a user) to the configured attribute causes either the value or quality to change, then a write of the configured attribute's value to the destination occurs during the next execute phase.

The quality must be Good or Uncertain for a write to occur. For writes to occur because of a quality change, the quality change must be a transition from Bad or Initializing to Good or Uncertain. The attribute called WriteValue is publicly exposed.

When the configured object is Off Scan, quality is always Bad and user sets are accepted.

Using the History Feature

Any attribute that exists at run time and is not already historized can be configured with a history Feature.



A history Feature can be added to a template or an instance attribute. If added to a template attribute, the existence of the history Feature is automatically locked in derived objects.

You can configure Writeable and Calculated attributes of the following data types with a history Feature:

- Float, Double (stored as a Float)
- Integer
- Boolean
- String stored as Unicode, 512 character limit
- Custom Enumeration stored as an Integer
- ElapsedTime stored as seconds

You can configure the following attributes for a history Feature:

History Feature Attributes	Description
Description	The attribute containing the description string, if any, associated with the attribute being configured. Enter the attribute manually, or use the browse button to open the Galaxy Browser to find a specific attribute. Example: For the description associated with Attribute001 that has been added to object Reactor31, you would enter or browse to "Reactor31.Attribute001.description". The description also can be a simple string without reference to an attribute.
Force Storage Period	Period after which the value, in milliseconds, must be historized even if the value has not changed. A value of 0 disables the parameter. Example: A setting of 3600000 indicates the value must be stored once per hour (measured from the time the object was last put onscan). If the force storage period value is less than the scan period of the host object, forced storage will occur every scan period (effectively equivalent to setting a value deadband of 0).

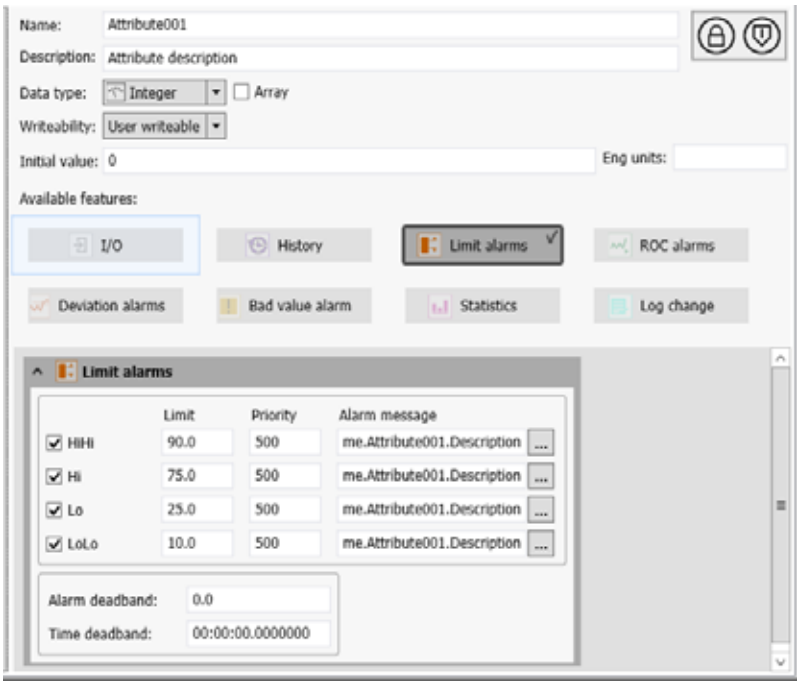
History Feature Attributes	Description
Value Deadband	The threshold value, measured in engineering units, that the absolute value of the difference between the new and last-stored values must differ before storing the new value to history. A value of zero (0) is valid and means that any level of change results in the new value being stored. A change in Quality always causes a new record to be stored, regardless of whether the Value has changed.
Trend High	The default top of a trend scale. This value must be greater than or equal to the low value for the trend. If this value is changed at run time, the maximum engineering unit change is not reflected in the Historian until you redeploy the object. This attribute applies only to numeric data types.
Trend Low	The default bottom of a trend scale. This value must be less than or equal to the high value for the trend. If this value is changed at run time, the minimum engineering unit change is not reflected in the Historian until you redeploy the object. This attribute only applies to numeric data types.
Enable swinging door	Enable and provide a valid swinging door rate deadband and the force storage period becomes the deadband override period. Boolean or string data types cannot be configured with a swinging door deadband.

History Feature Attributes	Description
Rate deadband	Available only if swinging door is enabled. The percentage rate of change deadband based on the change in the slope of incoming data values to the Historian. Example: A swinging door rate deadband of 10 percent means that data is saved to the Historian if the percentage change in slope of consecutive data values exceeds 10 percent. Default is 0.0, which indicates a swinging door rate deadband is not applied. Any percentage greater than 0.0 can be assigned to the rate deadband.

History Feature Attributes	Description
Interpolation type	The method used by the Historian to interpolate analog historical data. The interpolation type determines which analog value is selected during a Historian data retrieval cycle. Select an Interpolation type: **System Default**: The Wonderware Historian system-wide interpolation setting is used. The system-wide setting must be either stairstep or linear interpolated. **Stairstep**: The last known value is returned with the given cycle time. If no valid value can be found, a NULL is returned to the Historian. **Linear**: Historian calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.
Rollover value	A positive integer value that represents a tag's reset limit when the Historian operates in counter retrieval mode. In counter retrieval mode the Historian uses a tag's rollover value to calculate and return the delta change between consecutive retrieval cycles. The default value is 0.0. Boolean and string data types cannot be configured with a rollover value.

Using the Limit Alarms Feature

Select the **Limit alarms** Feature to add and configure a Limit Alarm on an attribute of Integer, Float, or Double data type. You can add a Limit alarm Feature to a template or instance. If added to a template attribute, the Limit alarm Feature is automatically locked in derived objects. Limit alarm Features cannot be added to attribute arrays.



You can enable up to four categories of Limit alarms. When enabled, alarms will be triggered when the value reaches the configured limit.

- HiHi
- Hi
- Lo
- LoLo

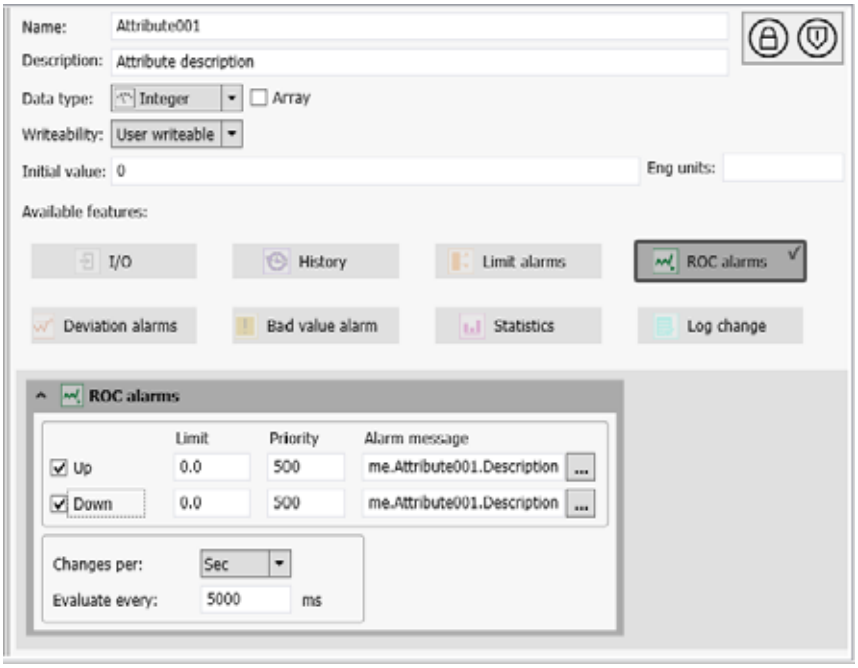
For each Limit alarm, specify the following parameters:

Limit Alarm Feature Parameters	Description
Limit	The limit that the attribute value must exceed to trigger an alarm. Enter the limit value for each enabled Limit alarm category.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see "Mapping Alarm Severity to Priority" on page 273.
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Alarm deadband	The amount, in engineering units, that the attribute value must exceed the configured HiHi, Hi, Lo, or LoLo limit before an alarm is triggered.
Time deadband	The time, in seconds, that must elapse after the attribute value exceeds an alarm limit before the alarm is triggered.

For more information about using Alarms, see "Working with Alarms and Events" on page 231.

Using the ROC Alarms Feature

Select the **ROC alarms** Feature to add and configure a Rate of Change (ROC) alarm on an attribute of Integer, Float, or Double data type. You can add a ROC alarm Feature to a template or instance. If added to a template attribute, the ROC alarm Feature is automatically locked in derived objects. ROC alarm Features cannot be added to attribute arrays.



You can enable up to two categories of ROC alarms. When enabled, alarms will be triggered when the value reaches the configured limit.

- Up: Rate of increase
- Down: Rate of decrease.

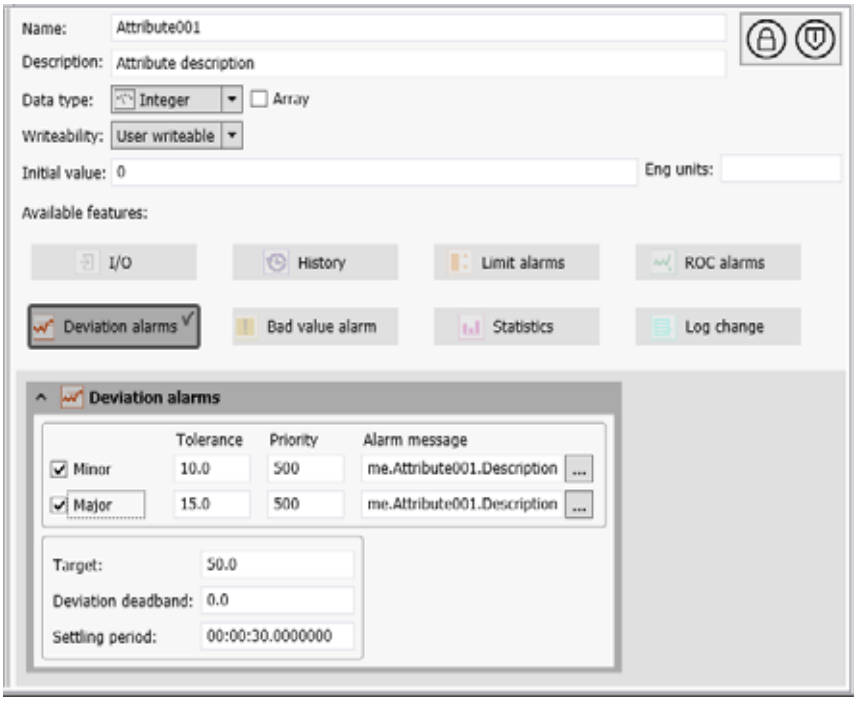
For each ROC alarm, specify the following parameters:

ROC Alarms Feature Parameters	Description
Limit	The limit that the attribute rate of change value must exceed to trigger an alarm. Enter the limit value for each enabled alarm category. The down limit is a positive value, but indicates a negative rate of change, or a descending rate Hi limit.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see "Mapping Alarm Severity to Priority" on page 273.
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Changes per	Select a unit of time for the rate of change calculation – seconds, minutes, hours, or days.
Evaluate every	The time interval, in milliseconds, at which the rate of change calculation and detection is performed.

For more information about using Alarms, see "Working with Alarms and Events" on page 231.

Using the Deviation Alarms Feature

Select the **Deviation alarms** Feature to add and configure a Deviation alarm on an attribute of Integer, Float, or Double data type. You can add a Deviation alarm Feature to a template or instance. If added to a template attribute, the Deviation alarm Feature is automatically locked in derived objects. Deviation alarm Features cannot be added to attribute arrays.



You can enable up to two categories of Deviation alarms. When enabled, an alarm will be triggered when the attribute level deviates from a target value by a configured amount.

- Major: The major alarm deviation tolerance.
- Minor: The minor alarm deviation tolerance.

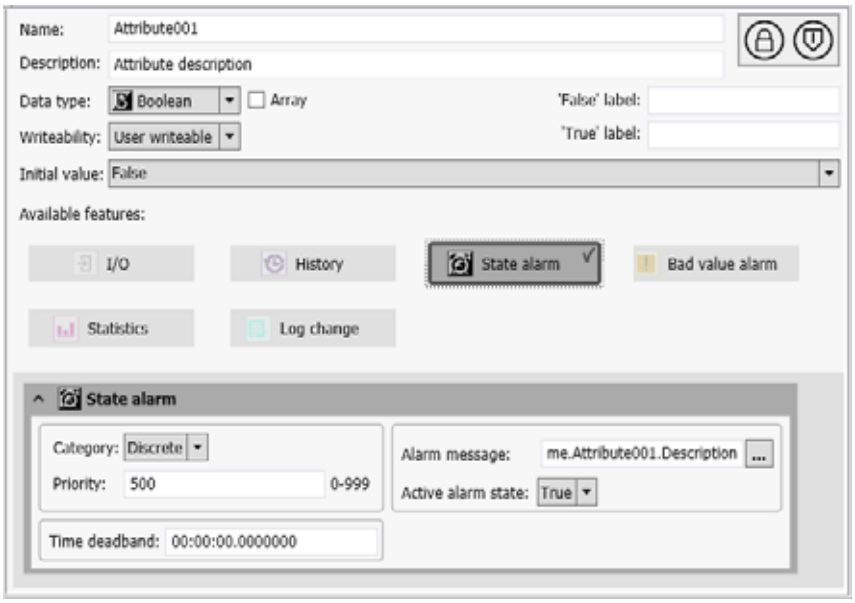
For each Deviation alarm, specify the following parameters:

Deviation Alarms Feature Parameters	Description
Tolerance	The major or minor deviation tolerance that the attribute value must exceed to trigger an alarm.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see "Mapping Alarm Severity to Priority" on page 273.
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Target	The setpoint value for deviation calculation.
Deviation deadband	The amount the attribute value must drop (measured in engineering units) below any of the deviation limits before the respective condition is set to FALSE.
Settling period	The time, in seconds, allowed for the attribute value to return within allowable range after a Target change is made before the deviation alarm is triggered. A deviation alarm cannot occur again until the alarm clears. The value you specify will be converted to the following format: days hh:mm:ss.ffffff.

For more information about using Alarms, see "Working with Alarms and Events" on page 231.

Using the State Alarm Feature

Select the **State alarm** Feature to add and configure a State alarm on an attribute of Boolean data type. You can add a State alarm Feature to a template or instance. If added to a template attribute, the State alarm Feature is automatically locked in derived objects. State alarm Features cannot be added to attribute arrays.



You can enable up to two categories of Deviation alarms. When enabled, an alarm will be triggered when the attribute level deviates from a target value by a configured amount.

- Major: The major alarm deviation tolerance.
- Minor: The minor alarm deviation tolerance.

For each Deviation alarm, specify the following parameters:

State Alarms Feature Parameters	Description
Category	The alarm category. Select a category from the list box.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see "Mapping Alarm Severity to Priority" on page 273.

State Alarms Feature Parameters	Description
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Active alarm state	The state of the active alarm, True or False, when the alarm is triggered.
Time deadband	The time, in seconds, that must elapse after the attribute value exceeds an alarm limit before the alarm is triggered.

For more information about using Alarms, see "Working with Alarms and Events" on page 231.

Using the Bad Value Alarms Feature

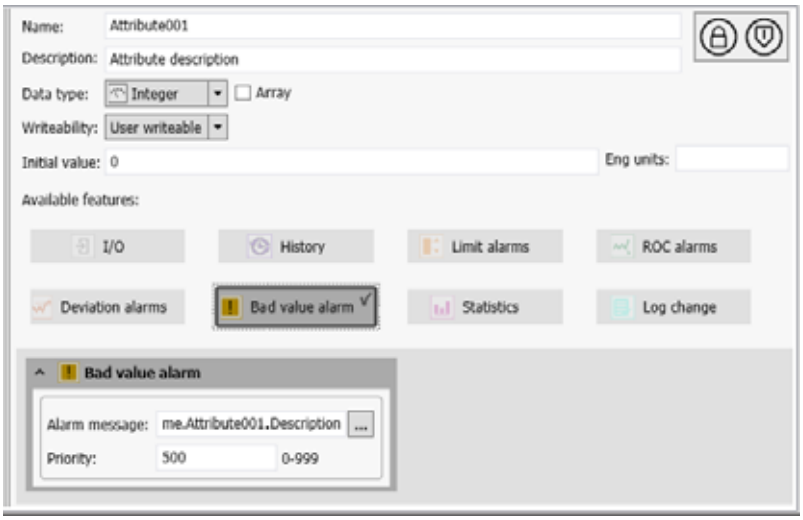
Select the **Bad value alarms** Feature to add and configure a Bad Value alarm on an attribute of Boolean, Integer, Float, or Double data type. If enabled, alarms will be triggered when the attribute has bad data value or data quality.

A bad value results from bad data quality. Bad quality can occur when:

- The PLC indicates that quality is bad (for example, if communications with a physical device is lost).
- There is a communications error (for example, an OPC or I/O server failure).
- There is a configuration error (for example, a broken I/O reference).
- An out of range value is recorded.
- A script sets data quality to bad.

You can configure graphic elements to highlight when a bad data quality state occurs, and its underlying cause. See "Using Quality and Status Styles" on page 183 for more information.

You can add a Bad Value alarm Feature to a template or instance. If added to a template attribute, the Bad Value alarm Feature is automatically locked in derived objects. Bad Value alarm Features cannot be added to attribute arrays.



In the **Alarm message** box, you can browse and select an existing attribute or you can type a text string as an alarm message. This text string appears in the InTouch alarm view.

Specify a **Priority** for this alarm, a numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent.

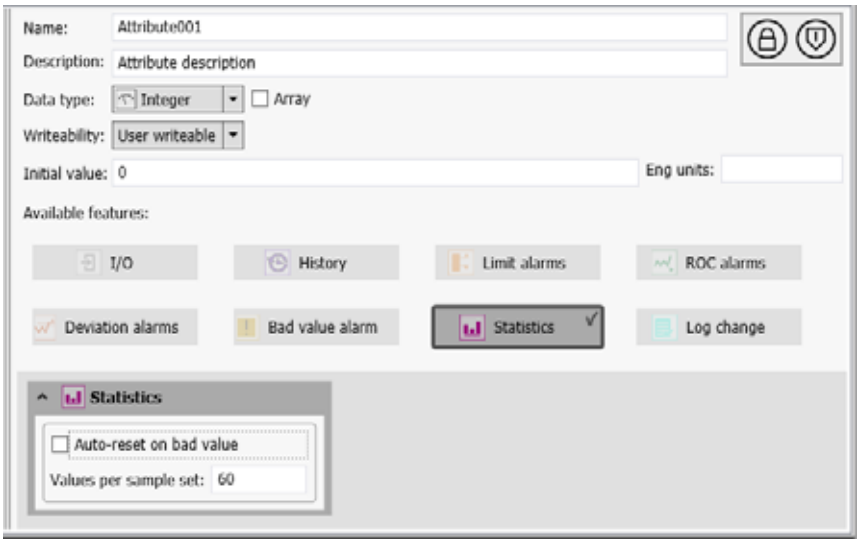
For more information about pre-configuring alarm priorities, see "Mapping Alarm Severity to Priority" on page 273.

For more information about using Alarms, see "Working with Alarms and Events" on page 231.

Using the Statistics Feature

Select the **Statistics** Feature to add and configure calculation of statistics on an attribute of Boolean, Integer, Float, or Double data type. If enabled, the following statistics are calculated:

- Time since last transition: If no change in the input value, the elapsed time value increments to show the total elapsed time since the last transition. A change in input value resets the elapsed time since last transition.
- Average: Defined as the sum of the sample size divided by the number of sample size attributes in the data set.
- Standard deviation: Defined as the amount of variation or "scatter" from the average.

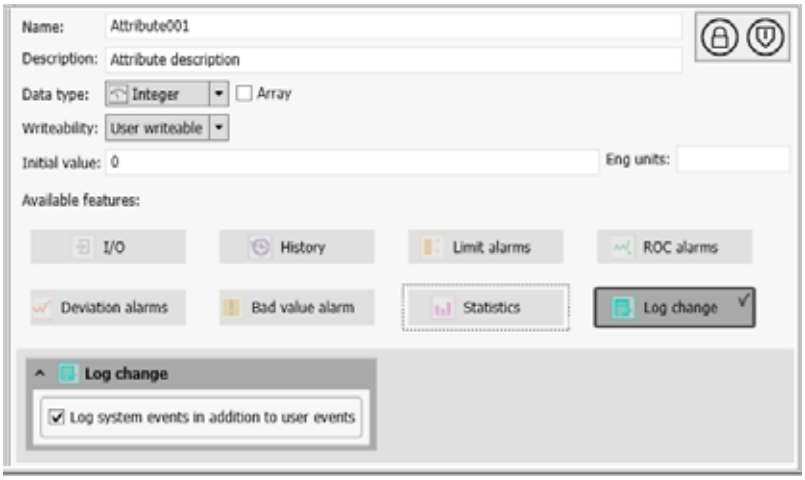


You can configure the following parameters for the Statistics Feature:

Statistics Feature Parameters	Description
Auto reset on bad value	If enabled, the input automatically triggers a reset command when the quality of an input goes from BAD or UNCERTAIN to GOOD.
Values per sample set	The maximum number of samples to collect in calculating the mean and standard deviation. The minimum number of samples is 2. A large sample set will impact run-time performance. This attribute is unavailable for a Boolean data type.

Using the Log Change Feature

Select the **Log Change** Feature to log system events for an attribute of Boolean, Integer, Float, or Double data type.



When enabled, the Log Change feature logs data change events as well as user events.

Using I/O Auto Assignment

Instead of configuring I/O references manually, or writing scripts to set them at run time, you can use I/O auto assignment. Manual configuration of I/O references can be time consuming. Scripting these references eliminates the issues of manual configuration, but can significantly increase the time needed for deploying objects. With I/O auto assignment, you do not need to check out individual objects to configure I/O references, and you do not experience the run-time penalties associated with scripting.

Note: I/O auto assignment is the default setting for application and other system objects, such as area objects. Device Integration (DI) Objects must be manually configured.

When you add input or output attributes to an area or application object in the **Attributes** page of the Object Editor, the default setting prepares these attributes for I/O automatic assignment. The auto assignment reference appears in the I/O section of the **Attributes** page if you have enabled the I/O attribute feature. The default string for an input reference is:

```
<IODevice>.[ObjectName].[AttributeName].InputSource
```

where `[ObjectName]` is the hierarchical name of the application or system object, and `[AttributeName]` is the name of the attribute.

The default string for an output reference is:

```
<IODevice>.[ObjectName].[AttributeName].OutputDest
```

The string `<IODevice>` is a placeholder that indicates the I/O reference will be built automatically in a process called auto binding. The reference resolves automatically as an auto-bound reference when you link the object to a scan group and DI object, without having to manually enter or script the reference.

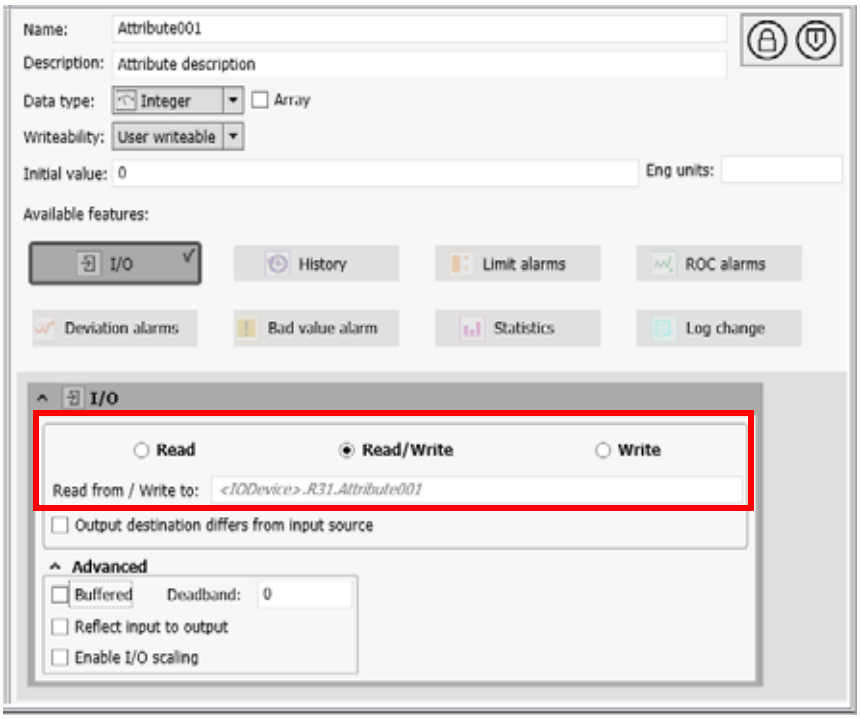
The following is an example of an I/O reference string before the object has been assigned to a scan group and DI object:

```
<IODevice>.Mixer.Tank.Inlet.InputSource
```

Once you assign the object to a scan group, the reference resolves to include the assigned Device Integration (DI) object and scan group. For example, assigning the object to the scan group "Fast" under DI object "OPC001" will change the reference to:

```
OPC001.Fast.Mixer.Tank.Inlet.InputSource
```

Important: Do not lock InputSource or OutputDest attributes when using I/O auto assignment. If either InputSource or OutputDest attributes, or both, are locked in the parent template, the attributes cannot be updated with the resolved auto-bound reference when the object is deployed, and the run-time value will be "---Auto---".



I/O auto assignment is configured in the **IO Devices view**. Use this view to associate application and system objects with DI objects and scan groups.

Scan groups are contained by DI objects and help categorize devices that are associated with them on the basis of how often their I/O points are scanned. For basic information about DI objects, refer to "Device Integration Templates" on page 48. For basic information about scan groups, refer to the OPCClient Object help file.

Note: A DI object will not appear in the **IO Devices view** unless at least one custom scan group has been defined for it.

Assign objects that need to be closely monitored or that have a faster rate of change to a scan group with a faster scan rate. Objects with slower rates of change can be assigned to a scan group with a slower scan rate. The scan rate is the rate at which the ArchestrA run time will perform scans. This may be different than the scan rate a PLC uses to monitor the hardware attached to it.

Note: Fast scan groups consume more network and computing resources than slow scan groups. In a large Galaxy, load balancing may require you to use multiple scan groups with similar scan rates.

The I/O references for the objects' attributes set to use I/O automatic assignment are dynamically generated when you assign an object to a scan group. The auto-bound reference replaces the placeholder string and is displayed in the IDE. It is generally in the following format:

```
<DIObject>.<ScanGroup>.<Object>.<Attribute>
```

`<DIObject>` and `<ScanGroup>` correspond to the name of the DI object and scan group to which you assign the object. For example, if the DI object name is `DI01` and the scan group name is `FastSG`, then the auto-bound reference will start with `DI01.FastSG`.

`<Object>` may consist of multiple elements if there are contained objects. The following is an example of an object string:

```
Tank3.Valve1.Attr2
```

Here, `Tank3.Valve1` corresponds to <object>. The complete auto-bound I/O reference would be:

```
DI01.FastSG.Tank3.Valve1.Attr2
```

Although you assign objects, references are generated for each I/O attribute associated with the object, and not for the object itself. Therefore, you may have to change the I/O references for a subset of an object's attributes. Once objects are associated with DI objects and scan groups, you can validate the auto-bound assignment for each of the object's attributes in the **IO Device Mapping view**, and if necessary, you can enter override values in this view.

Assigning Areas and Objects to Scan Groups

The **IO Devices view** displays all application and system objects, as well as DI objects that have custom scan groups associated with them. Templates are not shown in this view, nor are DI objects that do not have custom scan groups defined for them (DI objects with only the default scan group are not displayed in the IO Devices view).

Note: You cannot assign application and system objects to a default scan group; these assignments can only be made for custom scan groups.

In the **IO Devices view**, objects and areas are initially unassigned and placed in a flat view under the "Unassigned IO Devices" folder. You then select one or more of these unassigned objects or areas and drag and drop them onto a scan group. Deployed application and system objects cannot be assigned to a scan group. Undeploy them first.

Note: DI objects can remain deployed during the assignment process.

Once an object is attached to a scan group, the I/O references for the objects' attributes set to use I/O automatic assignment are automatically generated for you by concatenating the DI object, scan group, object (including contained objects), and attribute names.

You can assign multiple objects for auto-binding at once. Hold down the **shift** or **ctrl** key to select multiple objects and drag the selected items to a scan group. Object assignment is based on hierarchy and area inheritance, as follows:

- If all the objects you select are at the same hierarchical level, all contained (subordinate) objects that have not been assigned, except sub-areas, are also selected and will move to the same scan group.
  - To assign a sub-area, it must be specifically selected.
  - If some contained objects were previously assigned to a different scan group, these will retain their original assignments. To change an existing assignment, you must specifically select the assigned object and move it to a different scan group.
- If you select objects at different hierarchical levels for assignment to a scan group – for example, if you select an area, an object within the same area, and an object from another area – then only the selected items move to the scan group.
- If you select an area that contains another area, only the area selected will move. You must specifically select an area to assign it, regardless of its hierarchical level.

When an area or containing object is assigned to a scan group, the objects it contains are assigned as follows:

- Objects within the area that share the area's device linkage will be reassigned to the new device along with the area. This remains true even in the case where both the area and the contained objects are not currently linked to a device. Unassigned objects within the area will use the area's assignment.
- Objects within the area that are already assigned to a scan group that is different from that of the hosting area will retain their original assignments.
- If necessary, you can reassign contained objects to a different scan group after the initial assignment, or override assignments in the **IO Device Mapping view**.

Note: You cannot change scan group or DI object assignments for deployed objects in the **IO Devices view**. You must undeploy the object first.

Deleting a scan group will delete the I/O assignments of any objects that were previously assigned to it, including override values.

Caution: Creating or deleting scan groups within a derived DI object template may overwrite I/O assignments when these changes are propagated to instances of the DI object template. If this happens, all I/O auto assignment information for objects linked to the updated DI objects, including override values, will be lost. The objects will move to the "Unassigned I/O Devices" folder.

To assign an object or area to a scan group

1. In the **IO Devices view** or **Model view**, select application objects and areas for assignment.
2. Drag and drop the objects and areas to a scan group in the **IO Devices view**.
3. The object or area and all contained objects will be moved to the selected scan group.

Note: If the area or object has contained objects that were previously assigned to a different scan group, these will retain their original assignments. Contained areas will not move and must be overtly selected.

4. The I/O references for each attribute set for I/O auto assignment will be dynamically configured and displayed as auto-bound in the **IO Device Mapping view**.

Renaming Application, System, and DI Objects

You can rename application, system and DI objects in the **IO Devices**, **Model** and **Deployment** views. If an area or object has already been assigned to a scan group, its I/O references are updated automatically.

Renaming Scan Groups

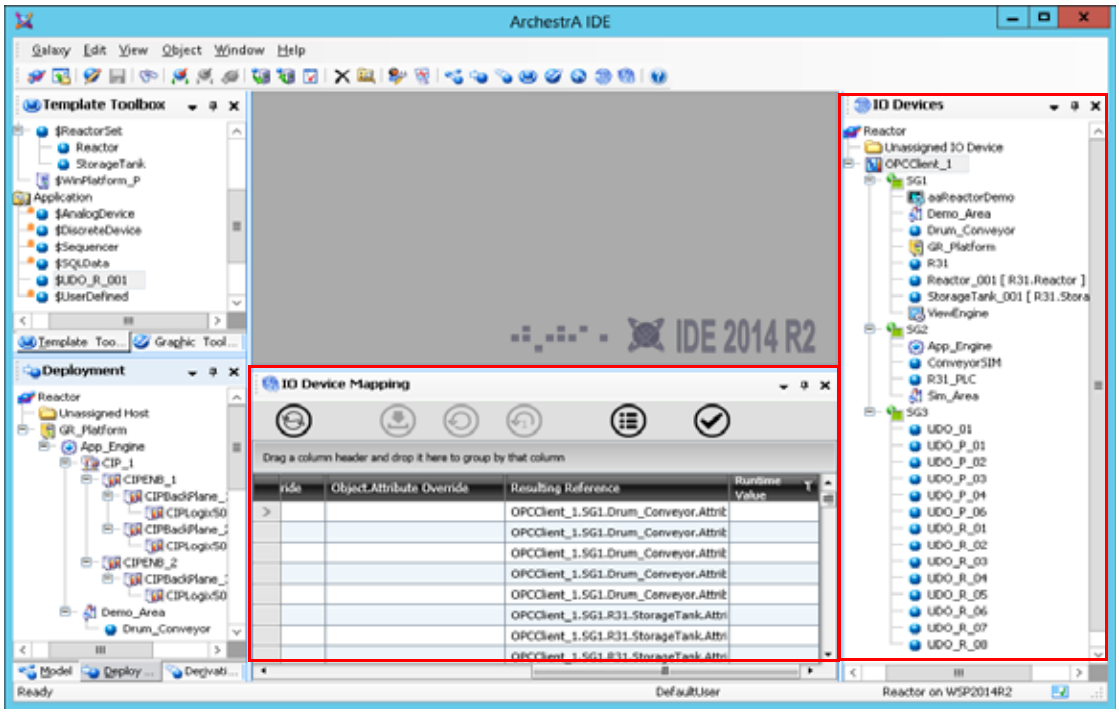
Changes to scan groups (renaming, adding, or deleting them) can only be done by editing the DI object that contains the scan group in the Object Editor. Use the DI object's **Scan Group** page to make any necessary changes.

If scan groups are renamed at the template level, the changes will propagate to the DI object instances and I/O assignments will be preserved.

Validating and Editing I/O Assignments

The **IO Device Mapping view** is a table that displays I/O auto assignment references for application and system objects that are linked to DI object scan groups. Only auto-bound references are displayed in the **IO Device Mapping view**. Application and system objects not yet assigned to a scan group, and manually configured references and are not shown. To be visible in the **IO Device Mapping view**, the object must be selected in the **IO Devices view**.

When you initially open the **IO Device Mapping view** after starting the IDE, the table will scroll so the far right column is in view.



- Selecting a DI object in the **IO Devices view** lists I/O auto assignment attributes for all auto-bound objects linked to all scan groups under it.

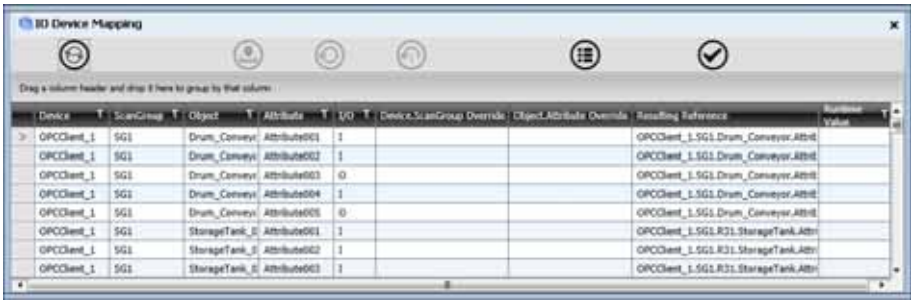
- Selecting individual scan groups restricts the scope of the information displayed in the **I/O Device Mapping view** to the auto-bound objects that have been linked to the selected scan groups.
- Selecting individual application or system object further restricts the scope of attributes displayed to only those associated with the selected object.

Note: You can select multiple items in the **IO Devices view**. Selected items can be at different hierarchical levels. Selecting a subordinate object will exclude non-selected objects within the device hierarchy, even though the parent object is selected.

In the **IO Device Mapping view**, you can view and validate I/O references for each automatically generated attribute, and you can override the automatically generated I/O references. As is the case in the **IO Devices view**, you do not have to check out objects to change their I/O assignments.

Using the IO Mapping View

The IO Mapping view is divided into columns, each of which displays information for an I/O attribute that has been auto assigned.



Column Heading	Definition
Device	Device Integration Object (DIO) name
ScanGroup	Scan group name
Object	Application object or area object name
Attribute	Attribute name
I/O	Input or Output
Device.ScanGroup Override	Override setting: enter an override value to replace the DIO and scan group that was automatically assigned
Object.Attribute Override	Override setting: enter an override value to replace the object and attribute names

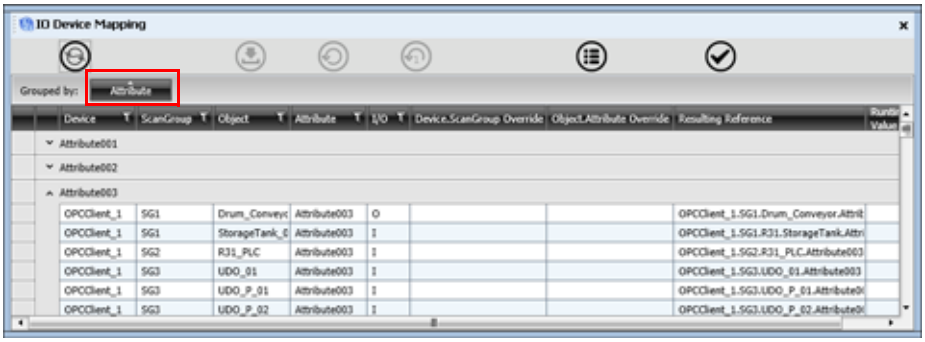
Column Heading	Definition
Resulting Reference	Concatenated I/O reference string (for example, "`DIO1.SG2.Mixer3.Attr5`" or "`DIO1.SG2.Conveyor.40001`")
Runtime Value	Data returned from run time when the **Validate References** button has been clicked

You can change how I/O attributes are displayed by sorting, grouping, or filtering the attributes.

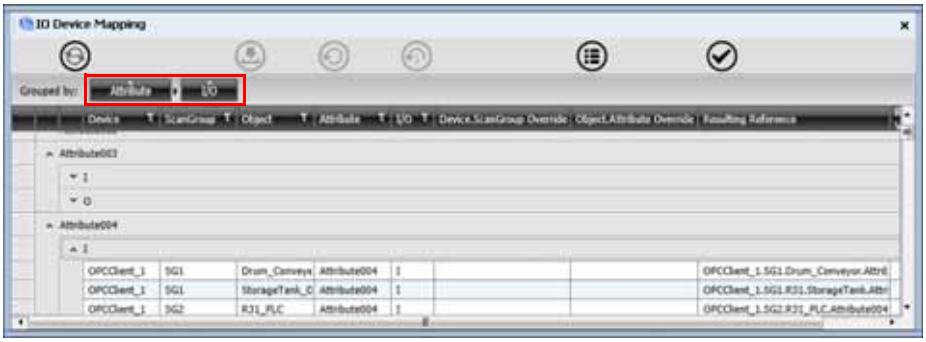
Attributes are initially sorted by Device name (DI object). To sort attributes using a different column, click on the column heading once to sort in ascending order. Click a second time on the same column heading to sort in descending order. A third click on the column heading turns off sorting for that column.

To group objects by I/O assignment criteria, drag a column heading to the top of the table in the **IO Device Mapping view**. For example, if you drag the heading "Attribute" to the top of the table, I/O assignments will be grouped by attribute name. In the figure below, I/O assignments are grouped by attribute name. Note each attribute grouping has been expanded.

Note: Grouping is not available for the **Device.ScanGroup Override**, **Object.Attribute Override**, or **Resulting Reference** columns.



You can use more than one column heading to group I/O assignments. In the figure below, assignments are grouped first by attribute name, and then by I/O type.



Columns can also be filtered to display only certain entries by clicking the filter symbol and then selecting the filtering criteria. When a filter is applied, the color of the filter symbol changes to orange.

Note: Filtering is not available for the **Device.ScanGroup Override**, **Object.Attribute Override**, or **Resulting Reference** columns.

The table contains additional information that you can reveal by clicking the **Show Details** button.

Details Column Heading	Definition
Attribute Data Type	Attribute type as defined by the attribute definition set in the Object Editor
Target Data Type	Attribute type of the target, as determined by the run time
Runtime Quality	Data quality from run time

Validating I/O References

You can validate auto-bound I/O attributes that have been assigned in the **IO Device Mapping view**. To begin I/O reference validation, press the **Validate References** button.

An estimate of the time needed for validation is displayed. The time is updated as validation progresses. When validation completes, the total number of unresolved I/O references is displayed along with the total number of auto-bound references checked, for example, "2 out of 5000 reference(s) failed to validate. To view references that did not validate during the validation period, click the "Show Advanced Columns" button. This will reveal the **Runtime Quality** column.

You can sort the **Runtime Quality** column to aggregate unresolved I/O references at the top of the list of references. See "Using the IO Mapping View" on page 162 for additional information on sorting. You can then move the associated objects to a different scan group or apply overrides. Recheck the references after completing any required reassignments and overrides.

Overriding I/O Auto Assignments

The **IO Device Mapping view** contains two override columns. To change an I/O assignment, enter the new parameter in one or both of the two override columns.

The information in other columns cannot be edited directly. You can also reassign an object to a different scan group in the **IO Devices view**, in which case the new assignment will be reflected in the **IO Device Mapping view**.

Note: If an object is already deployed, changing any of its I/O assignments in the **IO Mapping view** will change the object's status from "Deployed" to "Deployed with pending configuration changes."

To override the DI object and scan group assignment that was made in the **IO Devices view**, enter the value you want in the column **Device.ScanGroup Override.** Naming restrictions for the Device.ScanGroup Override value are as follows:

- The value must contain one DI object name and one scan group name contained by the DI object, separated by a period (.).
- The value cannot start or end with a period.
- The value cannot have any leading spaces.
- The I/O reference cannot be longer than 329 characters.

To override the object and attribute name, enter the value you want in the column **Object.Attribute Override**. Naming restrictions for the Object.Attribute Override value are as follows:

- The value cannot have any leading spaces.
- The I/O reference cannot be longer than 329 characters.

You can change attribute information using the Object Editor (after checking out the object), instead of overriding values in the **IO Device Mapping view**. However, if you enter the specific I/O reference for an attribute in the Object Editor (manual configuration), it will no longer appear in the **IO Device Mapping view**. Only attributes configured for I/O automatic assignment appear in this view.

Overriding Large Numbers of Attributes

If you have a large number of overrides to enter, information can be copied and pasted between the **IO Device Mapping view** and Microsoft Excel. However, you must be careful when moving data to and from the **IO Device Mapping view**. Observe the following conditions:

- Only data in the two override columns is editable. All other columns are read only.
- Even though you can only change the override columns, always copy all columns. This will help you verify that data is being copied correctly when you paste back into the **IO Device Mapping view**.
- When you copy data from the **IO Device Mapping view**, column headers are automatically included to help you while editing in Excel. When you copy your edited data back to the **IO Device Mapping view**, do not copy this header row, only copy the data.
- If you need to sort attributes, sort them in the **IO Device Mapping view** before you copy them into Excel.
- Do not change the sort order of the attributes in the **IO Device Mapping view** until you have pasted the edits made in Excel and applied them.
- Do not sort the attributes in Excel. Attributes can only be sorted in the **IO Device Mapping view** and the sort order cannot change until your edits have been applied.

WARNING! The IO Device Mapping view does not verify data that is copied to it. Therefore, if the order of attributes is different in Excel than it is in the IO Mapping view, the I/O references will be broken. Once changes are applied, there is no way to undo them.

To validate and edit I/O assignments

1. Click the **Validate References** button to check that the I/O assignments in the IO Device Mapping table are valid. This action retrieves data and quality information from the run-time data acquisition subsystem.

Important: I/O assignments can only be validated if the Platform, AppEngine, and associated DI object have been successfully configured and deployed.

2. Check the data returned in the **Runtime Value** column. If the assignment is not valid or if the quality is bad, no data will be returned (the Value column will be empty), and "Not Connected" will be displayed in the **Runtime Quality** column (click the **Show Details** button to reveal this column).

3 To isolate invalid assignments, select the filter icon at the top of the **Runtime Quality** column filter for "Not Connected" messages.

4 Edit the I/O assignment of objects as needed.

- To override a scan group and DI object assignment, enter the new value in the **Device.ScanGroup Override** column. The override value for this column cannot contain more than one period (.) to separate the DI object name from the scan group name.
- To override an object and attribute name, enter the new value in the **Object.Attribute Override** column.

5 Click **Apply Pending Overrides** to save your changes.

- To discard the last unsaved override setting, click **Undo Last Pending Override**.
- To discard the all unsaved override settings, click **Undo All Pending Overrides**.

Uploading Run-Time Configuration Changes for Auto Assigned Objects

If changes are made during run time to an auto assigned I/O reference, you can upload these changes as overrides to the original assignments. Invalid changes are ignored, and an error message is sent to the SMC Logger.

- Run-time changes to scan groups or DI objects are saved as **Device.ScanGroup** overrides.
- Run-time changes to objects or attributes are saved as **Object.Attribute** overrides.

When you look at the object in the **IO Devices view**, you will see that the original I/O auto assignment setting is retained and the object remains auto assigned to its original DI object and scan group.

The changes uploaded from run time will appear in the **Resulting References** column of the **I/O Device Mapping view**. The same information from run time will appear when you examine the object's I/O attributes in the **Object Editor**. These changes are preserved across subsequent redeployments.

To upload run-time configuration changes

1 From the **Model** or **Derivation** view, select the object(s) with run-time changes you want to preserve.

2 Click **Upload Runtime Changes** from the **Object** menu. The **Upload Runtime Changes** dialog box will appear and provide status of the upload process.

I/O Auto Assignment Workflow Example

This section describes the basic steps for implementing I/O auto assignment, from start to finish. The basic steps are:

1. Use the **Attributes** page of the **Object Editor** to prepare system objects and application objects for I/O auto assignment.
2. Use the **Object Editor** to configure DI objects and scan groups.
3. Use the **IO Devices** view to assign system and application objects to scan groups (objects must be undeployed).
4. Use the **IO Device Mapping** view to validate I/O references. You can enter overrides in this view, if necessary.

To prepare objects for I/O auto assignment

1. Create a derived template from the $UserDefined base template and open it in the **Object Editor**.
2. Add attributes as needed. For each attribute that requires input or output, click the I/O button.

   The default I/O mode is **Read/Write**, and the default string indicating that the attribute is primed for I/O auto assignment appears in the **Read from / Write to** field.
3. Change the I/O mode as needed.

Important: Do not change the default string if you want to use auto assignment for the attribute. The default string indicates that auto assignment is not yet complete and the object is not linked to a PLC.

4. Save and check in the template, then close the Object Editor.
5. In the **Template Toolbox**, create instances from the UDO derived template. The instances appear in the **Model view**, under Unassigned Area.

To configure DI objects and scan groups

1. In the **Template Toolbox**, create a device integration (DI) object from the $OPCClient, $DDESuiteLinkClient, or $Redundant DIObject template. The DI object appears under "Unassigned Area" in the **Model view** and under "Unassigned IO Devices" in the **IO Devices** view.
2. Add one more scan groups to the DI object you created in the previous step. The naming convention for the scan groups must conform to the naming convention of the PLC with which you are connecting.

   The scan groups will show as belonging to the DI object in the **IO Devices view**.

To assign system and application (user created) objects to scan groups - Method 1

1. Create a new area and set it as the default. Do not deploy the area yet.
2. In the **IO Devices view**, drag and drop the area to a scan group.
3. Create new application objects. As you create new objects, they will appear in the **IO Devices view** under the default area.

   The IDE will complete the references for each I/O attribute for you.
4. If you need to change the scan group assignment for an object, drag and drop it to the new scan group.

To assign system and application (user created) objects to scan groups - Method 2

1. Create a new area and assign it to a scan group.
2. Create new application objects. These will appear under the Unassigned folder in the **Model view**.
3. Assign the application objects to the area assigned to the scan group in step 1. All application objects assigned to the area will inherit the area's linkage to the DI object and scan group.
4. If you need to change the scan group assignment for an object in the area, drag and drop it to the new scan group.

To validate I/O references (and override auto assigned values, if necessary)

1. Open the **IO Device Mapping view**. This view opens automatically if you select an object, scan group, or DI object in the **IO Devices view**.

   Each row in the IO Device Mapping table contains the I/O reference for one attribute.
2. Enter an override value for the attribute in one of the override columns.

Note: The **DI Object.Scangroup Override** column can only contain the DI object name and scan group name, separated by a period (.).

3. Deploy the platform and associated DI objects.
4. Click the **Validate References** button to check that communication is established with the PLC.
5. Deploy the objects.

Using I/O Auto Assignment with OPC Clients

Using I/O auto assignment with the OPC communication protocol may require that you add overrides to allow communication between the IDE and certain PLCs, such as Allen-Bradley Logix and Siemens S7 controllers.

If you are using auto assignment to communicate with a device connected to an OPC Server on a different node (for example, a DAServer in a remote Galaxy), you will use overrides to build valid I/O references with all required parameters to fulfill the OPC hierarchical path. If bulk edits are required, you can copy and paste between the **IO Device Mapping view** and Microsoft Excel.

I/O auto assignment uses the syntax: <*DI Object.ScanGroup*>. <*Object.Attribute*>. It is the first part of the syntax that limits how the OPC client builds the I/O auto assignment reference, since additional parameters in the device hierarchy, such as Ethernet ports and backplanes are not included. While these parameters may be logically viewed as part of the *DI Object.ScanGroup* reference, you must add the parameters to the **Object.Attribute override** column (not the **DI Object.ScanGroup override** column) to create the fully qualified I/O reference.

Using DDESuiteLink instead of OPC will accommodate the automatically generated I/O reference to these PLCs, and in most cases, overrides will not be needed.

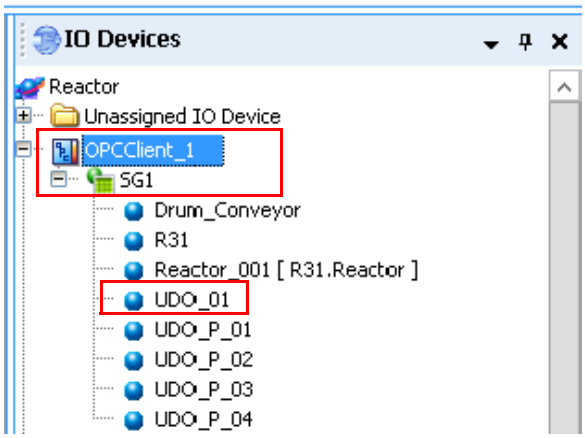
Caution: Do not change the order of references, either in Excel or in the IO Device Mapping view by sorting, grouping, or filtering the reference list while editing references.

Override Scenario 1 — Siemens S7 Controllers with an OPC UA Client

The following scenario illustrates using I/O auto assignment with a Siemens S7 PLC. The Siemens S7 PLC in this example scenario includes a TCP/IP port that is in addition to the OPC DI object and scan group names captured by I/O auto assignment in the **IO Devices view**.

To configure the Siemens S7 PLC with the OPC Client

1. Select the OPCClient object or scan group that contains the PLC in the **IO Devices view**.



I/O attributes for each object associated with the OPCClient or scan group (depending on which is selected in the **IO Devices view**) are shown in the **IO Device Mapping view**.

2 In the **IO Device Mapping view**, the resulting reference will be in the form:
`OPCClientName.ScanGroupName.ObjectName.AttributeName`

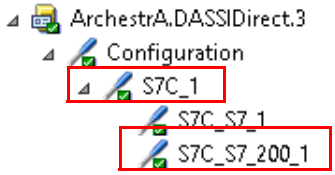
Example (**Resulting Reference** column):

`OPCClient_01.SG1.UDO_01.Attribute_001`

3 Do not edit the **Device.ScanGroup Override** column (leave it blank). This leaves the DI Object and scan group assignment as defined in the **IO Devices view**.

4 Enter the fully qualified field reference in the **Object.Attribute Override** column. For example:

`S7C_1.S7C_200_1.UDO_01.Attribute001`



5 Apply the override. The resulting reference will be displayed. For example:

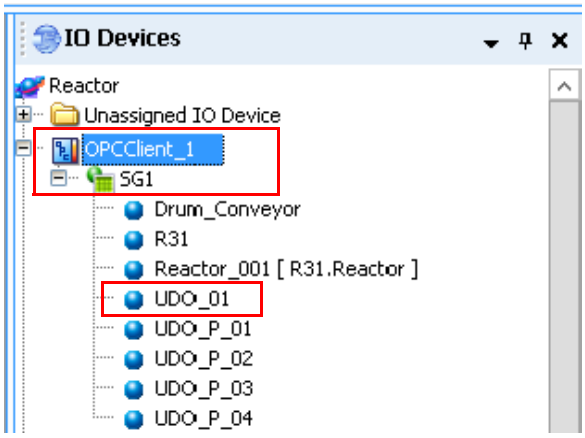
`OPCClient_01.SG1.S7C_1.S7C_200_1.UDO_01.Attribute001`

Override Scenario 2 — Allen-Bradley CIP Controllers with an OPC UA Client

The following scenario illustrates using I/O auto assignment with a Allen Bradley Logix PLC. The Allen Bradley Logix PLC in this example scenario includes a device hierarchy with a port object, ethernet port, backplane, and controller. These are in addition to the OPC DI object and scan group names captured by I/O auto assignment in the **IO Devices view**.

To configure the Allen-Bradley Logix PLC with the OPC Client

1 Select the OPCClient object or scan group that contains the PLC in the **IO Devices view**.



I/O attributes for each object associated with the OPCClient or scan group (depending on which is selected in the **IO Devices view**) are shown in the **IO Device Mapping view**.

2 In the **IO Device Mapping view**, the resulting reference will be in the form:
`OPCClientName.ScanGroupName.ObjectName.AttributeName`

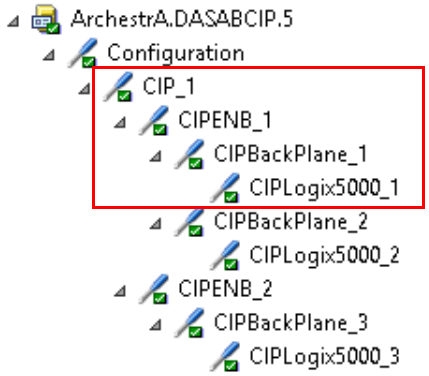
Example (**Resulting Reference** column):

`OPCClient_01.SG1.UDO_01.Attribute_001`

3 Do not edit the **Device.ScanGroup Override** column (leave it blank). This leaves the DI Object and scan group assignment as defined in the **IO Devices view**.

4 Enter the fully qualified field reference in the **Object.Attribute Override** column. For example:

```
CIP_1.CIPENB_1.CIPBackPlane_1.CIPLogix5000_1.UDO_01.
   Attribute001
```



5 Apply the override. The resulting reference will be displayed. For example:

```
OPCClient_01.SG1.CIP_1.CIPENB_1.CIPBackPlane_1.
   CIPLogix5000_1.UDO_01.Attribute001
```

Writing and Editing Scripts

You can write scripts to extend and customize your objects. You can write scripts that run commands and logical operations based on specified criteria being met. For example, a script starts when a key is pressed, a window is opened, or the value of an attribute changes.

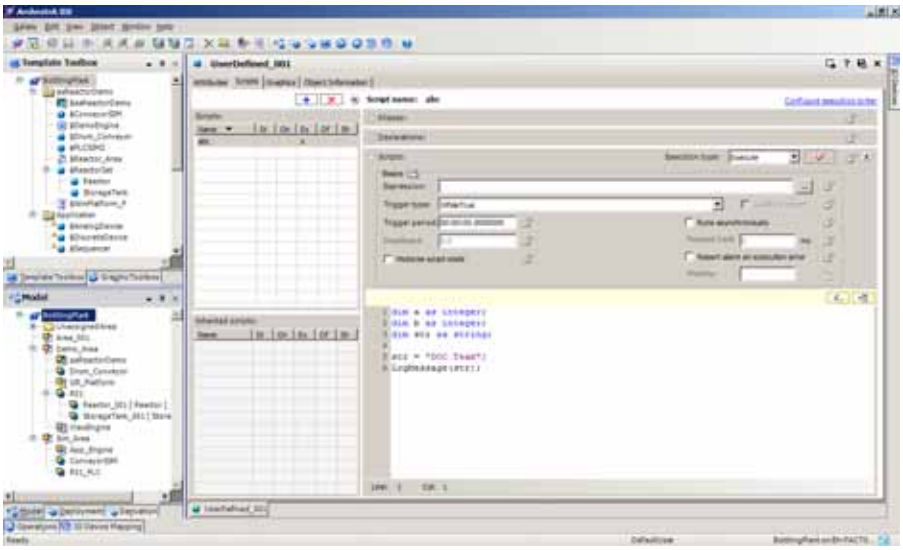
Using scripts, you can create a variety of customized and automated system functions. A script adds behavior that runs when the object that contains the script is deployed and the object is either:

- On scan in the run-time environment or
- Changes scan or start/shutdown state

A script typically runs based on attributes of the object that contains it, but can be started by another script based on changing values of attributes of more than one object.

When a script condition is true, the script runs at least once immediately. The maximum length of a script trigger period is 49-days. A script never runs if the trigger period exceeds 49-days.

For specific information about writing scripts, including the scripting language, syntax, commands, and using .NET, see the *Wonderware® Application Server Scripting Guide.*



For more information about the scripts page, see "About the Scripts Page" on page 81.

Important: You cannot pass attributes as parameters for system objects. Instead, use a local variable as an intermediary or explicitly convert the attribute to a string using an appropriate function call when calling the system object.

About Scripts

The following characteristics apply to the scripting environment:

- Script text has no length limitations.
- Selecting a script function from the **Script Function Library** dialog box adds it and its syntax to the script text where you can edit it.
- You can save a script with syntax errors, but the object cannot be deployed until you correct the script syntax errors.
- You can validate your scripts before using them. This helps you avoid syntactically correct but semantically incorrect combinations such as two statements declaring the same variable. Variables can be declared only one time in a single block.
- You can change the name of a script at any time by renaming it in the Object Editor.
- In the run-time environment, a script execution error stops the script's current execution. Script execution is retried on the next AppEngine scan.

Script Execution

The existence and execution order of scripts associated with an object are inherently locked at each stage of development in the template, derived template, and instance. For example, a set of scripts associated with a template are treated as an ordered block in the **Configure Execution Order** dialog box when configuring execution order in a next-generation derived template.

New scripts in the derived template can be ran in any order before and after the template's block of scripts. The derived template's execution order is treated as a block in any downstream derived templates or instances. Scripts cannot trigger any faster than the scan period of the AppEngine the script is associated with or faster than the scan period of the AppEngine that hosts the object that the script is associated with.

Scripts run in one of two modes:

- Synchronous scripting mode is the default for running scripts in the run-time environment. This mode runs scripts in order while an object is running on scan.

- Asynchronous scripting mode is a group of scripts running on the same, lower priority execution thread. These scripts only support Execute triggering and run independently from each other. Set the maximum number of independent threads in the AppEngine configuration editor. To use either scripting mode, you must select **Execute** as the **Execution Type** in the **Scripts** area on the **Scripts** page.

To create and associate a script with an object

1 Add a script. On the **Scripts** page of the Object Editor, click the **Add** button. A script is added to the **Scripts Name** list.

2 Type a name for the script and press **Enter**. Script names can be up to 32 alphanumeric characters, including periods. At least one character must be a letter.

Note: For detailed information about each item on the **Scripts** page, see "About the Scripts Page" on page 81.

3 Select a trigger that starts the script in the run-time environment.

Execution Type triggers include: Startup, On Scan, Execute, Off Scan and Shutdown.

- If you select **Startup**, **On Scan**, **Off Scan**, or **Shutdown**, the **Basics** group is unavailable. The script is triggered when the object starts, goes on scan, goes off scan, or shuts down.

  If you select **Execute**, the **Basics** group is available.

- If you selected **Execute** as the script trigger, select a **Trigger Type**. Depending on the type selected, you are required to enter an **Expression** and/or **Trigger Period** and **Deadband** values. When the combination of **Expression**, **Trigger Type**, **Trigger Period** and/or **Deadband** is satisfied in the run-time environment, the script starts running. See the following table for more information.

  The **Trigger Period** format is as follows:
  Hours:Minutes:Seconds:Milliseconds
  For example, 3 hours, 5 minutes, and 10.5 seconds is:
  03:05:10.5000000

Expressions are limited to one language statement in length and calling only synchronous mode script functions. Avoid using script functions with side effects in expressions because subtle behaviors can occur.

Trigger Type	Description
Periodic	When the object containing the script is going On Scan, a Periodic script evaluates its expression at the next scheduled scan period of the AppEngine. The script then runs periodically at the trigger interval specified in the **Trigger Period** box. A time interval of zero (0) starts the script during every scan. This trigger does not require an expression.
While True	When the object containing the script is going On Scan, a While True script evaluates its expression at the next scheduled scan period of the AppEngine. The script runs if true and then periodically thereafter at the trigger interval. The script continues running as long as the **Expression** value evaluates to true. A Trigger Period is required. Zero (0) evaluates the expression at the AppEngine scan period and non-zero means the expression is evaluated at the specified time interval.
On True	When the object containing the script is going On Scan, an On True script evaluates its expression at the next scheduled scan period. The script starts at the transition between the expression going from false to true.
On False	When the object containing the script is going On Scan, an On False script evaluates its expression at the next scheduled scan period. The script starts at the transition between the expression going from true to false.

Trigger Type	Description
Data Change	Scripts run when the value or quality of the expression changes. The expression must evaluate to a single, non-arrayed value of the following types: integer, real, time, elapsedtime, string, double, Boolean, custom enumeration and quality. To allow execution based on quality, select the **Quality changes** check box. A deadband can be specified for all data types. Deadband units for time and elapsedtime types are milliseconds. Deadband is always ignored for strings because any change (even from "ABC" to "abc") is considered a change. Only major changes in quality (from Good/Uncertain to Bad/Initializing or vice versa) are considered changes. After the object is put on scan, Data Change-triggered scripts start running at the AppEngine's next scan period and then at each subsequent scan period in which the value or quality changes.
While False	When the object containing the script is going On Scan, a While False script evaluates its expression at the next scheduled scan period, runs if false, and then periodically thereafter at the trigger interval. The script runs as long as the **Expression** value evaluates to be false. A Trigger Period is required. Zero (0) evaluates the expression at the AppEngine scan period and non-zero means the expression is evaluated at the specified time interval.

4 Select one or more of the following:

- Set the **Runs Asynchronously** and associated **Timeout Limit** parameters, as needed.
- Select **Report Alarm on Execution Error** and set a **Priority** for the alarm if you want the alarming function to alert you if a script execution failure occurs.
- Select **Historize Script State** to store the state of the script in your application's historian.

5 In the **Declarations** area, type variable declarations about the script you are writing.

6 Set aliases for the reference strings in the **Aliases** area. This can simplify the script code and allows script code to be created and locked at a template level using alias names. When an individual instance of that template is created, you can link external attributes to the alias names.

In the **Aliases** area, click the **Add** button to add a new alias. An alias is added to the list. The name is shown in edit mode. Double-click the **Reference** entry, and enter a reference string for the alias. You can also click the **Browse** button at the end of the **Reference** block to open the Attribute Browser for easy selection of an object's attributes.


7 Write the script in the **Script Creation** box. Use the **Display Script Function Browser** and **Display Attribute Browser** buttons to help you insert script functions and object attribute references in your script. For help with the specific commands and syntax, see Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

Click the **Validate Script** button to validate if your script contains any syntax errors.

8 Order the scripts. If you have more than one script associated with a single object, click **Configure Execution Order**. Ordering does not apply to asynchronous scripts. If a script is added to an instance derived from a template that contains scripts, the new script automatically defaults to running after the derived scripts.

9 When you are done creating your script and setting its execution triggering parameters, save and close the Object Editor.

Locking Scripts

When you lock a script in a template, the following rules apply:

- The name of a script and its existence is implicitly always locked. This means:
  - You cannot delete the script in derived objects.
  - You cannot change the name of the script in derived objects.
  - If you rename the script in the template, the name changes in all derived objects.
  - You can delete a script in the template after you create derived objects. The script disappears from the derived objects.
  - You can add a script to the template after you create derived objects. The script appears in the derived objects.

  - You can add scripts to derived objects. Adding scripts to derived objects does not affect parent object scripts.

- You can lock or unlock the script text in a template. There is script text for **Declarations, Execute, Startup, Shutdown, On Scan** and **Off Scan**. You cannot separately lock each script in the script editor. You use a single group lock to lock or unlock all at once.

  After you lock a script, derived templates and instances cannot modify any of the script text.

- When the script text is locked in a template, the alias names are automatically locked. The alias references are never locked. Locking of aliases is not specified separately.

  Locking aliases means the entire list of alias names is locked, including the number of items in the list. You cannot add new alias names in derived templates or instances when the alias list is locked. The alias references are always editable in derived templates and instances even when the entire list of alias names is locked. This is the primary objective of aliases.

- The script description, runs asynchronous flag, expression, trigger type, trigger period, deadband and execution error alarm are individually lockable and can be locked separately from the script text. A group lock is provided for this group of attributes.

- When you add a script to a template, all properties of the script are editable.

- When you add a script to an instance, all properties of the script are editable except for the lock properties. A lock is never editable in an instance.

Important: An expression typically uses attribute references. To lock the expression and the associated script in a template, use aliases in both the expression and the script. This allows you to specify the attributes that the aliases point to on a per instance basis while the script code is locked.

The following rules apply to the derivation behavior of locked script attributes:

- If an attribute is locked in a template, then all templates and instances derived from the template share the copy of the value of the locked attribute. A change to the value is only allowed in the template that locked it. The change propagates to all derived templates and instances.

  For scripts, locking an attribute of the script, such as its script text or execution type, in a template means all derived templates and instances point to that locked attribute. Future changes to that locked attribute's value, such as modifying the script text, propagate and appear in all derived templates and instances.

If instances are deployed, they are marked pending update status. After they are redeployed, the change to the locked attribute in the template exists in the deployed instance.

- If an attribute is not locked in a template, then all templates and instances derived from that template receive their own copy of the value of that unlocked attribute. A change to that unlocked value is allowed in derived templates and instances because they own their own copy. Any change to the unlocked attribute value in the template does not propagate to any derived template or instance.

  An unlocked attribute in a script (such as expression or script order) in a template means that all derived templates and instances have their own copy and the value of that unlocked attribute can change. Future changes to that locked attribute's value (for example, modified expression) in the template does not propagate to any derived template or instance. If instances are deployed, their status does not change to pending update. Redeploying them does not cause the value to change in the deployed instance.

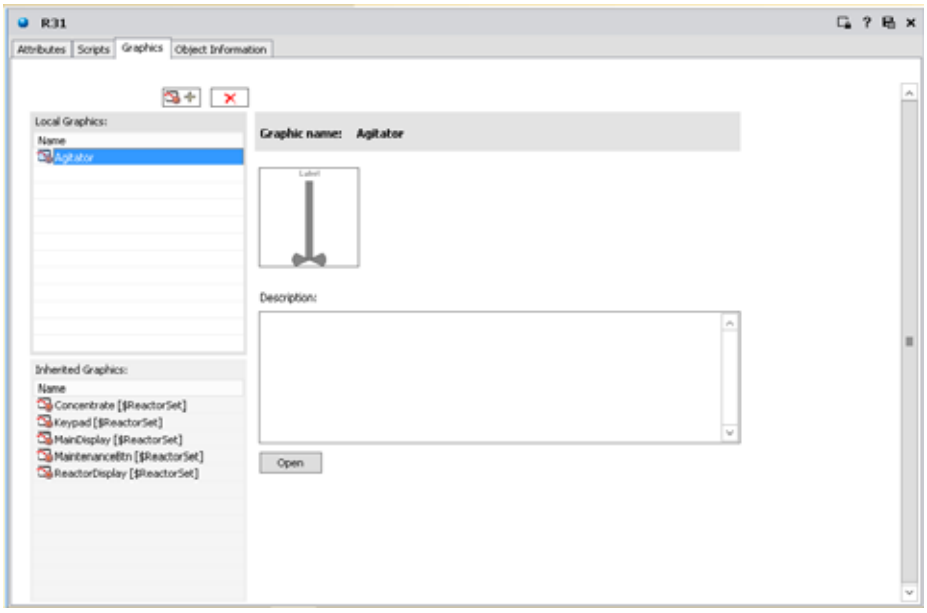
Creating and Working with Graphics

Use the **Graphics** page to add, modify, rename or delete local graphics; or to view inherited graphics. You must have the derived template or object instance checked out in order to add, modify, or delete local graphics; otherwise, you can only view local graphics.

You can view graphics inherited from parent templates and object instances. You cannot add, modify, or delete inherited graphics. This means that in order to edit an inherited graphic, you must check out the derived template or object instance where the graphic is local.

You can only add graphics to derived templates and object instances. You cannot add graphics to a base template.

On the **Graphics** page, use the **Open** button to start the Symbol Editor for the selected graphic.



Adding Graphics

You can add graphics to an object.

To add a graphic symbol to the object

1. On the **Graphics** page of the Object Editor, click the **Add** button. A graphic name is added to the **Name** list.
2. Type the new local graphic symbol name.
3. In the **Description** box, type a description for the graphic symbol being added.
4. Click **Open**. The graphics tool box opens. For instructions on using the Symbol Editor, see Chapter 1, "About Creating and Managing ArchestrA Symbols," in the *Creating and Managing ArchestrA Graphics User's Guide.*

Modifying Graphics

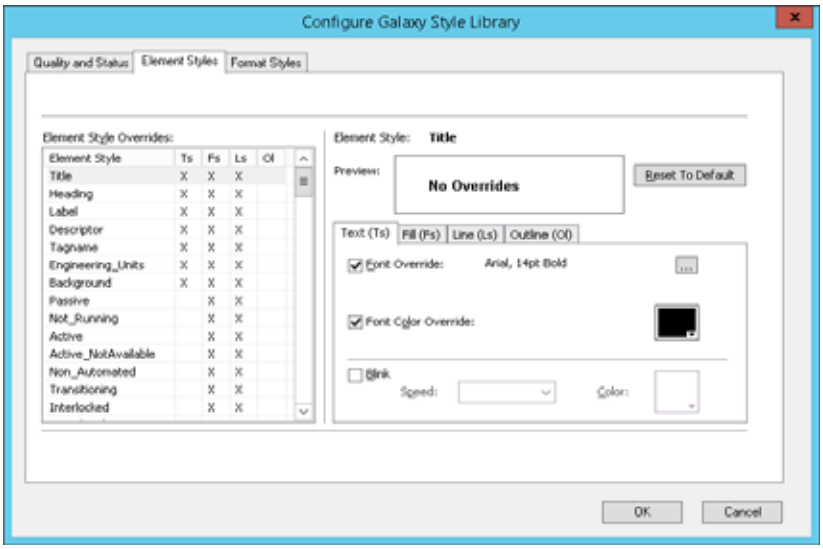
All modifications made to graphic symbols referenced by other objects are visible in the referenced objects.

To modify a graphic symbol

1. On the **Graphics** page of the Object Editor, click the **Name** of the graphic symbol to be modified.
2. Click **Open**. The graphics tool box opens, showing the selected graphic symbol.
3. Make changes. see Chapter 1, "About Creating and Managing ArchestrA Symbols," in the *Creating and Managing ArchestrA Graphics User's Guide.*

Modifying Graphics with Element Styles

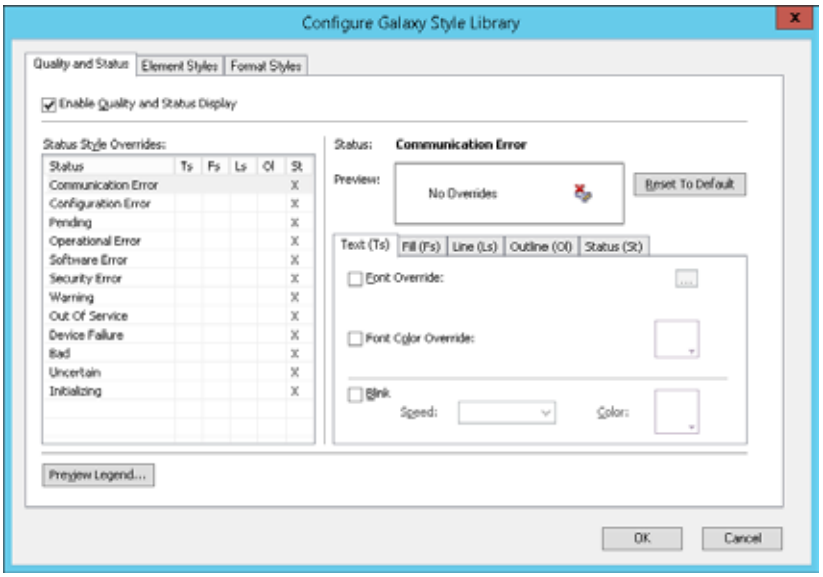
Element Styles define one or more visual properties such as fill, line, text, blink, outline, and status properties of graphic elements. You can apply an Element Style to a graphic element to set preconfigured visual properties defined in that Element Style. Element Styles establish consistent visual standards for symbols.



For more information about Element Styles, see "Managing Galaxy Style Libraries" on page 55 and Chapter 7, "Working with Element Styles," in the *Creating and Managing ArchestrA Graphics User's Guide.*

Using Quality and Status Styles

In the same way that Element Styles can be used to preconfigure consistent visual properties of graphic elements, Quality and Status styles can be preconfigured to provide visual elements specific to quality and status states.



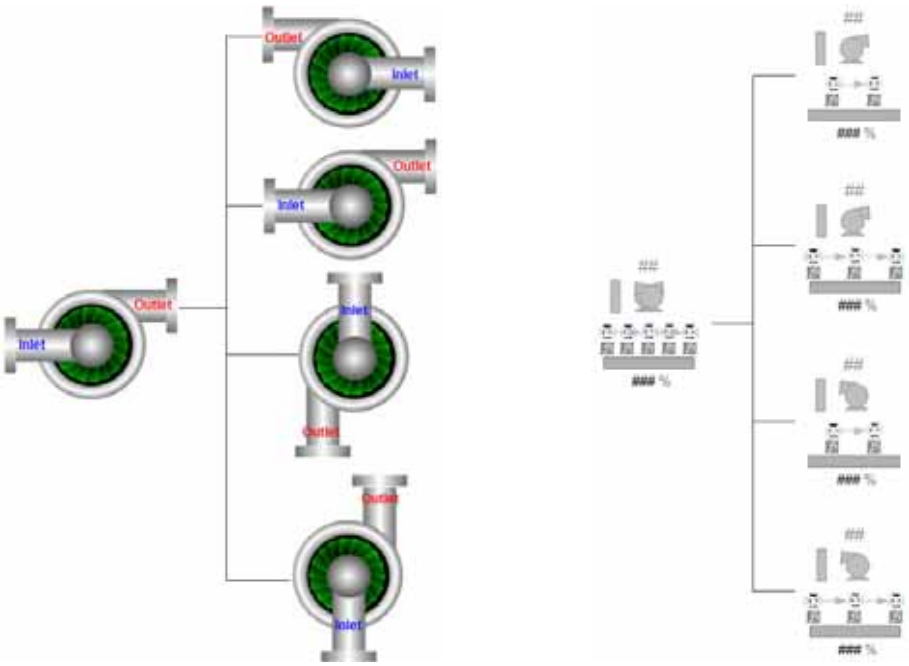
You can configure text, fill, line, outline, blink, and associated display icons for the following Status styles:

- Communication Error
- Configuration Error
- Pending
- Operational Error
- Software Error
- Security Error
- Warning
- Out of Service
- Device Failure
- Bad
- Uncertain
- Installing

Using Symbol Wizards

The ArchestrA Symbol Editor includes the Symbol Wizard Editor, which can be used to create symbols containing multiple visual or functional configurations called Symbol Wizards. A Symbol Wizard can be embedded into managed InTouch applications like standard ArchestrA symbols.

A Symbol Wizard's configuration is selected to meet the requirements of an application. Incorporating multiple configurations in a single Symbol Wizard reduces the number of symbols needed to develop an ArchestrA application.



The Symbol Wizard Editor can create Symbol Wizards from traditional ArchestrA Symbols and Situational Awareness Library symbols. Both types of symbols are located in the Graphic Toolbox in a separate set of folders. Symbol Wizards are saved in the IDE's Graphic Toolbox and are not associated with any specific ArchestrA object template or object instance. Except for the ability to select a specific symbol configuration, Symbol Wizards behave like standard ArchestrA Symbols.

Situational Awareness Library symbols provide an additional benefit of including defined properties and their associated attributes to more easily create configurations. Some Situational Awareness Library symbols include a Type property to assign a specific function for a symbol configuration. For example, a meter symbol can be configured to represent a thermometer, a pressure meter, or flow meter by simply changing the attribute assigned to the Type property.

Typically, the process of creating and embedding a Symbol Wizard in an application requires the involvement of a Designer and a Consumer. A Designer creates Symbol Wizards using the Symbol Wizard Editor. A Consumer selects a configuration of a Symbol Wizard and embeds the instance of the symbol into managed InTouch applications.

Creating Symbol Wizards

A Designer uses the Symbol Wizard Editor to define the various required symbol configurations based on a set of rules and symbol layers. A Designer defines a set of layers, which are used to group a set of graphic elements, custom properties, and named scripts. Graphic elements and other symbol properties can be assigned to no layers or multiple layers. Graphic elements that are not assigned to any layer always appear in all symbol configurations.

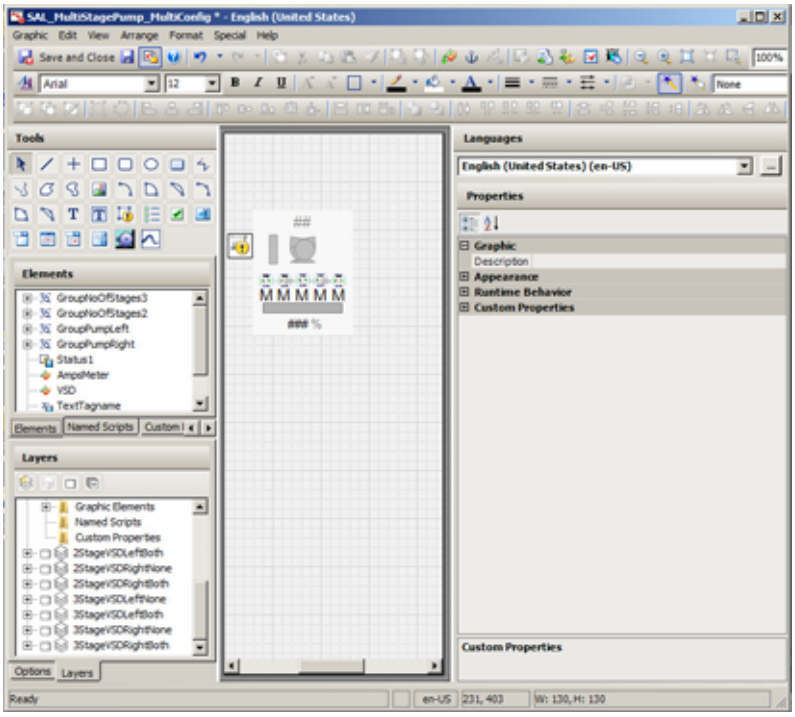
A Designer can create a rule for each layer that defines the conditions when the layer is included in a symbol configuration. Rules are assigned with Choice Groups, Choices, and Options. A Designer selects a configuration to be the symbol default that appears when the symbol is embedded in a managed InTouch application.

After creating all symbol configurations, the Designer verifies how each configuration of a symbol using the Symbol Wizard Preview. Designers set values in the **Wizard Options** view to verify that each configuration appears as designed based on the layer rules set for the symbol.

When a Symbol Wizard is finished, the Designer saves it to the Galaxy library so that it is available for use in managed InTouch applications. For more information about creating Symbol Wizards, see Chapter 17, "Working with Symbol Wizards," in the *Creating and Managing ArchestrA Graphics User's Guide.*

Embedding Symbol Wizards into an Application

Symbol Wizards are stored in a Galaxy library just like standard ArchestrA Symbols. When a Consumer selects a Symbol Wizard and embeds it into a managed InTouch application, the Symbol Wizard's default configuration is selected. The Consumer can change the symbol's configuration by changing the options from the Symbol Wizard's **Wizard Options** section of the **Properties** view. Depending on the selected configuration, there can be additional configuration-related properties that can be selected by the consumer.



After selecting a symbol configuration and changing any properties, the Consumer saves the Symbol Wizard so that it can be embedded into a window from WindowMaker.

While the InTouch application is running, the Symbol Wizard appears as the configuration selected by the Consumer. A Symbol Wizard configuration cannot be changed during run-time.

For more information about how to embed Symbol Wizards into a managed InTouch applications, see Chapter 17, "Working with Symbol Wizards," in the *Creating and Managing ArchestrA Graphics User's Guide*.

Renaming Graphics

You can rename a graphic symbol.

To rename a graphic symbol

1. On the **Graphics** page of the Object Editor, click the **Name** of the graphic symbol to be renamed.
2. Type the new name, and press **Enter**. The new name is saved.

Deleting Graphics

Caution: Deleting a graphic symbol with embedded references breaks the links to their related objects. “Symbol Not Found” appears when you open objects whose embedded graphic symbols have been deleted.

To delete a graphic symbol

1. On the **Graphics** page of the Object Editor, click the **Name** of the graphic symbol to be deleted.
2.  Click the **Delete** button. The **Delete** dialog box appears. The left pane lists the graphic symbol selected for deletion. The right pane shows all embedded references to the selected graphic.

   Click **Yes**.

Chapter 6

Deploying and Running an Application

You can deploy and test your objects at any time during development. When you are ready to test or run the application in production, you deploy the Galaxy.

You can see what your application looks like in the Deployment view or the Model view. Both views show you the structure of your application. For more information, see “Using IDE Application Views” on page 28.

Planning for Deployment

Deploying your Galaxy copies the objects from the development environment to the run-time environment. This makes your objects “live” and functional.

Until you deploy your IDE configuration environment to the run-time environment, changes you make in the IDE do not appear in the run-time environment. To see run-time data associated with your objects, use Object Viewer or InTouch. For more information about using Object Viewer, see the *Object Viewer User's Guide*.

Important: Do not configure nodes with Dynamic Host Configuration Protocol (DHCP). All nodes in your Galaxy communicate with each other by using both IP addresses and node names.
Use static IP addresses for all Application Server nodes.

Deploying Objects from the IDE to Run Time

Objects deploy from the configuration environment to the run-time environment as follows:

IDE Config Environment		Object Viewer Run-time environment
Galaxy database	→	[Does not exist in run-time environment]
Templates	→	[Does not exist in run-time environment]
Instance objects	→	Instance objects *[Run-time configuration and behavior]*
Security: General permissions	→	[Does not exist in run-time environment]
Security: Operational permissions	→	Run-time permissions to acknowledge alarms and modify attributes
Scripts configuration	→	Scripts execution
Alarms configuration	→	Alarms generate and acknowledge
History configuration	→	History Logs *[Wonderware Historian]*

Allocating Run-time Resources

Distribute components across platforms and engines as you develop your IDE configuration environment. This will help to balance the I/O, memory, and CPU burden across computing resources at run time.

The operating system determines engine assignment to individual CPU cores, and attempts to load balance across CPU and other computing resources. If you need to change how resources are allocated to improve load balancing, undeploy the objects and reassign them in the IDE configuration environment.

If you are using redundant AppEngines, you can control how objects are loaded at runtime. For more information, see “CPU Load Balancing” on page 432.

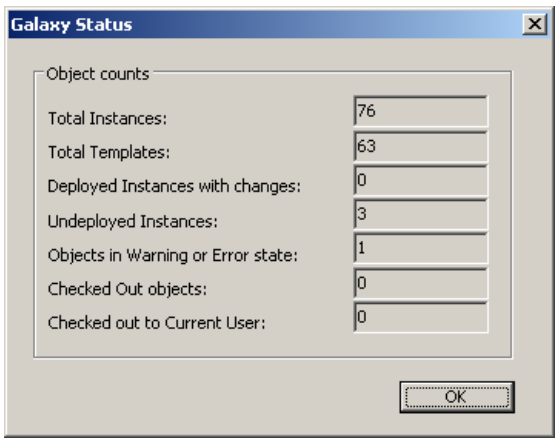
Determining Galaxy Status

You can see an overview of the condition of your Galaxy before you deploy. This lets you know if you have objects that are in warning or error status.

To determine the status of a Galaxy

1 Connect to the Galaxy.

2 On the **Galaxy** menu, click **Galaxy Status**. The **Galaxy Status** dialog box appears.



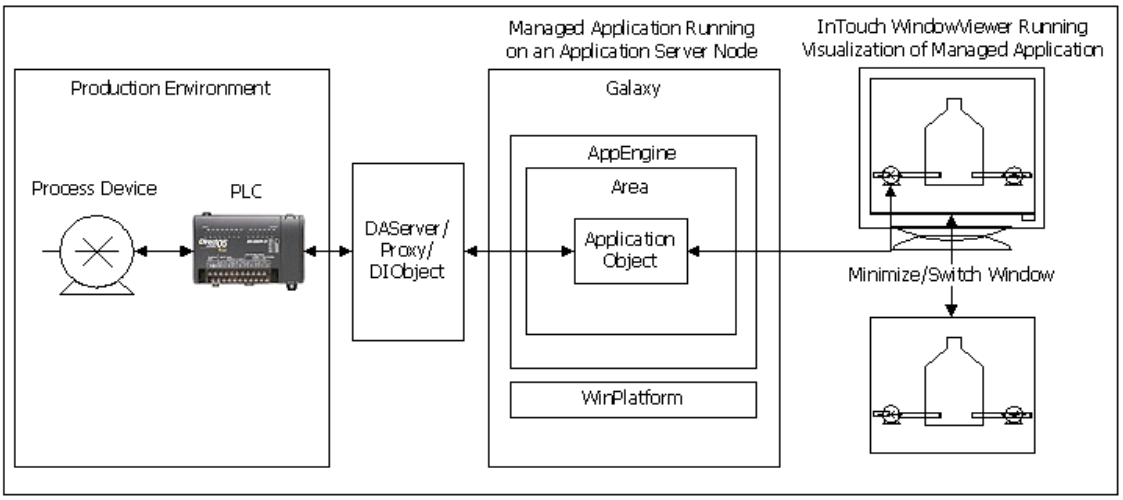
The **Galaxy Status** dialog box shows the counts of total instances, total templates, deployed instances with changes, undeployed instances with changes, objects that have an error or warning state, objects that are checked out, and object you have checked out.

3 Click **OK**.

Configuring Advanced Communication Management

Advanced Communications Management minimizes network traffic and CPU usage of a DI Object and DAServer when a particular DI attribute is currently subscribed to, but its value is not currently shown. For example, scanning for updated values from a DI attribute representing pump RPM is suspended when an operator minimizes the application window containing a pump graphic containing attribute references that subscribe to the DI Object's pump RPM attributes.

Note: This dependence on the attribute being referenced by a running application is only the case when the attribute subscription is not already active due to configuration for history, alarming, or scripting. For example, if an attribute is configured for history, the subscription to the field device will always be active regardless of what window is active in InTouch, and regardless of whether InTouch is running at all.



When the items on the current window are no longer active because the window is minimized, a Suspend operation is performed for each item. Suspending scanning causes data updates to stop flowing to the application without deleting the subscriptions for each item. This reduces the overhead of unsubscribing and subscribing again to items when windows are minimized and restored.

In Advanced Communication Management, applications monitor the number of outstanding references to attributes that are updated with values from an external source object. When an attribute's external reference count reaches zero, the application sends a Suspend message to the source object requesting it to suspend updates. For example, a UserDefined object named PumpRPM has an attribute named RPM, which has an integer data type. The input source of RPM is plc1.n7:11, where plc1 is an ABPLC DI object.

When there are no active subscriptions to ud1.RPM, the subscription between input source and PumpRPM is suspended. When a client like ObjectViewer or a managed application running in WindowViewer subscribes to ud1.RPM, an Activate message is sent to the DI object to start sending data between the plc1 object and the PumpRPM object. This enables ObjectViewer or a managed client application running in WindowViewer to receive current plc1 data.

Unlike defined static attributes, some scan groups contain *dynamic* attributes. Dynamic attributes within a scan group provide the means for a running application's objects to dynamically subscribe to data from an external device. Dynamic attributes are created, activated and added to the scan group using ArchestrA object attributes or InTouch indirect tags and scripts that include the IOSetRemoteReferences function.

When a dynamic attribute is poked, the time stamp is updated to the timestamp passed in with the value if available, or the current time provided by the hosting AppEngine is used. If the data provider passes in a Value, Time, Quality (VTQ) triplet of data in the callback, the time stamp is associated with the value and quality used to update the attribute.

To enable Advanced Communication Management, you must:

- Select Advanced Communication Management for your Galaxy.
- Set the scan mode for each scan group that belongs to your device integration objects within the Galaxy.

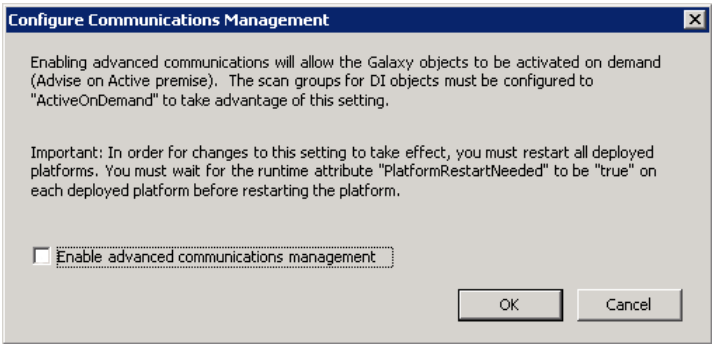
Selecting Advanced Communication Management

Advanced Communication Management is a Galaxy-wide configuration setting that you set from the ArchestrA IDE. Advanced Communication Management is disabled by default.

When upgrading a Galaxy or restoring a Galaxy from a .cab file, the configuration setting that was previously selected is retained.

To configure Advanced Communication Management

1. Connect to the Galaxy.
2. On the **Galaxy** menu, click **Configure** and then click **Communications Management**. The **Configure Communications Management** dialog box appears.



3. If necessary, select **Enable advanced communication management** and click **OK**.

Note: Any deployed platform and application engines must be restarted when the Advanced Communication Management state changes. Restarting should be done by means of the SMC.

Configuring Scan Modes

A scan group is a collection of object attributes with associated data items. Scan group attributes reflect I/O points in the address space of Data Access Server whose values are polled from devices at a common update interval. The update interval for each scan group is inherited from the scan groups configured in the source DIObjects.

You can configure scan groups for OPCClient, InTouchProxy, DDESuiteLinkClient, and RedundantDIObject device integration objects. Within each scan group, you can specify data items to use as the object attributes. At run time, scan group items are updated with the latest values from the OPC DAServer according to the update interval.

The OPCClient, RedundantDI, DDESuiteLinkClient, and InTouchProxy objects contain a Scan Mode attribute that can be set for each scan group. The assigned scan mode value determines if objects are continuously updated with current data when an operator minimizes or closes the application window.

You can assign three different scan modes to a scan group:

- **Active**
  - Items are deleted and added to the scan group as requested (referenced) by the clients.
  - Items that exist when a scan mode change occurs are not deleted unless the previous mode was ActiveAll and the items are no longer referenced.
  - The Active scan mode polls all points that are referenced, whether active or inactive.
  - In Active scan mode an attribute is always in the active state. When the last reference to the attribute is unregistered/unadvised, the attribute is deleted.
- **ActiveAll**
  - Scanning is continuous and dynamic attributes are never removed when unsubscribed.
  - Items are activated by client requests, but continue to exist even after the client are not interested in them anymore.
  - Items are not removed when the scan mode changes.
  - The scan group polls the field device for all points irrespective of whether they are currently active, inactive, or even subscribed to by items.

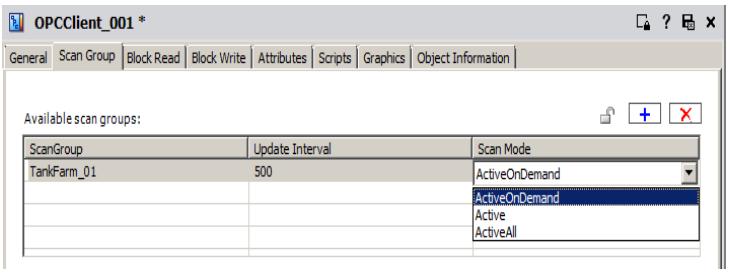
- **ActiveOnDemand**
  - When the scan mode is configured as ActiveOnDemand, attributes that are not actively being referenced by any client or object are made Inactive and are not scanned.
  - New Items are deleted and added to the scan group as requested (referenced) by clients.
  - Items that exist when a scan mode change occurs are not deleted unless the previous mode was ActiveAll and the items are no longer referenced.
  - ActiveOnDemand scan mode polls the field device for currently active items only. Inactive items are not scanned.

The following table shows scan mode scan states for dynamic attributes based on whether items are referenced or not.

Dynamic Attribute State	Scan Mode Active	Scan Mode ActiveAll	Scan Mode ActiveOnDemand
Reference/ Some active	Scan	Scan	Scan
Reference/All Inactive	Scan	Scan	No Scan
Not Referenced	Delete	Scan	Delete

To set a scan mode for a scan group

1. Edit the device integration object.
2. Show the editor page containing scan groups by completing one of the following:
   - Click the **Scan Group** tab if you are editing a RedundantDIObject or OPCClient object.
   - Click the **Topic** tab if your are editing a DDESuiteLinkClient object.
   - Click the **Items Configuration** tab if your are editing an InTouchProxy object.



3 If necessary, add a scan group by clicking the plus sign box above the list of available scan groups.

4 If necessary, set the length of the scan group update interval in milliseconds. The default update interval is 500.

5 Double-click in the **Scan Mode** box of the scan group to show a drop-down list of available scan modes. ActiveOnDemand is the default scan mode.

6 Select a scan mode from the list.

7 Save your changes to the device integration object and exit from the editor.

Deploying Objects

You deploy object instances for three reasons:

- Testing.
- Placing the application into production to process field data.
- Updating an existing application with changes you made.

When you are ready to deploy, make sure the following conditions are met:

- Bootstrap software is installed on the target computers.
- The objects being deployed are not in an error state in the Galaxy database.
- You created, configured, and checked in objects to the Galaxy.
- Objects are assigned to a host.
- The object's host is already deployed. A cascade deploy operation, which deploys a hierarchy of objects, deploys all objects in the correct order. This deploys an object's host before the object is deployed.
- Any associated script libraries are ready for use on the target computer. For more information, see "Importing Script Function Libraries" on page 121.

Note: DINetwork objects have specific configuration limits. For example, whether more than one object can be deployed to a single WinPlatform. The IDE does not check for these limits. For more information, see the help file for the DINetwork object for specifics on configuration limits.

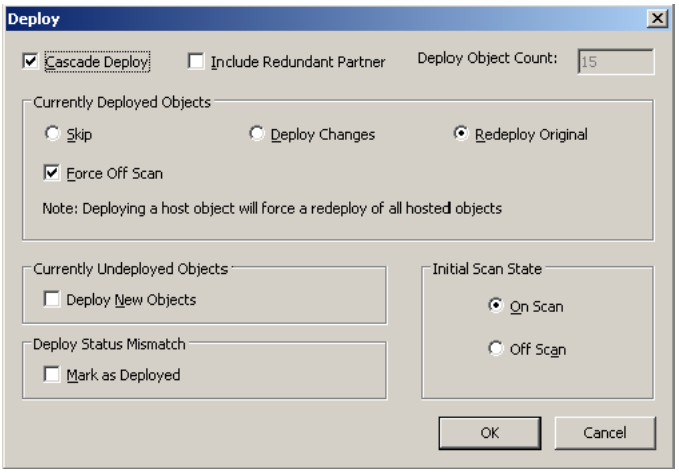
You can tell if you have objects that need to be deployed by looking at the icons next to the objects. Deployment status icons include:

	Not deployed
	Deployed
	Deployed, but pending configuration changes exist that have not been deployed.
	Deployed, but software modifications exist that have not been deployed.
	Applies only to redundant AppEngines. An AppEngine is undeployed, but its redundant pair is deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, but its redundant pair is not deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, its redundant pair is not deployed, and pending configuration changes exist that have not been deployed.
	Applies only to redundant AppEngines. An AppEngine is deployed, its redundant pair is not deployed, and software modifications exist that have not been deployed.
[No icon]	Good. No additional icon is displayed for deployed objects running on-scan or off-scan with a good status.
	Warning
	Error. The object in an Error state and cannot be deployed.
	InTouchViewApp application files are being asynchronously transferred to the target node. This icon is normally visible for only a few moments at the end of an InTouchViewApp deployment operation, unless the object is deployed on a slow network. This icon is larger than the other icons and completely replaces the original while it is being shown.

To deploy an object

1 Select the object in an **Application view**.

2 On the **Object** menu, click **Deploy**. The **Deploy** dialog box appears.



3 Select one or more of the following:

- **Cascade Deploy**: Select this check box to deploy the object selected for deployment as well as any objects it hosts. This option is selected by default if the object is a host. If you are deploying an individual host object, clear the check box. Objects being deployed across multiple platforms are deployed in parallel.
- **Include Redundant Partner**: Select this check box to also deploy an AppEngine's redundancy partner object. This option is selected and unavailable when the redundant engine has pending configuration changes or software updates.

4 In the **Currently deployed objects** area, select one or more of the following options. These options are not available if the selected object has not been deployed before.

- **Skip**: If one of the objects you are deploying is currently deployed, selecting **Skip** makes no changes to the already-deployed object.
- **Deploy Changes**: If one of the objects you are deploying is currently deployed, this option updates the object in question with new configuration data. The run-time state from the run-time file is preserved and the state is modified with any changes.

- **Redeploy Original**: If one of the objects you are deploying is currently deployed, this option deploys the same version as previously deployed. For example, use this option to redeploy an object that is corrupted on the target computer.
- **Force Off Scan**: If one of the objects you are deploying is currently deployed, this option sets the target object to off scan before deployment occurs.

5 In the **Currently undeployed objects** area, select the **Deploy New Objects** check box to start a normal deployment.

6 In the **Deploy Status Mismatch** area, select the **Mark as Deployed** check box to mark the object as deployed in the Galaxy. A mismatch happens when the object is previously deployed to a target node, but the Galaxy shows the object is undeployed. Clear this option to redeploy the object to the target node.

7 In the **Initial Scan State** area, select one of the following:

- **On Scan**: Sets the initial scan state to on scan for the objects you are deploying. If the host of the object you are deploying is currently off scan, this setting is ignored and the object is automatically deployed off scan. When you are deploying multiple objects, the deploy operation deploys all of the selected objects "off-scan." After all of the objects are deployed, the system sets the scan-state to "on-scan."

  Objects can run only when both the host/engine is "on scan" and the object is "on scan." If either the host/engine or the object is "off scan," the object cannot run.

  Always deploy Areas to their host AppEngines on scan. Because Areas are the primary providers to alarm clients, deploying Areas off scan results in alarms and events not being reported until they are placed on scan.

- **Off Scan**: Sets the initial scan state to off scan for the objects you are deploying. If you deploy objects off scan, you must use the ArchestrA System Management Console Platform Manager utility to put those objects on scan and to function properly in the run-time environment.

Note: The System Management Console controls the state of the host/engine. The ObjectViewer controls the state of the objects.

The default scan setting is set in the **User Default** settings in the **Configure User Information** dialog box. For more information, see "Configuring User Information" on page 39.

8 Click **OK** to deploy the objects. The **Deploy** progress box appears. If you see error messages, see "Deployment Error Messages" on page 200. When the deploy is complete, click **Close**.

Deployment Error Messages

If the object being deployed has configuration problems that were not known during configuration, the **Deploy** progress box shows messages.

The **Progress** dialog box also shows the affected instances and any error and warning messages. The target object can take any actions necessary to achieve a valid state, including changing attribute values provided during deployment.

WinPlatforms are the only objects whose configuration designates its deployment location. Deployment problems unique to WinPlatforms are the following:

- The target computer could not be found on the network. Ensure the WinPlatform was configured properly and the target computer is properly connected to your network.
- Another WinPlatform is deployed already to the target computer. Resolve this problem by undeploying the existing WinPlatform before deploying the new one.
- The target platform is running on the old version of the product. To resolve this problem, the user must upgrade the remote platform.
- If a WinPlatform object is deployed on a slow network and it does not respond to the IDE before the 30-second message time-out, a communication error occurs and the object is shown as not deployed. You may need to adjust the message time-out for the WinPlatform object to accommodate the slow network speed.

Publishing Managed InTouch Applications

You can publish a managed InTouch application using the ArchestrA IDE. As part of this process, you can create a compressed, self-extracting package file that contains all relevant files and set-up procedures to install an InTouch application on another computer. For more information, see the *InTouch HMI Application Management and Extension Guide.*

Redeploying Objects

Redeploying is similar to deployment. While you are testing, you frequently redeploy your application to see changes you make. The redeploying process undeploys the object and then deploys it back.

You may have an object with a Pending Update deployment state. The Pending Update state means the object changed since it last deployment. When you deploy those changes, the new object is marked as the last deployed version in the Galaxy.

To redeploy

1 On the **Object** menu, click **Deploy**.

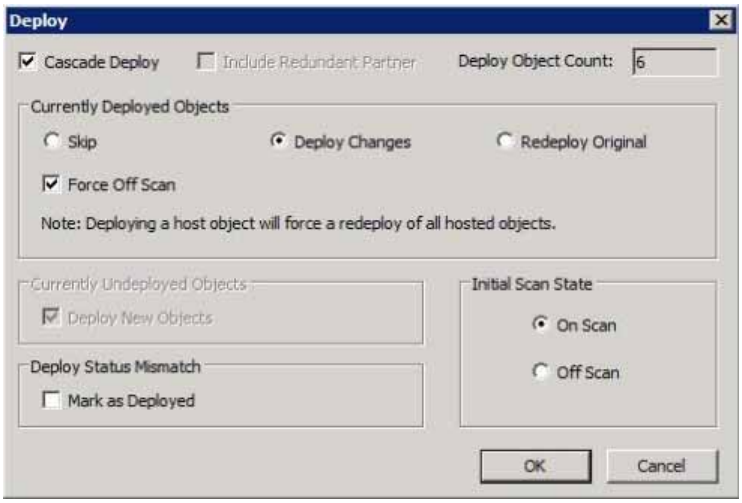
2 Follow the procedure for “Deploying Objects” on page 196.

Redeploying the WinPlatform Object

You can redeploy a WinPlatform object without undeploying and redeploying the objects hosted on WinPlatform. When you make changes to the Platform object, you can propagate the changes by redeploying WinPlatform alone without being required to undeploy all the objects hosted on the Platform.

To redeploy a WinPlatform without redeploying hosted objects

1 Select a Platform object on the **Object** menu, click **Deploy**. The **Deploy** dialog box appears.



2 Clear the **Cascade Deploy** check box to prevent other hosted objects from being redeployed. By default, the check box is selected.

Note: If you want to redeploy all hosted objects, do not clear the **Cascade Deploy** check box.

3 In the **Currently Deployed Objects** area, click the **Deploy Changes** option.

For more information on deploying objects, see “Deploying Objects” on page 196.

4 Click **OK**. The **Deploy** progress box appears. If an error occurs during deployment, an error message appears. For more information on error messages during deployment, see “Deployment Error Messages” on page 200. When deployment is complete, click **Close**.

Important: All engines will be shut down during the Platform redeploy operation. If the Platform hosts redundant engines, you can choose to fail over the redundant engine to the redundant partner while the Platform redeploy operation is in-progress. Depending on the timing configuration of the engine, the engine may also automatically fail over if the failover timeout is detected by the standby engine.

Undeploying Objects

You may need to undeploy one of more objects. Undeploying removes one or more objects from the run-time environment.

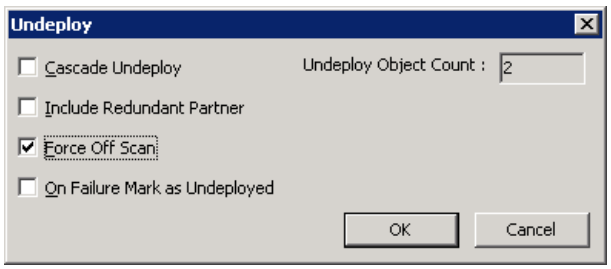
Before you start, you need to select the object or objects you want to undeploy in the IDE.

Before you delete or restore a Galaxy, undeploy all objects in the Galaxy.

Undeploying can fail if the target object has objects assigned to it. Make sure you select **Cascade Undeploy** in the **Undeploy** dialog box.

To undeploy

1 On the **Object** menu, click **Undeploy**. The **Undeploy** dialog box appears.



In the upper right of this dialog box, the **Undeploy Object Count** box shows the number of objects being undeployed. You can select a single object in **Application view** and, if you selected **Cascade Undeploy** and other objects are assigned to the selected object, the total number of objects appears in this box.

2 Select one or more of the following. Some of these options might not be available, depending on the kinds of object you select.

- **Cascade Undeploy**: Select to undeploy the selected object as well as any objects it hosts.

- **Include Redundant Partner**: Select to also undeploy an AppEngine's redundancy partner object.

Note: The AppEngine in a redundant pair that was configured as the Primary can be undeployed alone because objects hosted by it run on the deployed Backup AppEngine, which becomes Active.

- **Force Off Scan**: If one of the objects you are undeploying is currently on scan, selecting **Force Off Scan** sets the target object to off scan before undeployment. If you do not select **Force Off Scan** and the target object is on scan, the undeployment operation fails.
- **On Failure Mark as Undeployed**: Marks the object as undeployed in the Galaxy when the object targeted for undeployment is not found.

Uploading Run-time Configuration

You can upload run-time configuration changes to the Galaxy database. This lets you keep any attribute values that changed during run time.

Note: A node cannot upload run-time changes to the Galaxy if the version of Application Server it is running is older than the version running on the GR node.

Certain attribute values can be set in the configuration environment, but they can also change at run time. As a result, the values of these attributes can differ between the run-time and configuration environments. These types of attribute writeability are:

- Calculated
- Calculated Retentive
- Object Writeable
- User Writeable

For example, you create an object with a User Writeable attribute called `myInteger` and specify an initial value of 30 in the Object Editor.

Then you deploy the object. At run time, you write a new value to `myInteger` of 31. If you redeploy this object, the original value of 30 overwrites any value assigned during run time. To avoid losing changes made during run time, upload changes before redeploying an object.

Run-time configuration changes to auto assigned I/O references are uploaded as overrides to the original assignments. The I/O reference will continue to be auto assigned and the run-time changes are retained when the object is redeployed. See "Uploading Run-Time Configuration Changes for Auto Assigned Objects" on page 167 for additional information.

Objects whose configuration are successfully uploaded have a new version number and a change log entry for the upload operation. The run-time object's version number also has a new version number. That version number matches the version in the configuration database.

To upload run-time changes to the Galaxy

1. Select one or more objects in the **Model** view or **Deployment** view. For example, you could select an entire hierarchy from AppEngine down.
2. On the **Object** menu, click **Upload Runtime Changes**. The run-time attributes of the selected objects are copied over those in the Galaxy database.

Uploading Restrictions

Certain run-time changes to the Galaxy cannot be uploaded. Before uploading changes, make sure the selected objects are:

- Not checked out
- Not edited and checked in since last deployment or upload
- Not in Pending Update state

If you select an object that is currently checked out to you, a warning appears during run-time upload. If you continue, you lose all configuration changes you made to the checked out object. The Galaxy performs an Undo Check Out operation on it before the run-time attributes are copied to the Galaxy database.

Note: You cannot upload run-time changes for objects checked out to other users.

Undeployment Situations

The following situations occur in these specified undeployment scenarios:

- After undeploying a WinPlatform, only the Bootstrap software remains on the target computer. All other WinPlatforms are notified of the undeployment of the WinPlatform and stop trying to communicate with it over the network.
- Alarm Clients know immediately that an area is undeployed and is no longer available. They remove the area from selection lists. Alarms associated directly with the area (not as a result of containment) are immediately removed from current alarm views.

- Undeploying an object that has Pending Updates status removes that status. It is now marked as undeployed.
- If cascade undeploy fails on one object, then the undeploy continues to the extent possible on other objects. The entire operation is not terminated because the undeploy fails for one object. However, a host might not be undeployable if one of its assigned objects cannot be undeployed.

Assume an ApplicationObject is hosted by the Active AppEngine in a redundant pair and a number of subscriptions is configured in that ApplicationObject that refers to items in a DIObject. If you undeploy the ApplicationObject in question, the items are not removed immediately from the item count of the DIObject. How fast those items are removed depends upon the value of the **Maximum time to maintain good quality after failure** option (Redundancy.StandbyActivateTimeout attribute) on the **Redundancy** page in the AppEngine's editor. This behavior does not apply to the undeployment of ApplicationObjects hosted by non-redundant AppEngines.

Associating All Galaxy Graphics with an InTouchViewApp

Associating all Galaxy graphics with an InTouchViewApp template enables deployed and published InTouch applications to execute "show graphic" requests made of any graphic in the Galaxy without having to embed them in the application.

- The ShowGraphic() function uses the graphic as a parameter. Associating all Galaxy graphics ensures that the graphic is deployed and available at run time whether or not it is referenced by InTouchViewApp.
- Associating all Galaxy graphics ensures that template symbols referenced by the ShowGraphic() function are deployed and available at run time.

Note: The term "graphic" includes any symbol or client control present in the Graphic Toolbox, and any symbols owned or inherited by templates and instances.

About Associating All Galaxy Graphics with an InTouchViewApp

Associating all Galaxy graphics with an InTouchViewApp (ITVA) exhibits the following deployment and publishing behavior:

- The associate all Galaxy graphics option—"Include all Galaxy graphics"—changes only the ITVA template.The option is inherited by the template instances, and changes the template only if none of its instances are deployed.
- A deployed ITVA with the "Include all Galaxy graphics" option enabled will be marked "pending change" under the following scenarios

Scenario	Operations Performed
Creating graphics	Create new graphics in the graphics toolbox Check in new symbol(s) in an automation object Import automation object with symbol(s) Import graphics toolbox graphics Create a new derived template or instance object that inherits symbols GR Load new symbols into the galaxy
Modifying graphics	Edit and check-in symbol(s) Modifying symbols through an Import operation Modifying symbol localization through an Import operation
Deleting graphics	Delete graphics from the graphics toolbox Check in an application object from which symbols have been deleted Delete template or Instance object with symbols
Renaming graphics	Rename graphics in the graphics toolbox Check in an application object, any of whose symbols have been renamed
Rename object	Rename Instance or Template which has owned or inherited symbols
Object containment	Containment of automation objects that own or inherit graphics

- Deploying an ITVA with the "Include all Galaxy graphics" option enabled for the first time will deploy all checked-in graphics in the Galaxy. Client controls will be deployed whether or not they are embedded in a symbol.Undeploy behavior does not change.
- Deploying changes with an ITVA that has the "Include all galaxy graphics" option enabled will deploy the Galaxy graphics changes since the previous ITVA deployment.

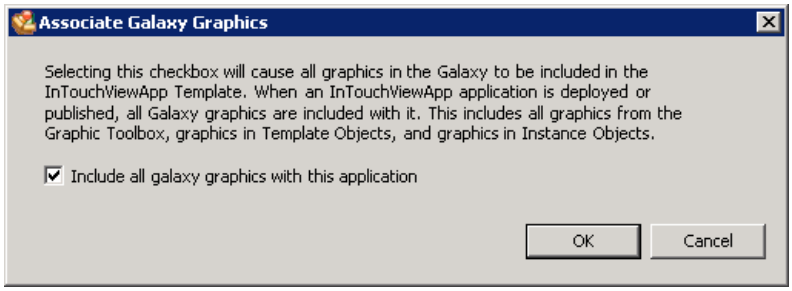
- Redeploying the original ITVA that has the “Include all Galaxy graphics” option enabled will not account for graphics that were added, deleted, renamed or modified since the original deployment.
- Publishing an ITVA with the “Include all galaxy graphics” option enabled will copy all the checked-in galaxy graphics to the published application folder. The ITVA will retain its “pending change” status as this is not a deploy operation.
- The “Include all Galaxy graphics” enabled setting will persist through a Galaxy backup and restore operation, but will not persist through an ITVA import or export, or through GR dump and GR load operations.

Configuring the Include All Galaxy Graphics Option

Access the option through the InTouchViewApp right-click context menu.

To include all Galaxy graphics with an InTouchViewApp

1. Right-click the InTouchViewApp template you wish to configure. The InTouchViewApp context menu appears.
2. Select the **Associate Galaxy Graphics** menu item. The **Associate Galaxy Graphics** dialog box appears.



   a. When the **Associate Galaxy Graphics** dialog appears, the $InTouchViewApp template will be checked out.

   b. If the $InTouchViewApp template is not yet initialized or checked out, the **Include all Galaxy graphics with this application** option will be disabled.

3. Click the check box and click **OK**. A status box displays check in progress.

   If an InTouchViewApp instance is deployed when you attempt to modify the **Include all Galaxy graphics** setting, you will see an information message to undeploy all deployed instances.

Chapter 7

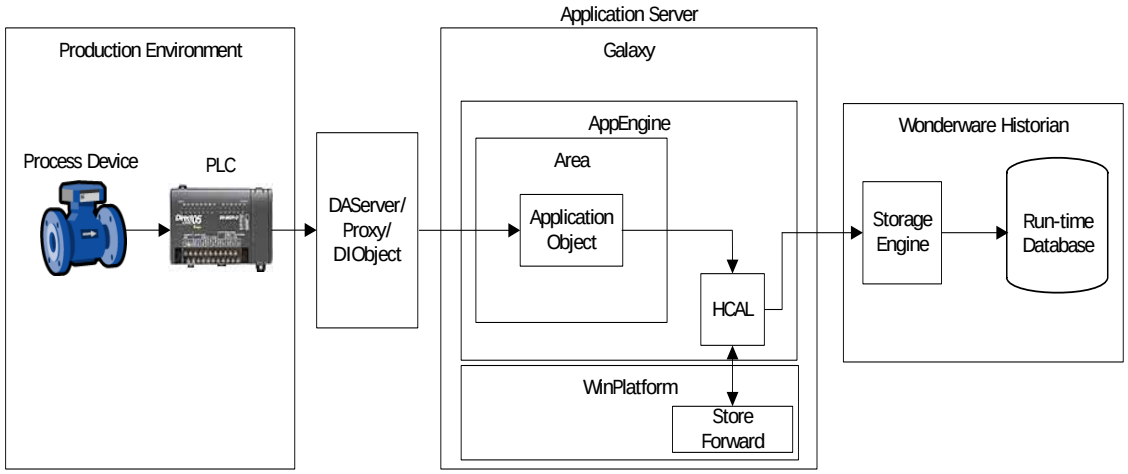
Working with History

You can configure application objects to store process data in the Wonderware® Historian (formerly called IndustrialSQL Server). Historical data from Application Server can be retrieved and viewed using standard Wonderware Historian database utilities. Also, you can use historical data to produce reports shown from your client applications. InTouch includes a set of ActiveX controls designed to show historical data from the Wonderware Historian in trends or graphs that can be embedded in client application windows.

Application Server History Components

To save your process data to a historical database, you must install the Wonderware Historian Server. A Historian Server database can be installed on any computer outside the Galaxy, but on the same network. In a production environment, the Wonderware Historian should be installed on a dedicated computer without other ArchestrA products running on it.

The following figure shows the major ArchestrA components to save process data from a production device to the Wonderware Historian.



A single Wonderware Historian installation can receive historical data from a single Galaxy. A push model is used to send and save new historical updates to Wonderware Historian. Each system object Engine (Platform, AppEngine, ViewEngine) includes a historian feature that sends all history updates for all hosted objects to the historian. The historian feature receives the history updates from objects on the same engine only. All Engine objects include an attribute to specify the node name of the computer hosting Wonderware Historian.

The figure shows a single Wonderware Historian. This may be a common configuration, but other Application Server configurations support multiple Wonderware Historian databases for a Galaxy. However, each Engine object only sends its historical data to one Wonderware Historian.

There is a one-to-one relationship between a historical object attribute and a tag in Wonderware Historian.

Sending Historical Data Between Application Server and Wonderware Historian

Application Server communicates with Wonderware Historian through an interface called the Historian Client Access Layer (HCAL).

HCAL can establish and maintain a connection to one or more historians either synchronously or asynchronously. If a disconnection from the historian occurs, HCAL attempts to restore the connection.

If HCAL cannot communicate with the historian, all data currently being processed can be stored locally on the computer running HCAL. This hard drive location is called the store-and-forward path. Historical data is stored until the threshold capacity of the path is reached or communication to the historian is restored. In the event that all store forward disk capacity is used to store historical data, no more data is stored. An error message is logged. Remote store-and-forward paths are not supported.

HCAL can go into store-and-forward mode even if it has never been connected to the historian. After the tags are configured successfully on the historian, current data will start to be sent, along with the locally-stored data.

You can specify how disconnects between Application Server and the Wonderware Historian should be reflected in the data until the disconnect period can be backfilled with store-and-forward data.

If you select the **Reconnect as soon as possible & do not mark disconnects** option for an AppEngine, NULL values are injected into the data stream for the disconnect period. For a trend, this means that a line gap appears during the period of NULL values. The tag remains in store-and-forward mode until the timestamps become greater than the startup time of the server or the time that the connection was restored. If you do not enable this option, NULL values are not injected and no gap is shown in client-side trends.

For more information about HCAL, see the Wonderware Historian documentation.

Saving Object Attribute Data to the Historian

Some attribute values can be saved as historical data when values change. Data quality and time are also associated with attribute values. When available, Application Server uses the attribute's value, the timestamp when the value changed, and the calculated quality of the data to create a Value, Time, Quality (VTQ) packet sent to the Historian. If there is no timestamp associated with the attribute value, the current scan time from the AppEngine hosting the object is used instead.

Saving Process Values as Historical Data

For attribute data to be stored in the historian, a host AppEngine must be configured to send history data to a Wonderware Historian node. For each object, you can configure attributes of the following data types to be saved to the Wonderware Historian.

- Float (numerical).
- Double (numerical) maps to a Wonderware Historian float data type. If the value of the double exceeds the range of a float, its value is clamped to the maximum value for the float and the quality is set to Uncertain.
- Integer (numerical)
- Boolean (non-numerical)
- String – Unicode (non-numerical). Limited to 512 characters. Characters that exceed the 512 maximum length are truncated. If truncation occurs, the quality value of the packet is set to Uncertain.
- CustomEnum (non-numerical) maps to a Wonderware Historian integer.
- ElapsedTime (numerical) maps to a Wonderware Historian float and is converted to seconds.
- Arrays or parts of arrays are not supported.
- Enum type attributes are saved as Wonderware Historian integer ordinal values.
- IEEE NaN values for float and double data types are converted to null values prior to being sent to the historian. NaN values are associated with a Bad OPC data quality.
- All numerical attributes sent to the Wonderware Historian are in the engineering units specified for the attribute. The Wonderware Historian does not scale numerical values.

Saving Data Quality as Historical Data

Wonderware Historian supports the OPC Quality definition. A 16-bit value for OPC data quality is sent to Wonderware Historian in the VTQ packet. Within the 16-bits, the low-order byte is the standard OPC portion. Wonderware reserves the high order byte and it is currently unused. The Good, Bad, Initializing (which is a form of Bad) and Uncertain quality states are specified in the low-order byte.

Any change in the mapped Wonderware Historian Quality Detail saves a new record to Wonderware Historian, regardless of whether the attribute value has changed. If an attribute value remains constant over a period of time with varying changes in quality, multiple records are stored in Wonderware Historian. The OPC quality stored in Wonderware Historian is the actual quality of the attribute in Application Server without modification.

Wonderware Historian can create and insert additional (non-ArchestrA generated) VTQ records that modify quality in the database. These additional VTQ records provide additional information about network outages or other events to client applications that use historical data from Wonderware Historian.

Additional Quality Data Saved to the Historian

Wonderware Historian also has two additional data quality fields (quality and quality detail) that are non-OPC compliant. Both fields are stored with each record in addition to the OPC quality. Wonderware Historian maps the value of these fields in a manner consistent with its quality definition for those fields. The values of those fields are driven by the OPC quality sent by Application Server. For example, an OPC Good quality results in Wonderware Historian quality detail of Good.

Wonderware Historian Quality is a 1-byte flag that summarizes the quality of the associated data value in the packet sent by Application Server. Wonderware Historian Quality is an enumerated type with the following values:

Hex	Decimal	Name	Description
0x00	0	Good	Good value.
0x01	1	Bad	Value was marked as invalid.
0x10	16	Doubtful	Value is uncertain. Only fabricated at retrieval time.
0x85	133	Initial Value	(Good) Initial value for a delta request.

OPC Qualities from Application Server result in storage of an Wonderware Historian Quality as follows:

OPC Quality (substatus form)	Wonderware Historian Quality Decimal (1-byte)
Good (any)	0 - Good
Bad (any)	1 - Bad
Bad (Initializing form)	1 - Bad
Uncertain (any)	0 - Good

Wonderware Historian QualityDetail contains the detail of data quality as a 32-bit value. QualityDetail is derived from OPC quality. Each source is allocated a byte position within the QualityDetail 32-bit word, which is formatted as:

0xXXRRDDDD

Only the low-order two bytes (DDDD) are mapped to OPC quality for data sent by Application Server and saved to the historian:

OPC Quality (any substatus form)	Results in any one of these Wonderware Historian Quality Details
Good (any)	192 - Good value 150 - Initial value (good) 151 - Initial store/forward value (good) 252 - First value received (good) 44 - First good value received by storage after the connection to HCAL has been restored. 248 - First value in the second stream during a store-and-forward.
Bad (any)	24 (DAServer disconnect)
Bad (Initializing form)	24 (DAServer disconnect)
Uncertain (any)	192

Saving the Data Timestamp as Historical Data

Application Server includes a timestamp in the VTQ packet sent to the historian for each attribute value/quality update that is saved as a historical record. Application Server propagates the timestamp associated with an attribute value. The timestamp is propagated throughout all Application Server components as UTC in FILETIME format.

If an object attribute has a Value DeadBand specified, a quality change or value change from previous scan cycle greater than Value DeadBand causes the attribute's Value, Time and Quality (VTQ) to be saved as historical data in the Historian. When a Value DeadBand is not specified, any change to the attribute's value, timestamp, or quality cause the attribute to be saved as historical data.

When an attribute is configured to periodically save a historical record, the attribute's current time is used as the timestamp. If the attribute does not support a timestamp, the hosting AppEngine's current scan time is used as timestamp included in the packet.

If slow updates for real-time data values are received from an I/O source and an intermittent network disconnect occurs, the timestamp of the first data value of a historized attribute uses the AppEngine's timestamp instead of the source's timestamp. In the following example, in row number 4, the value "30" is sent again to the historian, but the timestamp is the AppEngine timestamp and not the actual timestamp of "2008-11-24 18:50:41.061" when the value of 30 was generated from the I/O source.

DateTime	TagName	Value	Quality	QualityDetail	OPCQuality
2008-11-24 18:48:30.081	Realtime_ Client.ival	20	133	44	192
2008-11-24 18:50:41.061	Realtime_ Client.ival	30	0	192	192
2008-11-24 18:51:38.666	Realtime_ Client.ival	NULL	1	24	0
2008-11-24 18:52:06.670	Realtime_ Client.ival	30	0	192	192 –> Row number 4
2008-11-24 18:53:22.988	Realtime_ Client.ival	40	0	192	192

If slow updates for late data values are received from an I/O source and an intermittent network disconnect occurs, the timestamp of the first data value of a historized attribute is modified to maintain the time sequence. In the following example, in row number 4, the value “30” is sent again to the historian, but the timestamp is modified by adding 5 milliseconds to previous timestamp of the NULL data value. Note that a QualityDetail of 704 is stored for the data value in row number 4.

DateTime	TagName	Value	Quality	QualityDetail	OPCQuality
2008-11-24 18:48:30.081	Latedata_ Client.ival	20	133	44	192
2008-11-24 18:50:53.624	Latedata_ Client.ival	30	0	192	192
2008-11-24 18:50:53.625	Latedata_ Client.ival	NULL	1	24	0
2008-11-24 18:50:563.630	Latedata_ Client.ival	30	0	704	192 –> Row number 4
2008-11-24 18:53:26.863	Latedata_ Client.ival	40	0	192	192

Saving Alarms and Events as Historical Data

Alarms and events detected by the Application Server at run time are saved as historical data to the Wonderware Historian for the host engine. Alarms are a type of event that can be historized. In addition, alarm-related events such as Acknowledge are also saved as historical alarm data.

You can save alarm counts by severity level as historical data. For more information about mapping alarm severities to priority ranges and saving alarm counts aggregated by severity level to the Historian, see “Mapping Alarm Severity to Priority” on page 273.

Deploying and Undeploying Attributes

When an object is deployed with historical attributes, the history feature causes dynamic reconfiguration of Wonderware Historian. Each history feature causes a new tag to be created and configured automatically in Wonderware Historian at deployment time. The Wonderware Historian storage system that will be used is determined by the configuration of the host engine object.

This dynamic configuration causes appropriate row/column configuration in the Wonderware Historian schema.

If the connection to the Wonderware Historian is down at deploy time, the attempt to dynamically reconfigure Wonderware Historian is achieved when the connection is restored. In other words, automatic retry is built in. However, any data generated by the object during the time between deployment and the recovery is lost and cannot be recovered by store forward.

When an object configured for history is redeployed, changes to an object's attributes makes the historian reconfigure storage. For example, if the engineering units string of an object changes from "Deg F" to "Deg C" when the object is redeployed, the Wonderware Historian configuration database shows the change.

When an object configured for history is undeployed, all history remains in the Wonderware Historian. The history data can be viewed in the future even if the object is no longer deployed.

Saving Historical Data During Run Time

While an application is running, data is saved to the historian as follows:

- If no previous historical attribute value exists in the historian, the first value is always saved to the historian.
- Numerical attributes (double, float or integer): If the value for the attribute changes and that change is more than the value deadband, or the value's quality changes (for example, from Good to Bad), the data is saved to the historian.
- If the attribute type is qualitative (enumeration, string, or Boolean) and the attribute's value or quality changes.
- If the current time exceeds the last Forced Store period occurred for the attribute by the time period that was set as the Force Storage Period. If no last Forced Store occurred since starting the object, a new store occurs immediately.

Note: If enabled, the Value Deadband mechanism resets itself based on this new stored value.

- All new attributes value, timestamp, and quality are sent to the Wonderware Historian for storage.
- Wonderware Historian stores the historical records to the database.

Advanced Communication Management

To ensure that attribute data saved to the historian is always current and continuous, Application Server registers a reference to the attribute. By registering a reference to the attribute, it prevents Message Exchange from suspending historical updates during the period when the client application has minimized the window containing the object. Historical data continues to be saved to the Wonderware Historian even during the period when the object is in an Advanced Communication Management Suspended state.

Store-and-Forward Mode

If the Wonderware Historian shuts down or the network connection to it is lost while an application is running, historical data continues to be stored locally on the computer hosting the WinPlatform object.

When the Wonderware Historian node recovers, data is forwarded from the local node to the Wonderware Historian node at a low priority.

If an AppEngine loses connectivity to the Wonderware Historian node, the Wonderware Historian reports bad data quality to clients. When you undeploy an object with attributes configured for history, the Wonderware Historian stores the final data points with Bad quality.

Buffered Behavior

The buffered data feature enables efficient accumulation and propagation of VTQ (Value, Time, and Quality) data updates, without folding and data loss, to data consumers such as objects, alarms, the Historian, and scripts from field devices that support buffering.

Buffered data is defined as data captured and stored locally on a remote device for later transfer to a supervisory system for processing, analysis, and long-term storage. The Buffer property is input-only.

When an object is associated with an attribute that supports buffered data (HasBuffer property is true), the object monitors and processes the Buffer property of the attribute. Each VTQ entry in the buffer that meets value deadband criteria is pushed to the Historian.

Attributes that are buffer-enabled are registered to Historian in the same manner as non-buffered attributes are registered. The Historian does not differentiate a buffered from a non-buffered registered attribute. An attribute is considered registered after a Historian key is returned from HCAL to the object.

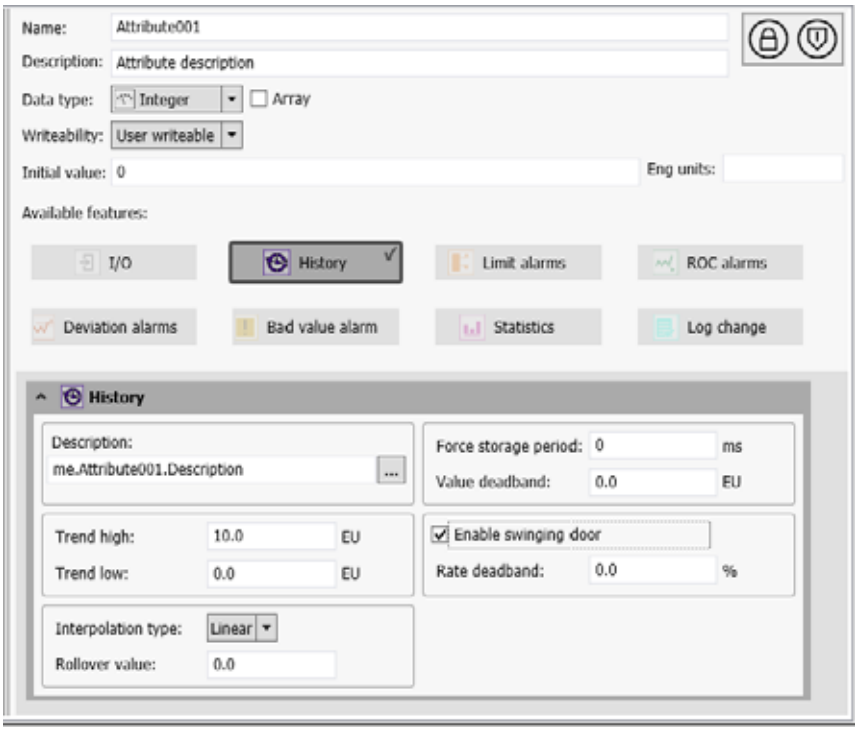
All the buffered VTQs that are received by the parent attribute while Historian registration is pending are ignored and discarded.

For more information about buffered data, see “Working with Buffered Data” on page 321. For further reading on historizing events and attributes that are buffering-enabled, see “About Buffered Data and History” on page 331.

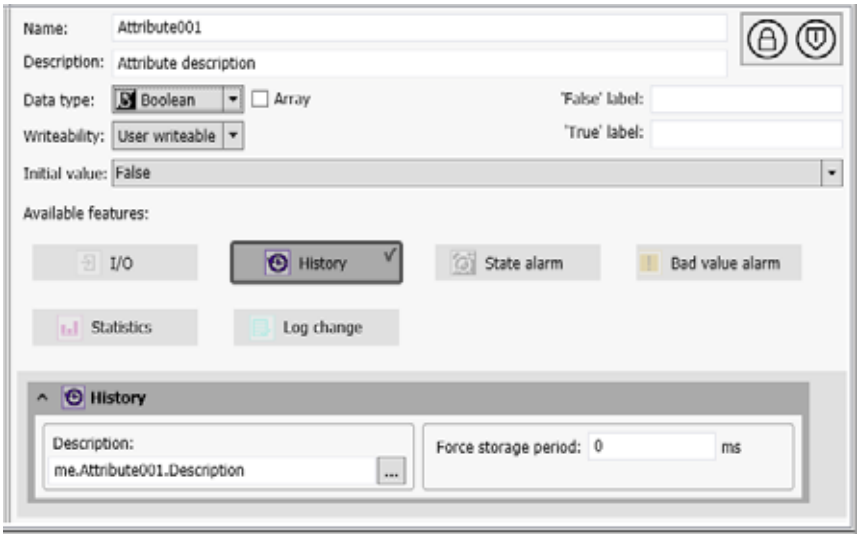
Configuring Common Historical Attributes

Each application or system object that you select from the Template Toolset include a set of common attributes you configure to save data to the Historian. These are configured after activating the **History** feature in the **Attributes** page.

The historical attributes you configure are based on the data type of the object. For example, the following figure shows the common historical attributes for numerical attributes (integer, float, or double).



If you are configuring a boolean or string attribute, you see a smaller set of historical attributes. Similarly, a discrete field attribute will also have this same, smaller set of historical attributes.



This section describes the common historical attributes that you can configure for your system and application objects. Refer to this section as you complete the procedures to configure your objects.

- **Force Storage Period**

  Interval in milliseconds in which an attribute value is saved to the Historian, regardless of whether the value exceeds its value deadband setting or not. An attribute value is saved to the Historian at every Force Storage interval. The default value of zero (0) disables the Force Storage period.

- **Value Deadband**

  Threshold value in engineering units, which is the absolute difference between the current and most recent saved historical values. The current value must exceed the absolute deadband to be saved as historical data.

  The Value Deadband filters out small, momentary value changes from being saved to the Historian. A new value that is within the range of the deadband is not saved to the Historian. The default value of 0.0 disables the Value Deadband and any change to a value is saved as historical data.

- **Trend Hi**

  Initial maximum trend value in engineering units for clients. The default is 100.

- **Trend Lo**

  Initial minimum trend value in engineering units for clients. The default is 0.0.

- **Enable Swinging Door**

  A flag that indicates whether the swinging door rate deadband is enabled or disabled. The default is disabled.

  Enable Swinging Door is disabled if the historical attribute data type is Boolean or a string.

- **Interpolation Type**

  List of methods used by the Historian to interpolate analog historical data. The interpolation type determines which analog value is selected during a Historian data retrieval cycle. Interpolation types include System Default, Stairstep, or Linear. The default interpolation type is System Default.

  - Stairstep

    No interpolation occurs. The last known value is returned with the given cycle time. If no valid value can be found, a NULL is returned to the Historian.

  - Linear

    The Historian calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.

  - System Default

    The Wonderware Historian system-wide interpolation setting. The system-wide setting must be either stairstep or linear interpolated.

- **Rollover Value**

  Positive integer value that represents a tag's reset limit when the Historian operates in counter retrieval mode. In counter retrieval mode the Historian uses a tag's rollover value to calculate and return the delta change between consecutive retrieval cycles. The default value is 0.0.

  The Rollover value applies only to numeric attributes. The Rollover value is disabled if the historical attribute data type is Boolean or a string.

- **Rate Deadband**

  Percentage rate of change deadband based on the change in the slope of incoming data values to the Historian. For example, specifying a swinging door rate deadband of 10 percent means that data is saved to the Historian if the percentage change in slope of consecutive data values exceeds 10 percent.

  The default is 0.0, which indicates a swinging door rate deadband is not applied. Any percentage greater than 0.0 can be assigned to the rate deadband.

Configuring System Objects to Store Historical Data

The following table shows the historical attributes for Application Server system objects.

History Attributes	Application Server System Objects				
	WinPlatform	AppEngine	Area	ViewEngine	InTouchViewApp
Description	●		●	●	
Enable compression	●	●		●	
Reconnect as soon as possible & do not mark disconnects		●			
Enable Storage to Historian	●	●		●	
Enable Swinging Door	●	●	●	●	
Enable Tag Hierarchy	●	●		●	
Engineering units	●	●	●	●	
Force Storage Period	●	●	●	●	
Historian	●	●		●	
History store forward directory	●	●		●	
Interpolation Type	●	●	●	●	
Pre-processing buffer size	●	●		●	
Rate Deadband	●	●	●	●	
Rollover Value	●	●	●	●	
Store forward threshold	●	●		●	
Store forward minimum duration	●	●		●	
TCP Port	●	●		●	
Throttling network bandwidth	●	●		●	
Trend High	●	●	●	●	
Trend Low	●	●	●	●	

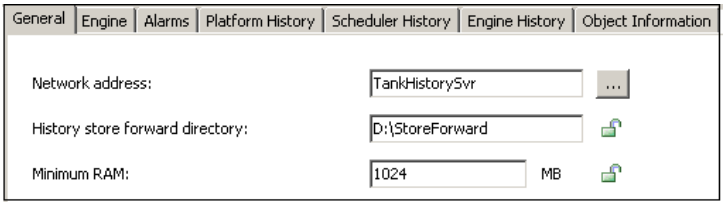
History Attributes	Application Server System Objects				
	WinPlatform	AppEngine	Area	ViewEngine	InTouchViewApp
Value Deadband	●	●	●	●	
Wait to send incomplete packets	●	●		●	

Configuring the WinPlatform Object to Store Historical Data

A WinPlatform object contains attributes to configure how historical store-and-forward data is cached locally and then pushed to the Historian. Also, you can select the WinPlatform object's platform, scheduler, and engine data to be saved to the Historian. Finally, you can select the WinPlatform's attribute values to be saved to the Historian.

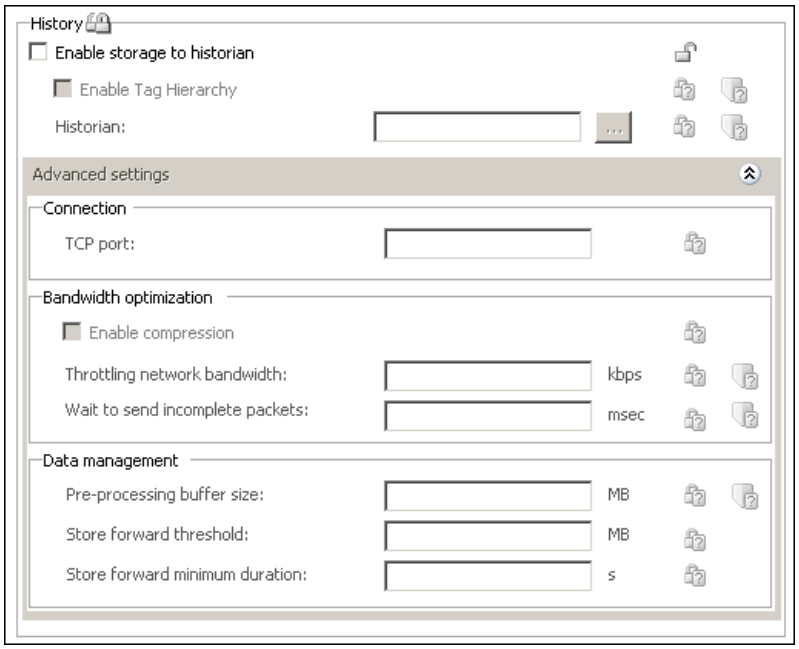
To configure a WinPlatform Object to store historical data

1. Double-click on an WinPlatform derived template or instance to open the Object Editor.
2. Click the **General** tab to show history attributes.



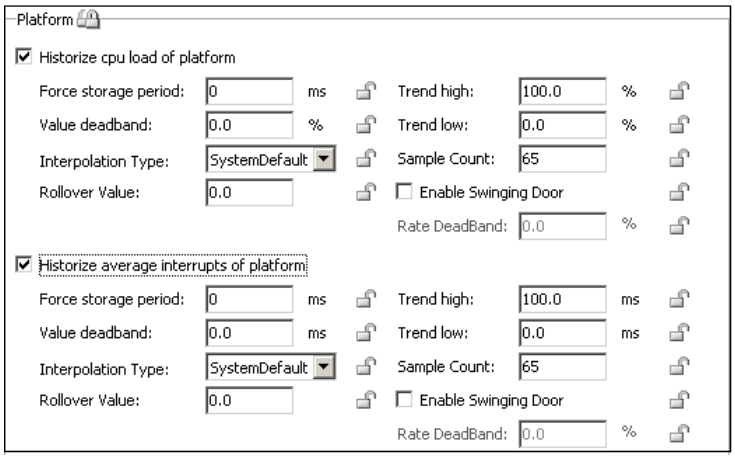
3. In the **Network address** box, type or select the node name of the computer assigned to the WinPlatform object.
4. In the **History store forward directory** box, enter the path to the folder to save store and forward historical data. The folder must exist on the computer specified in the **Network address** box.

5 Click the **Engine** tab to show history attributes.



6 In the **History** area of the **Engine** page, configure the attributes. For more information on each attribute, see the WinPlatform object Help.

7 Click the **Platform History** tab to show the attributes to save system data from the computer hosting the WinPlatform object to the historian.



8 Select the check box next to each attribute whose data you want saved in the historian. For more information about these historical attributes, see “Configuring Common Historical Attributes” on page 219.

9 Click the **Scheduler History** tab to show the attributes to save data to the historian.

10 Select the check box next to each scheduler attribute whose data you want saved in the historian.

11 Click the **Engine History** tab to show the attributes to save data from AppEngines hosted by the WinPlatform object.

12 Select the check box next to each engine attribute whose data you want saved in the historian.

13 Save your changes and check in the WinPlatform object.

14 Deploy the object to its target computer in an on scan state.

Configuring an AppEngine Object to Store Historical Data

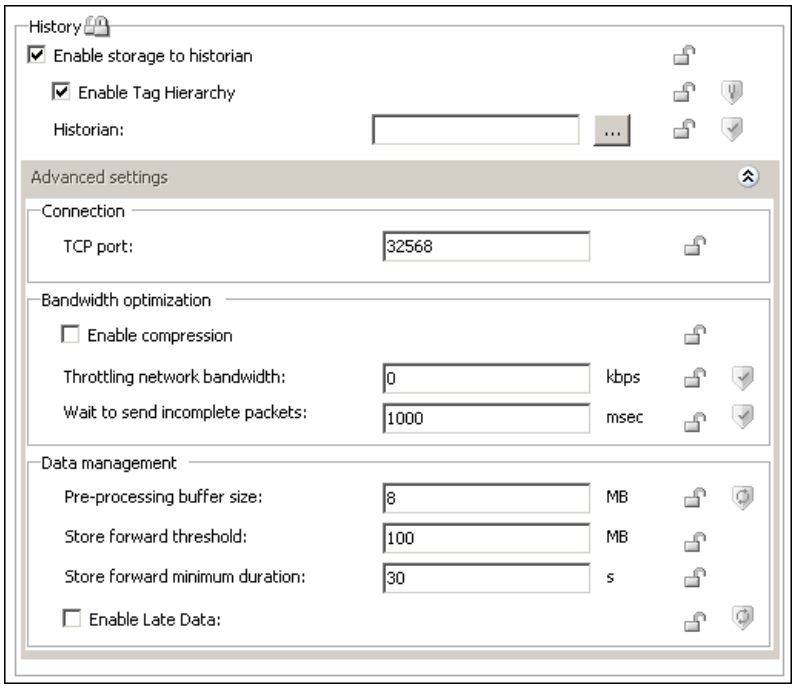
If an AppEngine is deployed before Wonderware Historian is started, history data can be stored locally by HCAL until the objects successfully register with the Wonderware Historian.

Note: Except for Late Data, the ViewEngine object contains the same historical attributes as the AppEngine object.

To configure an AppEngine object to store historical data

1 Double-click on the AppEngine instance to open it within the Object Editor.

2 Click on the **General** tab to show history attributes.



3 In the **History** area of the **Engine** page, configure the attributes. For more information on each attribute, see the AppEngine object Help.

4 Click the **Scheduler History** tab to show the attributes to save data to the historian.

5 Select the check box next to each scheduler attribute whose data you want saved in the historian.

6 Click the **Engine History** tab to show the attributes to save data from AppEngines hosted by the WinPlatform object.

7 Select the check box next to each engine attribute whose data you want saved in the historian.

8 Save your changes to the AppEngine object.

Configuring an Area Object to Save Alarm Counts as Historical Data

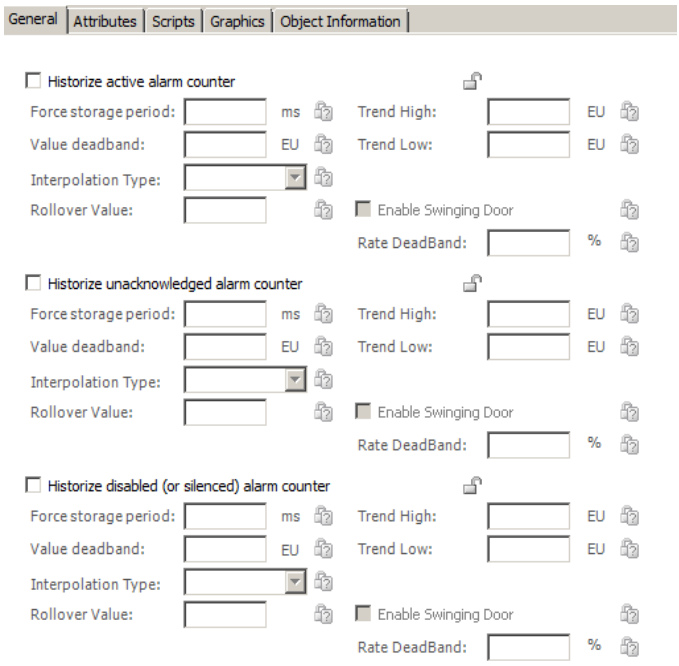
An Area object represents a plant area to group objects for modelling and report alarms. You can configure a set of attributes to save the counts of an area's alarm states to the historian. An Area object contains a set of attributes to save the counts of the following alarm states to the historian:

- Active alarms
- Unacknowledged alarms
- Disabled or silenced alarms

To configure an Area object to store historical data

1 Double-click on the Area instance to open it within the Object Editor.

2 Click on the **General** tab to show attributes for area alarm counts that can be saved as historical data.



3 Select the check box next to each alarm state counter that you want to save to the historian.

4 Set the attributes for each alarm counter that you select. For more information about assigning values to historical attributes, see “Configuring Common Historical Attributes” on page 219.

5 Save your changes and close the Object Editor.

Configuring Application Objects to Save Historical Data

The following table shows the historical attributes for application objects. These history attributes are described in “Configuring Common Historical Attributes” on page 219.

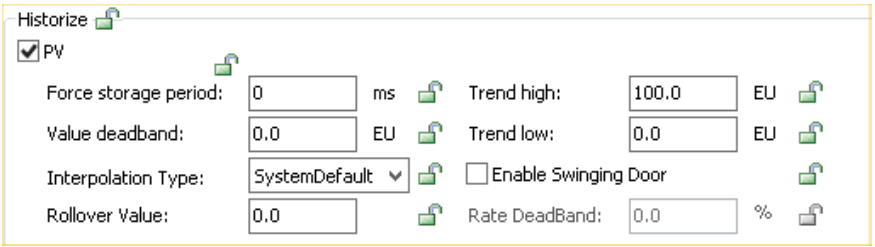
History Attributes	Application Server Application Objects										
	AnalogDevice	Boolean	DiscreteDevice	Double	FieldReference	Float	Integer	Sequencer	String	Switch	UserDefined
Enable Swinging Door	●	●		●	●	●	●				
Force Storage Period	●	●	●	●	●	●	●		●	●	
Interpolation Type	●	●		●	●	●	●				
Rate Deadband	●	●		●	●	●	●				
Rollover Value	●	●		●	●	●	●				
Trend High	●	●		●	●	●	●				
Trend Low	●	●		●	●	●	●				
Value Deadband	●	●		●	●	●	●				

Note: The historical attributes available from the UserDefined object are based on the data type associated with the selected attribute.

To configure an application object to store history

1 Open the application object in the Object Editor.

- If you selected a UserDefined object, click the **Attributes** tab.
- If you selected an AnalogDevice object, click the **History** tab. Then, click the General tab to show the history attributes for the object’s PV value.



2 Enable historization.

- For UserDefined objects, create an attribute and then activate the **History** feature. Assign values to the history parameters. For more information about assigning values to the history feature, see “Configuring Common Historical Attributes” on page 219.
- For AnalogDevice objects, select the **Historize** feature to enable the common history attributes shown in the **History** area of the page.

  The History page includes check boxes to save **PV**, **SP**, **PV mode**, **Control mode**, and **Control Track Flag** data to the historian.

  Assign values to the PV history attributes that appear in the **History** area of the page. For more information about assigning values to the common history attributes, see “Configuring Common Historical Attributes” on page 219.

3 Save your changes and close the Object Editor.

Chapter 8

Working with Alarms and Events

You can create ArchestrA applications that generate alarms and events to provide the status of a running application.

- **Alarms** warn about process conditions that can potentially cause problems. Typically, you set up an alarm to become active when a process value exceeds a defined limit. For example, you can set an alarm for a pump that warns when no fluid pressure is detected.

  An alarm is an abnormal condition that requires immediate attention. An operator usually acknowledges an alarm. ArchestrA handles the real-time reporting of alarms and provides special clients for viewing them.

- **Events** represent normal system, application, or user occurrences that produce status messages. A typical event occurs when an operator logs on to an application at the beginning of a work shift. Application Server can detect events, store them as historical data, and report them to client applications.

ArchestrA objects include built-in event and alarming reporting capabilities. You must configure alarms for each object in the IDE to use the event and alarm functions.

Understanding Events

An event indicates a significant occurrence that is detected, reported, and saved as historical data. Events can be detected by any Application Server component including automation objects and the SMC.

Events are more general purpose than alarms and provide a means for any software component to log information about a system, application, or operator action. Application Server components use an event API to send event messages to Alarm/Event distributors and the Wonderware Historian. Also, events are sent to client applications like InTouch HMI to be shown in real-time trends or reports.

Types of Events

Application Server creates several types of run-time event messages, which are saved as historical data.

- System events

  Event messages are logged when a Platform, Engine, Area, DI Network or DI Device start or stop. Any system action that affects a large number of objects is logged as an event.

  A System event message contains the following information:

  - Event type, which is System.
  - Tagname, which is the name of the object generating the event.
  - Tag description, which is a short description of the object generating the event.
  - Area, which is the name of the area that contains the object generating the event.
  - Event description, which describes the system action and can be either Started or Stopped.
  - Timestamp, which is the current system time.

- Security-audit (operator change) events

  Event messages are logged to the Alarms and Events subsystem when an operator logs on to or logs off from an application. Also, security-audit events are logged when an operator changes actions by means of User sets.

  A Security-audit event message contains the following information:

  - Event type, which is operator change.
  - Timestamp, which is the date and time when the operator change event occurred. The timestamp of the event is the current AppEngine scan time.

- Tagname, which is the name of object generating the event.
- Prim.attr, which is the reference string of the attribute being changed.
- Tag description, which is a short description of the object. For Secured and Verified writes, this will contain the following:
  - Type of write (Secured or Verified)
  - Comment from the user, if any
  - Reason description, if any
- FieldAttribute description, if the attribute is a FieldAttribute and has a description; otherwise, this is the Object description.
- Area, which is the name of the area that contains the object generating the event.
- UserEngine, which is the name of the view engine or other user engine requesting the operator change.
- Operator1, which is the full name of the primary operator requesting a change. The full name is an attribute of the UserProfile.
- Operator2, which is the full name of the secondary operator validating the change, if any.
- Old value, which is the previous value of an attribute.
- New Value, which is the new value of an attribute.

- Application (or process) related events

  Application event messages are generated by application objects in response to process actions. For example, application event messages are created when a process pump starts or stops.

  An Application event contains the following information:

  - Event type, which is data change.
  - Timestamp, which is the date and time when the application event occurred. The timestamp assigned to the event is the timestamp of the attribute associated with the event, if available. Otherwise, the event timestamp is the current AppEngine scan time.
  - Tagname, which is the name of the object generating the event.
  - Prim.attr, which is the reference string of the attribute being changed.
  - Tag description, which is a short description of the object.
  - Area, which is the name of the area that contains the object generating the event.
  - Old value, which is the previous value of an attribute.

- New Value, which is the new value of an attribute.

Understanding Alarms

Application and system objects can detect and generate an alarm. To detect an alarm, a system or application object sets a Boolean Attribute flag to indicate whether the object's alarm condition is currently true or false.

To report an alarm, the object must contain an alarm feature. The alarm feature makes a reference to the object's Boolean flag to determine whether the alarm condition is true. It then combines this information with the current alarm mode to determine whether to report a this as an active or inactive alarm state. An alarm feature is dedicated to reporting a single alarm condition's state. The alarm feature send alarm notification messages to ArchestrA alarm and event distributors.

Every alarm notification includes a set of fields containing data that describes the alarm. Some alarm notification data is saved as historical data. The following list describes all fields sent with an alarm notification.

- TagName, which is the name of the object generating the alarm. Saved as historical data.
- Name, which is the name of the alarm. Saved as historical data.
- InAlarm, which is a Boolean value that indicates whether the object's alarm state is currently active or inactive. Saved as historical data.
- Quality, which is the current quality of the data upon which the alarm is based. Saved as historical data.
- OnTimeStamp, which is the time when the attribute value transitioned into an alarm state. The attribute's value timestamp is used, if available. Otherwise, the timestamp is the AppEngine scan time. Saved as historical data.
- OffTimeStamp, which is the time when the alarmed attribute value returns to normal. The attribute's value timestamp is used, if available. Otherwise, the OffTimeStamp is the AppEngine scan time. If an active alarm is disabled, it forces a return to normal and the timestamp is the current time. Saved as historical data.
- Category, which is an integer between 1 and 14 that identifies the type of alarm and source of the alarm. These values are associated with Internationalized category labels set by Wonderware.

  Alarm category labels can be localized to other languages. Application Server uses the default Galaxy language to retrieve these strings and send them to InTouch. The alarm category labels

appear in InTouch and InTouch history as the default Galaxy language strings. Saved as historical data.

- Priority, which is an integer value from 1 to 999 indicating the severity of the alarm. An alarm priority of 1 is most urgent and 999 least urgent.
- TargetValueReference, which is an optional field that makes a reference to the target value of the alarm. Not saved as historical data.
- ActualValueReference, which is an optional field that makes a reference to the actual attribute value for the alarm condition. Not saved as historical data.
- TargetValue Snapshot, which is an optional field containing the attribute's target value at the time when the alarm became active. Saved as historical data.
- ActualValueSnapshot, which is an optional field containing the attribute's actual value at the time when the alarm became active. Saved as historical data.
- EngUnitsReference, which is the reference to the engineering units string for the condition. Saved as historical data.
- AcknowledgedFlag, which indicates whether the alarm is acknowledged or not. If this flag is FALSE, the alarm is still unacknowledged. Saved as historical event data.
- AcknowledgeTime, which indicates the time when the alarm was acknowledged if the AcknowledgedFlag is TRUE. Saved as historical data at the time of acknowledgement.
- AcknowledgeUserId, which is the string containing the name of the user who acknowledged the alarm. Saved as historical data at the time of acknowledgement.
- AlarmMode, which indicates whether the alarm mode is Enabled, Silenced or Disabled. Saved as historical data at the time when the alarm mode changes.
- Message text describing the alarm, which can be statically or dynamically constructed. The message typically contains the alarm description, the exceeded limit value, and possibly the variable value. For an alarm feature, this message is retrieved by means of a reference to a string/international string attribute in the object. The reference is setup at ObjectDesigner time for alarms. If none is specified, then the common feature short description attribute is utilized. This field is also provided in the alarm feature section and can be dynamically generated and scripted. Saved as historical data.
- Area, which is the name of the area that contains the object generating the alarm. Saved as historical data.

Types of Alarms

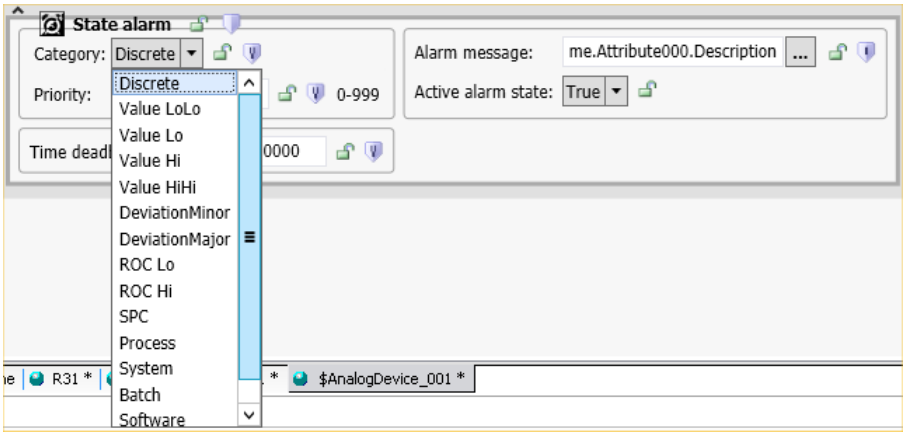
Application Server supports the following types of alarms:

- State alarms, which are also known as Boolean alarms
- Limit alarms
- Target deviation alarms
- Rate of change alarms

The type of alarm that you can configure is based on the data type of the attribute's value.

State Alarms

A state alarm corresponds to a discrete tag with two possible states. When you create a state alarm, you configure whether the active alarm state corresponds to the TRUE or FALSE state of the attribute.

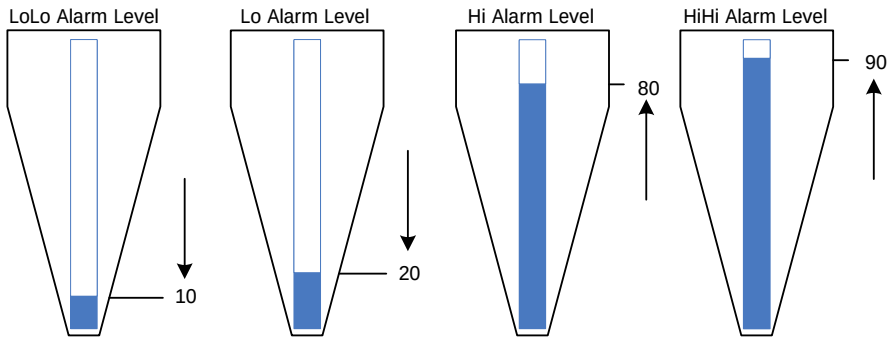


You can set an alarm message and Priority for a state alarm. The time deadband sets the length of time that an attribute value must continuously remain in an alarm or unalarmed state. The time deadband filters out rapid, transitory value spikes.

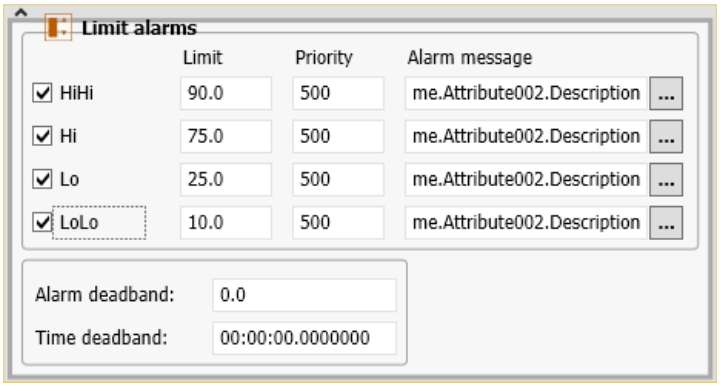
The timestamp when a state alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Limit Alarms

A limit alarm compares the current value to one or more predetermined alarm limits within the attribute's full range of values. If the value exceeds a limit, an alarm occurs.



You can individually select and configure values and priorities for the LoLo, Lo, Hi, and HiHi alarm limits. You can set individual messages for each alarm limit.



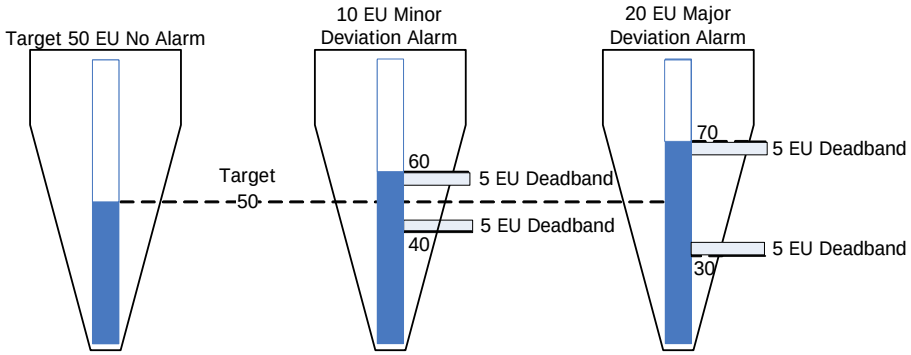
You can also configure alarm and time deadbands for limit alarms. The alarm deadband is expressed as a percentage of the attribute's full value range. The deadband range sets the percentage of the total range that the attribute value must change to reset a limit alarm to the inactive state. For example, if the HiHi alarm limit is 80 and the alarm deadband is 5, then the attribute value must decrease to 74 to reset a HiHi alarm to an inactive state.

The time deadband sets the length of time that an attribute value must continuously remain in an alarm or unalarmed condition. The process variable must remain above or below the indicated limit for at least the indicated deadband time before the application object updates the status of the alarm CONDITION Boolean. Then, standard Alarm feature logic determines whether to take that updated alarm condition and report changes to the alarm state or not.

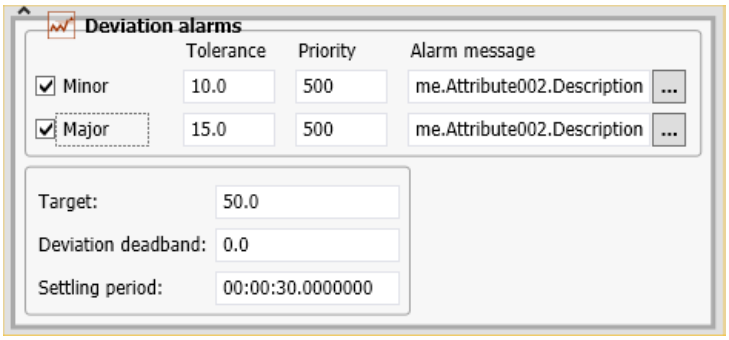
The timestamp when a limit alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Target Deviation Alarms

A target deviation alarm compares the current attribute value to a target Engineering Units value. Then, the absolute value of the difference is compared to one or more alarm deviation limits expressed in EngineeringUnits.



You can individually select and configure values and priorities for the minor deviation limit and the major deviation limit. You can set individual messages for the minor and major deviation alarm limits.



The deviation alarm's settling period is the time allowed for the attribute value to reach an expected target value after a device starts. No alarm can occur during the settling period.

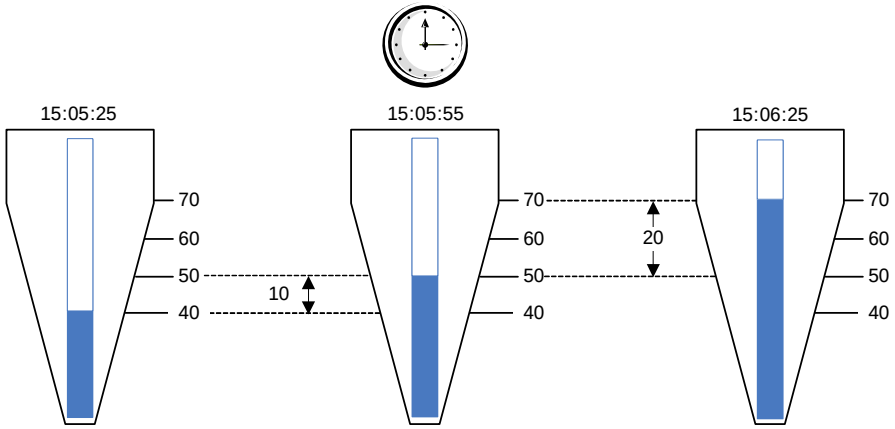
You can also configure a value for a deviation deadband, which is expressed in Engineering Units. The deadband range sets a threshold that an attribute value must change from a deviation limit to reset the alarm to the inactive state.

The timestamp when a deviation alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

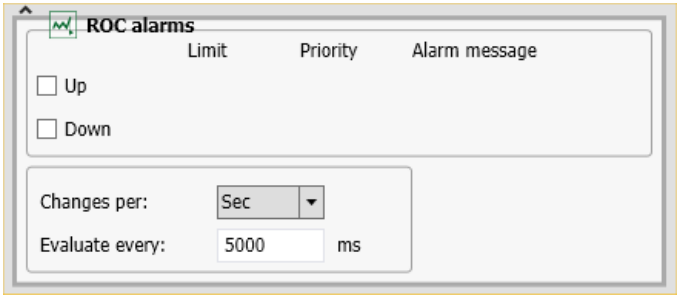
Rate of Change Alarms

A rate of change alarm identifies when an attribute value is changing too quickly over time. For example, you can set a rate of change alarm for a tank level that indicates when the pump inlet pressure is too high.

Rate of change is the calculated slope, which is the absolute difference between the current and previous attribute values divided by a specified interval. When the slope (positive or negative) exceeds a specified value, a rate of change alarm occurs. For example, if a tank volume increases from 17 to 45 liters over a 5 minute interval, the calculated slope is 5.6 liters per minute. If you set your rate of change alarm limit to 5.0 liters per minute, a rate of change alarm condition exists.



Alarm limits are expressed in the Engineering Units of the attribute's value over an interval, which can be per second, minute, hour, or day.



You can select and configure the value and priority for the upward and downward ROC limits. You can set individual messages for ROC alarms that exceed the upward or downward limits.

The timestamp when a rate of change alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Statistical Alarms

A statistical alarm is one in which a statistic is calculated, based upon an attribute. If the statistic exceeds some pre-set limit, the object flags the alarm condition as TRUE.

Setting Alarm State with Object Attributes

The Application Server alarm enable/disable mechanism includes four attributes to set an object alarm mode and report alarm status:

- AlarmModeCmd Attribute
- AlarmInhibit Attribute
- AlarmMode Attribute
- _AlarmModeEnum Attribute

AlarmModeCmd Attribute

AlarmModeCmd is a writable attribute that sets the current commanded alarm mode for the object. You set the AlarmModeCmd to enabled, silenced, or disabled with a script, user input, or from an attribute configured to read from an input source.

AlarmInhibit Attribute

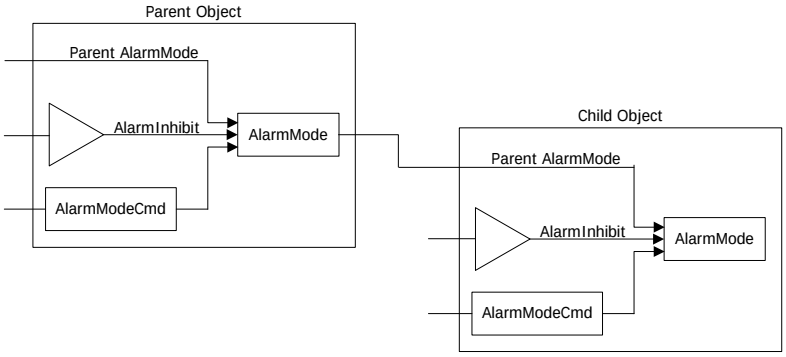
The AlarmInhibit attribute disables one or more alarms when set to TRUE. The value of the AlarmInhibit attribute is typically set by a script, manually by the user, or from an attribute configured with an input feature. If the AlarmInhibit attribute is set TRUE, all alarms of the object and of any contained objects are disabled.

When the AlarmInhibit attribute is set to FALSE, alarms are not inhibited and the object AlarmMode and parent object AlarmMode determine whether alarming is enabled, silenced, or disabled.

AlarmMode Attribute

The AlarmMode is a calculated attribute that identifies the object alarm mode and is based upon the current values of an object's:

- AlarmModeCmd attribute
- AlarmInhibit attribute
- Parent object AlarmMode attribute



Application Server checks the AlarmModeCmd and AlarmInhibit attributes of an object and the AlarmMode status of the parent object. Application Server then updates the object's AlarmMode attribute to reflect the most restrictive setting.

All individual alarms use the object's AlarmMode status to determine whether they are enabled, silenced, or disabled.

_AlarmModeEnum Attribute

The _AlarmModeEnum attribute can be used by any object to obtain an enumeration of permissible settings for the AlarmModeCmd attribute.

Alarms and Buffered Data

An object with alarms enabled detects the alarm condition and sets an associated alarm condition attribute to true. If the object detects the alarm condition based on an attribute with buffered data (HasBuffer property set to true), it enables buffer support for the condition attribute and sets the condition attribute's Buffer property to a VTQ buffer of true/false values representing alarm conditions.

Alarm Detection

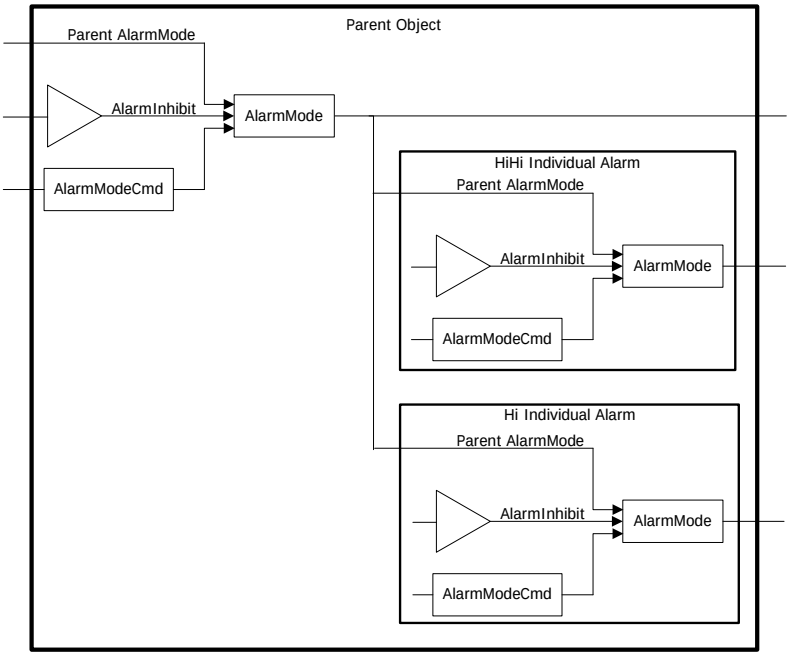
The Buffered Data feature will pass buffered alarm VTQs to the alarm subsystem in the same manner as with buffer data not enabled. When enabled, the Buffered Data feature can generate multiple alarm messages in the same scan.

For more information on alarm detection, see “About Buffered Data and Alarms and Events” on page 332.

Setting Alarm State for Individual Alarms

You can set individual alarms within an object for each type of alarm. For example, you can set alarms for each of the limits of a level alarm.

The following figure shows an object's individual alarms for the HiHi and Hi alarm limits.



The calculated AlarmMode attribute value of an individual alarm uses the same inputs as an object alarm. The parent AlarmMode attribute is from the object itself. As with object alarms, the individual alarm mode is set to the most restrictive input state. For example, if the object's AlarmMode state is disabled and the individual alarm's AlarmInhibit is FALSE, the individual alarm is disabled.

Each individual alarm is autonomous from other individual alarms in an object. The AlarmMode of an individual alarm is not propagated to other alarms. Unlike inhibit for the entire object, inhibit of an individual alarm does not affect the alarms of any contained objects. You can selectively enable, silence, or disable an individual alarm and not set other alarms to the same value within the object hierarchy.

Enabling, Silencing, and Disabling Alarms

Alarms can be enabled, disabled, or silenced while an application is running. An object's alarm state can be set at the Area level, at the container object level, or at the individual object. In addition, individual alarms within a single object can be enabled, silenced, or disabled.

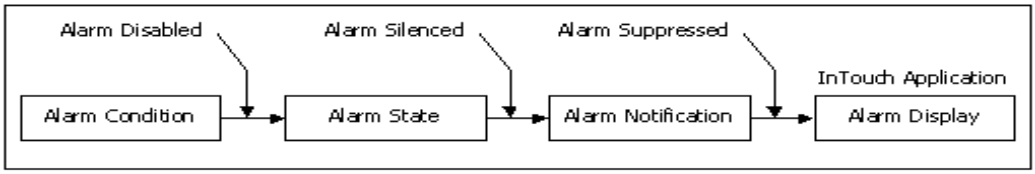
- Enabled: All alarms for an object are reported to client applications and saved as historical data. The enabled state is less restrictive than the silenced or disabled alarm states.

- Silenced: All alarms for an object are detected and logged in the Historian alarm and event history database. Silenced alarms are not logged to the InTouch alarm database, and are not shown in alarm clients displaying current alarms or recent history alarms. The silenced alarm state is more restrictive than the enabled state, but less restrictive than the disabled state.
- Disabled: No alarms for the object are detected. The alarm is return-to-normal until the alarm is re-enabled. The disabled state is more restrictive than the silenced and enabled alarm states. A disabled alarm does not require acknowledgement.

When an alarm state changes from silenced to enabled, the following applies:

- Only the last occurrence of the alarm appears in alarm clients.
- Only the last occurrence of the alarm is logged into the alarm database.
- Only the last occurrence of the alarm is saved to history.

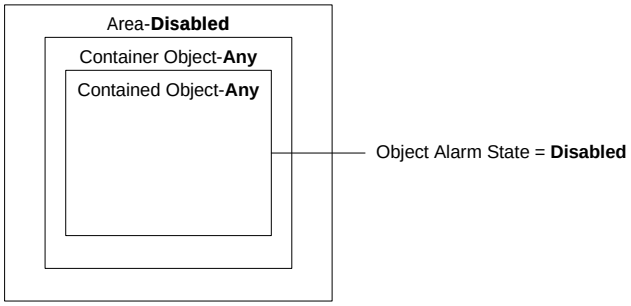
The following figure shows how the different alarm modes affect the different phases of an alarm. In the case of Alarm Suppressed, selected alarms can be filtered out of the InTouch AlarmViewer display by an operator, or programmatically by a script.



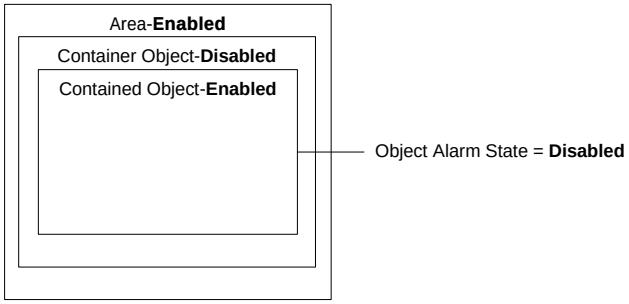
To ensure that alarmed attributes always have current data, the alarm feature always registers a reference to the alarmed attribute. This guarantees that Message Exchange never suspends updates for this attribute. Even if alarms are disabled for a particular attribute, the alarmed attribute cannot enter an Advanced Communication Management Suspended state.

The object hierarchy and alarm states determine the final alarm condition of an object.

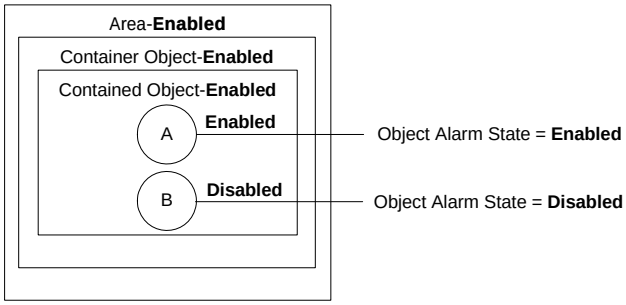
- An Area object's alarm state determines the alarm state for all alarms of objects that belong to the Area.



- The most restrictive setting within an object hierarchy determines the object's alarm state.



- When an individual object alarm is silenced or disabled, it applies only to that alarm and not to other alarms belonging to the object.



The alarms on any contained object are not affected. The disabled or silenced state of an individual alarm does not propagate downward through the object hierarchy to the alarms of any contained or assigned object.

Enabling Alarms

To enable an object's alarms, you must ensure that the AlarmModeCmd and AlarmInhibit attributes are enabled for the object, its container, and its area. An event, including the user's name, is generated indicating the object's alarms are enabled.

When object alarms are enabled, you can enable, silence, or disable an individual alarm.

Silencing Alarms

When object alarms are silenced, an individual alarm that is enabled or silenced is forced to be silenced. When object alarms are silenced, an individual alarm can be disabled.

Disabling Alarms

When object alarms are disabled, an individual alarm that is enabled or silenced is forced to be disabled.

When object alarms are enabled and an individual alarm is enabled or silenced, the individual alarm can be inhibited. This forces the individual alarm to be disabled.

When object alarms are silenced and an individual alarm is enabled or silenced, the individual alarm can be inhibited. This forces the individual alarm to be disabled.

When object alarms are inhibited, an individual alarm that is enabled or silenced is forced to be disabled.

Shelving Alarms

You can shelve alarms to temporarily hide them from displays for a fixed period. Alarms continue to be historized, even when they are shelved.

Shelving typically is used to reduce alarm “noise”, or to temporarily suppress alarms that might be triggered during system modifications or repairs, allowing you to focus on correcting other more urgent alarms.

Shelving is similar to silencing an alarm, but shelved alarms differ from silenced alarms in the following ways:

- Shelved alarms have a built-in associated timeout. Shelved alarms are automatically unshelved when the configured shelving period expires. You can also manually unshelve alarms and return them to an active, displayed state.
- Alarm shelving must be enabled at an area level, but shelving applies only to individual alarms. You cannot shelve a hierarchy of alarms for an entire area or for an entire object. You cannot propagate alarm shelving through the Model View hierarchy.
- The system enforces role-based limitations on permission to shelve alarms, alarm severity levels that can be shelved, and the total number of alarms a user can shelve.

The system tracks who shelved the alarm, from what workstation, the reason for shelving the alarm, when shelving began, and when shelving will expire. Shelved alarms aggregate in similar fashion to silenced alarms. For information about alarm aggregation, see “Obtaining Aggregated Alarm Severity Status Information at Run Time” on page 273.

A set of seven attributes provide run-time alarm shelving information and control:

AlarmShelveCmd	User writeable. Use this attribute to shelve or unshelve an alarm. Default values: Duration = 0, Reason = “”
AlarmShelved	Read-only, Boolean value. Shows True if alarm is shelved, False if alarm is unshelved. Default value: False
AlarmShelveStartTime	A read-only date/time stamp indicating when alarm shelving began, based on the engine time when the shelving request was received, Default value: Blank
AlarmShelveStopTime	A read-only date/time stamp equal to the AlarmShelveStartTime plus the duration for which the alarm is to be shelved. Default value: Blank
AlarmShelveReason	A read-only string value providing the reason for which the alarm was shelved or un-shelved by the Alarm Shelve command. Default value: Blank (See AlarmShelveCmd attribute.)

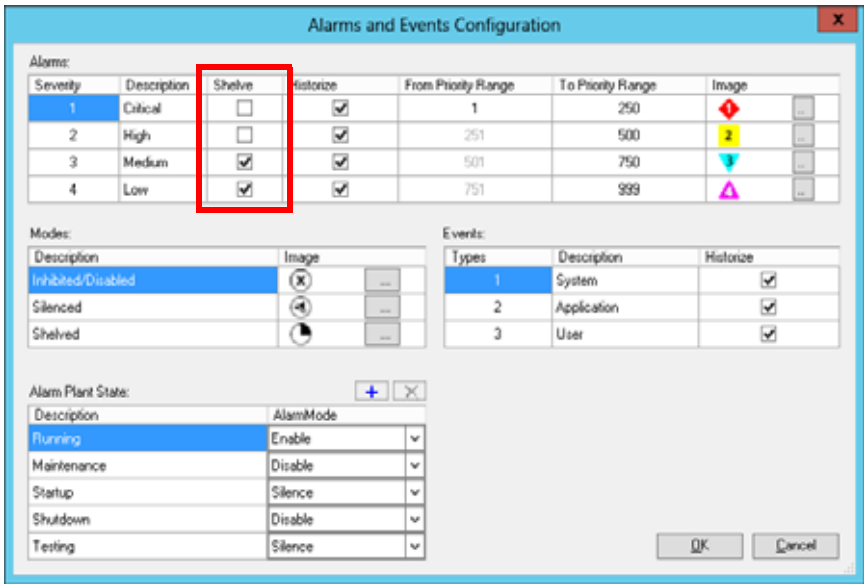
AlarmShelveUser	Read-only, the name of the user who most recently shelved or un-shelved the alarm with the Alarm Shelve command. Default value: Blank
AlarmShelveNode	Read-only, the name of the computer node from which the user most recently shelved or un-shelved the alarm with the Alarm Shelve command. If the node is hosted in a Terminal Server client session, the node and the TSE ID are both identified. Default value: Blank

Enabling Alarm Shelving

Alarm shelving is configured from the IDE **Configuration** menu, and is enabled on the Area object.

To enable alarm shelving

1. Enable at least one security role to configure alarm shelving. Associate the relevant user with that role, if not already associated, before proceeding to step 2.
2. On the IDE main menu, click **Galaxy**, then click **Configure**, then click **Alarms and Events Configuration**. The **Alarms and Events Configuration** dialog appears.



3 Select the severity level in the shelving column. By default, severity levels 3 (medium) and 4 (low) are enabled.

4 In the IDE, open the relevant Area object editor. Click the **Shelve Alarms** checkbox to enable shelving for that area. Shelving is enabled by default.

Shelving Alarms at Run Time

Use a run-time client to shelve and unshelve alarms.

From Application Server, you can use Object Viewer to monitor and control shelved alarm attributes.

From a Managed InTouch application, you can use an embedded ArchestrA Alarm Control and at least one animation to write to the AlarmShelveCmd attribute. For information about using the ArchestrA Alarm Control, see the *Wonderware ArchestrA Alarm Control Guide*.

To shelve or unshelve an alarm in Object Viewer

1 In Object Viewer, select the object in the **Console Tree** (left pane) which contains the alarm you want to shelve. The object attributes display in the **Details Pane** (right pane).

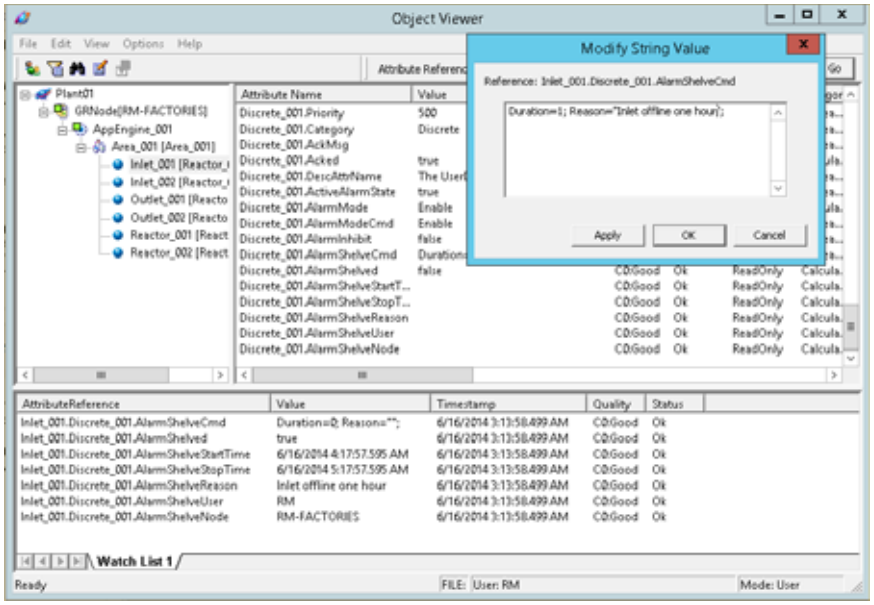
2 In the Attributes list, find the attribute configured with the alarm you want to shelve.

For example, “Inlet_001.Discrete_001”.

3 In the Attributes list, below the attribute in step 2, find the **AlarmShelveCmd** attribute.

For example, “Inlet_001.Discrete.001.AlarmShelveCmd”.

4 Double-click the .AlarmShelveCmd attribute. The **Modify String Value** dialog opens.



a To shelve an alarm, enter a duration in hours or in decimal fractions if the duration is to be less than one hour. The duration cannot be blank.

b Enter a reason between the quotation marks for shelving or unshelving. The reason cannot be blank. Click **Apply**.

c To unshelve a shelved alarm, enter a value of “0”.

The six read-only alarm shelving attributes will display the current shelving status information.

Throttling Alarms

Alarm throttling prevents network and message queues from flooding during periods of high alarm activity. Throttling sets a maximum number of transitions into and out of an alarm state within a defined period. Alarm transitions that exceed the throttling limit are not reported.

The WinPlatform object includes tuning parameters that specify the maximum alarm rate per second on an engine. The Alarm throttle limit attribute must be used in conjunction with the scan period to determine the maximum number of alarms that can occur in a scan cycle. You can set the alarm throttle limit when you configure the WinPlatform object. For more information about setting the scan period and alarm throttle limit, see “Configuring WinPlatform Object Alarms” on page 254.

Alarm messages can be throttled for alarms going into alarm and out of alarm state. Alarm acknowledgement messages are not throttled. Users can still acknowledge alarms even when alarm throttling is active. Users can still disable alarms for objects or areas when the alarm rate causes throttling to occur. Also, the alarm inhibit and the disable/enable/silence messages are not throttled.

If an active alarm is disabled, the going out of alarm and disabled messages are sent to the notification distributor. If alarms are being throttled, the going out of alarm message can be throttled. The disabled message is accepted regardless of the throttling status. The going out of alarm message is not raised again. This can result in never logging a message for that alarm indicating it went out of alarm. The final alarm list in the Notification Distributor still shows the correct alarm state.

Propagating Timestamps with Alarms and Events

An alarm feature always registers a reference to the alarmed attribute. This registered reference guarantees that Message Exchange never suspends updates for the alarmed attribute. Even if alarms are disabled for a particular attribute, the Advanced Communication Management feature cannot suspend the attribute.

Be aware that the time stamp propagates to all masked bits of an integer attribute, even if only one of the bits changes.

For example, you have an Integer address in a PLC that represents 16 different alarm states. You assign ObjA.Attr_Integer to point to the PLC address. You then split the bits to different alarm attributes, adding one attribute for each alarm and naming them ObjA.Attr_Alarm00 to ObjA.Attr_Alarm15. Each attribute has an input source that refers to a different bit of ObjA.Attr_Integer. For example, ObjA.Attr_00.InputSource -> me.Attr_Int.00, and so on. At run time, when bit 00 is changing in the PLC, all of the attributes (ObjA.Attr_Alarm00 to ObjA.Attr_Alarm15) get a new time stamp, as all the bits changed. This can cause incorrect time stamps for alarms.

Alarms or Events Become Active

For alarms or events, if an attribute has no timestamp, the current AppEngine scan time is used instead. If an attribute has a timestamp, the timestamp of the value that caused the alarm to occur is used as the value for the object's AlarmOnTime attribute.

When an alarm is silenced or disabled, and an alarm condition becomes TRUE, when the alarm is later enabled, the timestamp for AlarmOnTime is the timestamp of the most recent attribute change, not the time at which the alarm was enabled.

For an event, if an attribute has a timestamp, the timestamp of the value that caused the event to be reported is used for the EventTime. A System Event timestamp is the current system time.

Alarms Become Disabled

If an active alarm becomes disabled, the alarm is forced to return to normal. The timestamp corresponds to the current time when the alarm became disabled, not the timestamp of the attribute nor of the alarm condition. Otherwise, the assigned timestamp is the AppEngine scan time.

Alarms Revert to Normal

If an attribute value has a timestamp, the timestamp of the value that caused the alarm to revert to normal is used for the AlarmOffTime.

If alarm is based upon several attributes or upon several values of a single attribute, the most recent timestamp is used when assigning values to the AlarmOnTime or AlarmOffTime.

Alarm Acknowledgement

Alarms are acknowledged by users who view unacknowledged alarms from their client applications like InTouch HMI. Only alarms of certain priority levels need to be acknowledged.

The basic workflow to acknowledge an alarm consists of the following general steps:

- The alarm is detected and reported to any subscribed alarm clients. The alarm is unacknowledged unless it is of a priority level that does not require acknowledgement.
- An authorized user attempts to acknowledge the alarm from the client. The user can type an optional comment when acknowledging the alarm.
- The user's acknowledge request is sent to the detecting object's alarm feature. The acknowledgment must pass through standard security checks first. The acknowledge request contains the user's name and any alarm comment.

For the alarm utility, the alarm is acknowledged within the alarm feature immediately. The user name, comment, and acknowledged time are also saved. Alarm comments can be localized into any supported language. See “Working with Languages” on page 373.

The acknowledge is considered an alarm state change, which is sent to all subscribed clients. When an alarm is acknowledged, the current AppEngine timestamp is used as the acknowledgement time.

After an alarm is acknowledged for the first time, any additional (extra) acknowledgement attempts for the same alarm are rejected and an error is returned.

Acknowledging Alarms with Signature Required

You can use the SignedAlarmAck( ) script function in ArchestrA Graphics to require a signature to acknowledge alarms. Add the script function to a symbol as described in the *Creating and Managing ArchestrA Graphics User's Guide*, and then embed the symbol in the WindowMaker window.

When you acknowledge the alarms at run time, the script function checks for a required signature. If a signature is required, it checks Galaxy security settings and alarm priority. If any alarm in the list falls within the configured conditions, you must provide your signature to acknowledge the alarms. You must sign in to acknowledge the alarms if no user is logged on to the InTouch application.

If any alarm in the list is requires a signature, then a signature is required for the entire list of alarms.

Note: The SignedAlarmAck() script function supports only ArchestrA alarms.

By inspecting the Alarm Viewer or the Alarm Control client, you can determine if the alarm acknowledgements were successful.

Configuring Alarms

Alarming capabilities are a part of object templates, but they are not implemented until you configure the object in the IDE. After alarms are configured, you can view Application Server alarms in a client visualization application like InTouch HMI.

Configuring an object to be an alarm provider includes the following general sequence of steps:

- Decide whether alarm notification is needed for each possible alarm condition of an object. For example, a command time-out alarm for a valve if the output command fails to move the valve.
- Edit the object and set an attribute that specifies alarming.
- Edit the object from the IDE and assign values to alarm attributes.
- Configure the alarm properties. Typically, the fields that require configuration are Category, Priority and Description.
- Configuring any limit fields to set an alarm. For example, the feedback time-out time limit.

You can add alarm detection and reporting capabilities to objects that were not originally developed to detect alarms. You do this by configuring alarm features for the object's attributes.

Configuring Alarming for System Objects

To enable alarming for your application, you need to configure your Galaxy's WinPlatform and AppEngine objects as alarm providers. Both system objects report their own alarms.

Client applications subscribe to application object alarms by the area containing the objects. Client applications can also subscribe to WinPlatforms and AppEngines directly. These are called "pseudo-areas." They do not need to be assigned to an area for a client to see the alarms, although the user may want to assign them to an area, such as for simplifying the alarm query in the InTouch Alarm Viewer or Alarm DB Logger.

The following list shows the alarms for each system object.

- WinPlatform
  - Excess CPU load alarm
  - Low disk space alarm
  - Excessive page faults alarm
  - Low memory alarm
  - Engine failure alarm
  - Engine checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition

  A WinPlatform object includes a general communication alarm when it loses contact with the areas to which it is subscribed.
- AppEngine
  - Checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition
  - Redundancy failover alarm
  - Redundancy Standby unavailable
  - Redundancy Standby not ready

- ViewEngine
  - Checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition

The Area and InTouchViewApp system objects do not include any alarms.

Configuring WinPlatform Object Alarms

You select the areas of your Galaxy to monitor for alarms from the WinPlatform object. Also, you can select specific alarms for the status of the WinPlatform itself.

You must specify that the WinPlatform object is an InTouch alarm provider to subscribe to alarms from the various areas (and pseudo-areas) of the Galaxy and report them to the InTouch Alarm Manager.

Configuring Communication Failure Alarm Priority

InTouch Alarm Provider reports any communication outage to the InTouch Alarm Manager. Communication alarms are displayed and must be acknowledged. By default, the priority of a communication failure is “1”, the highest priority.

There can be instances when a communication failure is not a high-priority alarm, and should not require the attention of a high-priority alarm. Shutting down or undeploying a platform as a normal operation can result in communication failure alarms, for example.

To indicate whether the alarm is of high or low importance, the communication failure alarm priority is configurable at design time as an integer of range 1–999. InTouch Alarm Provider must be enabled in the WinPlatform object editor, **General** tab.

Selecting the Register using “Galaxy_<GalaxyName>” instead of “Galaxy” option

When you select the **Register using “Galaxy_<Galaxy name>”** instead of **“Galaxy”** option, you enable ITAlarmProvider contained in WinPlatform to register the ITAlarmSubsystem using a provider name of Galaxy_GalaxyName.

For example, if your galaxy is named Andromeda, the ITAlarmSubsystem registers as Galaxy_Andromeda.

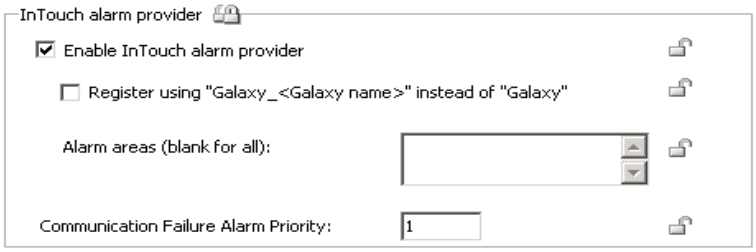
This configuration can be useful if you are using an Alarm Client to display alarms from multiple galaxies. For more information about multiple galaxies, see Chapter 9, "Working with Multiple Galaxies."

- This means that all InTouch applications which query alarms from this platform should be modified to use Galaxy_GalaxyName!AreaName.
- Select this option when the ArchestrA Alarm Control, the embedded alarm client or EAC, or the AlarmViewControl in InTouch queries alarms from different galaxies.

To configure a WinPlatform object to be an alarm provider

1 Open the WinPlatform object in the Object Editor.

2 Click the **General** tab.



3 Select the **InTouch alarm provider** check box.

4 Select the **Register using "Galaxy_<GalaxyName>"** instead of **"Galaxy"** check box to enable Galaxy_<GalaxyName> registration for alarm comment language switching. An information box appears. Click **OK** on the information box to continue. See "Working with Languages" on page 373.

5 In the **Alarm Areas** box, type the names of areas to subscribe to for alarms.

If you leave the **Alarm Areas** box blank, the WinPlatform subscribes to all areas in the Galaxy.

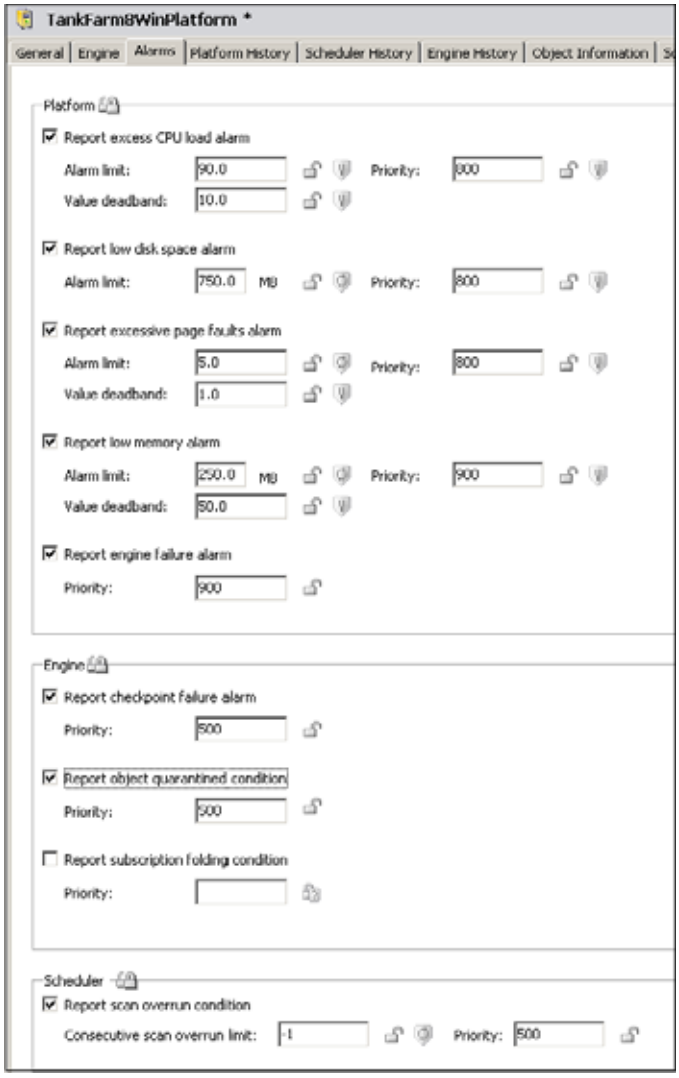
If you want to subscribe to only selected areas within the Galaxy, insert a space between each area name. For example:

Area1 Area2 Area3

6 Enter a number from 1 to 999 to configure the **Communication Failure Alarm Priority**.

7 Click the **Engine** tab to show the **Alarm throttle limit** box. Either accept the default throttle limit of 2000 alarms per second, or enter another value. A value of 0 disables alarm throttling. For more information about alarm throttling, see "Throttling Alarms" on page 249.

8 Click the **Alarms** tab to show platform, engine, and scheduler alarms that can be set for the WinPlatform object.



9 Select the check box next to each alarm that you want to enable for the WinPlatform object.

10 Set the limit, value deadband, and priority for each alarm you selected.

11 Save and close the Object Editor.

12 Check the object in to the Galaxy.

13 Deploy the object in an on scan state.

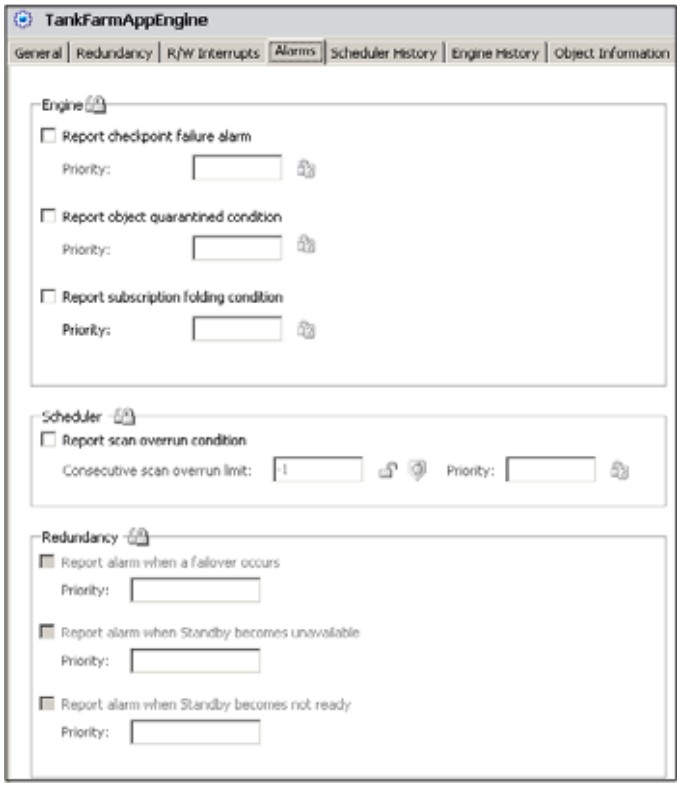
Configuring Alarms for an AppEngine Object

You can set AppEngine attributes that determine whether alarms are enabled for an engine, scheduler, and redundant failover engine.

To configure AppEngine object alarms

1 Open the AppEngine object in the Object Editor.

2 Click the **Alarms** tab to show engine, scheduler, and redundancy alarms that can be set for the AppEngine object.



3 Select the check box next to each alarm that you want to enable for the AppEngine object.

4 Set the priority for each alarm you selected.

5 Save and close the Object Editor.

6 Check the object in to the Galaxy.

7 Deploy the object in an on scan state.

Configuring Alarms and Events for Application Objects

The following table shows the different types of alarms that can be specified for application objects. The table shows the application objects containing native alarm attributes.

You can also set bad value and state alarms for an object's attributes. For more information about setting alarms for attributes, see "Setting Alarms on the Attributes Page" on page 261.

	Alarm Types				
Application Objects	State	Limit	Target Deviation	Rate of Change	Statistical
AnalogDevice	●	●	●	●	
Boolean					
DiscreteDevice	●				●
Double					
FieldReference					
Float					
Integer					
Sequencer	●				
String					
Switch	●				
UserDefined	●	●	●	●	

The following list shows the types of alarms for each application object in more detail.

- AnalogDevice
  - Level alarms (HiHi, Hi, Lo, LoLo) [limit alarms]
  - Rate of Change alarms (Up, Down)
  - Target Deviation alarms (Minor, Major)
  - PV Bad Value alarm [state alarm]

- DiscreteDevice
  - Uncommanded change alarm [state alarm]
  - Command time-out alarm [state alarm]
  - Active1 state alarm [state alarm]
  - Active2 state alarm [state alarm]
  - Fault state alarm [state alarm]
  - Active1 state duration alarm [statistical alarm]
  - Active2 state duration alarm [statistical alarm]
- Sequencer
  - Execution halted [state alarm]
  - Condition trigger failure [state alarm]
  - OnEntry output failure [state alarm]
  - OnExit output failure [state alarm]
- Switch
  - PV State alarm [state alarm]
- UserDefined (Attributes can be alarmed)
  - PV State alarm [state alarm]
  - PV Bad Value alarm (that is, bad quality) [state alarm]
  - Attribute alarm features

    State alarms

    Limit alarms (HiHi, Hi, Lo, LoLo)

    Rate of Change alarms (Up, Down)

    Deviation alarms (Minor, Major)

    Bad Value alarm (that is, bad quality)

There are no built-in alarms for these application objects:

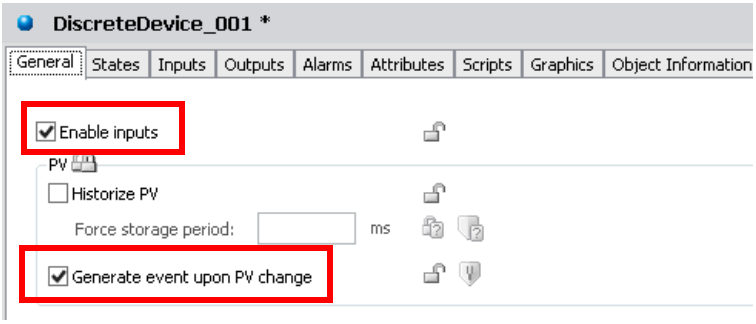
- FieldReference
- Boolean
- Double
- Float
- Integer
- String

You can also configure your application objects to generate an event each time the object's PV value changes. In addition, you can configure an alarm feature on any object for any Boolean attribute.

To configure alarming and events for application objects

1 Open the application object with the Object Editor.

2 For a DiscreteDevice, click the **General** tab, enable inputs, and then enable **Generate event upon PV**.

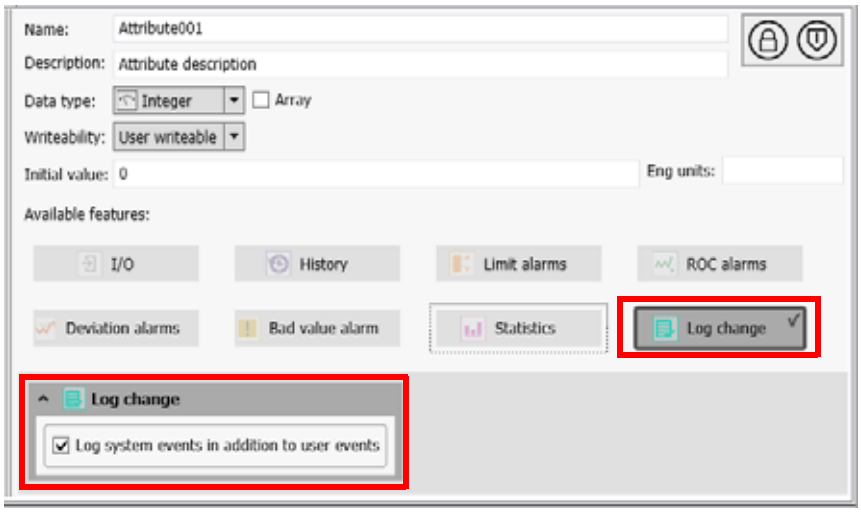


For objects that do not include this check box, use the **Attributes** page.

Note: If you are using field attributes instead of Attributes, use the **Field Attributes** page.

a Enable the **Log change** feature.

b Enable **Generate event upon change**.



3 Select or clear the check box based on whether you want to generate an event each time the object's PV value changes.

4 Click the tab that lists alarm attributes.

- For the AnalogDevice object, click **Alarms**.
- For the DiscreteDevice object, click **Alarms**.
- For the Sequencer object, click **Settings**.
- For the Switch object, click **General**.

- For the UserDefined object, click **Attributes** (or **Field Attributes**, if you are using field attributes). See “Setting Alarms on the Attributes Page” on page 261 for more information.

5 Select the check box that enables alarming for the object.

6 Assign values to the attributes for the type of alarm you selected by completing the following steps:

a Assign values to the alarm limits based on the type of alarm.

b Assign an alarm priority (1-999) for each limit you set.

c Accept the default alarm message or include another message for each alarm limit.

d Assign values to the remaining attributes based on the type of alarm you selected. For more information about other alarm attributes, see “Types of Alarms” on page 236.

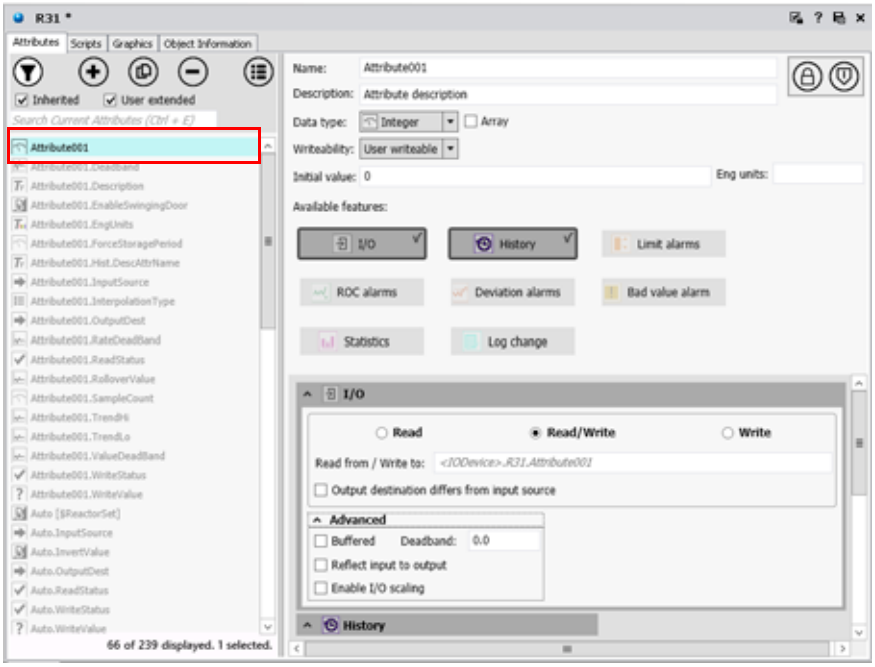
7 Save your object changes and close the Object Editor.

Setting Alarms on the Attributes Page

You set alarms for all Application Server objects in the Attributes page in the Object Editor.

To specify alarms for an object’s attributes

1 On the **Attributes** page of the Object Editor, select an attribute from the **Attributes List**.



2 Select the alarm features you want to configure for the attribute. For Boolean data types, you can select:

- **State alarm**
- **Bad value alarm**

For Integer, Float and Double data types, you can select:

- **Limit alarms**
- **ROC alarms**
- **Deviation alarms**
- **Bad value alarms**

3 For each alarm feature that you activate, enter the settings for each alarm.

a Assign values to the alarm limits based on the type of alarm.

b Assign an alarm priority (1-999) for each limit you set.

c Accept the default alarm message or include another message for each alarm limit.

d Assign values to the remaining attributes based on the type of alarm you selected. For more information about other alarm attributes, see “Types of Alarms” on page 236.

4 To configure event logging for an object, select the **Log change** feature and click the checkbox “Generate event upon change.”

5 Save your object changes and close the Object Editor.

Distributing Alarms and Events

After you configure object instances for alarm detection, deploy the instances and put them On scan. The instances begin checking for alarm conditions.

When an alarm is detected, or an event occurs, a notification is reported to its alarm and event distributor, which is running on the same AppEngine.

These alarm and event distributors include:

• Area objects	Area objects report detected alarms through the Area, which distributes them to alarm and event clients.
• WinPlatform objects	Report their own alarms and events.
• AppEngine objects	Report their own alarms and events.
• Device IntegrationObjects	Report their own alarms and events.

The Area object plays a key role in alarm and event distribution. All objects belong to an Area. Areas can contain sub-Areas. Alarm and event clients are configured to subscribe to a set of Areas.

Areas provide a key organizational role in grouping alarm information and assigning it to users who use alarm and event clients to monitor their Areas of responsibility.

WinPlatforms, AppEngines and Device Integration objects do not report their alarms and events to Area objects even though they belong to Areas. This allows alarm clients to receive alarm notifications without any dependencies on Area objects. For example, a deployed and running WinPlatform can report alarms even though its Area is not deployed and running.

Alarm-event distributor objects maintain a list of all currently active alarms and inactive but unacknowledged alarms. They do not maintain a list of events, which are routed to clients that are currently subscribed at the time of the event.

You can configure a WinPlatform to act as an InTouch Alarm Provider in the run-time environment.

The WinPlatform sends an alarm through the InTouch Distributed Alarm System to InTouch clients when the WinPlatform loses communication with an Area that it subscribes to. This condition typically occurs during a network outage with computers hosting those Areas.

In a network outage, the WinPlatform InTouch Alarm Provider sends an alarm for each disconnected Area that it subscribes to, including all of its alarm distribution hierarchy. Each of these alarms is a high priority alarm that contains the name of the Area to which communication is lost. These communication problem alarms must be acknowledged.

Although they still appear in the historical record, any current alarms from the disconnected Area drop from the InTouch client's summary list. They can no longer be acknowledged.

When communication to the disconnected Areas is restored, any unacknowledged alarms generated in those Areas are sent to the alarm client.

Subscribing to Alarms and Events from a Client

Clients indicate interest in alarms and events by subscribing to an Area. When subscribing to an Area, the subscription is actually to all notification distributors within that Area.

For example, if an Area contains sub-Areas, those sub-Areas are subscribed to. If WinPlatforms, AppEngines or Device Integration objects belong to an Area, those objects are also directly subscribed to.

When a notification distributor receives an alarm and event subscription from a client, the notification distributor provides the client with the following:

- A list of all current alarm conditions, including unacknowledged return-to-normal conditions.
- An alarm condition state change. A state change includes transitions into or out of alarm (return to normal) and change in acknowledged flag.
- An event occurrence.

Alarm and event subscription requests do not include filters, for example, only show alarms greater than a specific priority value. All alarm and event messages received by the notification distributor are sent to all subscribed clients. Filtering is provided as a display option by clients.

Using InTouch HMI as the Alarm and Event Client

InTouch run-time clients subscribe to event reports from a Galaxy. Application Server reports alarms to the InTouch Distributed Alarm System, which subscribe to alarm and event reports from a Galaxy.

An InTouch client application can visualize Application Server components. An InTouch alarm client can show alarm information for new, unacknowledged alarms, including all required fields.

The new alarm is in the unacknowledged state. An operator can view alarms, acknowledge alarms, disable alarms, and enable alarms from the client application running in InTouch WindowViewer.

Understanding the Syntax of Alarm Queries

InTouch alarm queries subscribe to alarm and event information from objects within a Galaxy. The alarm and event queries can be in the form of user input or a script.

Alarm query syntax must be in one of the following forms:

\Provider!Area

or

\\Node\Provider!Area

You can have one or more references in a query separated by spaces.

You can also optionally append a tagname filter at the end, separated by another exclamation mark:

\Provider!Area!Filter

\\Node\Provider!Area!Filter

The filter can have a wildcard * character at the beginning or at the end, but not both.

The \\Node at the beginning is only important if you want to query for alarms from a provider on another computer. Otherwise, you can leave it off and the reference is assumed to be a provider on the local computer. The provider name Galaxy refers to alarms and events that get reported by the WinPlatform configured as an InTouch alarm provider on that computer node.

Alarm Query Syntax when Register Using Galaxy_<GalaxyName> is Enabled

In the WinPlatform object, when you enable InTouch alarm provider, you can enable **Register using Galaxy_<GalaxyName>** instead of **Galaxy**. This option will register the platform to the alarm subsystem using the Galaxy name prefixed by "Galaxy_" instead of just the word "Galaxy". This allows an InTouch application to monitor alarms from multiple Galaxies and avoid name conflicts.

Syntax changes slightly when Galaxy_GalaxyName is enabled:

- Use \\ for computer name.
- Use \ for Galaxy or Galaxy_<GalaxyName>.
- Use ! for Area.

For example: \\Galaxy\MyGalaxy!Area001.

If Galaxy_GalaxyName is not enabled in WinPlatform, then the default behavior described in the previous section applies.

You can determine if Galaxy_<GalaxyName> has been enabled by monitoring the run-time attribute of the platform ITAlarmProvider.ProviderNameAsGalaxyNameEnabled.

Examples of Alarm Queries

- You can submit a query to get all alarms from Area1 and all other alarms within Area1, as reported by the WinPlatform object on the local computer.

  ```
  \Galaxy!Area1
  ```

  The query returns all alarms and events from all objects directly contained in Area1 and any sub-areas contained by Area1. This hierarchy is determined by what is configured in the Model View in the IDE.

- If Area1 and Area2 are two separate mutually exclusive areas, you can submit a query for alarms from both areas.

  ```
  \Galaxy!Area1  \Galaxy!Area2
  ```

- If you're on NodeA and the WinPlatform is on NodeB, you can submit a query for the alarms from the remote computer.

```
\\NodeB\Galaxy!Area1
```

- You can submit a query for all alarms from objects whose name begins with “Tank” in the TankFarm1 area.

  ```
  \Galaxy!TankFarm1!Tank*
  ```

  The trailing wildcard character matches alarms from all objects with names that begin with “Tank” like Tank001, Tank002, TankUpper, or TankLower.

- You can submit a query for specific alarm types. For example, you can submit a query for all HiHi alarms in the TankFarm1 area.

  ```
  \Galaxy!TankFarm1!*.HiHi
  ```

- You can submit a query for all types of alarms from a specific object within an area.

  ```
  \Galaxy!TankFarm1!Tank752.*
  ```

  The trailing wildcard character matches all alarm types for Tank752.

Alarm Requirements for InTouch Client Applications

For Application Server alarming to function, the following conditions must be met:

In Application Server:

- One or more Area objects are deployed and running.
- The source object is on scan.
- The source object’s Area is on scan.
- Alarming must be enabled for the target object.
- An InTouch alarm provider on any WinPlatform in the Galaxy.

In InTouch:

- The InTouch client application is running in WindowViewer.
- An InTouch alarm ActiveX control is placed in a window and configured as an alarm consumer for the Galaxy.
- The user is logged into InTouch using ArchestrA security and is authorized to acknowledge alarms for the object that is in the alarm state. If the user only wants to view alarms, security authorization is not required.

Application Server validates the user has sufficient security privileges to acknowledge the alarm.

If the user does not have privileges to acknowledge alarms, the user can attempt to acknowledge the alarm, but the Galaxy rejects the acknowledgment request.The alarm remains unacknowledged in the InTouch Alarm display.

The rejected alarm acknowledge event is recorded in InTouch Event History if the user attempting the acknowledgement has a valid Galaxy user account. Otherwise, the rejected acknowledgement is not recorded as an event.

You can acknowledge multiple alarms by providing a valid signature. At run time, the ArchestrA Graphics SignedAlarmAck() script function checks a set of alarms and conditions to find out if any alarm from the list requires a signature in order to acknowledge it. If so, you must provide your user name, password and domain or your Smart Card details and PIN to authenticate yourself and acknowledge the alarms. To be able to provide the Smart Card details, your computer must be configured for Smart Card authentication.

Note: Smart Card authentication is not supported in multi-galaxy environments for read/write operations to remote galaxies.

If even one alarm in the list requires a signature, you must provide a signature to acknowledge the alarms. The SignedAlarmAck() function will acknowledge ALL the alarms in the list.

With the SignedAlarmAck( ) function, you can provide credentials and acknowledge an alarm or group of alarms even if you are not the logged-on operator.

Alarms and Events in the InTouch HMI and in Application Server

Both InTouch HMI and Application Server implement alarms and events. The following table shows the similarities and differences between the two products.

Item	InTouch	Application Server
Alarm configured or detected by	Within a tag	Within an object
Alarm Classes (client column)	Only certain classes of alarms are supported or detected: DSC, VALUE, DEV, ROC, SPC.	No system-wide distinction for classes. Alarms are tied to a Boolean that can be triggered from any logic.
Alarm Type (Sub-class) (client column)	Discrete, LoLo, Lo, Hi, HiHi, MinorDev, MajorDev, ROC, SPC. Client column.	No sub-class. The Alarm feature name is the closest concept. For example, ".PVHiAlarm". Mapped from Category.

Item	InTouch	Application Server
Priority (client column)	1-999 (1 most urgent)	Priority 0-999. 0 most urgent. 0 is mapped to 1 in InTouch.
Name (client column)	Alarm name = Tag name.	Object.attribute
Comment (client comment)	Separate alarm comment, which is different from the tag comment.	Object short description or alarm message where available.
Group	Alarm group allows client-side filtering. Sub-groups must be on same InTouch.	No alarm group. But Area provides mappable concept. Sub-Areas can be on different nodes.
State	Four states, which are combinations of ACK/UNACK and ALARM/RTN • UNACK/ALARM (usually displayed as UNACK) • ACK/ALARM (usually displayed as ACK) • UNACK/RTN (usually displayed as UNACK_RTN) • ACK/RTN (usually displayed as ACK_RTN) These states have a 1:1 correspondence with states of the Alarm feature, which keeps track of whether the alarm is InAlarm and IsAcked. Alarms in the state ACK/RTN are not shown in the SUMMARY alarm display because they do not need any further action from the operator. But, all four states appear in the HISTORY display, and in the Alarm Database.	Alarm state provides equivalent concept and can be mapped.
Value, CheckValue	Only static values sent with alarm message.	Static values and dynamic references are provided.
Ack	All alarms sent to client and require acknowledgement regardless of priority.	All alarms sent to client and require acknowledgement regardless of priority.
History	Alarm state changes are logged to event history and shown on historical client.	Alarm state changes are logged to event history and shown on historical client.

Configuring Plant State-Based Alarms

You can map alarm modes on a per-Galaxy basis to different plant operational states to control how alarms are reported. Five plant states are pre-defined, and have default alarm states associated with them:

Plant State	Default Alarm State	Available Alarm States
Running	Enable	Enable
Maintenance	Disable	Enable / Silence / Disable
Startup	Silence	Enable / Silence / Disable
Shutdown	Disable	Enable / Silence / Disable
Testing	Silence	Enable / Silence / Disable

You can define new plant states, rename plant states, or remove existing plant states, except the “Running” state (you can, however, rename “Running”). The alarm state for Running is read-only and cannot be changed from Enable.

After you have defined plant state-based alarm configurations for the Galaxy, you can assign plant state-based alarming to area objects in the Galaxy. This is done by setting the PlantState attribute for each area object that will use plant state-based alarming. The area object will automatically update its PlantAlarmMode attribute to match the alarm state that is set for the PlantState currently assigned to it.

Attribute	Definition
PlantState	The name of the currently assigned plant state (Running, Maintenance, etc.).
PlantStateAlarmMode	The alarm state of the assigned plant state (enable, silence, or disable). This attribute is read-only at run time.

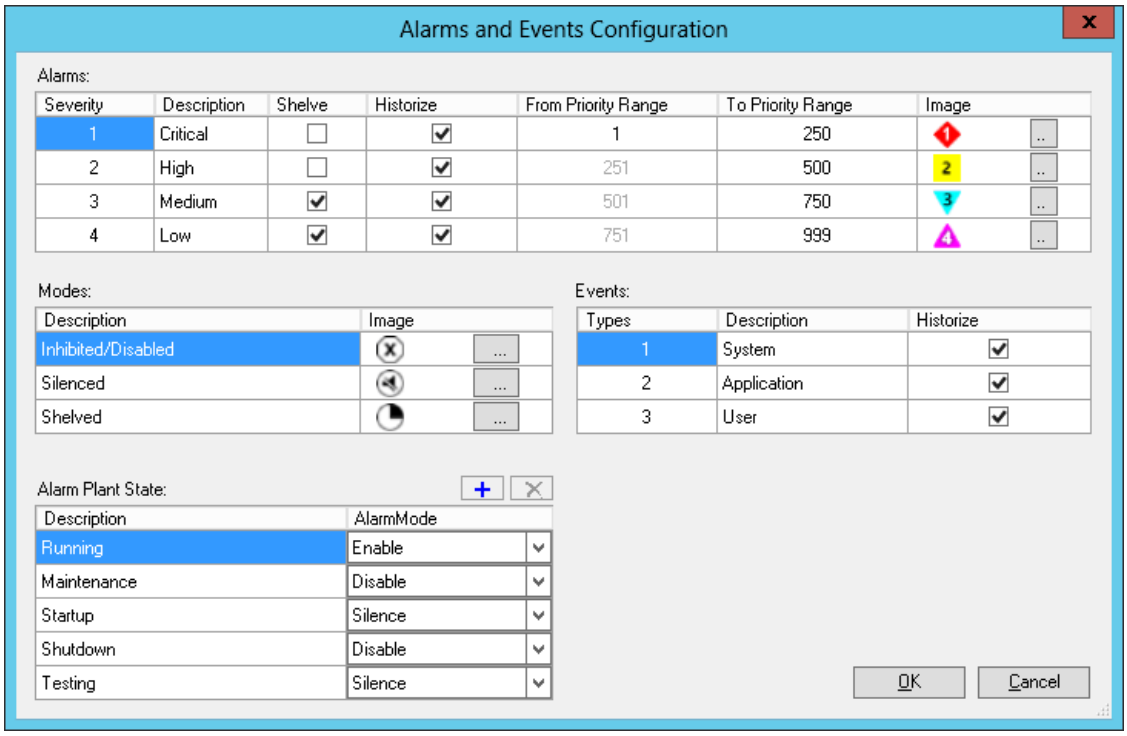
Mapping Alarm Modes to Plant States

Define plant states and map them to alarm modes in the IDE. To access the configuration dialog:

1. Click **Galaxy** on the IDE menu.
2. Select **Configure**, then select **Alarms and Events Configuration**. The **Alarms and Events Configuration** dialog opens.

Note: Settings in this dialog are not shared across Galaxies in a multi-Galaxy environment. Each Galaxy in the environment will have its own Alarms and Events Configuration.

To use the default values, you do not have to set anything in this dialog. Simply enable plant state-based alarms for each area object that is to utilize this feature. See “Configuring State-Based Alarming on an Area Object” on page 271 for additional information.



Change Alarm Modes and Modify Plant States

- To change the AlarmMode value, click the AlarmMode associated with the Plant State that you want to change. Then, select the new value (enable, silence, or disable).

Note: AlarmMode for Running is read-only and cannot be changed.

- To add a new Plant State, Click the **add** button. A new row is created at the bottom of the table.

  Enter the name of the new plant state. The maximum name length is 64 characters.

  Select a value for the AlarmMode (enable, silence, or disable).

- To delete an existing Plant State, click the **remove** button. The selected plant state and its associated value are removed from the table.

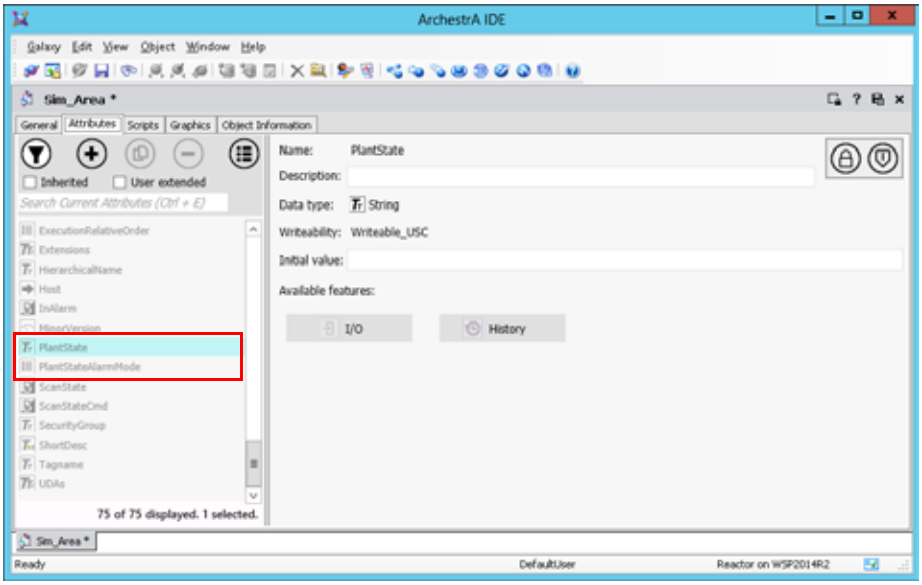
Note: The first row of the Alarm Plant State table is read-only and cannot be deleted. Running is the default PlantState name in the first row.

- To rename an existing PlantState, double click the plant state you want to rename, then enter the new name for the plant state. The maximum name length is 64 characters.

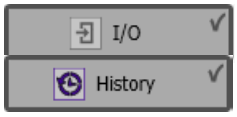
Note: All Plant States, including Running, can be renamed.

Configuring State-Based Alarming on an Area Object

State-based alarm modes apply to String attributes, with writeability set to Writeable_USC (these cannot be changed). To use state-based alarming for one or more areas, you must configure the PlantState attribute for each area that will use plant state-based alarming.



Configure plant state alarming



1. Open the area object and click the Attributes page.
2. Select the PlantState attribute.
3. Enable the **I/O** feature and select **Read**, **Read/Write**, or **Write**.
4. Enable the **History** feature and configure it.
5. Select the PlantStateAlarmMode.
6. Repeat steps 3 and 4 to configure the **I/O** and **History** features.

Viewing Plant State-Based Alarms at Run Time

You can view plant state-based alarming at runtime through clients such as Object Viewer and InTouch applications in WindowViewer.

User Access and Security During Run Time

Operators with the appropriate permissions can change the PlantState of an area through a run time client. An operator may have to change PlantState from Startup to Running, for example. Scripts can also be used to implement modifications.

To change a PlantState, the operator enters a string to match one of the defined PlantState values (for example, “Running”). Before a change to PlantState can be implemented, the system checks that the user belongs to a role that has the permission “Can Modify Plant State of an Area” and is therefore authorized to make the change. If the operator enters a string that does not match a defined PlantState, the change is rejected.

At run time, the AlarmMode attribute is read-only. Therefore, the AlarmMode for a PlantState cannot be changed through a run time client (for example, changing from enable to silence). It can only be changed through the IDE.

If an area is assigned to a PlantState and the PlantState is deleted through the IDE, the area will remain in that PlantState until it is changed. If AlarmMode for the deleted PlantState is anything other than enable, the AlarmMode will change to enable.

Note: Operators with appropriate permissions can use the AlarmModeCmd attribute to change AlarmMode of an area at run time. However, the AlarmModeCmd can only be used to set a more restrictive condition than the AlarmMode of the area’s corresponding PlantState. For more information about changing alarm modes, see “Enabling, Silencing, and Disabling Alarms” on page 242.

Obtaining Aggregated Alarm Severity Status Information at Run Time

Obtain aggregated alarm severity status information by:

1. Mapping alarm severity levels to priority ranges,
2. Configuring alarms on objects,
3. Enabling alarm aggregation by Area,
4. Viewing aggregated alarm status information by means of run-time clients and applications.

Mapping Alarm Severity to Priority

Use the IDE to map each of four alarm severities to priorities expressed as numeric ranges. Alarms can then be aggregated and displayed with the specified severity at run time.

Note: Mapped alarm severities are not shared across galaxies in a multi-galaxy environment. Each galaxy in the environment will have its own priority to severity mapping.

Configuring Alarm Severity to Priority Mapping

The **Alarms and Events Configuration** dialog allows you to map alarm severities to priority ranges, to enable alarm shelving by priority range, and to enable or disable historization of both alarm severities and event types. All alarms are historized by default.

Default Alarm Mapping and Historization Values

Severity	Description	Shelve	Historize	From Priority Range	To Priority Range
1	**Critical**	N	Y	1	250
2	**High**	N	Y	251	500
3	**Medium**	Y	Y	501	750
4	**Low**	Y	Y	751	999

Shelving, historization and priority ranges are configurable.

- Severity 1 starts at priority 1 by default. Severities can be mapped to priorities in ascending or descending order. For example, severity 1 can map to priority range 1–250 or it can map to priority range 999-751.

- Severity 4 ends at 999 by default, but 999 is not required to be the end-of-range number. For example, severity 4 can end at 900, leaving a priority range above 900 unmapped to any severity level.
- A priority outside the configured priority ranges will return severity 0, not mapped. Since the priority is unmapped, it cannot be historized.

Configuring Historization for Alarms and Events

Historization is configurable for both events and alarms. Event types and the default historization setting for each type are:

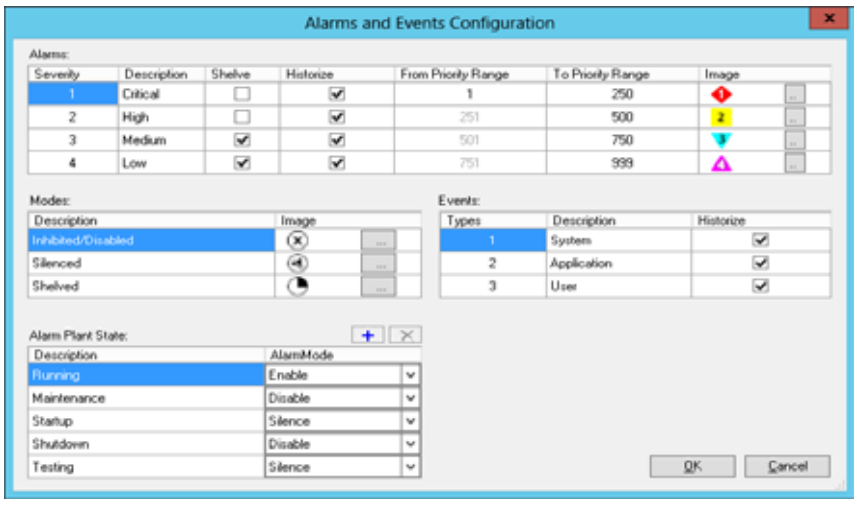
Default Event Type Historization Values

Types	Description	Historize
1	**System**	Y
2	**Application**	Y
3	**User**	N

Note: In prior versions of Application Server, historization of alarms and events was configured through the Alarm DB Logger Manager (an InTouch component) and utilized the SQL database "WWALMDB." The method described here replaces the Alarm DB Logger method.

To configure alarm and event mapping

1 On the IDE main menu, click **Galaxy**, then click **Configure**, then click **Alarms and Events Configuration**. The **Alarms and Events Configuration** dialog appears.



2 Configure alarms:

   a Edit the priority range for each severity level, or accept the default values. If you edit priority ranges and there are

   b Accept the default (enabled) historization or click to clear the check box and disable historization for each severity level.

3 Configure events: Accept the default historization values, or click to clear the check box and disable historization for each event type.

If you want to change the default Alarm Border indicator images shown in the **Alarms and Events Configuration** dialog, see “Changing Alarm Border Indicator Icons” on page 310 in the *Creating and Managing ArchestrA Graphics User’s Guide.*

Monitoring Alarm Severities at Run Time

Monitor alarms by severity, expressed as integers 1–4, using clients such as Object Viewer and InTouch applications in WindowViewer. You can monitor individual alarms and you can monitor alarms aggregated by containment level. For more information about aggregating alarm states, see “Aggregating Alarm State Information” on page 276.

Typically, alarm priorities and priority to severity mapping are not changed during run time, but it is possible to change alarm priority configuration and severity mapping without closing the client application.

Understanding How Alarms are Ranked at Run Time

The most important alarm has the lowest severity number, but other criteria are taken into account when ranking alarms by urgency at run time.

Alarm Mode	Enabled is more urgent than silenced.
Alarm Shelved	FALSE (not shelved) is more urgent than TRUE (shelved).
InAlarm	TRUE (InAlarm) is more urgent than FALSE (normal).
Acked	FALSE (unacked) is more urgent than TRUE (acked).
Severity Level	1 is most urgent; 4 is least urgent.

Aggregating Alarm State Information

Alarm aggregation provides an efficient way to identify whether any alarms on an object are currently in the InAlarm state, the overall status of the most important of those alarms, and how many alarms are active at each level of alarm severity and at each level of containment. This makes it possible to follow a trail from one level to the next to find the underlying cause of a complex object's alarms.

You can view alarm aggregation statuses in run-time clients such as Object Viewer. You can model alarm aggregation in Managed InTouch applications by using animations such as the alarm border animation with Situational Awareness Library symbols or with ArchestrA symbols.

Alarm aggregation functionality can be described for an object, for an area, and for an attribute.

Object	Aggregation represents all alarms on the object, on all contained objects, and on their descendents down to the lowest level of the model view. Alarm aggregation values on child objects are added to the values of the parent object or objects. All objects have this alarm aggregation functionality.
Area	Aggregation represents the alarms on the Area object itself, on all objects assigned to the area, and on all sub-Areas, down to the lowest level of the model view. If the Area's **AlarmMode** is silenced, all alarms on all Objects in that Area will be silenced. For more information about setting alarm modes, see "Understanding Alarms" on page 234, "Setting Alarm State with Object Attributes" on page 240, and "Enabling, Silencing, and Disabling Alarms" on page 242.
Attribute	Aggregation represents all the alarms on the attribute itself. This is the lowest level of aggregation.

Alarms are aggregated if they are in one of three states:

- UNACK_ALM
- ACK_ALM
- UNACK_RTN

Alarms in the ACK_RTN state are not aggregated.

Alarms in silenced mode are aggregated, even though they do not appear in a run-time client.

A set of five attributes provide run-time aggregated alarm status information:

AlarmMostUrgentSeverity	Displays the severity as an integer 1–4 of the most important current alarms on an object and its descendents. If no alarms are in the InAlarm state or waiting to be acknowledged, the value is 0.
AlarmMostUrgentMode	Displays the mode (enabled or silenced) of the most important current alarm. Alarms in disabled mode are not aggregated.
AlarmMostUrgentInAlarm	Displays the InAlarm status (true or false) of the most important current alarm.
AlarmMostUrgentAcked	Displays the acknowledgement status (true or false) of the most important current alarm.
AlarmCntsBySeverity	Displays an array of all four severity levels with a count of current alarms for each severity.

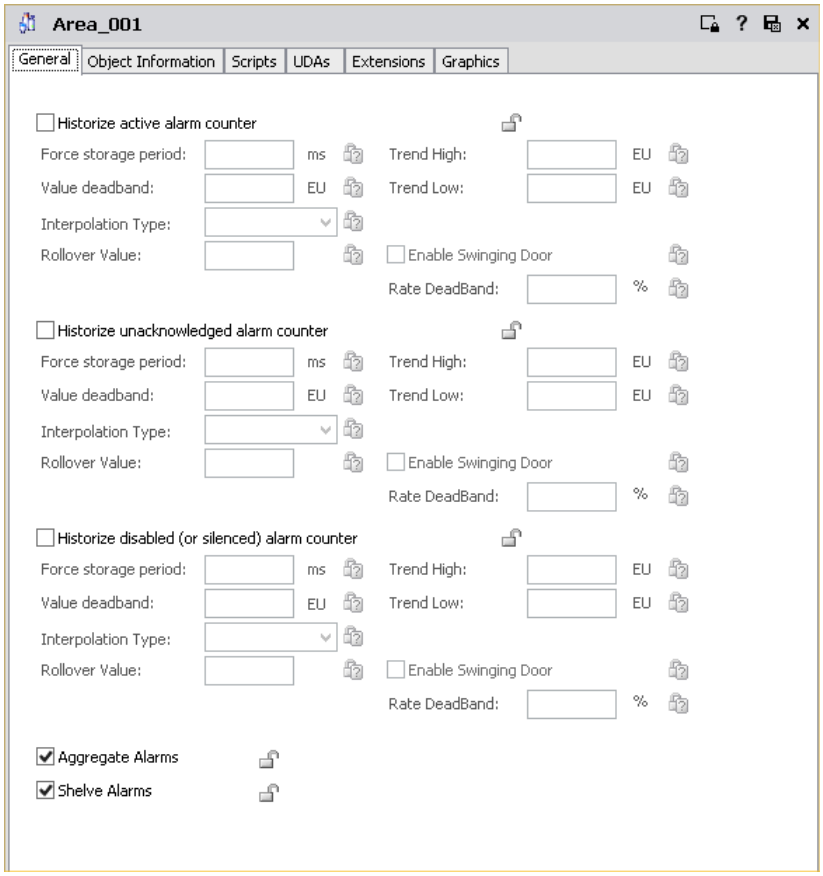
Configuring Alarm State Aggregation

Configuring alarm state aggregation consists of normal alarm configuration procedures plus an added step of enabling the aggregation feature on each relevant Area object.

To configure alarm state aggregation

1. Accept the default settings or configure alarm severities for each of the four severity levels you want to aggregate. This is a global configuration.
2. Configure an object with one or more alarms. All alarms configured on all objects will be aggregated.
3. Set the **AlarmModeCmd** of the object to enabled or silenced, but not to disabled.
4. Set the **AlarmModeCmd** of at least one alarm on the object to enabled or silenced, but not to disabled.
5. Set the **AlarmModeCmd** of the Area object to enabled or silenced, but not to disabled.

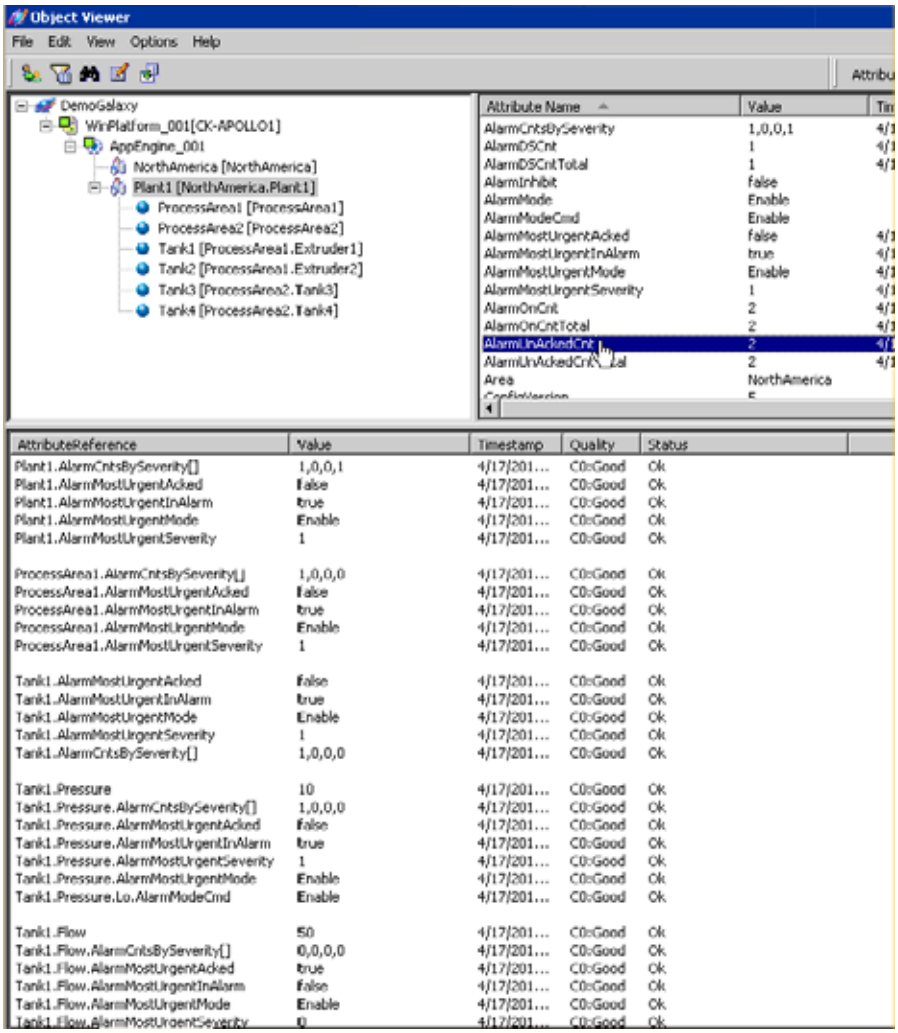
6 Check the **Enable Alarm Aggregation** checkbox on the Area object editor to enable alarm aggregation. This sets the value of the **AlarmAggregationStateCmd** attribute to True.



Monitoring Alarm State Information at Run Time

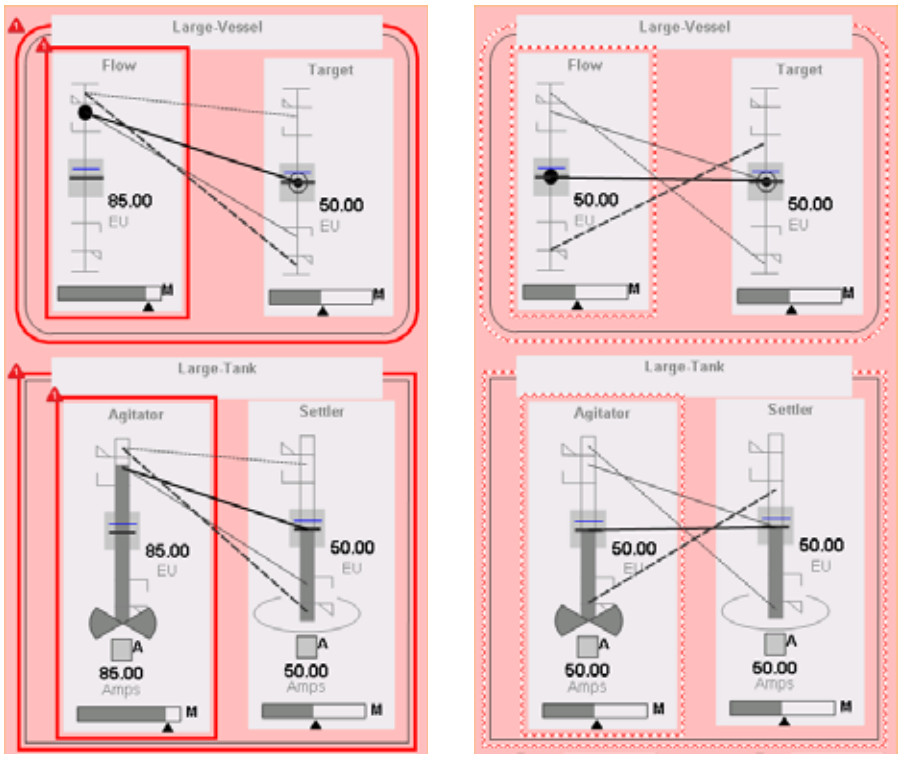
You can obtain run-time alarm state information using clients such as Object Viewer or InTouch Tag Viewer. You can also create InTouch applications with Situational Awareness Library symbols configured with alarm border animations that display alarm state aggregation status information.

For example, you have configured a galaxy with ProcessArea1 and ProcessArea2 contained in Plant1. You have two tanks in each process area, with alarms configured on each tank. You can watch the aggregated alarm severity counts at each containment level in Object Viewer.



You can visualize the same objects in an InTouch application, using the Alarm Client along with the alarm border animation to represent the aggregated alarms at each level of containment.

For example, an alarm border animation applied to a group of symbols can indicate a priority 1 alarm with a red border, flashing or solid for different alarms. The alarm border animation can also indicate a return to normal (RTN) for the same symbols.



Alarm clients configured in InTouch applications will display the severity of each alarm in the **User1** field. You can change “User1” to a more descriptive heading, such as “Severity”.

For information about configuring alarm animations, see “Configuring an Alarm Border Animation” on page 304 in the *Creating and Managing ArchestrA Graphics User's Guide.*

Chapter 9

Working with Multiple Galaxies

You can use ArchestrA Service Bus IDE extensions to configure multi-galaxy communication. You can then use standard tools such as the Galaxy Browser, Object Viewer, and InTouch applications to browse and subscribe to attributes across multiple galaxies.

Understanding the Multi-Galaxy Environment

Multi-Galaxy communication enables you to create an environment—a distributed automation system—comprised of multiple galaxies. You can configure connectivity for multiple Galaxy Repositories (GRs), browse attributes in multiple ArchestrA namespaces, and subscribe to attributes at run time across remote GRs.

With this core functionality and scalability, you can build multiple-galaxy topologies tailored to your manufacturing processes and operations.

About the ArchestrA Service Bus

The ArchestrA Service Bus (ASB) is a framework that hosts services based on general principles of a service-oriented architecture (SOA).

- SOA provides a mesh of interoperable services that can be used within a single system or extended to external systems based on an established trust relationship between the systems.
- SOA also generally provides a way for services to discover other available SOA-based services.

ASB components are installed with Wonderware Application Server as part of the Wonderware System Platform installation. ASB functionality appears as menu items in the IDE. For information about the ASB components in the IDE, see “Setting Up a Multi-Galaxy Environment” on page 285.

ASB components include a set of core run-time services, a catalog of user-configurable services, and a scheme that enables connectivity, interoperability and exchange of data among the services as well as with internal and external applications.

Defining Service Discovery

The discovery servers provide an infrastructure for ASB services to find and communicate with one another, either on the same local node or on a remote node. Multi-Galaxy communication uses three tiers of discovery servers.

- **Local Discovery Server**

  The Local Discovery Server provides discovery for a single, local node. All services register with the Local Discovery Server, and all clients query it. It is not user configurable.

- **Local Galaxy Server**

  The scope of the Local Galaxy Server encompasses the services in an entire galaxy. You can designate primary and secondary servers running on different nodes for redundancy.

  At least one Local Galaxy Server must be configured and online to enable multi-galaxy communication.

- **Cross Galaxy Server**

  The Cross Galaxy Server encompasses the services available across multiple galaxies. You can designate primary and secondary servers running on different nodes for redundancy. In any multi-galaxy environment, there can be only one designated primary and one designated secondary Cross Galaxy Server.

The Cross Galaxy Server can reside on any node in the multi-galaxy environment, but all galaxies in the environment must configure the Cross Galaxy Server to run on the same node.

Defining User-Configurable ASB Services

The following user-configurable services are installed with Application Server as part of the ASB, or are separately installed, as the case with the Event Service, EventHistorization Service, and ASBOPCUAClient Service. You can begin configuring your ASB services with the default instances of each service provided as part of the installation.

- **ASBMxDataProvider Service**

  The ASBMxDataProvider service connects to ArchestrA run-time components and provides access to run-time data.

- **ASBGalaxyBrowsing Service**

  The ASBGalaxyBrowsing service connects to the ArchestrA namespace to enable browsing of ArchestrA objects and attributes.

- **ASBAuthentication Service**

  Users are authenticated against Active Directory user accounts. You can configure more than one ASBAuthentication service instance to point to different domain servers as needed.

- **Event Service**

  The Event service raises, monitors, and routes events. The Event Service is not required to enable or configure multi-galaxy communication.

- **EventHistorization Service**

  The EventHistorization service stores events in a configured database. It is not required to enable or configure multi-galaxy communication.

- **ASBOPCUAClient Service**

  The ArchestrA OPC UA Client Service provides OPC UA data access connectivity to OPC UA servers. The OPC UA Client is supplied as a separately installed service in the ASB infrastructure. Multiple instances of the OPC UA Client service are supported within the same ArchestrA Galaxy.

Note: Close the IDE before uninstalling the OPC UA Client Service. If you uninstall the OPC UA Client Service with the IDE running, the service will not function properly when you re-install it.

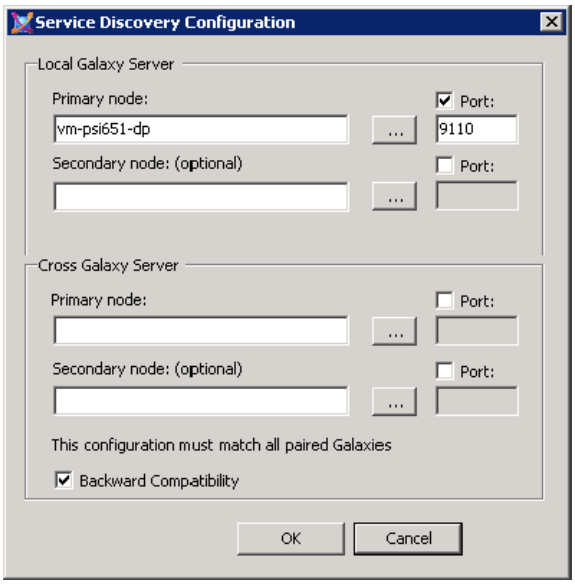
Understanding Multi-Galaxy Communication Workflow

You can browse objects and attributes in an ArchestrA namespace, and subscribe to run-time data from a remote galaxy. The tools, such as Object Viewer, Galaxy Browser, and Managed InTouch applications, and their respective procedures, are nearly identical to working within a single galaxy.

Performing such tasks in a remote galaxy requires some basic steps to set up the multi-galaxy environment. The multi-galaxy communication workflow, using a two-galaxy pair as an example, is as follows:

1. Prepare the multi-galaxy environment by adjusting galaxy names as necessary. Each galaxy in a multi-galaxy environment must have a unique name.
2. Install or upgrade the Wonderware System Platform with Application Server, including the Galaxy Repository.
3. Start Application Server on “Node1”, and connect to “Galaxy1”.
4. Start Application Server on “Node2”, and connect to “Galaxy2”.
5. Configure Service Discovery on each GR node in the multi-galaxy environment.
   a. Configure the **Local Galaxy Server** on each GR node.

Note: For backward compatibility with prior versions of Application Server or existing galaxies that have custom port assignments, the **Service Discovery Configuration** dialog box includes fields to enter port numbers. Clear the **Backward Compatibility** check box to use the assigned default port numbers. With the **Backward Compatibility** check box cleared, only the Microsoft standard port 808 is used by ASB.



b Configure an optional secondary local galaxy server for redundancy as necessary.

c Configure the **Cross Galaxy Server** on each GR node. This server is configured to be the Galaxy Repository node of the local galaxy by default, but can be changed. Configure a secondary server for redundancy as necessary.

There can be only one Cross Galaxy Server in a multi-galaxy environment. All GRs in the environment must point to the same Cross Galaxy Server node.

When pairing, the GR node initiating the pairing gets a copy of the primary Cross Galaxy Server node if none has been configured.

6 Configure **Multi-Galaxy**.

a Enable pairing.

b Pair galaxies.

7 Configure and deploy additional instances of **ArchestrA Services** as required for scalability and enhancing performance during configuration or run-time.

For detailed information about each step in this workflow, see “Setting Up a Multi-Galaxy Environment” on page 285 and “Accessing Multiple Galaxies” on page 293.

Setting Up a Multi-Galaxy Environment

Setting up the multi-galaxy environment requires enabling pairing, designating the Cross Galaxy Server node, and specifying a galaxy or galaxies to be paired.

Naming Galaxies in a Multi-Galaxy Environment

Setting up a multi-galaxy environment requires a unique name for each galaxy in the environment. This may require you to rename existing galaxies if you plan to include in your multi-galaxy environment galaxies that currently have the same name.

We recommend performing any necessary renaming operations prior to upgrading to ASB-enabled Application Server software.

For more information about creating and backing up galaxies, see “Getting Started with the IDE” on page 19 and “Managing Galaxies” on page 403.

Important: You must create a new galaxy with a new, unique name, from a backup .cab file rather than by creating a galaxy and performing a restore of the backup .cab file.

To rename a galaxy for use in a multi-galaxy environment

1. Select a galaxy you want to rename, undeploy it and back it up to create a .cab file.
2. Use the .cab file as a "template" by placing it in one of the following directories, according to your operating system:
   - (32-bit)
     \Program Files\ArchestrA\Framework\Bin\BackupGalaxies
   - (64-bit)
     \Program Files (x86)\ArchestrA\Framework\Bin\
     BackupGalaxies
3. Create a new galaxy based on the backup .cab file, with a new name. The name must be unique, not in use anywhere else in the multi-galaxy environment.
4. Repeat the preceding steps for each galaxy to be renamed with a unique name.
5. Redeploy each newly created galaxy.
6. Delete the original galaxy from the Galaxy Repository node.
7. Upgrade to ASB-enabled Wonderware Application Server using the latest Wonderware System Platform installation program.

Your galaxies can now be configured for use in a multi-galaxy environment.

Installing the ASB

The ASB is installed silently—without specific user interaction—as part of the Wonderware System Platform installation. The ASB elements installed on a given Application Server node depend on the type of node you have selected.

User-configurable ArchestrA Services are installed only on a GR or run-time node.

Bootstrap node	• ASB core services • ASB user documentation (*Wonderware Application Server User Guide* and online help)
GR node	• ASB Service Repository (SR) database • IDE extensions (ArchestrA Services Management) • ASB core services • ASB user-configurable services, including Galaxy Browser Service • ASB user documentation
IDE node	• IDE extensions (ArchestrA Services Management) • ASB core services • ASB user documentation

Start Application Server and connect to galaxies as usual.

Configuring Service Discovery

The Local Galaxy Server is the discovery service for a single Galaxy. All browsing and data access functions can be performed within a galaxy by means of the Local Galaxy Server and ArchestrA Services infrastructure.The Local Galaxy Server default configuration minimizes intra-galaxy network traffic when not using the multi-galaxy communication functionality.

To enable and use galaxy pairing, you must first identify the Local Galaxy Server by its Galaxy Repository node name or IP address for each galaxy.

The Cross Galaxy Server hosts the discovery service for the entire multi-galaxy environment. The Cross Galaxy Server node can be any of the nodes in the multi-galaxy environment. But, it cannot exist on more than one node in the environment. Each galaxy to be paired must use the same Cross Galaxy Server.

You must configure the Cross Galaxy Server as the GR you have selected to be the server for your multi-galaxy environment.

You can designate a secondary server for each type (Local Galaxy and Cross Galaxy) for redundancy purposes.

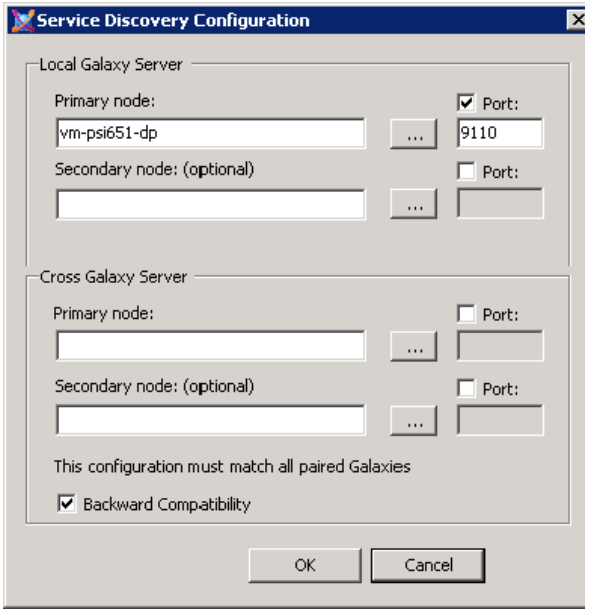
To configure service discovery

1 Click **Galaxy** on the IDE menu, then click **Configure**, then click **Service Discovery**.

The **Service Discovery Configuration** dialog box appears.

2 If you want to communicate with Galaxies created in System Platform 2012 or System Platform 2014, select the **Backward Compatibility** check box.

When **Backward Compatibility** is selected, the specified port numbers must agree with the port numbers used on nodes running previous versions of Application Server. Typically, default port numbers are used, and no change is necessary.



Note: If you intend to use redundant secondary Local and Cross Galaxy servers, the primary and secondary servers must be configured on separate computers.

3 Configure the **Local Galaxy Server**.

a Enter a node name or IP address in the **Primary node** text box.

b If you want to enable redundancy, enter the secondary computer's node name or IP Address in the **Secondary node (optional)** text box.

4 Configure the **Cross Galaxy Server**.

   a Enter a node name or IP address in the **Primary node** text box.

   b If you want to enable redundancy, enter the secondary computer's node name or IP address in the **Secondary node (optional)** text box.

5 Click **OK**.

Important: After setting either the Local Galaxy Server or the Cross Galaxy Server, you must wait 20 seconds before you deploy a platform, either the local GR or remote, to allow the multi-galaxy support services to complete a restart.

Configuring Multi-Galaxy Pairing

Pairing must be enabled on one of the galaxies to be paired in the multi-galaxy environment. All galaxies to be paired must use the same passphrase.

Providing the Pairing Passphrase

Pairing galaxies establishes a trust between the two galaxies. This is accomplished by means of an initial passphrase used by each galaxy to be paired. This passphrase is temporary. It enables the pair of galaxies to access and share their underlying authentication credentials. The pairing passphrase is disposable, and can be changed as often as required.

The pairing passphrase must be the same for each galaxy to be paired. The pairing passphrase must meet a minimum complexity standard:

- At least eight alphanumeric characters
- At least one upper case letter
- At least one lower case letter
- At least one number
- At least one symbol: ! @ # $ % ^ & * ( )

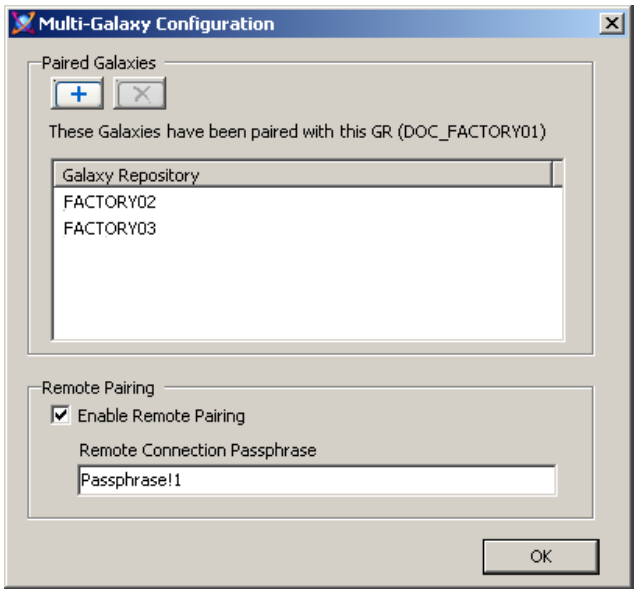
Enabling Galaxy Pairing

You must enable pairing on one of the galaxies to be paired before attempting to pair GR nodes.

To enable galaxy pairing

1 Click **Galaxy** on the IDE menu, then click **Configure**, then click **Multi Galaxy**.

The **Multi-Galaxy Configuration** dialog box appears.



2 Click **Enable Remote Pairing**. This enables the **Remote Connection Passphrase** text box.

3 Enter a passphrase that meets the passphrase complexity criteria in the text box. For passphrase requirements, see “Providing the Pairing Passphrase” on page 289.

4 Click **OK**. Pairing is now enabled for this galaxy.

Pairing with an Enabled Galaxy

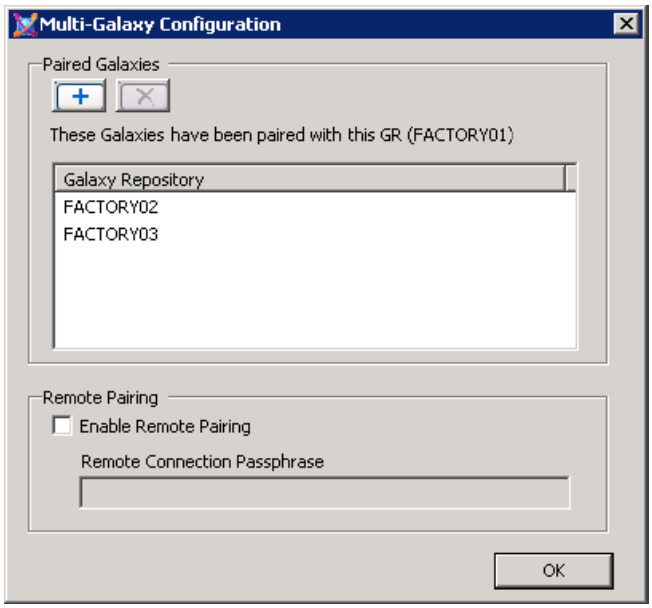
One pairing-enabled galaxy can pair with many other galaxies, but pairing more than two galaxies is not transitive. Galaxy A paired with galaxy B, and galaxy B paired with galaxy C, does not mean that galaxy A is also paired with galaxy C. Galaxy A must be paired with galaxy C as a separate configuration action.

Pairing is commutative. The results are the same if galaxy A pairs with galaxy B or the other way around. The galaxies still are paired.

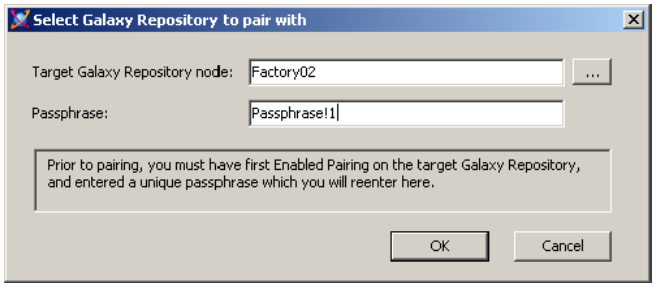
To pair with enabled galaxies

1 Click **Galaxy** on the IDE menu, then click **Configure**, then click **Multi Galaxy**.

The **Multi-Galaxy Configuration** dialog box appears.



2 In the **Paired Galaxies** area, click the **Add (+)** button. The Select **Galaxy Repository to pair with** dialog box appears.

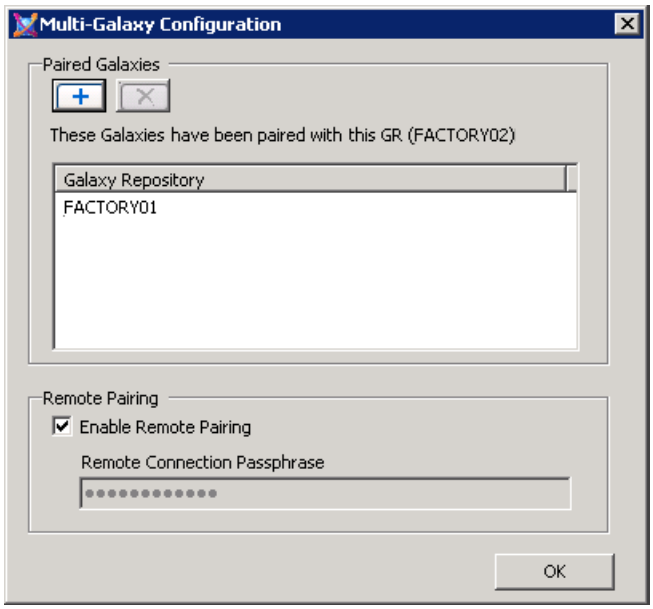


3 Enter the name of the Galaxy Repository node with which you want to pair in the text box, or select a name by clicking the **Browse (...)** button.

4 Enter the same passphrase you used to enable pairing on the galaxy whose Galaxy Repository node you just entered.

5 Click **OK**. The name you entered appears in the **Galaxy Repository** list pane.

The name of the galaxy from which you paired appears in the **Galaxy Repository** list pane of the target galaxy.



6 Repeat these steps to pair with additional Galaxies.

Important: Pairing should be disabled by clearing the **Enable Remote Pairing** check box as soon as all galaxies are paired. Leaving pairing enabled throughout configuration and run-time operations creates a security vulnerability.

Unpairing Galaxies

You can modify your multi-galaxy configuration by unpairing individual galaxies.

To unpair galaxies

1 Click **Galaxy** on the IDE menu, then click **Configure**, then click **Multi Galaxy**.

The **Multi-Galaxy Configuration** dialog box appears.

2 In the **Paired Galaxies** area, select the Galaxy Repository you want to unpair, then click the **Delete (x)** button.

Under certain conditions, the unpairing operation can fail. For more information, see the troubleshooting topic “Unpairing galaxies” on page 320.

Renaming Paired Nodes

Renaming a node that is paired with other nodes will disable communication with all other nodes to which the renamed node has been paired.

If a paired node is renamed, particularly if it is the designated Cross Galaxy Server, you must unpair all paired nodes and re-pair them using the new node name.

If you need to rename a node that is already paired, you should unpair the node first, then re-pair it after renaming it.

For information about naming Galaxies for use in a multi-galaxy environment, see “Naming Galaxies in a Multi-Galaxy Environment” on page 285.

Accessing Multiple Galaxies

You can browse from one galaxy to another in a multi-galaxy environment.

Using the Galaxy Browser is described in “Referencing Objects Using the Galaxy Browser” on page 93.

Using Object Viewer is described in the *Object Viewer User's Guide*.

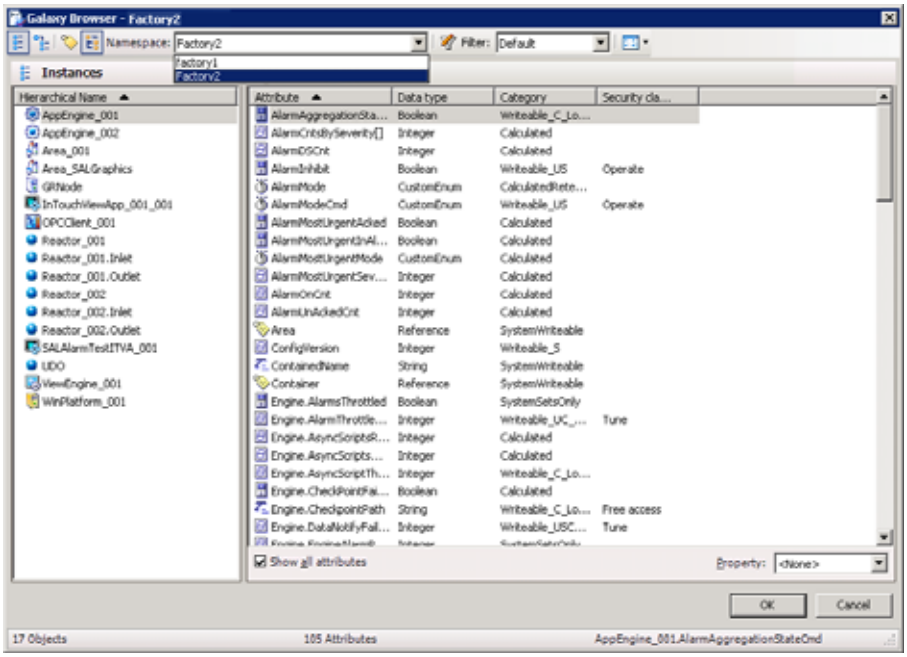
Using the ArchestrA OPC UA Client Service is described in the *ArchestrA OPC UA Client Service Guide*, and in “Browsing with an ArchestrA OPC UA Client” on page 95.

Configuring and running Managed InTouch applications is described in the *InTouch HMI Application Management and Extension Guide*, the *InTouch HMI Data Management Guide*, and other InTouch HMI user guides and online help.

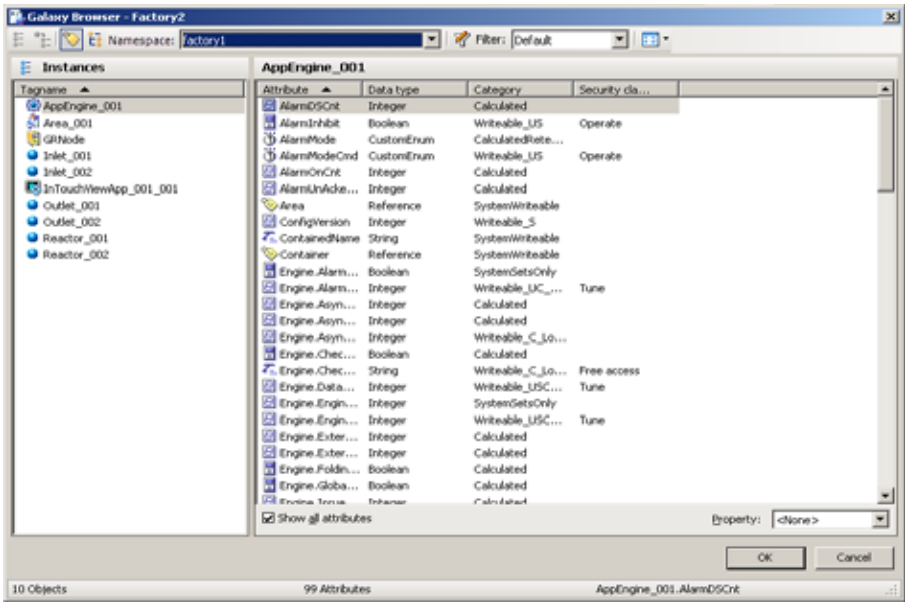
Using the Galaxy Browser with Multiple Galaxies

Open the Galaxy Browser as usual to display the objects and attributes in your galaxy. For illustration purposes, this galaxy's name is Factory2 and you have paired it with Factory1.

The names of all paired galaxies will appear in the **Namespace** selection box. Select Factory2 to view the Tagnames and Attributes in Factory2.



Using the same instance of the Galaxy Browser, you can select Factory1 from the **Namespace** selection box. The Factory1 Tagnames and Attributes will appear.



You can work with the Factory1 Tagnames and Attributes in the same way as with those of Factory2.

For example, you are working in the Object Editor scripts tab, and you want to add a reference from Factory1 in a script. You can open the Galaxy Browser, select Factory1 in the **Galaxy Selection** box, select the Tagname “Outlet_001”, select the Attribute “udInt3”, then click OK. The Attribute reference is inserted into the script as

```
Factory1:Outlet_001.udInt3
```

at your cursor location. This functionality is unchanged except for the addition of the GR node prefix to the Tagname and Attribute.

Using an instance of the ArchestrA OPC UA Client Service to browse a Galaxy namespace requires the use of the specific service instance name as a prefix.

For example, using an instance of the UA Client Service named “OPCUAClient_010” to browse the preceding example would be addressed as follows:

```
OPCUAClient_010:Factory1:Outlet_001.udInt3
```

Using Object Viewer with Multiple Galaxies

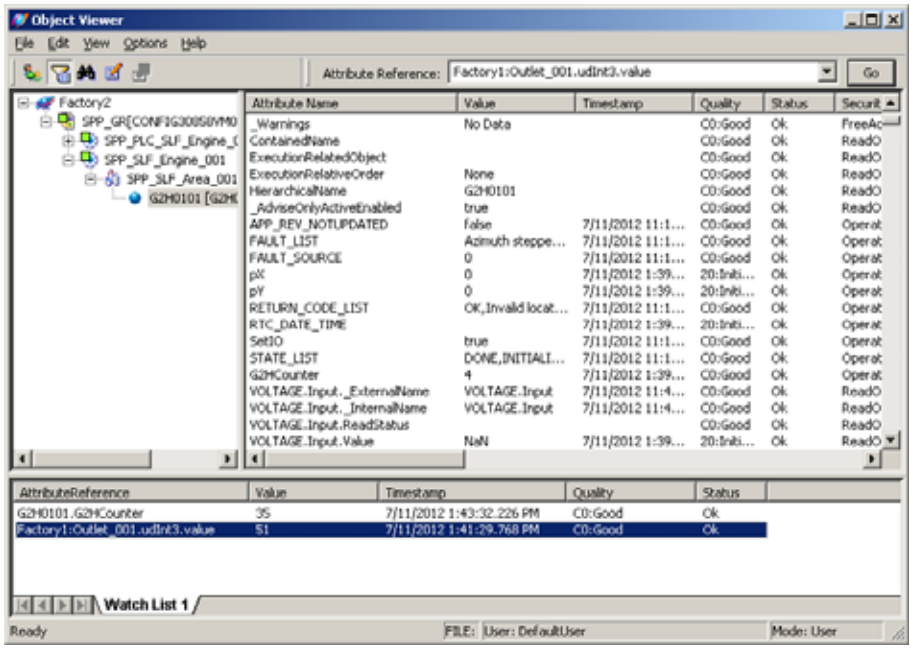
Open Object Viewer as usual to view data types, quality, values, timestamps, and status of the Application Objects, to perform diagnostic testing, and to modify selected attributes.

Object Viewer will display the console tree hierarchy of the GR node on which you have opened it. In this example, open Object Viewer from the Factory2 GR node. The attribute names, values, timestamps, and other data will display for a selected object. Add and manage attribute references to the Watch Window from Factory2 as usual.

In this example, if you want to add the attribute reference “Outlet_001.udInt3”.value to the Watch Window, you must prefix it with the paired galaxy name as follows:

```
Factory1:Outlet_001.udInt3.value
```

Everything following the prefix is addressed as usual. Adding the attribute reference to the Watch Window is the same as usual. The attribute reference will appear in the Watch Window with the GR node name prefix.



Using an instance of the ArchestrA OPC UA Client Service in Object Viewer requires the use of the specific service instance name as a prefix.

For example, using an instance of the UA Client Service named “OPCUAClient_010” in the preceding example would be addressed as follows:

```
OPCUAClient_010:Factory1:Outlet_001.udInt3.value
```

Using InTouch with Multiple Galaxies

You can use an InTouch application as a client to access data in a multi-galaxy environment in the same way you would access data within a single, local galaxy.

You can use the InTouch Tag Browser to select an Application Server object as a tag source and browse the Galaxy database in a multi-galaxy environment in an almost identical fashion to the traditional browsing workflow described in the *InTouch HMI Data Management Guide*.

Application Server object attributes or attribute properties can be used in remote references or as items for InTouch I/O tags in a multi-galaxy environment.

The address syntax changes for use in a multi-galaxy environment. The following is an example of a Galaxy reference used within a single galaxy:

```
Galaxy:Pump1.PV
```

The following is the same reference with the same access name, but used in a multi-galaxy environment to target a remote galaxy named “Factory2”, containing object Pump1 with attribute PV:

```
Galaxy:"Factory2:Pump1.PV"
```

The entire expression, enclosed in quotation marks, becomes an item name. This syntax can be used in ArchestrA Graphics .NET scripting, and in InTouch scripting.

Using an instance of the ArchestrA OPC UA Client Service in this scenario requires the use of the specific service name as a prefix.

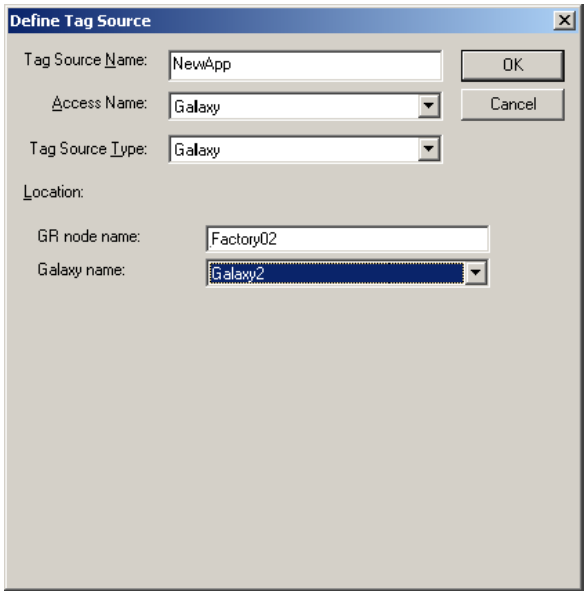
Scope Name/Service Name is specified in the ArchestrA UA Client service editor. This can be the identifier of the OPC UA server to which it is connecting.

For example, using an instance of the UA Client Service named “TestServer” would be addressed as follows:

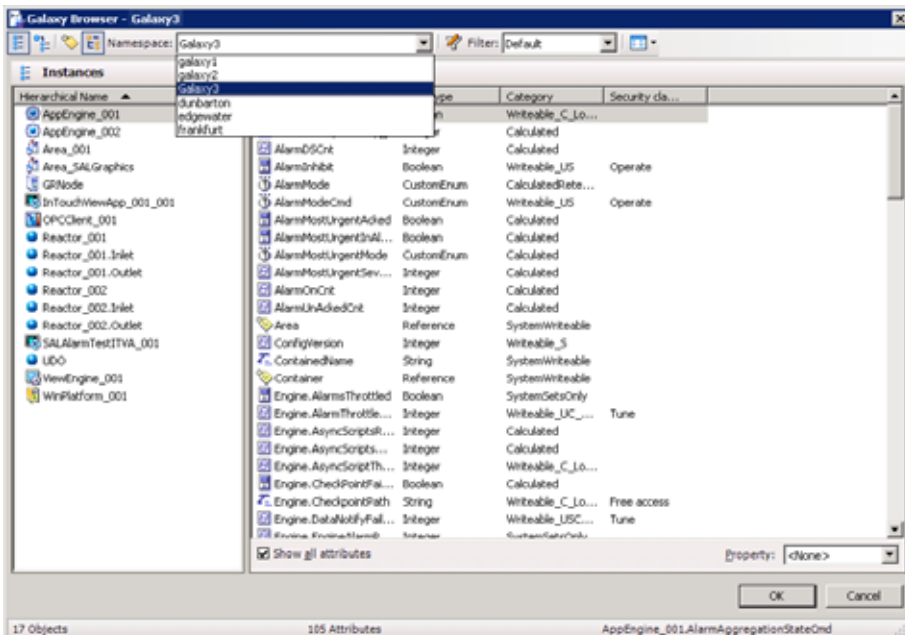
```
TestServer:Item1
```

Important: If you manually enter ArchestrA Graphic custom property overrides, do not enter the double quotation marks shown in the example. Browsing automatically provides the correct syntax, without the quotation marks, for custom property overrides. This syntax variation applies to custom property overrides only.

You can define a paired galaxy as a tag source, with an InTouchViewApp leveraging the galaxy pairing already configured in Application Server.



You can then use the Galaxy Browser to access the Galaxy namespaces of the paired galaxies.



Guidelines for Accessing Multiple Galaxies

Working with multiple Galaxies is basically the same as working with a single Galaxy, but there are some important variations.

Some general guidelines can help you to understand and work with those variations:

- Working with Security in a Multi-Galaxy Environment
- Pairing Galaxies with Multiple Network Interface Cards
- Working with IDE Extensions on a GR Node
- Working with Alarms

Working with Security in a Multi-Galaxy Environment

The multi-Galaxy environment supports the OS User and OS Group ArchestrA Security models.

The Galaxy ArchestrA Security model is not supported:

In order for writes to succeed in a multi-galaxy environment:

- The logged-on user must be a member of the domain, either OS User or OS Group.
- Both galaxies in the pair must be configured to use the same security model.

A user with domain security credentials in a galaxy from which a write is initiated can perform writes, secured writes, and secured and verified writes, depending on configured privilege level—to a remote galaxy without logging on to the remote galaxy.

- Use InTouch as a client to write to attributes in a remote galaxy whether or not the user has logged on to InTouch.
- Use Object Viewer to write to attributes in a remote galaxy.
- Use the MxAccess client to write to attributes in a remote galaxy.

For more information about security, see “Working with Security” on page 353.

Pairing Galaxies with Multiple Network Interface Cards

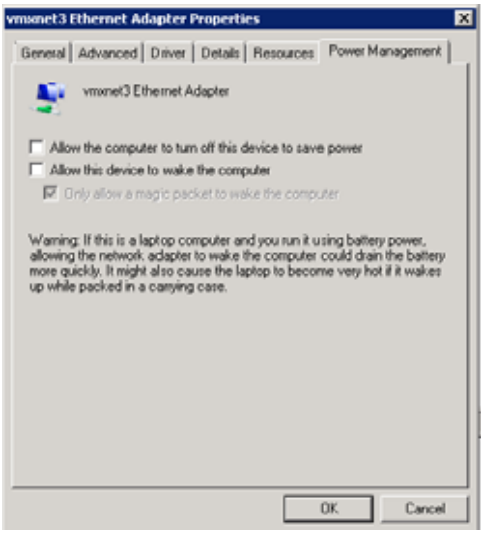
When using more than one Network Interface Card (NIC), galaxy pairing can fail if the NICs are not properly configured. For more information about configuring multiple NICs, see “Using Multiple Network Interface Cards” on page 424.

Preventing ASB Services Disruptions

Windows includes a power management option to place a NIC into sleep mode to save power during periods of low activity. When a NIC is placed into sleep mode, ASB services are disrupted and multi-Galaxy communication is no longer possible.

To prevent ASB Services disruptions

1. On the computer hosting a multi-Galaxy environment, click **Start** and select **Control Panel**.
2. Select **Device Manager** to show the list of installed hardware devices.
3. Click **Network Adapters** to show a list of installed NICs on the computer.
4. Right-click on a listed NIC and select **Properties** from the shortcut menu.
5. Click the **Power Management** tab.
6. Clear the **Allow the computer to turn off this device to save power** option.



7. Click **OK**.
8. Clear this option for all nodes in your multi-galaxy environment.

Working with IDE Extensions on a GR Node

Multi-Galaxy configuration options are either limited or are not available in the following scenarios:

- **IDE running on the GR node**

  When you run the IDE on the GR node, you can only configure multi-galaxy communication for the GR on the same node. From the GR node, you can connect the IDE to a different GR to perform other configuration operations, but any attempts to perform multi-galaxy configuration will result in an informational message that the functionality is not allowed.

- **IDE running on a deployed platform node**

  When you run the IDE on a deployed platform node, you can only configure multi-galaxy communication for the GR the platform is deployed from. From the platform node, you can connect the IDE to a different GR to perform other configuration operations, but the multi-galaxy configuration dialogs will be disabled.

- **Multiple instances of the IDE on a stand-alone node**

  On a stand-alone node, if you or other users run more than one instance of the IDE, multi-galaxy can only be configured for the GR to which the IDE last connected.

  The IDE node automatically pairs with the GR it connects to. While an IDE is connected to a GR node—and paired with it—if another IDE connects to a different GR on the same computer, the IDE node pairs with the new GR. The previously running IDE will continue to function, but multi-galaxy configuration dialogs will be disabled.

By default, installing Application Server deploys the ASBGRBrowsing service. Other user-defined services are not deployed during installation. When you deploy the platform, the ASBMxDataProvider and the ASBAuthentication services automatically deploy. These are default instances which you can modify. You can configure and deploy new instances of each of these services.

Important: The multi-galaxy environment does not support buffered data.

Working with Alarms

All of the available alarm controls are unchanged, and function in a multi-galaxy environment as described in their individual user guides and online help. See the user guides for information on how to configure the various controls, how to query alarms, and how to retrieve alarm information.

- *ArchestrA Alarm Control Guide*
- InTouch HMI Alarm Viewer Control online help
- InTouch HMI Alarm DB View Control online help.

The alarm controls offer varied results and functional limits in a multi-galaxy environment.

ArchestrA Alarm Control Function and Results

- Can take several seconds to display the **Alarm Statistics** dialog box with tens of alarm providers. Can take longer with 30–60 alarm providers.
- Can subscribe (query and retrieve) up to 20,000 alarms.
- Has an historical alarms limit for the maximum number of records of 32,766. Can show more than 32,766 in historical mode.
- Consumed high CPU resources at 200 alarms-per-second from 4 providers. During that time, might not update in a timely manner.

Alarm Viewer Control Function and Results

- Typically requires one second to display the Alarm Statistics dialog box under the same conditions as the ArchestrA Alarm Control.
- Can subscribe (query and retrieve) more than 20,000 alarms.
- Can refresh the alarm list every second.

Alarm DB View Control Function and Results

- Alarm queries are limited to 1,958 characters. Larger queries are truncated.
- Can generate alarms from 4 providers at 20 alarms per second, then will start to buffer alarms.
- Logger can store 20,000 entries, then the cache is full.

We recommend using the ArchestrA Alarm Control under the limits specified. In excess of 20,000 alarms, 20 alarm providers, and 100 alarms per second from multiple providers, we recommend using the Alarm Viewer Control.

Enhancing Multi-Galaxy Performance

Multi-galaxy communication can compound the volume of run-time subscriptions and data changes, potentially impacting system resources. You can manage potential performance degradation by creating and deploying additional ArchestrA Service instances, particularly the ASBMxDataProvider service.

Browsing operations typically place less stress on system resources. Operations tend to be fewer in number than run-time subscriptions and are performed sequentially rather than in volume, in parallel. The ASBGRBrowsing service can be instantiated and deployed to remedy any browsing performance issues.

Operations requiring authentication, such as secured writes, also tend to place less stress on system resources than run-time subscriptions. Performance could become a factor in some scenarios, such as acknowledging a large volume of alarms where the alarm acks require secured writes. The ASBAuthentication service can be instantiated and deployed to remedy any performance issues with secured writes or other operations requiring authentication.

You can take one of two general approaches to multi-galaxy performance: remedy performance issues after the fact, or plan the multi-galaxy environment before the fact. We recommend scaling the multi-galaxy environment before the fact, particularly in large systems.

Remedying Performance Issues After the Fact

Remedying performance issues as they occur involves monitoring subscriptions for data change rates. If your system begins to slow down or starts to show communication failures or bad quality data on subscribed attributes, do the following:

- Create a new ASBMxDataProvider service instance on the publishing side
- Configure the new service instance
- Assign the new service instance to a node and deploy it

  Perform the operation on any node in the multi-galaxy environment exhibiting performance issues.

In a smaller multi-galaxy environment, or when you are just starting to implement multi-galaxy communication, this approach can have the advantage of saving up front configuration time, and can also serve as a trial period to gather metrics about your system.

The disadvantages of this approach are that you can lose time dealing with a system that is slowing down at run time.

Scaling the Multi-Galaxy Environment for Optimum Performance

We recommend at least general advance planning of service and node deployment.

As a general guideline, a single instance of the ASBMxDataProvider service can subscribe up to 20,000 data changes per second. Beyond that number, you should create and deploy a new ASBMxDataProvider service on the publishing side.

An object running under an AppEngine subscribing to remote attributes or an InTouchViewApp, as a general guideline, can support up to 20,000 data updates per second from remote galaxies. Beyond that number, you should create and deploy a new ASBMxDataProvider service on the publishing side.

You can assign the new service instances to the same node as existing service instances provided each instance has a unique port number.

You can assign new nodes and deploy the new service instances to those nodes. In that scenario, a new service instance can have the same port number as an existing service instance on a different node.

Working with ArchestrA Services

The ArchestrA Services provide tools for multi-galaxy communication scalability and performance. For example, you can create additional instances of the ASBMxDataProvider Service if your run-time attribute subscription volume increases to a point where it begins to impact performance.

One default instance of each service is provided during the Wonderware System Platform installation. You can rename and modify these instances and you can create additional instances.

Configuring and Deploying ArchestrA Services

The configuration tool and workflow are the same for all user-configurable ArchestrA Services. To configure an ArchestrA Service, you will need to perform the following tasks:

1. Create an instance of an ASB service
2. Assign the service instance to a node
3. Deploy the instance

You can make multiple instances of a single service, and deploy them to multiple nodes. You can also deploy a single instance to multiple nodes. This approach helps reduce service management. On the same machine, you can have more than one service simply by creating new instances.

Creating an Instance of an ASB Service

You must create an instance of an ArchestrA Service (for example, ASBGRBrowsing Service) to configure the service.

Assigning the Service Instance to a Node

After you create an instance of the ArchestrA Service, you must assign the instance to the desired node. You must check out the instance before assigning it to a node. You must check in the instance after assigning it to a node.

Note: For the ASBGRBrowsingService, you can also specify the name of the Galaxy to assign an instance of the service.

Deploying the Instance

After assigning a service instance to a node, you must deploy the instance. A deployment confirmation message appears.

For information about configuring specific ArchestrA Services, see the following topics:

- “Configuring the ASBGRBrowsing Service” on page 307
- “Configuring the ASBMxDataProvider Service” on page 308
- “Configuring the ASBAuthentication Service” on page 309

Configuring ArchestrA Service TCP Ports

Important: As of Windows System Platform 2014 R2, there is no need to configure any TCP ports. All ArchestrA Service Bus communication is based on the Microsoft WCF shared port 808. This information is retained for existing applications built with a version of Windows System Platform prior to the 2014 R2 release that have been assigned custom port numbers.

To configure ports for OPC UA clients and servers, the following requirements apply:

- On the node running the ASBOPCUAClient service, you need to open two ports, one for IData and one for IBrowse.
- On a node running an OPC UA server, you need to open two ports, one for the OPC UA server and one for the OPC UA discovery service, if configured.

  See the following table for more information about OPC UA port configuration.

The ASB Core Services, installed with the ASB and started by the Watchdog Service, also use specific ports. The following table lists the default TCP ports assigned to the user-configurable ASB Services and the ports in use for the ASB Core Services.

- The ASB Core Services port numbers should not be used to configure instances of user services.
- The firewall must allow incoming connections to the ports in the following table for their corresponding services to function.

ArchestrA Service	Port Numbers in Use
User-configurable ASB Services	
ASBGRBrowsing Service	7500, 7501 (default) See note following this table.
ASBMxDataProvider Service	3572 (default)
ASBAuthentication Service	7779 (default)
Event Service	3575 (default, configurable)
EventHistorization Service	3586 (default, configurable)
ASBOPCUAClient Service	8666, 4600 (default, configurable) IData and IBrowse ports can be auto-assigned or user-assigned.
OPC UA Server	*nnnn* (configurable, server-dependent)
OPC UA Discovery Service	4840 (OPC Foundation service - not an ASB service)
ASB Core Services-Not configurable	
Local Discovery Server	9111
Primary Local Galaxy Server	9110
Secondary Local Galaxy Server	9210
Primary Cross Galaxy Server	9310
Secondary Cross Galaxy Server	9410
Galaxy Pairing	7085
Configuration Service	6332

ArchestrA Service	Port Numbers in Use
Content Provider Service	6011
Deploy Agent Service	6533, 6633
Service Manager Service	6111, 6113
System Authentication Service	9876
aaServiceASBSoftwareUpdate	7587

Important: If a Galaxy Repository (GR) has more than one Galaxy, two additional ports must be opened to enable a remote GR to browse to each additional galaxy. For example, two galaxies would require ports 7500, 7501, 7502, and 7503 to be open. Three galaxies would require ports 7500-7505 to be open.

Configuring the ASBGRBrowsing Service

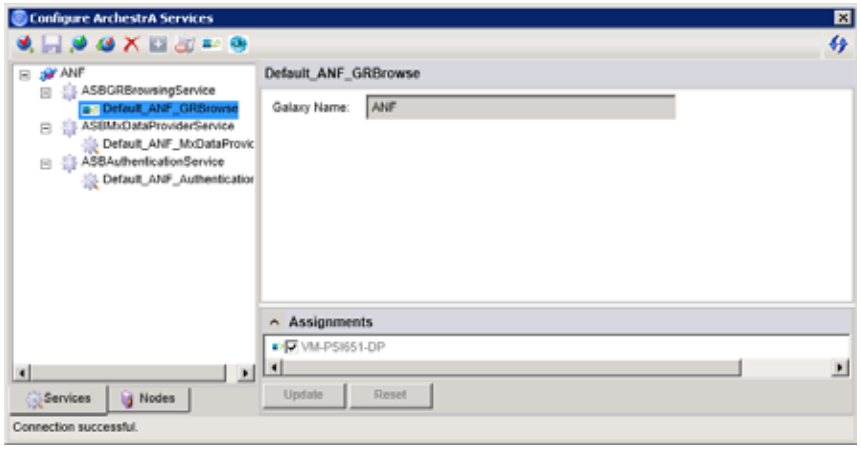
The ASBGRBrowsingService enables you to browse Application Server namespaces.

Important: The ASBGRBrowsing service is configured with one default service instance named "Default_<GalaxyName>_GRBrowse". Do not change the name of this default service instance or delete this service instance. The GRBrowsing service will stop working when you undeploy the GR Platform. If the default service instance has been renamed, revert the service instance name back to the default format. If the default service instance has been deleted, create a new instance with the correctly formatted default name.

To configure the ASBGRBrowsing Service

1 In the IDE, click **Galaxy** on the menu, then click **Configure**, then click **ArchestrA Services**.

The **Configure ArchestrA Services** utility opens.



2 Right-click **ASBGRBrowsingService**, and then click **Create**. You can also type CTRL+N. The new instance appears in the tree structure.

3 Right-click the instance name and click **Check-out**

4 Enter the Galaxy name in the **Galaxy Name** box.

5 In the **Assignments** area, select the node you want to assign to the instance, and then click **Update**.

Note: The **Update** button is enabled only when you select a node. You cannot delete a node that is already assigned to a service instance. To delete a node, you must first unassign the node from the service instance and then delete it.

6 On the left-pane, right-click the instance, and then click **Check-in**.

7 On the left-pane, right-click the instance, and then select **Deploy**. A message appears indicating whether the node is successfully deployed.

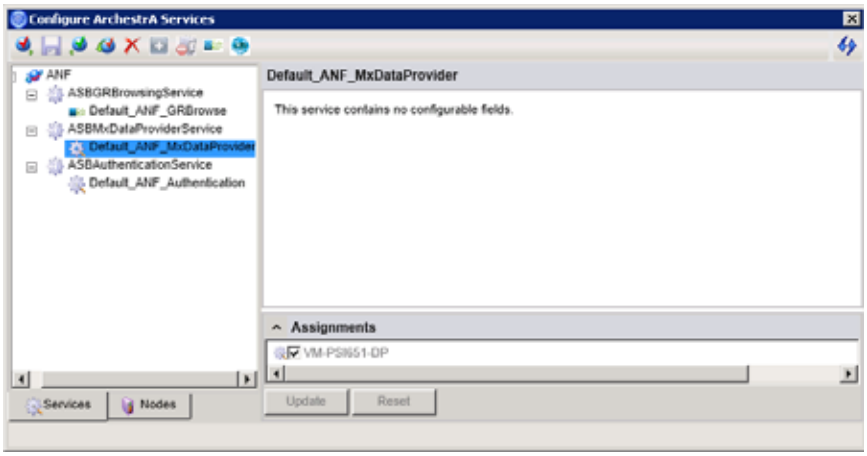
Configuring the ASBMxDataProvider Service

The ASBMxDataProvider Service acts as a gateway providing access to Application Server data.

To configure the ASBMxDataProvider Service

1 In the IDE, click **Galaxy** on the menu, then click **Configure**, then click **ArchestrA Services**.

The **Configure ArchestrA Services** utility opens.



2 Right-click **ASBMxDataProviderService**, and then click **Create**. You can also press CTRL+N. The new instance appears in the tree structure.

3 Right-click the instance name and click **Check-out**.

4 In the **Assignments** area, select the node you want to assign to the instance, and then click **Update**.

5 On the left-pane, right-click the instance, and then click **Check-in**.

6 On the left-pane, right-click the instance, and then select **Deploy**. A message appears indicating whether the node is successfully deployed.

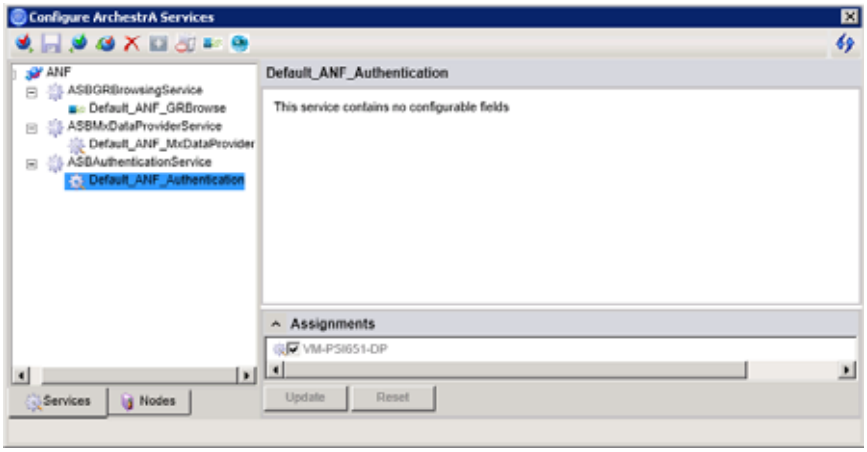
Configuring the ASBAuthentication Service

The ASBAuthenticationService provides user authentication.

To configure the ASBAuthentication Service

1 In the IDE, click **Galaxy** on the menu, then click **Configure**, then click **ArchestrA Services**.

The **Configure ArchestrA Services** utility opens.



2 Right-click **ASBAuthenticationService**, and then click **Create**. You can also press CTRL+N. The new instance appears in the tree structure.

3 Right-click the instance name and click **Check-out**

4 In the **Assignments** area, select the node you want to assign to the instance, and then click **Update**.

5 On the left-pane, right-click the instance, and then click **Check-in**.

6 On the left-pane, right-click the instance, and then select **Deploy**. A message at the bottom of the window appears indicating whether the node is successfully deployed.

Note: A run-time node will be offline if its hosted WinPlatform object is undeployed or if there is a pending software upgrade (SUP state). To change the status of the node to online and deploy new services, apply any required software upgrades and redeploy the WinPlatform object.

Managing Galaxies in a Multi-Galaxy Environment

The workflow for backing up and restoring a galaxy in a multi-galaxy environment is unchanged from those operations in a single-galaxy environment.

There are some important differences in behavior in galaxy management operations performed in a multi-galaxy environment:

- Uninstalling and Reinstalling Application Server
- Backing Up a Galaxy
- Deleting a Galaxy
- Restoring a Galaxy
- Upgrading a Galaxy
- Migrating a Galaxy
- Upgrading SQL Server

Uninstalling and Reinstalling Application Server

When you uninstall Application Server, the ArchestrA Services database is not deleted from the system. This means that any existing galaxy pairings will persist when you reinstall Application Server on the same system. The persistent pairing information is automatically available after the reinstallation.

This pairing information, however, can be out of date and multi-galaxy operations not working as the Discovery Service information is lost in the transition from uninstall to install.

To resolve out-of-date pairing information, you must unpair and re-pair to the required galaxies after performing uninstall and re-install operations on the same node.

For example:

1. Install the latest Application Server software on two nodes, Node1 and Node2.
2. Create a galaxy with a platform, and pair the nodes.
3. On Node1, uninstall Application Server and restart the system.

4 On Node1, reinstall Application Server and connect to the same galaxy.

5 Open the **Multi-Galaxy Configuration** dialog. Notice that all the pairing information has persisted through the uninstall-reinstall operation. Also, notice that multi-galaxy operations are not functioning. You can't browse to or subscribe from Node2 even though the nodes appear to be paired.

6 Unpair and pair Node1 to get the new Discovery Service and pairing information from Node2.

Backing Up a Galaxy

Use the Galaxy Database Manager in the SMC to perform a galaxy backup. The backup workflow is unchanged. There are minor differences in some details:

- The .cab file XML (_GalaxyInfo.txt) has changed to include the backup node name.
- The file repository folder of the .cab file now contains a file, <gr node name><galaxy name>_SR.csv, that lists all nodes and services configured for the galaxy whose backup the .cab file represents. The galaxy restore process uses this .csv file to recreate ASB nodes and services in the target galaxy.

You can back up and restore to a local node. You can back up and restore to a remote node.

Note: Galaxy pairing information is not included in the galaxy backup.

Deleting a Galaxy

Deleting and restoring a galaxy requires that there be no clients connected to it. In a multi-galaxy environment, any deployed browsing service is considered a connection.

The following preparatory steps occur prior to initiating the galaxy delete operation:

1 Use the **Configure ArchestrA Services** editor to undeploy and delete all service instances. The ASB Service instances created for a galaxy are not deleted when the galaxy is deleted.

2 Use the **Configure ArchestrA Services** editor to undeploy all browsing services associated with the galaxy being deleted. If clients are connected, the delete operation is aborted and all browsing services that were previously undeployed are redeployed.

If the delete operation can proceed, all services, including the browsing services, are deleted.

Restoring a Galaxy

The galaxy restore workflow remains unchanged. You can do the following restore operations:

- Restoring to an Existing Galaxy
- Restoring to a New Galaxy

Restoring to an Existing Galaxy

If the existing galaxy to which you are restoring already has deployed browsing services, the browsing services are automatically undeployed as a first step in the restore process.

These browsing services are redeployed if the galaxy restore operation cannot proceed for any reason. Reasons for a galaxy restore operation being aborted:

- There are clients connected to the galaxy.
- The galaxy has a deployed platform.

At the end of the restore process, the service nodes and instances in the existing galaxy are deleted and replaced by new ones defined in the .cab file.

The node corresponding to the GR node in the backup replaces the GR node being restored.

The workflow for restoring Galaxy2 to Galaxy1 remains unchanged. Use the SMC to restore the galaxy, and target the existing galaxy, Galaxy1, to be replaced. The destination galaxy name will remain unchanged, even though the nodes will be replaced with what is defined in the .cab file.

Restoring to a New Galaxy

Typically, you use the Galaxy Database Manager in the SMC to perform a restore operation. Specify the backup .cab file to be restored, and provide a galaxy name. You are restoring a backup, but the result is a new galaxy.

The exception to this workflow is when you must rename a galaxy to ensure that all galaxies in the multi-galaxy environment have unique names. For information about the renaming scenario and procedure, see “Naming Galaxies in a Multi-Galaxy Environment” on page 285.

Upgrading a Galaxy

Back up your galaxies before performing an Application Server software upgrade.

When you restore a non-ASB galaxy from a backup to a later version of the software that includes the ArchestrA Services and ASB functionality, the restore process will set up the default services and extensions.

When you back up the upgraded galaxy, the backup .cab file XML will have the node name. The .csv file will contain the list of services and instances as they exist in the upgraded galaxy.

Migrating a Galaxy

The migration workflow remains unchanged, but has an additional operation at the end of the migration process specifically for ASB.

Migrating from a non-ASB galaxy to an ASB-enabled galaxy, by default, does not create service instances and nodes during the migration process.

Creation of ASB service instances and nodes is deferred until the end of the migration process.

Upgrading SQL Server

We recommend undeploying ASB services before you upgrade SQL Server on your computer.

To upgrade SQL Server in an ASB services environment

1. Undeploy the galaxy.
2. On the menu bar, click **Galaxy**, then **Configure**, then **ArchestrA Services**. The **Configure ArchestrA Services** window appears.
3. Select the galaxy name in the left pane. Right-click and select **Undeploy** from the context menu, or click the **Undeploy** icon on the toolbar.
4. Verify that each service instance is undeployed, then close the configuration window.
5. Follow the relevant Microsoft instructions to install the new version of SQL Server.
6. When complete, you can deploy the ASB services, then deploy the galaxy.

Troubleshooting a Remote Galaxy Connection

Informational and error messages remain the same in a multi-galaxy environment as they are for single-galaxies. The diagnostic tools already available, such as Object Viewer and the SMC Logger, continue to function in a multi-galaxy environment as they do in a single-galaxy environment.

Object Viewer displays communication or configuration errors, and displays data quality. The SMC Logger displays informational messages about system events.

Important: Watchdog and other ASB core services must be running whether or not you are operating in a multi-galaxy environment. Galaxy deletion, configuration, or run-time operations can fail if ASB services are not running, even when not operating in a multi-galaxy environment. See **Local services** in the following troubleshooting table for more information.

Indications of communication or configuration errors can include:

- Galaxies in a multi-galaxy environment do not appear in the **Galaxy:** list box in the **Galaxy Browser**.
- **Attribute References** in an Object Viewer watch list show Bad quality.
- **Attribute References** in an Object Viewer watch list show communication errors.
- Specifying an **Attribute Reference** in the Object Viewer text box returns a configuration error.
- Specifying an attribute reference in an InTouch application returns a configuration error or fails to return run-time data.
- A galaxy in the multi-galaxy environment does not appear in the **Browse Node** dialog box when attempting to select a Galaxy Repository with which to pair.
- You cannot connect to a galaxy in the multi-galaxy environment.
- You cannot undeploy or deploy an instance of an ArchestrA Service.
- Writes from InTouch WindowViewer fail when security is enabled.

When communication or configuration issues arise in a multi-galaxy environment, you can perform the following series of quick checks and remedies to determine if the issues have been caused by a multi-galaxy configuration error.

Check ...	Description	Procedure
Local services	The ArchestrA Watchdog service is a utility to start all the ArchestrA Core Services. The Watchdog service starts automatically when you start your operating system. The Watchdog service and other ASB services are tightly coupled with Application Server configuration and run-time operations. • The Watchdog service must be running in order to create service instances and nodes, to deploy, undeploy, or delete user-defined ArchestrA services. • The Watchdog service must be running whether or not you are using a multi-galaxy environment.	Do not stop the Watchdog service even if you do not plan to use a multi-galaxy environment. These steps use the Windows 2008 R2 operating system for illustration. **1** Click **Start**, then **Administrative Tools**, then **Component Services**. The **Component Services** utility opens. **2** Click **Services (Local)** to populate the **Services (Local)** pane. **3** Click **ArchestrA Watchdog Service** to view its state. **4** If the Watchdog service is not running, click **Start the service**. **5** Exit the **Component Services** utility.
Service Discovery configuration	By default, the Local Galaxy Server primary node is not configured (blank), and must be designated before you can enable galaxy pairing. By default, the Cross Galaxy Server primary node is not configured (blank), and must be set to the designated Cross Galaxy Server before you can enable galaxy pairing. There can be only one Cross Galaxy Server in a multi-galaxy environment. All galaxies in the environment must designate the same Cross Galaxy Server primary node.	Do these steps for each galaxy in the multi-galaxy environment for which you are having connectivity issues. **1** In the IDE main menu, click **Galaxy**, then click **Configure**, then click **Service Discovery** to open the **Service Discovery Configuration** dialog box. **2** Verify or configure the **Local Galaxy Server Primary node** as the local galaxy GR node. **3** Verify or configure the Cross Galaxy Server Primary node as the node you have defined as the Cross Galaxy Server for all paired galaxies.

Check ...	Description	Procedure
Multi-Galaxy configuration	Multi-Galaxy configuration establishes and maintains the trust relationship between paired galaxies for the duration of the pairing session. Remote pairing must be enabled on each GR to be paired, each of which requires the same unique passphrase.	**1** In the IDE main menu, click **Galaxy**, then click **Configure**, then click **Multi-Galaxy** to open the **Multi-Galaxy Configuration** dialog box. **2** Verify that the galaxy or galaxies to be paired are listed in the **Galaxy Repository** list box. **a** If a particular galaxy is not listed, click the **Add** button to open the **Select Galaxy Repository to pair with** dialog box. **b** Enter the **Target Galaxy Repository node** in the text box. **c** Enter the **Passphrase** in the text box. **3** If pairing fails, ensure that remote pairing is enabled on the target GR, and that you have the correct passphrase entered.

Check ...	Description	Procedure
ArchestrA Services configuration	The user-configurable ArchestrA Services provide communication channels for attribute browsing and run-time data access. By default one instance of each service is configured and locally deployed. At least one instance of each service must be deployed on each galaxy in the multi-galaxy environment for multi-galaxy communication to function. **Note:** When you set or change the Local Galaxy Server or the Cross Galaxy Server, the core ASB services automatically restart. This can create a timing issue if you immediately deploy a platform. The ASBMxDataProvider and the ASBAuthentication services can fail to deploy. Run-time connectivity would then fail in the multi-galaxy environment.	For more information about configuring ArchestrA Services, see “Configuring and Deploying ArchestrA Services” on page 304. **For browsing issues**, verify the ASBGRBrowsing service instance. **For run-time data subscription issues**, verify the ASBMxDataProvider service instance on the publisher side, and ensure that the galaxy is deployed. **For secured write issues**, verify the ASBAuthentication service instance. **1** In the IDE main menu, click **Galaxy**, then click **Configure**, then click **ArchestrA Services** to open the **Configure ArchestrA Services** dialog box. **2** Expand the GR node in the hierarchy pane to view the installed services. Expand each service in the hierarchy to view the service instances. **a** Select the instance to view its configuration in the configuration pane. **b** Verify that the **Galaxy Name** is correct (ASBGRBrowsing service only). **c** Verify that the service instance is assigned to a node, visible in the **Assignments** pane, and that the node is correct. **d** Verify that the service instance is deployed. **3** Edit the service instance configurations as necessary.

Check ...	Description	Procedure
Client configuration	The Galaxy Browser, Object Viewer, and Managed InTouch applications can function as clients to browse attributes in the ArchestrA namespace, or to subscribe to run-time data. Using these clients in a multi-galaxy environment is described in “Accessing Multiple Galaxies” on page 293.	**For Object Viewer** communication errors, review the Attribute Reference syntax. In particular, ensure that the correct <GalaxyName>: prefix is present and points to the correct galaxy. **For Galaxy Browser** errors, ensure that you have selected the correct galaxy in the **Galaxy:** list box. **For Managed InTouch** errors: • Ensure that you have selected the correct galaxy when accessing tagnames and attributes from the Galaxy Browser. • Ensure that syntax is correct, including the “Galaxy:” access name prefix, and the full attribute reference in quotes, with the remote galaxy prefix as part of the expression. Correct any incorrect syntax. **Note:** If you selected InTouch HMI **Development and Run Time** during installation, you cannot create a multi-galaxy environment. This installation does not include a GR, required for multi-galaxy functionality.

Check ...	Description	Procedure
Security configuration	OS User with domain users only and OS Group (supports only domain users) are the only ArchestrA Security models supported in a multi-galaxy environment. Security issues, evident with failed writes, can occur for any of the following possible causes: • Security mode of galaxy is set to **Galaxy Security** • Security mode of InTouch is not set to **ArchestrA** • User has not logged into the remote Galaxy at least once • Security modes of local and remote galaxies don't match • User doesn't have sufficient permissions to perform the write • Default User Authentication service is not deployed on the GR node	For information about security configuration, see "Working with Security" on page 353. **In the IDE**, set the security mode to **Galaxy Security**. **In InTouch HMI**, set the security mode to **ArchestrA**. **In the multi-galaxy environment**, set local and remote galaxy security to the same mode for each galaxy. **In the multi-galaxy environment**, log into each remote galaxy at least one time (for each user who requires access to those galaxies and has sufficient write permissions). **In the Configure ArchestrA Services dialog**, verify the ASBAuthentication service instance. **1** Open the **Configure ArchestrA Services** dialog box. **2** Expand the GR node in the hierarchy pane to view the installed services. Expand the each service in the hierarchy to view the service instances. **a** Select the instance to view its configuration in the configuration pane. **b** Verify that the service instance is assigned to a node, visible in the **Assignments** pane, and that the node is correct. **c** Verify that the service instance is deployed. **3** Edit the service instance configurations as necessary.

Check ...	Description	Procedure
Unpairing galaxies	Unpairing galaxies requires that communication is established with the remote node to be unpaired. If communication with a remote galaxy is broken, unpairing with that node will fail. This condition can occur in different scenarios, including: • The remote node is down or not on the network. • The System Authentication service is not running. • The Watchdog service is not running • A mismatch or corruption has occurred in a security key.	When an unpair operation fails, you will see the **Un-pair operation with remote Galaxy Repository failed** error message and dialog box. This dialog box offers you a choice to force the unpair, or to wait until the remote GR is back online. • Click **Yes** to force the unpair. The forced unpairing occurs only on the local GR. When the remote GR is back online, any client running on the remote GR will be unaware that an unpairing has occurred, and will still be able to discover and connect to the service on the local GR from which you performed the failed unpairing operation. • Click **No** to end the unpairing operation and wait until the remote GR is back online. When you click **No**, the system takes no action, and pairing remains as configured. You can now troubleshoot why the remote GR is offline, then retry the unpairing operation.

Chapter 10

Working with Buffered Data

The buffered data feature enables efficient accumulation and propagation of VTQ (Value, Time, and Quality) data updates, without folding and data loss, to data consumers such as objects, alarms, the Historian, and scripts from field devices that support buffering.

Buffered data is defined as data captured and stored locally on a remote device for later transfer to a supervisory system for processing, analysis, and long-term storage. The Buffer property is input-only.

Enabling buffered data alters the run-time behavior of some Application Server features such as History and Alarms. The differences between buffered and non-buffered behavior are described.

About Buffered Data

Enabling the buffered data feature ensures that if a given attribute changes its value several times during a single scan cycle, there is no folding of data, which occurs when an accumulation of values of multiple data changes within a single scan are overwritten and only the latest value is stored. With buffered data enabled, the entire subset of values is propagated to data subscribers such as the alarm subsystem and the Historian if the attribute is configured to be alarmed or historized. Scripts can also take advantage of processing buffered data.

Components that Use Buffered Data

The following table summarizes component functions related to buffered data subscription and processing.

Subscribing to the Buffer property	
Attribute **Field Attribute** (if you are using field attributes)	A configurable Attribute (or field attribute) of a UserDefined object to support buffered data processing. When buffered data is enabled, the attribute (or field attribute) subscribes to the Buffer property and receives the buffered data from Message Exchange.
Scripts	**Script:** A script subscribes to an attribute and receives the buffered data. A specific syntax is used to properly process buffered data in scripting. Scripts can also read the buffer from an attribute or directly from the DIObject. For more information, see “Using ArchestrA Scripts to Process Buffered Data” on page 326.
Processing buffered data	
Data Access Server	The Data Access (DA) Server collects and stores VTQ data from field devices. DAServer functionality is unchanged by the buffered data feature: Protocol interfaces within the DAServer allow processing of data from the field devices that may have been buffered. The DAServers, in turn, can pass the buffered data as packets to the Application Engine.
DIObject	For attributes whose Buffer property is being subscribed, the DIObject collects VTQ values from the DAServer and packs them into buffer collections based on scan cycle completion or on end of data received.
Application Engine	The Application Engine hosts and executes the ApplicationObjects’ logic.

Message Exchange	Message Exchange is the inter-object communications protocol used by Application Server. For more information, see “Using Message Exchange and Attributes” on page 338. The buffered data, a stream of VTQs, is assembled and appended in caches to avoid data folding, and is then published through Message Exchange to consumers such as the Historian, alarms and scripts.
Data Consumers	When buffered data is enabled, data consumers such as the Historian and alarms subsystems subscribe to the Buffer property of the configured attribute to receive published buffered data.

Note: The buffered data feature does not support out-of-sequence data. Out-of-sequence data occurs when the time stamps of the VTQ sequence for a specific reference is not arranged in incremental order from older to newer.

Configuring and Processing Buffered Data

You can configure buffered data for an attribute by setting the **Buffered** feature from the **Attributes** tab (or the **Field Attributes** tab, if you are using field attributes (of the UserDefined object’s editor page. An attribute that functions as a data source for an attribute using buffered data must also be configured for buffered data. A configuration error is logged if the source attribute is not configured to provide buffered data.

Configuring Buffered Data for an Attribute

To configure an attribute to support processing buffered data

Note: If you are using field attributes with a User Defined Object (UDO), enable buffered data on the UDO’s **Field Attributes** page.

1. Open the object you have created in the **Object Editor** window of the IDE.
2. Click the **Attributes** page.

3 Select any of the existing user defined attributes already configured or create a new attribute by clicking the **Add (+)** button.

Buffered data can be configured for all attribute data types except InternationalizedString.

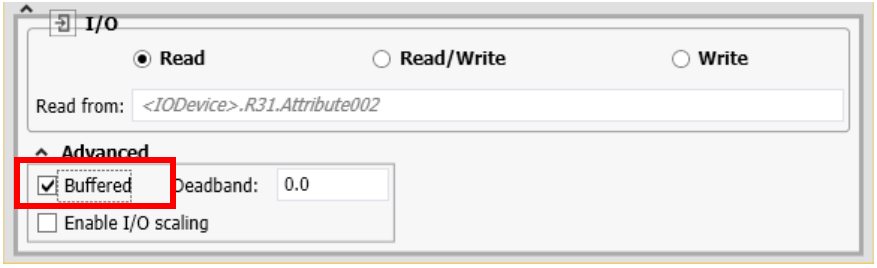
4 Click **I/O** beneath the **Available features** area on the page.

The **Attributes** page shows the fields for **I/O** features. You may need to click **I/O** in the shaded area beneath the **Available features** area to expand and show the feature fields.

5 If necessary, click **Advanced** to show a set of advanced **I/O** features.

6 Select **Buffered** to enable buffered data for the attribute.

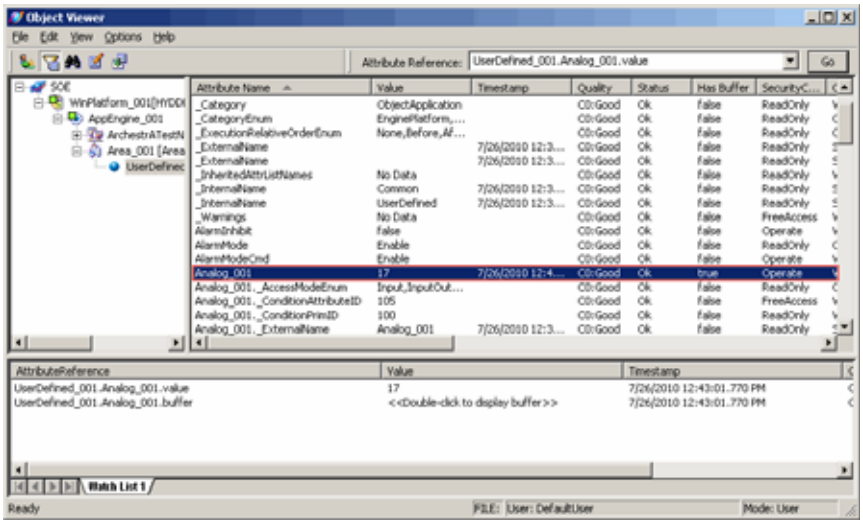
Note: The linked I/O attribute must also have buffering enabled.



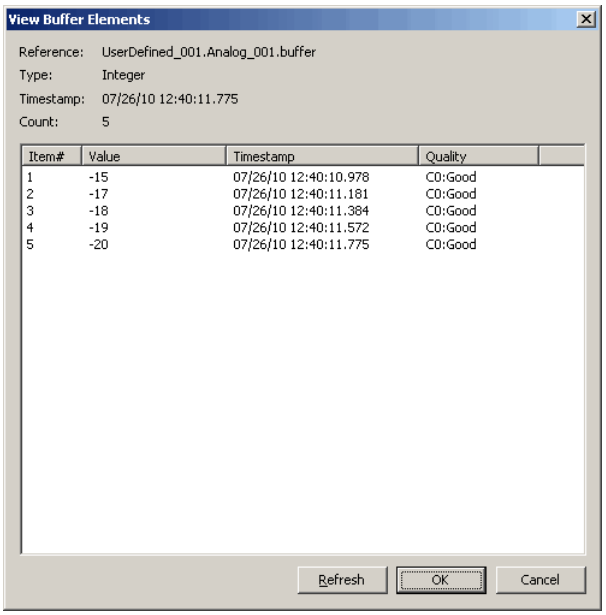
Using Object Viewer with Buffered Data

You can use the Object Viewer utility to view the attribute values of a selected object at run time.

The **Object Viewer Attributes List** displays a **Has Buffer** column to reflect whether an attribute has a buffer. The value displayed in the column is True if the attribute has a buffer and False if the attribute has no buffer, as shown in the following illustration.



Add the **<attributename>.buffer** property to the **Object Viewer Watch Window**. The **Value** column displays the message <<Double-click to display buffer>>. Double-click the attribute to view the buffer elements. The **View Buffer Elements** dialog box appears.



The **View Buffer Elements** dialog box header displays the Attribute Reference, Type, Timestamp, and Count. A details window displays Value, Timestamp, and Quality (VTQ) columns for each item. The data is displayed as a snapshot. Refresh the window to retrieve another snapshot.

The source attribute must also have buffered data enabled. If buffered data is not enabled on the source attribute, the data quality will return as bad, and a configuration error message is displayed.

See the *Object Viewer User's Guide* for more information.

Using ArchestrA Scripts to Process Buffered Data

ArchestrA scripts can be synchronous or asynchronous. The synchronous mode is the default choice and represents serial script execution by the ApplicationEngine in the course of calling the Execute method of all ApplicationObjects that are on-scan in the ApplicationEngine.

Synchronous mode requires that all scripts execute deterministically and quickly enough to prevent an ApplicationEngine over-scan condition.

Important: For scripts that process buffered data, it is strongly recommended that you configure scripts as asynchronous execution type. Processing buffered data may be time consuming and exceed the script execution time limit. See "Scripting Tips and Best Practices" on page 328 for more information.

Scripting Basic Functions

The following is a collection of basic functions used to read buffer data from the script accompanied by script snippets.

Declaration

```
dim vtq as ValueTimeQuality;
```

To access samples in each buffer

```
FOR EACH vtq in me.U1.buffer
```

Note: The only syntax that can be used to process buffers in scripts is FOR EACH.

To write the buffer to text file

Declaration:

```
dim strWriter as System.IO.StreamWriter;
```

Usage:

```
strWriter = System.IO.File.AppendText("C:\\sample.txt");
```

To write the buffer to Logger (or SMC)

LogMessage();

Sample Script and Output

Read Buffered Data from the VTQ Buffer

```
dim vtq as ValueTimeQuality;

dim strWriter as System.IO.StreamWriter;

LogMessage("Script Start");
   strWriter = System.IO.File.AppendText("C:\\sample.txt");
FOR EACH vtq in me.UD1.buffer
   strWriter.WriteLine("DAS value is "+vtq.value);
   strWriter.WriteLine("Time stamp of the value is "+vtq.time);
   strWriter.WriteLine("Quality of the value is "+vtq.quality);
   strWriter.WriteLine("-------------------------------------
   ");
   'Below messages will be logged to the Logviewer (or SMC)
   LogMessage("DAS value is "+vtq.value);
   LogMessage("Time stamp of the value is "+vtq.time);
   LogMessage("Quality of the value is "+vtq.quality);

next;
   strWriter.Flush();
   strWriter.Close();

LogMessage("Script End");
```

For each VTQ in this script sample, Value, Timestamp, and Quality are written to the sample.txt file.

For purposes of illustration, UD1 in the script sample is a buffer-enabled Integer attribute with input extended to a DDE item.

Sample Output Text Written to Text File

```
DAS value is 2

Time stamp of the value is 7/15/2010 11:28:54.880 AM

Quality of the value is 192
```

Sample Output Text Written to Logger

```
Info        ScriptRuntime   UD1.S1: Script Start

Info        ScriptRuntime   UD1.S1: DAS value is 2

Info        ScriptRuntime   UD1.S1: Time stamp of the value is
  7/15/2010 11:28:54.880 AM

Info        ScriptRuntime   UD1.S1: Quality of the value is 192

Info        ScriptRuntime   UD1.S1: Script End
```

Scripting Tips and Best Practices

Use Asynchronous Scripts for Buffered Data

Synchronous scripts may read buffered data. However, if the buffer is expected to take too much time for processing it would be better to mark the script as asynchronous.

Buffered Data Script Syntax and Configuration

Scripts can process buffered data by using the FOR EACH syntax to iterate over an attribute's Buffer property.

To process all the buffers without any loss, asynchronous scripts need to have the following configuration:

- Execution type: Execute
- Trigger type: While True
- Expression: True
- Time period: 00:00:00.0000000

Using BindTo for Buffered Data

When working with indirect scripts to read the .buffer property of an attribute, the BindTo call needs to be used in startup instead of using it in execute. Using BindTo in execute can lead to losing the data in alternate buffers. The following snippets show an example of using an indirect to process a buffer. In this example, the BindTo call is in Startup and FOR EACH is in Execute.

When binding to an on-engine attribute, the whole script could be put in Execute. When binding to an off-engine attribute, use BindTo in startup instead of in execute, as shown in the example.

In Declarations:

```
dim x as indirect;
```

In Startup:

```
x.BindTo("me.attribute1.buffer");
```

In Execute:

```
dim vtq as ValueTimeQuality;
for each vtq in x
  LogMessage(vtq.value);
next;
```

Buffered Data Run-time Behavior

This following table describes how enabling buffered data affects the run-time behavior of the Scan Group, Inputs, and Redundancy for all phases:

- On Startup
- On Scan
- Execute
- Off Scan
- Shutdown

History run-time behavior and Data and Alarms run-time behavior also change when buffered data is enabled. The run-time behavior of those features is described in “About Buffered Data and History” on page 331 and “About Buffered Data and Alarms and Events” on page 332.

Important: Buffered data is not supported in a multi-galaxy environment.

Feature	Phase	Behavior
Scan Group		When the buffer property of a scan group's dynamic attribute is subscribed to at run time, the buffer data support is enabled. The scan group accumulates all of that item's updates into a buffer of VTQs. The scan group supports buffering up to 10,000 VTQs per item per scan. Higher data throughput rates are not supported
	On Scan	When the object is placed On Scan, the system enables all available items and resets item error count.

Feature	Phase	Behavior
	Off Scan	When the Scan Group goes Off Scan, it will clear all buffers associated with each item in the Scan Group. When buffered data is enabled and the host object goes Off Scan, a VTQ with Bad Quality is added to the buffer. On a subsequent scan, the buffer property of the **Value** attribute is cleared.
Inputs	On Startup	When buffer is enabled on the attribute, buffer support for the input is enabled. No data is processed during this phase.
	Execute	The buffered attribute is registered, but no data is processed or stored until the buffer reference finishes initializing.
	Shutdown	Interfaces are unregistered. (No change in behavior for buffered data-enabled.)
Redundancy		**Application Engine** Buffered attributes are not synchronized with a redundant partner. When a failover occurs, unprocessed buffered data is lost.
		Redundant DIObject When buffering is enabled, the RedundantDIObject (RDI) subscribes to the buffer property of its DI source items. On failover, all advised items are moved to the backup node, whether or not buffered data is enabled. Buffered attributes are not synchronized with a redundant partner. When a failover occurs, unprocessed buffered data is lost.

About Buffered Data and History

When an ApplicationObject starts up, if configured to save historical data, it registers its configuration data with the Historian using a Historian supplied interface.

The following table describes buffered data-enabled history behavior.

History Function	Behavior
Buffer Monitoring	The ApplicationObject monitors and historizes the attribute value. If the attribute being historized has a buffer property, the ApplicationObject monitors the attribute's buffer instead of its value, and historizes each VTQ in the buffer.
Attribute Registration with the Historian	Attributes that are buffer-enabled are registered with the Historian in the same manner as non-buffered attributes are registered.
Push Attribute Changes to the Historian	After successful registration of an attribute with the Historian, the buffer-enabled attribute pushes the data changes to the Historian. If the buffer-enabled attribute reads an empty buffer, and if this is the first scan cycle after the attribute was registered, it gets the attribute's Value, Time, and Quality properties, and if Quality is Good, sends this VTQ to the Historian. If the buffer-enabled attribute reads an empty buffer, and if this is not the first scan cycle after the attribute was registered, it does nothing. For more information, see "Working with History" on page 209.

About Buffered Data and Alarms and Events

Application Server's alarm and event capabilities automate the detection, notification, historization, and viewing of either application alarms and events or system alarms and events.

An attribute, when configured for alarming and buffer-enabled, can receive a packet of VTQs in a single scan. This packet can contain values that will trigger an alarm condition. Alarm conditions will be generated based on the VTQs in the buffer.

Alarm Functions and Buffered Data

The attribute, if configured for alarms, detects the alarm condition and sets an associated alarm condition attribute according to the base attribute being monitored. If the base attribute is configured with buffered data enabled, the condition attribute will also be enabled with buffered data support.

At run time, the condition attribute's buffer property will contain a VTQ buffer of true or false values representing alarm conditions corresponding to the base attribute buffer contents.

Only the most current alarm condition can be acknowledged.

For more information, see "Working with Alarms and Events" on page 231.

The following table describes buffer-enabled behavior for alarm functions.

Alarm Function	Behavior
Detection	With buffered data enabled for the condition attribute, the alarm condition change is detected using the buffer property of the condition attributes. Multiple alarm messages can be generated based on the buffer contents. The alarm is time-stamped with the time the alarm was recorded at the field device, not the time at which the values were received and processed within the object. A set of alarm messages corresponding to the alarm state transitions are sent to its Notification Distributor. The TimeAlarmOff attribute's value is set to the time of the last off transition, and TimeAlarmOn to the last on transition. The InAlarm value, time and quality properties are set to the last alarm transition's state, time and quality.
Notification	When the Notification Distributor receives a set of alarm messages, it updates the state of the record for the corresponding alarm. The set of alarm messages are forwarded to all registered alarm clients. The Notification Distributor does not send a throttling signal as a result of receiving multiple alarm messages. When the Notification Distributor receives a set of event messages from ApplicationObjects as the result of processing an attribute's buffer, the set of event messages are forwarded to all registered event clients.

Alarm Function	Behavior
Throttling	Alarm throttling settings are ignored for alarm messages generated based on buffered data. Alarms will be generated for all alarm conditions in the buffers. If a potential throttling point is reached, the current version of Application Server provides a functionality for all alarms to be processed without system overload.
Acknowledgement	If an alarm is associated with buffered data, acknowledging an alarm only acknowledges the last alarm, and not all alarms contained in the buffer.
Disabled Alarms	If an alarm is disabled for a buffered attribute, the alarm's Buffer property is not processed. Alarm buffers for disabled alarms are cleared. If an alarm is silenced for a buffered attribute, the alarm's Buffer property is processed. Alarms will be generated but will not be displayed by the alarm client.
Outages	Buffered alarms and events can be lost when there is an outage because the Notification Distributor will only keep and re-send the last on-alarm message and the last off-alarm message.

Alarm and Event Types and Buffered Data

The following table describes buffered data-enabled behavior of alarm and event types.

Alarm/Event Type	Behavior
Data Change Event	Application (data change) Events have the following fields: • Event type – data change (historized) • Timestamp – date/time of event (historized) When Log Event is checked for the buffered data-enabled attribute or field attribute, a data change event will be reported for every VTQ in the attribute's buffer property.
Limit Alarm	The limit alarm can provide a buffer of alarms on an attribute with buffered data enabled. If the base attribute has an associated Buffer property, then the limit alarm calculation is performed for each VTQ in the buffer. This results in a buffer of Boolean alarm conditions with associated time of the VTQ matching the times of the VTQ in input buffer, which is stored in the respective condition attribute's buffer property. The order of the condition buffer matches the order of the input buffer. The respective condition value and time reflects the last VTQ in the input buffer. **Value Deadband:** If value deadband is configured for limit alarms, then the alarm condition is calculated using VTQs in the input buffer. All VTQs in the buffer are processed. Value deadband is used during execution. **Time Deadband:** If time deadband is configured for limit alarms, when the alarm condition is met, the duration of the condition is calculated using the VTQs in the input buffer. If the last condition change remains undecided until the end of processing the current buffer, then a late decision will be made when the next input buffer is processed. If no more input buffer is received, the last alarm condition change is reported when the time deadband has elapsed as engine time. The time deadband is applied only to the On Rise event of the alarm state. It is not applied when the alarm state falls outside the limit condition.

Alarm/Event Type	Behavior
State Alarm	The state alarm can provide a buffer of alarms on a Boolean attribute when buffered data is enabled. If the base attribute has an associated Buffer property, then the state alarm calculation is performed for each VTQ in the input buffer. This results in a buffer of Boolean alarm conditions with associated time of the VTQ matching the times of the VTQ in input buffer, which is stored in the respective Condition attribute's buffer property. The order of the condition buffer matches the order of the input buffer. The respective condition value and time reflects the last VTQ in the input buffer.
Rate of Change Alarm	If the input has an associated Buffer property, then the Rate of Change (ROC) calculation is performed for each VTQ in the input buffer. Interval is calculated using the difference of two successive VTQs. These calculations result in an array of Boolean alarm conditions with associated timestamps which are stored in the buffer property of the respective condition alarm attribute. The condition value reflects the last VTQ in the respective condition buffer.
Deviation Alarm	If buffered data is enabled for a deviation alarm's attributes or field attributes, buffered data is also enabled on the configured condition attributes Dev.Minor.Condition or Dev.Major.Condition or both. When buffered data is enabled on a deviation alarm's attributes or field attributes and the AppEngine goes Off Scan, the buffers for the configured condition attributes, Dev.Minor.Condition or Dev.Major.Condition or both, are cleared of all VTQs. If the input has an associated Buffer property, then the deviation calculation is performed for each VTQ in the input buffer. These calculations result in an array of Boolean alarm conditions and associated timestamps which are stored in the buffer property of the respective condition alarm attribute. The condition value reflects the last VTQ in the respective Condition buffer.

Chapter 11

Working with References

References allow identification and communication between objects in the ArchestrA environment. Every object, every attribute, and every property can be uniquely referenced. Those references are communicated over the messaging system. The Message Exchange is the object-to-object communications protocol used by ArchestrA and the Wonderware Application Server.

Note: ArchestrA is the framework for supervisory control and manufacturing information systems. It is an open and extensible technology based on a distributed, object-based design. For example, if you are using Application Server with InTouch, these products communicate with each other using the ArchestrA framework.

This section describes the concept of references and how to use reference strings in creating your Application Server application.

Using Message Exchange and Attributes

All object attributes have properties, such as Value and Quality. Any data read or written to or from these attributes over ArchestrA Message Exchange is tracked. If an operation cannot be performed, the requesting client is notified.

Message Exchange provides the following features and information:

- Guaranteed response
- Name signatures
- Status and data quality
- Message order preservation within a priority system
- AppEngine-to-AppEngine buffering
- Publish-subscribe heartbeats

Reference Strings

Reference strings refer to an object or to data within an object's attributes. A reference string consists of an object's reference string plus an attribute's reference string.

```
object Reference + Attribute Reference
```

A reference string is the object name plus the attribute name: `ObjectName.AttributeName`.

In a multi-galaxy environment, a reference string to a remote galaxy is the galaxy name plus the object name plus the attribute name:

```
GalaxyName:ObjectName.AttributeName.
```

Use references and reference strings in a multi-galaxy environment as you would in a single-galaxy environment, with the exception of the galaxy name prefix, <GalaxyName>, followed by a colon as in the preceding example. For more information about using references and reference strings in a multi-galaxy environment, see “Accessing Multiple Galaxies” on page 293.

In `TIC101.PV`, `TIC101` is the object reference and `PV` is the attribute reference. The `AttributeName` can be omitted in a reference string, `PV` being assumed in such cases.

Note: Some objects have a `PV` attribute, while others do not.

Reference strings are concatenated substrings, each no more than 32 characters separated by periods. A substring cannot contain a period. Mathematical operator characters are not allowed. At least one character in each substring must be non-numeric.

Avoid assigning objects and attributes names such that the same reference string can refer to two different things. For example, you have two objects named A1 and B2, and inside A1 you create an attribute named B2 with a data type of Float. A1.B2 refers to the attribute Float named B2. If you then assign object B2 so that A1 is a container of object B2, the reference A1.B2 could refer either to the object B2 or the attribute B2 Float.

Important: The Galaxy resolves reference strings. If the GR is not available, resolution is done on a peer-to-peer level. After initial resolution, an object is provided an alias that handles references to its location across your network. If an object is relocated or renamed, the reference string resolution is repeated and a new alias provided.

Relative References

References that go up the hierarchy to parent objects are called relative references. For more information about hierarchy, see “ApplicationObject Containment” on page 60.

Relative references, such as `Me`, are valid reference strings. A valid reference string must always contain at least a relative reference or one substring.

The following are valid relative references that refer to the current object:

- Me
- MyContainer
- MyArea
- MyPlatform
- MyEngine
- MyHost

Relative references are especially useful in templates because absolute references typically do not apply or make sense.

When you use relative references, like MyContainer, you can refer to contained objects within that container. For example, a reference to `MyContainer.InletValve.PV` is equivalent to `Tank1.InletValve.PV` in the following hierarchy:

`Tank1`	Cannot reference at this level because this is not contained
`Inlet Valve (InletValve)`	Can reference at this level because this object is contained
`Outlet Valve (OutletValve)`	Can reference at this level because this object is contained

Property References

Certain property names are reserved for ArchestrA. If a string has a reserved property name in the ArchestrA environment, you can still use it. The PROPERTY keyword must be part of the string, for example, PROPERTY(propertyName). In all other cases, the case insensitive PROPERTY keyword is not required.

The Value property is assumed if no property reference is specified.

The following are property references:

- .Name
- .Value
- .Type
- .Quality
- .Time

syntax:

```
obj.int.PROPERTY(quality)
```

where:

obj = object specifier

int = attribute

PROPERTY = keyword

(quality) = property specifier

For example, you can address the time of an attribute in a scan group for a DIObject from within an InTouch application, as follows:

```
Galaxy:"<DIObject>.<scangroup>.<attribute>.Property(Time)"
```

This is the same as if you used .Time:

```
Galaxy:"<DIObject>.<scangroup>.Attribute(<attribute>).Time"
```

You can directly address an item without having an attribute in the scan group. For this example, the item is MB1:

```
Galaxy:"<DIObject>.<scangroup>.MB1.Property(Time)"
```

and

```
Galaxy:"<DIObject>.<scangroup>.Attribute(MB1).Time"
```

For objects with a default scan group, you must refer to the .Time, .Value, and .Quality properties using the .Property(time), .Property(value), and .Property(quality) notation.

The following is an example of the correct use of property:

```
LogMessage(ATTRIBUTE("MyEngine.tagname.PROPERTY(securitycla
ssification)"))
```

The following is an example of the incorrect use of property:

```
Logmessage(abtcpplc5_001.fast.changingpoint.property(qualit
y));
```

Handling Time Zones with the Time Property

If you need to share time stamp values across different time zones (Platforms), use the Time data type in every time zone location. However, if you need to share it as string, remember that when converting the Time data type to a string (for example, in a script), it is automatically converted to local time, so you lose the ability to adjust it in a different time zone.

For example, to convert the Time property to a string GMT:

```
Dim localDateTime As System.DateTime;

localDateTime = System.DateTime.Parse( obj.attr.Time );

obj.udStringGMTfromLocalTime=
  localDateTime.ToUniversalTime().ToString();
```

To convert the string GMT to a string of local time:

```
Dim univDateTime As System.DateTime;

univDateTime = System.DateTime.Parse(
  obj.udStringGMTfromLocalTime );

Obj.udStringLocalTimeFromGMT =
  univDateTime.ToLocalTime().ToString();
```

Preserving Time Stamps from the Publishing Source

In the following cases, if you want to pass only the time stamp, the subscriber gets the time stamp as converted to the local time zone of the publisher and not the time zone of the data source.

Examples of configurations that do not preserve the original time zone are as follows.

In this configuration, GalaxyB:Object1.TimeAttr shows the time adjusted to the local time zone of the GalaxyA FSGateway and not the time zone of the PLC:

PLC.Item <= GalaxyA Object1.IntAttr.Time <= FSGateway <= GalaxyB OPCClient <= Object1.TimeAttr

In this configuration, GalaxyB:Object1.TimeAttr shows the time adjusted to the local time zone of the InTouch application and not the time zone of the PLC:

PLC.Item <= GalaxyA Object1.IntAttr.Time <= InTouch App I/O Message Tag <= GalaxyB InTouchProxy <= Object1.TimeAttr

To avoid these problems, subscribe to the GalaxyA:Object1.IntAttr value property. This way, both the value and time stamp propagate to GalaxyB:Object1.IntAttr. You can then use the GalaxyB:Object1.IntAttr.Time. For example:

PLC.Item <= GalaxyA Object1.IntAttr <= FSGateway <= GalaxyB OPCClient <= Object1.IntAttr

PLC.Item <= GalaxyA Object1.IntAttr <= InTouch App I/O Integer Tag <= GalaxyB InTouchProxy <= Object1.IntAttr

In this configuration, the time property propagates from InTouch to Object.IntAttr.Time:

PLC.Item <= InTouch I/O Integer Tag <= Galaxy InTouchProxy <= Object.IntAttr

Arrays

A reference string can also refer to the Value property of an array attribute with an optional Array Element Reference that includes up to one dimension:

- [i] – individual element
- [] – entire array

The letter `i` represents an integer constant.

Formatting Reference Strings

These symbols apply to the reference strings that follow:

This...	means...
::=	can be replaced by
|	or
[]	contents optional
{}	contents can be left out, used one time or repeated

Quotation marks are not allowed in tag names, feature names, or attribute names.

Using Literals

Items outside of angle brackets “<>” are literals. For example:

- `reference_string ::= <Automation_object_reference><attribute_reference> | <tag_name>`
- `Automation_object_reference ::= <absolute_reference>|<relative_reference>`

- absolute_reference ::= <tag_name>{.<contained_name>}
- tag_name ::= <identifier>
- contained_name ::= <identifier>
- relative_reference ::= <relative_name> | <relative_contained_reference>
- relative_contained_reference ::= MyContainer.<contained_name> | MyArea.<contained_name>
- relative_name ::= Me | MyContainer | MyArea | MyHost | MyEngine | MyPlatform
- attribute_reference ::= <value_ref>|<property_ref>
- whole_attribute_ref ::= [.<feature>][.<attribute>] | [.<feature>][.ATTRIBUTE(attribute)]
- value_ref ::= <whole_attribute_ref>[<array_index>]
- array_index ::= <open_bracket> {<index>} <close_bracket> [<open_bracket><index><close_bracket>][<open_bracket><index><close_bracket>]
- property_ref ::= <whole_attribute_ref>.<property>
- property ::= Value|Type|Quality|BitField|Dimension1|SecurityClassification|Locked|Category | propertyref
- propertyref ::= PROPERTY(Value|Type|Quality|BitField|Dimension1|SecurityClassification|Locked|Category)
- BitField ::= .00, .01, .02, …, .31 (valid ONLY for attributes of type MxInteger; otherwise Configuration error occurs at run time)
- attribute ::= <static_attribute>|<dynamic_attribute>
- static_attribute ::= [<static_attribute>.]<identifier>
- <dynamic_attribute> ::= <any_char_but>{<any_char_but>}
- feature ::= [<feature>.]<identifier>
- identifier ::= <valid_char>{<valid_char>}
- valid_char ::= <letter>|<digit>|<special_character>
- letter ::= any letter in alphabet of any language
- digit ::= any numerical character
- special_character ::= any graphics char, except the following:

  . + - * / \ = ( ) ` ~ ! % ^ & @ [ ] { } | : ; ’ , < > ? " whitespace
- whitespace ::= CR, LF, Tab, Space, FF, as returned by iswspace()
- any_char_but ::= any character except whitespace
- open_bracket := [

- close_bracket :=]
- Galaxy_identifier ::= <letter> | <digit>

Notes

- <tag_name> is an object's unique name.
- <contained_name> is an object's optional contained name. It can be specified in a reference when an object is referred to as a contained child of another object.
- <index> is –1 or a positive integer from 1 to 32767.
- <identifier> is limited to a maximum of 32 characters.
- An <attribute> name or <feature> name can contain several <identifier> parts. The length of each <identifier> part can be up to 32 characters. Each <identifier> part is separated by a period. The maximum total length of the <attribute> name is 329. This name length applies to both static and dynamic attribute names. The maximum total length of the <feature> name is 329.
- <relative_name> and <property> replacements are case insensitive, including PROPERTY().
- If no attribute reference is specified, .PV is assumed. If PV is an attribute of type array, the resulting reference is invalid. For arrays, the .PV[] part must be explicitly supplied.

  The exception to this rule is a reference that is preceded with an @ sign. This reference refers to the object itself and not any particular attribute or property. Currently, this reference string format is used only in the **Execution Order** group on the **Object Information** page of the Object Editor.
- Do not use Property Names or InTouch Pseudo-Property Names for the names of features or attributes when enhancing an object's functionality on the **Scripts**, **Attributes** and **Field Attributes** pages.

  ArchestrA Property Names include: Locked, Category, HasruntimeSetHandler, Name, Type, Quality, Dimension1, Value, SecurityClassification, 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 and 31.

  InTouch Pseudo-Property Names include: #VString, #VString1, #VString2, #VString3, #VString4, #EnumOrdinal, #ReadSts, #WriteSts and #QString. For more information on InTouch Pseudo-Property Names, see the *InTouch® HMI Data Management Guide*.

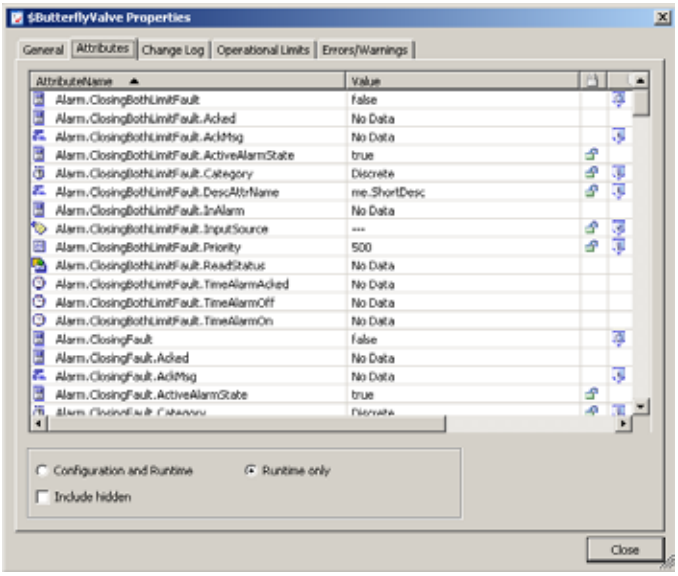
- If the object is a DIObject, use `obj.Attribute(attribute)[x]` rather than `obj.Attribute[x]` to avoid having to resolve the reference at the GR. Resolving an object reference at the GR can negatively impact performance.

Viewing Attributes in Objects

Within the ArchestrA IDE, you can view the attributes in an object. This lets you see what attributes are available.

To view the attributes in a selected object

1. Select an object.
2. On the **Galaxy** menu, click **Properties**.
3. To see the references for the selected object, click the **Attributes** tab.



This page shows you:

- **AttributeName** list — The selected attributes for the object, the current value, and locked and security status. The information shown depends on whether **Configuration and Runtime** or **Runtime Only** is selected.
- Value — Shows the value of the attribute.
- Locked/Unlocked — Shows if the attribute is locked or unlocked.
- Security — Shows the current security setting, if any.

4. To filter the view, select one or more of the following:
   - **Configuration and Runtime:** Select to show both configuration and run-time attributes for the selected object.

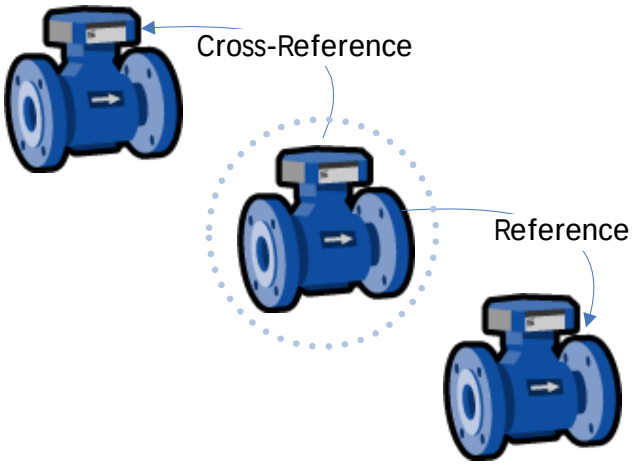
- **Runtime only:** Select to show only run-time attributes for the selected object.
- **Include hidden:** Select to show hidden attributes for the selected object.

5 When you are done, click **Close**.

Viewing References and Cross References

Objects have references and cross references.

- *References* are the objects the selected object is looking for.
- *Cross-references* are the objects looking for the selected object.



You can view references and cross-references for a selected object.

Some attributes are dynamic attributes. These are attributes that get created during run time and exist only in run-time. A device integration (DI) reference to a hardware register is a good example.

The attribute referencing a hardware register does not exist at configuration time by default. DI instances create the attribute dynamically during run time if the hardware register exists in the target device.

Note: References and cross-references shown in the **Properties** dialog box only refer to interobject communications. Area associations, containment, or host assignments are not shown.

To view references and cross-references

1 Select an object.

2 On the **Galaxy** menu, click **Properties**.

3 To see the references for the selected object, click the **References** tab. You see:

- Source Attribute — The attribute in the selected object referencing an attribute in another object in the application Galaxy.
- Attribute Reference — The reference string within the **Source Attribute**. This is either an absolute reference or a relative reference.
- Target Attribute — The absolute reference of the **Attribute Reference**. If this is a reference to a dynamic attribute, the **Target Attribute** only lists the Tagname name of the instance.

4 To see the cross references for the selected object, click the **Cross References** tab. You see:

- Target Attribute — The absolute reference of the **Attribute Reference**. If this is a reference to a dynamic attribute, the **Target Attribute** only lists the Tagname of the instance.
- Attribute Reference — The reference string within the **Source Attribute**. This is an absolute reference or a relative reference.
- Source Attribute — The attribute in the selected object that is referencing an attribute in another object in the application Galaxy.

5 When you are done, click **Close**.

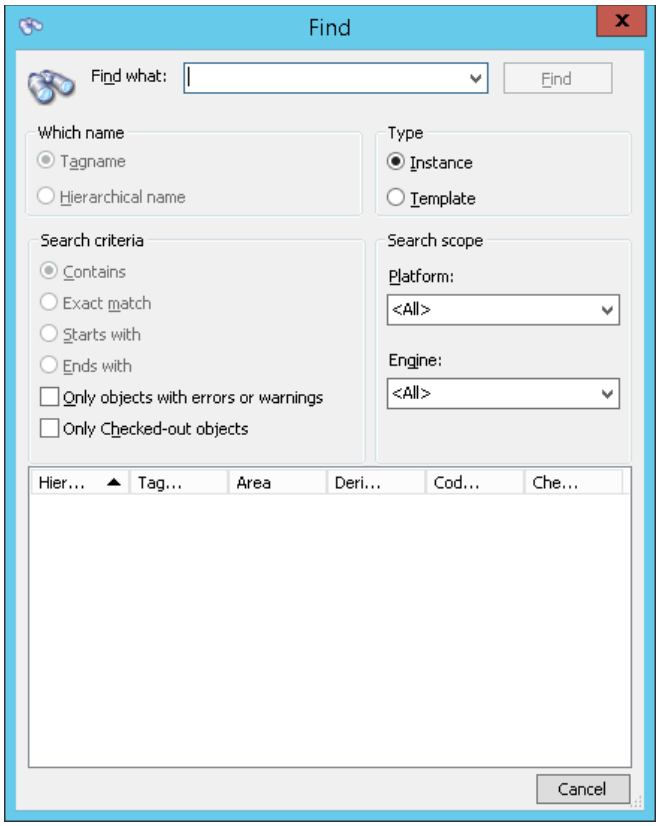
Finding Objects

Your Galaxy can get very large and can include many objects. It can become difficult to find a specific object. You can search for templates or instances, and you can search by part or all of a tag name or hierachical name. You can limit a search by platform and engine. Other search options are:

- Find objects with errors or warnings
- Find checked-out objects

To search for objects

1 On the **Edit** menu, click **Find**. The **Find** dialog box appears.



2 Do some or all of the following:

- In the **Find what** box, type some or all of the name of the object.
- In the **Which name** area, select either **Tagname** or **Hierarchical name** as the type of name you entered in the **Find what** box.
- In the **Type** area, specify if you are looking for an **Instance** or a **Template**. If you select **Template**, the **Which name** and **Search scope** groups are unavailable.
- In the **Search criteria** area, specify how to search for the name in the **Find what** box. The options are: **Contains, Exact match, Starts with** or **Ends with**.
- Search for objects in an error or warning state by selecting **Only objects with errors or warnings**.
- Search for objects that are checked out by selecting **Only Checked-out objects**.
- Limit the search scope by selecting from the **Search scope** lists.

3 When you are done specifying the search criteria, click **Find**. The search results appear in the bottom pane.

4 Double-click an object in the results pane. The object is located and selected for you in the **Application views** area.

If you double-click a Backup AppEngine, the IDE opens the **Deployment view** and the object is selected there. See "Working with AppEngine Redundancy" on page 434 for more information.

Using Galaxy References in InTouch

You can use Galaxy references in InTouch. Use the InTouch Tag Browser in unlimited selection mode to browse and include references from an Application Server Galaxy in InTouch applications you are developing.

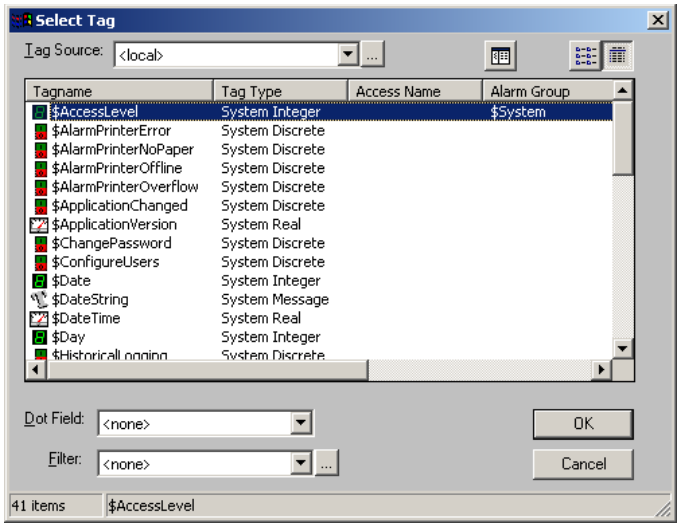
You also can use Galaxy references in InTouch in a multi-galaxy environment by using both the **Galaxy:** access name and the remote GalaxyName as part of the item name. For more information, see "Using InTouch with Multiple Galaxies" on page 297.

Note: You must install the Bootstrap and IDE on the InTouch node to create the TagSource to browse Galaxy objects.

The following lists the primary ways you can open the Tag Browser in InTouch to see unlimited selection mode:

- Double-click an animation link tagname or expression input box.
- Double-click an ActiveX or wizard tagname or expression input box.
- Double-click an empty area in any InTouch QuickScript window.
- In the InTouch QuickScript editor, select **Tagname** on the **Insert** menu.
- Press the **Alt+N** keys in the InTouch QuickScript editor.
- Double-click a blank **New Name** box in the **Substitute Tagnames** dialog box.

- Double-click the **Tagname** input box in the SQL Access **Bind List Configuration** dialog box.

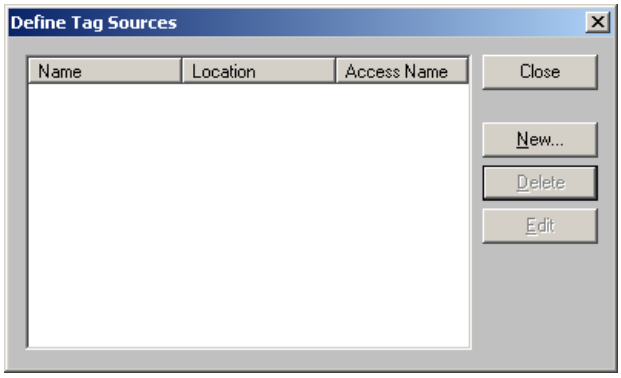


For complete information about using the InTouch Tag Browser, see the InTouch documentation.

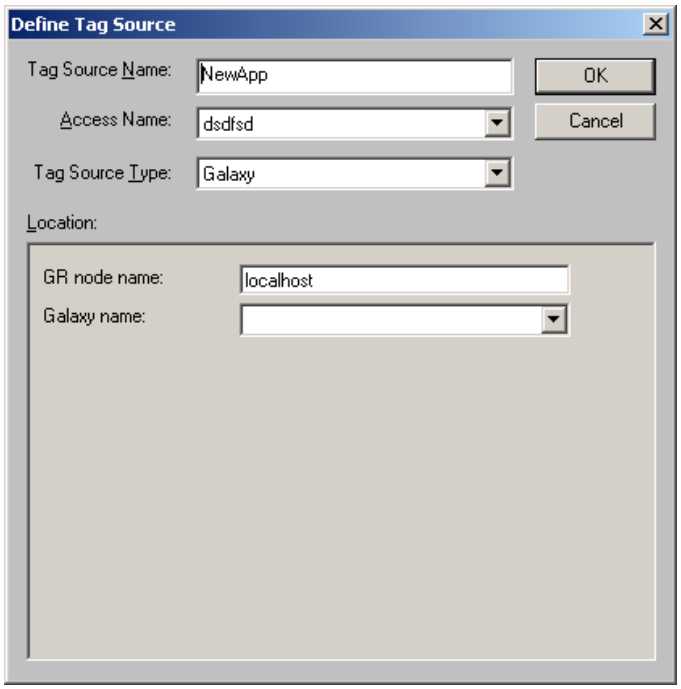
Before you can browse Galaxy references, you must define a new tag source.

To define a new tag source

1 In the InTouch HMI **Tag Browser**, click the **Browse** button to the right of the **Tag Source** list. The **Define Tag Sources** dialog box appears.



2 Click **New** to open the **Define Tag Source** dialog box.

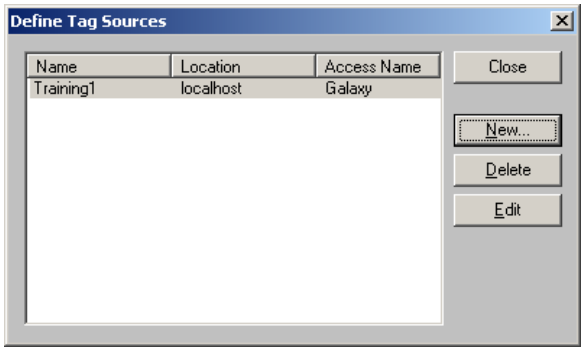


3 Type a **Tag Source Name**. This name appears in the **Tag Source** box of the Tag Browser. For example: Training1.

4 Select Galaxy from the list for **Access Name** and **Tag Source Type**.

5 In the **GR node name** box, type the Host Name of the computer on which the Galaxy is located. If the Galaxy is located on the same computer, use `localhost`.

6 In the **Galaxy Name** list, select the name of your Galaxy. For example: Training. Assuming the examples given in steps 3 and 5, the **Define Tag Sources** dialog box appears as follows.



7 Click **Close**. The Tag Browser appears. Now you can browse the TagNames.

To browse attribute references in a Galaxy

1. Open the **InTouch Tag Browser** in unlimited selection mode.
2. In the **Tag Source** box, select the tag source you created in the previous steps (Training1 in the example). The **Attribute Browser** appears.
3. Select the object and attribute you want to reference in your InTouch application and click **OK**.

Note: The next time you open the Tag Browser, the **Attribute Browser** automatically opens. To change that, exit the **Attribute Browser** without selecting anything by clicking the blue arrow at top right. The Tag Browser appears and it defaults to the InTouch Tagname Dictionary.

For more information about using the Attribute Browser, see “Referencing Objects Using the Galaxy Browser” on page 93.

Chapter 12

Working with Security

Galaxies are created without security. After a Galaxy is created, you can assign security to manage access. Using security lets you manage access to:

- IDE for configuring and managing objects.
- ArchestrA System Management Console (SMC) for performing maintenance and system administration functions.
- Any run-time operations.

This section describes the architecture of ArchestrA security and how to use it to manage access to configuration and run-time aspects of your Application Server application.

For more information on ArchestrA security, see the Schneider Electric Software Global Customer Support website.

About Security

Security settings let you control access to:

- User interfaces in the ArchestrA environment.
- Object attributes and the data they represent.
- Connection to the SQL Server database used for the Galaxy Repository.

Note: For additional information about SQL Server security, see the section on SQL Server Rights Requirements in the *Wonderware System Platform Installation Guide*.

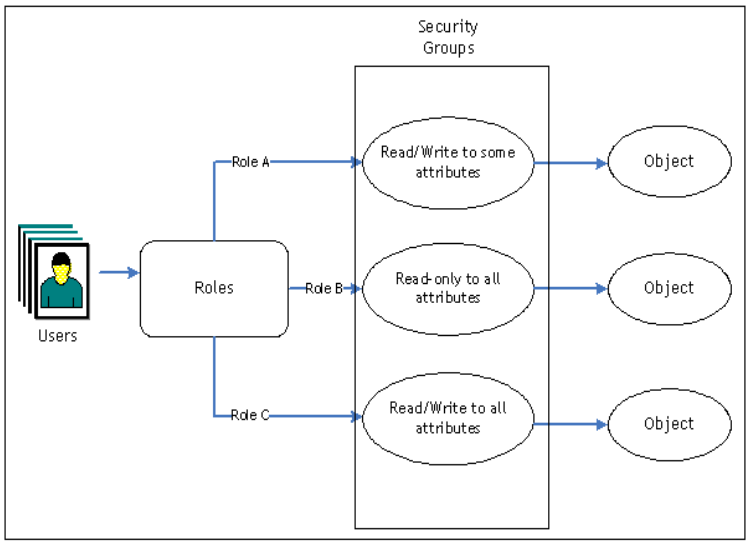
Each Galaxy in the Galaxy Repository manages its own security model. The security schema managed in a Galaxy is a three-level configuration model to create and maintain the following:

- Users associated with specific roles
- User roles associated with specific system administration, configuration and run-time (operational) permissions, which map to security groups
- Security groups associated with specific objects in the Galaxy

The default Galaxy Security model includes:

- Two users: DefaultUser and Administrator, both with full access to everything.
- One security group named Default.
- Two security roles: Default and Administrator, both with full privileges.

The security matrix defines a cascading model of users associated with specific roles that are associated with specific security groups that are associated with specific objects. User run-time permissions can vary from object to object, action to action, and process to process. The security icons associated with object attributes map directly to control points in the ArchestrA security model.



About Authentication Modes

You can select from three authentication modes to assign security:

- **Galaxy**: Uses local Galaxy configuration to authenticate users. All security for the Galaxy is specified and contained at the specific Galaxy level. When the user logs on, security credentials are checked and access to areas and activities is granted at the Galaxy level.
- **OS User Based**: Uses the operating system's user authentication system on an individual user level. All security for the Galaxy is specified and contained in the operating system (OS) on a user level basis. When the user logs on, security credentials are checked and access to areas and activities are decided at the OS user level.
- **OS Group Based**: Uses the operating system's user authentication system on a group basis. All security for the Galaxy is specified and contained in the user-to-roles mapping you created in the OS to assign security. When a user logs on, security credentials are checked and verified at the OS group level. OS groups are mapped to security roles in the Galaxy to allow access to areas and activities in the Galaxy.

Note: If you are using OS user-based security or OS group-based security and you have permissions to use the IDE, the **Log In** dialog box does not appear.

For more information about OS security, see “About OS Group-based Security” on page 367.

Multiple Accounts Per User

Regardless of the security system, a single user can have multiple accounts. For example, a user can have an account that provides permissions for working with instances but not templates. The same person can have another supervisory account for working with templates and managing users in the ArchestrA environment.

Each account requires a different user name and password. For example:

User Name	Password	Access
bsmith	password	Instances, not templates
bobsmith	super	Instances, templates, not managing users
Robertsmith	admin	Instances, templates, managing users

Changing Security Settings

After you change security for a Galaxy, you see the following behaviors and conditions:

- When you change the authentication mode security, the IDE restarts.
- To switch users, the person must log on as the new user by clicking **Change User** on the **Galaxy** menu.
- If you previously configured security under one authentication mode and then switch authentication modes, only those users created while configuring the new mode are available. Other users are not deleted, just unavailable in the new mode.
- Objects that are reassigned to different security groups are marked as “pending update” and require redeployment for the change in security group to take effect.
- If security was previously configured for an OS-based authentication mode, reconfiguring security synchronizes the user's full name and OS groups if some data in the OS has changed.

About Security Groups

Every object in the Galaxy belongs to only one security group. You can create and manage security groups that make sense for your organization. These security groups are mapped to roles on the **Roles** page.

Permissions determine what kind of access users have for each attribute. There are five basic operational permissions:

- Acknowledge alarms
- Change the value of attributes with security mode Configure
- Change the value of attributes with security mode Operate; this includes also security modes Secured Write and Verified Write
- Change the value of attributes with security mode Tune
- Confirm writes to attributes that require Verified Writes

By default, all currently used objects are assigned to a security group called Default.

A user who is a member of a role assigned to Security Role “Default” has permission to:

- Acknowledge alarms
- Change attribute values with “configure” security mode

- Change attribute values with “operate” security mode, including “secured write” and “verified write”
- Change attribute values with “tune” security mode
- Verify writes to Attributes with “verified write” security mode

For example, you want users in certain roles to only have permission to acknowledge alarms that are generated from objects contained in Area1. You have a role named Area1Acknowledgers. You need to:

1. Create a new Security Group, for example **SecGrpArea001**.
2. Assign all objects that are contained in area Area1 to Security Role **SecRoleArea001**.
3. On the **Roles** page, select the **Area1Acknowledgers** role. In the **Operational Permissions the Security Group** for **SecGrpArea001**, select **Can Acknowledge Alarms**.
4. Any user that belongs to the **Area1Acknowledgers** role can at least acknowledge alarms of objects contained in the security group **SecGrpArea001**. They do not have any other operational permissions for those objects.

About Roles

You can create and manage user roles that apply to your organization's processes and work-based authorities. Two roles are defined by default: Administrator and Default.

You can specify General and Operational Permissions for each role.

- General permissions relate to application configuration and administration tasks.
- Operational permissions relate to the security groups listed on the **Security Groups** page. By default, the Administrator has all permissions.

Note: You cannot modify the General permissions for the role of Administrator.

The **Operational Permissions** that can be associated with a role:

- Can Modify “Operate” Attributes: Allows users with operational permissions to do certain normal day-to-day tasks like changing setpoint, output and control mode for a PID object, or commanding a Discrete Device object.
- Can Modify “Tune” Attributes: Allows users to tune the attribute in the run-time environment. Examples of tuning are attributes that adjust alarm setpoints and PID sensitivity.

- Can Modify “Configure” Attributes: Allows users to configure the attribute’s value. Requires that the user first put the object Off scan. Writing to these attributes is considered a significant configuration change, for example, a PLC register that defines a Discrete Device input.
- Can Acknowledge Alarms: Allows users to manually acknowledge an alarm in the run-time environment.
- Can Shelve Alarms: Allows users to manually shelve and unshelve alarms.
- Can Modify Alarm Modes: Allows users to modify the mode of an alarm.
- Can Modify Plant States: Allows users to modify plant states for state-based alarming.
- Can Verify Writes: Allows users to provide an authentication signature for attributes configured with Verified Writes security classification. Only users with this permission can verify a task performed by users with the Can Modify “Operate” Attributes permission.

Important: For Galaxies that have security enabled and are migrated to Application Server version 3.5, the “Can Modify Operate Attributes” operational permission setting will be copied to the “Can Verify Writes” attribute. Starting with Application Server 3.5, Galaxies have the “Can Verify Writes” operational permission setting disabled by default.

About Users

If you select either OS based authentication mode, users with local accounts are added to the **Authorized Users Available** list in the following format: `.\<username>`.

If you select **OS Group Based** authentication mode, the local account must exist on each node in the Galaxy for successful authentication of that user on any computer in the Galaxy.

Two users are defined by default when a new Galaxy is created: Administrator and DefaultUser. These cannot be deleted in an open security setting and they are both associated with the default roles, Administrator and Default.

Important: When using OS-based security, do not use the local ArchestrA user account for Application Server user authentication purposes.

About SQL Server Security

SQL Server has its own security settings. When you install Wonderware Application Server, the following operating system groups and users and user accounts are created:

- aaAdministrators group
- ArchestrA user account
- aaGalaxyOwner user account

Caution: aaGalaxyOwner and ASBService are reserved OS user names. aaAdministrators and ASBSolution are reserved OS group names. Do not create users or groups with these names.

The aaAdministrators group, ArchestrA user account, and aaGalaxyOwner user account are defined for use with SQL Server, and are used for Galaxy operations. Do not delete this group or these accounts. See the section on SQL Server Rights Requirements in the *Wonderware System Platform Installation Guide* for additional information. This section also describes how to set the SQL Server Security Mode with the **aaConfig SQL** utility.

Configuring Security

Before you open the security editor for a Galaxy, make sure:

- No other user is connected to the Galaxy.
- All objects in the Galaxy are checked in.
- Your user profile has configuration permissions to change Framework Configuration/Modify Security Model, if security is previously configured.

If you try to open the security editor before these conditions are met, a warning message appears and you are denied access.

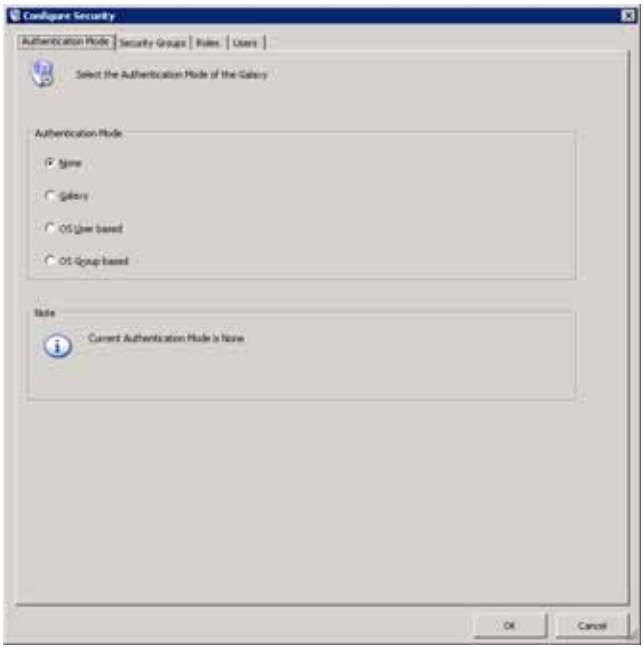
Caution: Do not configure security settings of the IDE while an IDE-managed InTouch application is opened for editing in WindowMaker.

Other users who try to open the Galaxy while you are configuring security are denied access to the Galaxy.

Caution: You can only change the ArchestrA administrator username or password using the Change Network Account Utility. The administrator account information is cached, and you may need to redeploy engines after you change the account so that the engines run using the current information. For more information about the utility, see “About ArchestrA User Accounts” on page 423.

To configure Galaxy security

1 On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears.



2 On the **Authentication Mode** tab, do the following:

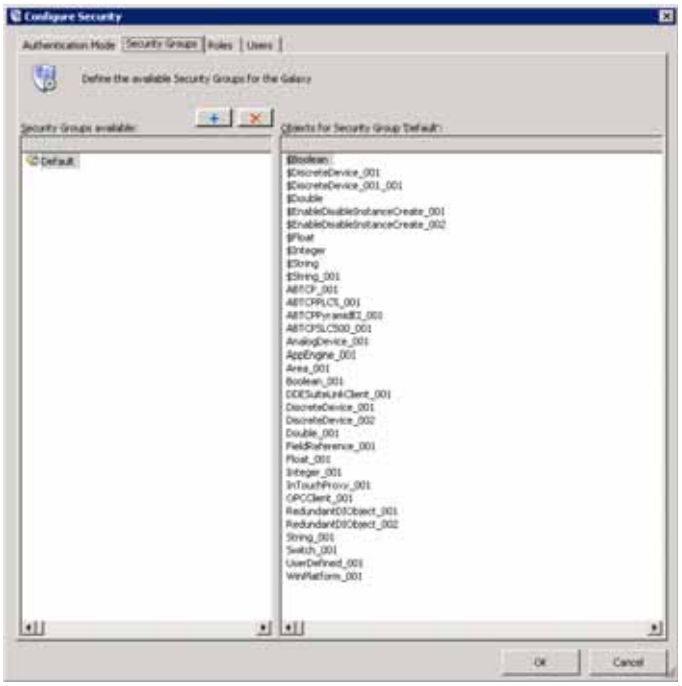
- Select the security type you want. Depending on what you select, more options become available.
- If you select **OS Group-based** and you are working on a slow or intermittent network, you can specify the intervals in milliseconds:

  Login Time: The time-out period (measured in milli-seconds) during which the system validates the user's membership against the OS groups selected as ArchestrA Roles. Minimum value is 0 (zero), maximum is 9,999,999. The default value is 1,000. If the login time is set to 0 (zero), which turns this feature off, the operation does not time out. Specify a value, based on the speed of your network and the number of groups configured in ArchestrA. The slower the network or the larger the number of groups, the greater the value.

  Role Update: The time between each validation attempt per OS group for the user's membership when a log on is attempted. The user membership update is done one role per **Role Update** interval to minimize network usage. The minimum allowed value is 0 (zero) and the maximum is 9,999,999. The default value is 0 (zero), which turns off this feature so the operation does not pause between validating user membership and groups. This option operates independently of the **Login Time** option. Even if **Login Time**

times out, the role update operation continues in the background and eventually updates user-to-role relationships for this user in the local cache. For more information about OS group-based security, see the Wonderware Development Network.

- Click **OK** or click the **Security Groups** tab.



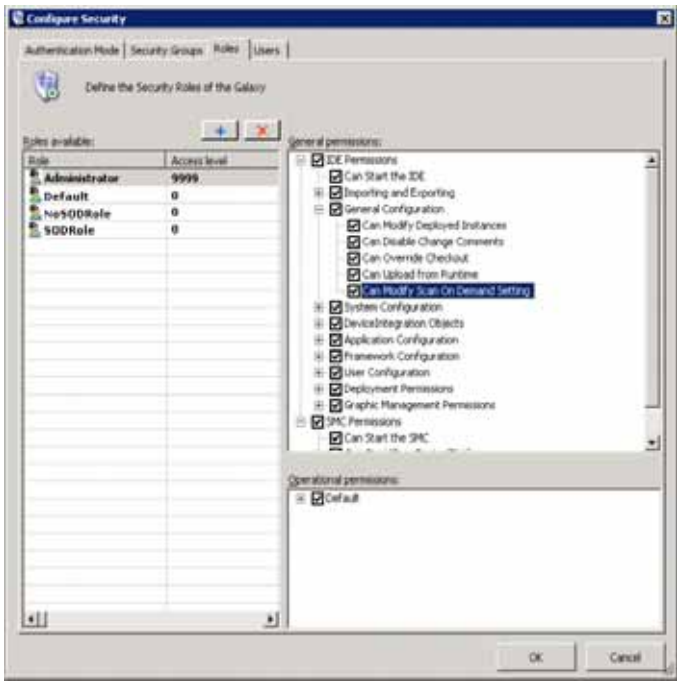
3 On the **Security Groups** page, do the following:

- Create a new security group by clicking the **Add** button. Type a unique name for the new group in the **Security Groups Available** pane. Security group names can be up to 32 alphanumeric characters, including a period. The name must include at least one letter and cannot start with $.

Note: Security group names are not case sensitive. `Admin` is the same as `admin`.

- Assign the objects you want the new group to have access to. Click the **Default** group. Drag the objects to the new security group.

- Click **OK** or click the **Roles** tab.



4 On the **Roles** page, do the following:

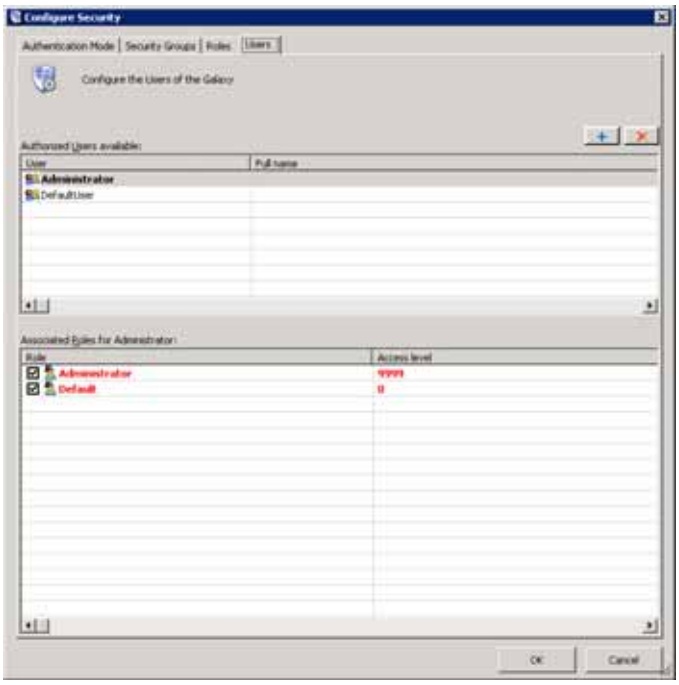
- Create a new role by clicking the **Add** button. Type a name for the new role in the **Roles Available** pane. Role names can be up to 512 alphanumeric characters, including a period.
- Select the **General** and **Operational Permissions** for the new role.

Important: In the **General permissions** area, clearing the **Can Start SMC** check box will still allow a user assigned to this role to start the SMC, but not to connect to Platform Manager.

Important: In the **General permissions** area, clearing the **Can Start/Stop the Engine/Platform** check box will still allow the user assigned to this role to set the engine or platform On Scan or Off Scan.

Important: If a role is given “Can Modify Deployed Instances” permission, make sure “Can Create/Modify/Delete…” permissions in the System Configuration, Device Integration Objects, and Application Configuration groups are also selected. This provides the role with check in and undo checkout abilities.

- Click **OK** or click the **Users tab**.



5 On the **Users** page, do the following:

- Create a new user by clicking the **Add** button.

  If you selected authentication as Galaxy, type a name for the user. User names can be up to 255 alphanumeric characters with no spaces.

  If you selected an OS-based authentication, click the **Browse** button in the **Roles Available** pane, select an existing user and click **OK**. The user name appears as `.\<username>`.

  While viewing Application Server events and alarms in InTouch, the "." appears as the user's domain if it is a local name. Otherwise, it appears as `<domain name>\<username>`.

Note: Users and **Roles** in bold red text are invalid with the selected **Authentication Mode**.

6 When you are done, click **OK**. You are prompted to log on to the currently open Galaxy.

Assigning Users to Roles

After you create users and roles, you can assign users to roles. On the **Users** page, all users in the Galaxy and the roles they are assigned are listed.

By default, the new user is associated with the Default role but not the Administrator role. This cannot be changed as every user belongs to the Default role. Double-click in a text box to change, if needed.

Note: All users are automatically assigned to the Default role in addition to any new roles you create and assign to them. The best way to manage permissions is to limit the permissions of the Default role to those permissions you want everyone to have. Then, add users to other roles with permissions for other parts of Application Server.

The Administrator user can log on any authentication mode except when security is disabled. When logged on as Administrator on the Galaxy Repository node, you can change the password of any Galaxy user without providing the old password.

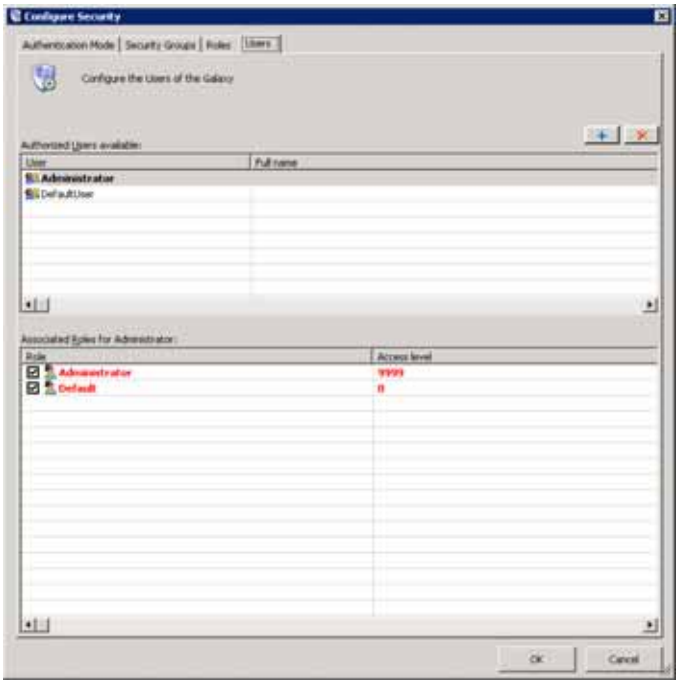
- In Galaxy authentication mode, you can edit the **User Name** in the **Change Password** dialog box.
- In OS-based authentication modes, the **User Name** of the OS user is shown. User information cannot be edited.

You can assign users to more than one role.

To assign a role to a user

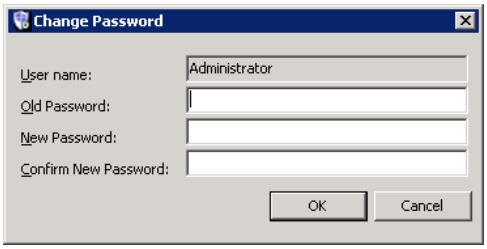
1 On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears.

2 Click the **Users** tab.



3 Select the user in the **Authorized Users available** area. Select a role in the **Associated Roles for <user name>** area.

4 Provide each user with a password by clicking **Change Password**. The **Change Password** dialog box appears.



Important: If an OS-based security mode is selected on the **Authentication Mode** page, changing a user's password changes the OS password for the user. Any ArchestrA password may be set to a maximum of 127 characters.

5 Enter the correct information. This information is used in the configuration, administration and run-time environment to authenticate users.

6 Click **OK**.

Deleting Security Groups

You can delete a security group you no longer need. Before you can delete a security group, make sure no objects are associated with it.

You cannot delete the Default security group.

Note: The aaAdminstrators group is required. If you accidentally delete it from Windows, you can run either the **Change Network Account** utility or the **aaConfig SQL** utility to restore it. If you delete the group from SQL Server, you must run **aaConfig SQL**. See the *Wonderware System Platform Installation Guide* for more information.

To delete a security group

1. On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears. Click the **Security Groups** page.
2. On the **Security** page, select the group you want to delete.
3.  Click the **Delete** button.

Deleting Roles

You can delete roles you no longer need. You cannot delete a role if any users are associated with it.

You cannot delete the Default and Administrator roles.

To delete a role

1. On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears. Click the **Roles** tab.
2. On the **Role** page, select the role you want to delete.
3.  Click the **Delete** button.

Deleting Users

You can delete users you no longer need.

Note: You cannot delete a user who is currently logged on.

To delete a user

1. On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears. Click the **Users** tab.
2. On the **User** page, select the user you want to delete.
3.  Click the **Delete** button.

Note: The ArchestrA user account and aaGalaxyOwner account are required. If you accidentally delete the ArchestrA user account from Windows, you can run either the **Change Network Account** utility or the **aaConfig SQL** utility to restore it. If you delete either account from SQL Server, you must run **aaConfigSQL** to restore it. See the *Wonderware System Platform Installation Guide* for more information.

About OS Group-based Security

If you use OS Group-based Authentication Mode, make sure you understand the Windows operating system, particularly its user permissions, groups and security features. ArchestrA OS Group-based security uses these Windows features. For more help, see the Microsoft online help or a 3rd party book about Windows security.

When you use local OS Groups as Roles, each node within the Galaxy must have the same OS Users, Groups, and user-group mappings to get the same level of access to the user at each node.

Note: Application Server uses Pre-Windows 2000 Group names for OS Group-based Authentication Mode, not the Active Directory Common Name.

Connecting to a Remote Node for the First Time

A newly-added user working on a computer with no access to the Galaxy Repository node cannot write to an attribute on a remote node if that user has never logged on to the remote node. This is true even if the user is given sufficient run-time operational permissions to do writes. To enable remote writing capabilities, log on to the remote node at least one time.

When attempting to access a remote Galaxy Repository node that contains an IDE, the user who is logging in must have admin privileges.

Cached Data at Log In

If you log on to ArchestrA on a workstation that belongs to Domain A and Domain Controller A fails, locally cached login data is used on subsequent log on attempts. When the domain controller returns to operation, your log on fails during the time period that trusts are being reestablished by the controller.

If, during the controller outage, your username/password data changed, you can use the old log on data if you intend to work locally.

If you want to perform remote operations, you should log on with the new log on data. If that fails, the trusts are being reestablished by the controller, and you should retry at a later time.

The user's full name is not available to any client (for example, an InTouch window) if the domain controller is disconnected from the network when the user logs on to ArchestrA for the first time.

If the user previously logged on to ArchestrA when the domain controller was connected, the user's full name is still available to the client from data stored in cache even if the domain controller is disconnected when the user subsequently logs on to ArchestrA.

Mixed or Native Domains

Your unique listing maps to the list of domains and user groups you see when you use the Windows tool for managing your domain. The list of domains and user groups appears differently in the group browser depending on whether your domain is configured as a Mixed or Native domain.

The two modes cannot be used interchangeably. We recommend using Native-mode domains for the mode's advantages in security and use of domain local groups.

Using Domain Local Groups

Domain local groups (DLGs) cannot be used in a mixed-mode domain. If you use DLGs to define role permissions, when group members attempt to access secured content, those group members might see "access denied" error messages.

This problem can occur if your domain is operating in mixed mode. In mixed mode, the scope of a DLG is limited to the domain controllers only. The DLG is not valid for member servers. The DLG can still appear valid as certain applications, such as SharePoint Portal Server resources, do not filter out invalid DLG entries.

If you grant the necessary role permissions to the domain user accounts individually, those users can gain access to the secured content, but this approach nullifies one of the advantages of using DLGs.

Using Security and Distribution Domain Configurations

You must select one of two group types when you create a new group.

- Security Group Type

  Use security groups to manage user and computer access to shared resources and to control which users receive group policy settings.

- Distribution Group Type

  Use distribution groups as email distribution lists with email applications such as Microsoft Exchange or Microsoft Office Outlook. You cannot use distribution groups to assign permissions or to filter group policy settings.

  For this reason, a domain group configured as a distribution type rather than a security type cannot be used for security purposes.

Using InTouch Access Levels Security

The **Roles** page includes the **Access Level** column. The **Access Level** is an InTouch function.

In InTouch, access levels are a schema for prioritizing run-time functions. In the ArchestrA security model, it only maps to InTouch values and has no prioritizing characteristics.

The maximum value is 9999 and the minimum is 0 (zero). If a user is assigned more than one role with different access levels, the higher access value is passed to InTouch.

Using Secured and Verified Writes

You can assign Secured Write or Verified Write security classification to a configurable attribute. These security classifications require authentication to perform run-time writes to the configured attribute. With authentication, users can write to such attributes by:

- Any assignment in a script that sets the value of the attribute, such as `"A=B";`

  where A references an attribute that is configured for Secured Write or Verified Write security classification.

- Any action on an animation graphic that alters the value of an attribute that has Signed Write or Verified Write security classification, such as a user input, a slider, an up/down button on a counter, or any other such actions.

- A script that uses the SignedWrite() function.

For information about the SignedWrite() function, see Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

The operator and verifier provide credentials for authentication by entering a valid security account (domain name, user name, and password), or by using a Smart Card if a Smart Card reader is attached to the system.

Note: Smart Card authentication is not supported in multi-galaxy environments for read/write operations to remote galaxies.

Operators can write to attributes configured with Secured Write or Verified Write security classification even if another user is logged on. This does not affect the session of the logged-on user.

- The operator must have the Can Modify Operate Attributes operational permission to perform the Verified Write.
- The verifier must have the Can Verify Writes operational permission to confirm the Verified Write.

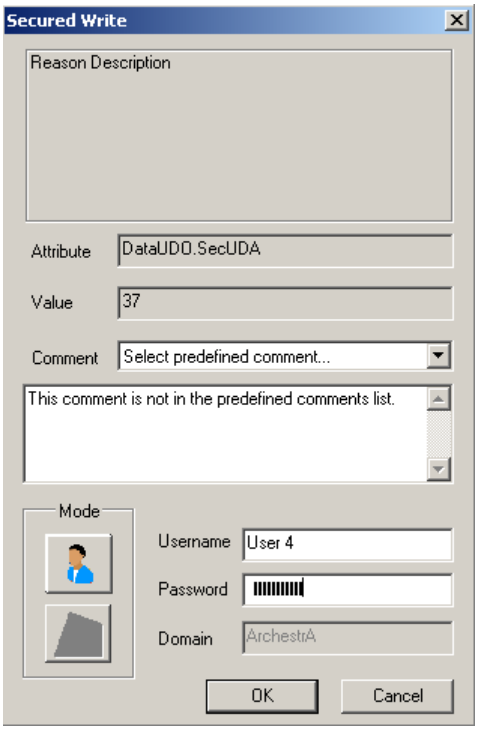
For more information, see “Setting Object Security” on page 75 and “About Roles” on page 357.

Note: The Secured and Verified Write features work only when security is enabled on both the Galaxy and InTouch applications and do not work if either Galaxy or InTouch security is set to None.

Using Configurable Descriptions and Comments for Secured and Verified Writes

At run time, the Secured Write or Verified Write dialog boxes appear when the operator attempts to write to an attribute configured for Secured Write or Verified Write. The dialogs enable configurable user input that can be used to provide information about the Secured or Verified Write.

Use the SignedWrite() script function to configure these inputs. The script function can be used only for ArchestrA attributes configured with Secured Write or Verified Write. It can be used only for symbol scripts and not for application object scripts.



Reason Description

The reason description is specific to an ArchestrA attribute. It explains the purpose of the attribute and the impact of changing it. If you do not configure a reason description, the reason description area is blank.

It is possible to use a script to directly assign a value to an attribute or field attribute that requires Secured or Verified Write. When the script is executed, the Secured or Verified Write dialog box appears and the reason description area displays an Attribute description, if there is one. If the Attribute does not have a description, then the reason description area displays the description of the Application Object to which the attribute belongs.

Predefined Comment List

The predefined **Comment** list enables the operator to comment on changing the attribute by selecting from a predefined list of comments.

Editable Comment Box

You can enable the operator to enter a comment in the **Comment** box.

For more information on configuring the reason description and the predefined **Comment** list and box, see "Working with the SignedWrite() Function for Secured and Verified Writes" in Chapter 3, "Managing Symbols," in the *Creating and Managing ArchestrA Graphics User's Guide*.

Note The reason message and **Comment** box and list are displayed in the **Secured Write** or **Verified Write** dialog box only in InTouch WindowViewer and not in ObjectViewer.

About Secured and Verified Writes Logged Information

Following a Secured or Verified Write a security Event is written to the event log, including the following information:

- The signee name
- Verifier name, if any
- Type of write: “Secured Write” or “Verified Write”
- Comment, if any entered by user
- Reason Description, if any provided
- Attribute or field attribute description, if any, or the Short Description of the Application Object, if no Attribute description exists

Chapter 13

Working with Languages

Application Server language features enable you to develop applications that can be switched to another language at run time. To enable run-time language switching, you must:

- Configure multiple languages for the application.
- Export your application text for offline translation.
- Translate one or more exported dictionary files.
- Import one or more translated dictionary files.

This section describes the features and procedures for configuring languages and enabling run-time language switching.

Defining and Configuring Galaxy Languages

You can set the Galaxy default language and you can define additional languages for purposes of switching symbol and alarm comment languages at run time.

The default language is set and languages are added only at the Galaxy level.

Graphics Language Switching

Run-time language switching applies to all portions of the graphics or animations that you create or configure using the ArchestrA Symbol Editor or InTouch WindowMaker. The language to display is set when the application is being shown at run time.

You can only view translations at design time for ArchestrA Symbols in the Symbol Editor. You cannot switch languages in WindowMaker at design time to see the translations.

Alarm Comment Language Switching

The alarm comment language switching feature allows you to format and export the configured alarm comments from attributes in a Galaxy to external files. The purpose of exporting the alarm comments to external files is to produce InTouch-compatible localized files to support language switching of alarm comments in the InTouch runtime environment.

You can import the alarm comment files back into the Galaxy after translation for convenient updating and re-export.

If you change an alarm comment after exporting the alarm comments, you must re-export the alarm comments. The translated alarm comments imported into InTouch and displayed at run time do not change dynamically when the alarm comment is edited.

Workflow

A typical workflow for a large Galaxy might consist of the following:

1. Export the Galaxy elements to be translated as a single file that is more manageable by the translator.
2. Import the translated file back into the Galaxy.
3. Export the translated file by Areas for use by InTouch applications.
4. Import the translated language file into InTouch.
   - In stand-alone InTouch, you can perform a single import operation.
   - In managed InTouch, you must import the language file into each managed InTouch application.
5. For InTouch applications that only target subareas of the Galaxy, exporting by Area is a more optimal workflow.

Important: Exporting, translating and importing the translated files back into the Galaxy are important steps for translating and managing translated files. Note that run-time language switching will not function if you do not import the translated language file as described in number 4.

Configuring Languages for a Galaxy

The language settings of the Galaxy control which languages are available to symbols and alarm comments. You cannot add a language at the symbol or attribute level. Languages are only added at the Galaxy level using the IDE.

When you open a symbol for editing using the Symbol Editor or open a managed InTouch application in WindowMaker, the language settings are retrieved from the Galaxy.

For example, you configure English and French languages for the Galaxy. You open symbol S1 in the Symbol Editor. Symbol S1 is now configured with English and French languages. Using the IDE, you add German to the list of configured languages. You must close the Symbol Editor and open symbol S1 again to see the German language available for the symbol.

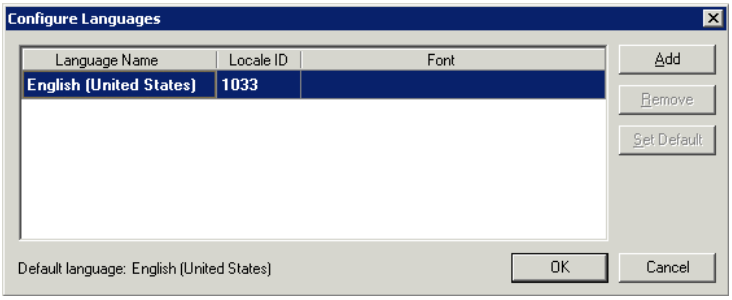
For more information about language switching in ArchestrA Symbols, see the *Creating and Managing ArchestrA Graphics User's Guide.*

Adding a Language to a Galaxy

Every Galaxy is associated with a base language. You must configure any additional languages that you want to support.

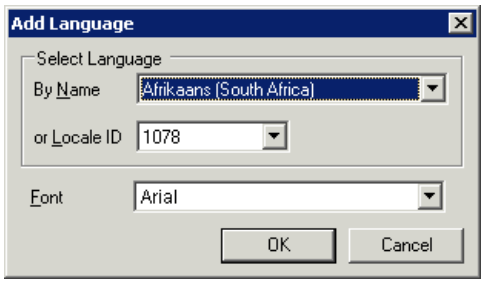
To add a language for a Galaxy

1. Using the ArchestrA IDE, open the Galaxy for which you want to add a language.
2. On the **Galaxy** menu, point to **Configure**, and then click **Languages**. The **Configure Languages** dialog box appears.



The **Configure Languages** dialog box shows the base (default) language of the Galaxy.

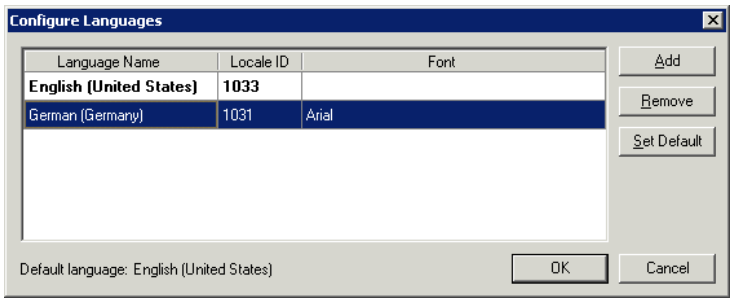
3 Click **Add**. The **Add Language** dialog box appears.



4 Specify the language and font for the translated text.

- In the **By Name** or the **Locale ID** list, click the language to add. If you select the language by name, the corresponding locale ID appears in the **Locale ID** list, and vice versa.
- In the **Font** list, click the name of the font.

5 Click **OK** to close the **Add Language** dialog box. The language you configured is listed in the **Configure Languages** dialog box.



6 To add more languages, repeat steps 3 through 5.

7 To remove a language, select the row in the list and then click **Remove**.

8 To specify a particular language as the default, select the row in the list and then click **Set Default**.

9 When you are done, click **OK**.

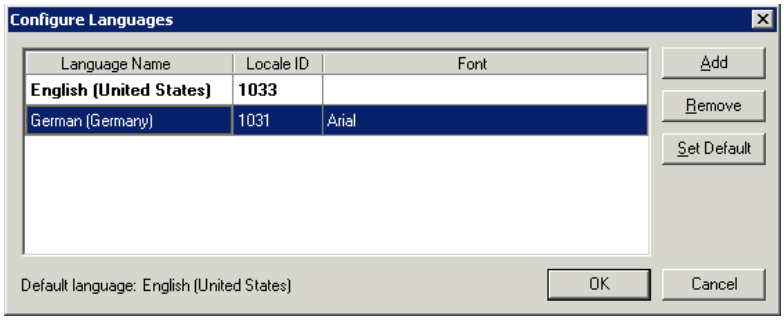
Removing a Language from a Galaxy

You cannot remove the default language. At least one language must be configured for a Galaxy.

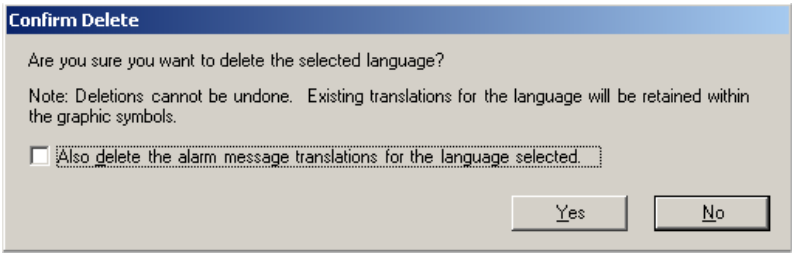
To remove a language for a Galaxy

1 Using the ArchestrA IDE, open the Galaxy for which you want to remove a language.

2 On the **Galaxy** menu, point to **Configure**, and then click **Languages**. The **Configure Languages** dialog box appears.



3 Select the row of the language to remove and then click **Remove**. A **Confirm Delete** dialog box appears.



4 The dialog box provides an additional language deletion option. Select the check box to delete alarm comment translations for the language selected. Leave the check box empty (unselected) if you want to keep the alarm comment translations for the selected language.

5 Click **Yes** to confirm deletion of the selected language as well as the alarm comment translations for that language if also selected.

Modifying the Font for a Language

The default font for all languages is Arial. You can change the font setting for a language that you have already added. You cannot edit the font for the default language.

The configured font for a language is used in design time when a specific translation is supplied for a language. It is also propagated to all translations for secondary languages when the fonts are not specifically overridden by you.

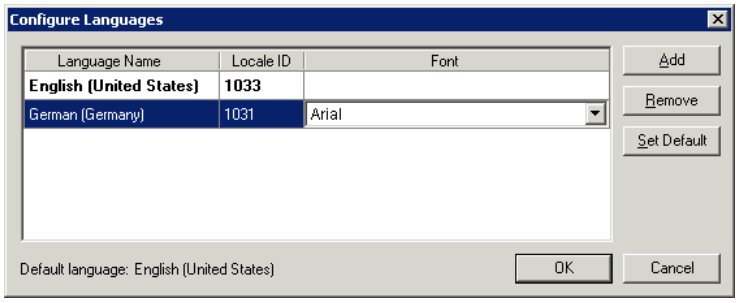
For example, if you create a text element and provide translation for the secondary language without modifying the font, the text is shown using the font from the Galaxy. However, if you manually modify the font, then the text will always use the chosen font. This same behavior also applies to run time.

For an ArchestrA Symbol to show the updated font from the Galaxy, WindowViewer must be restarted. No notification is provided to WindowViewer when the Galaxy font changes.

All managed InTouch applications are updated to use the new font for the selected language.

To change the font settings for a configured language

1. Using the ArchestrA IDE, open the Galaxy for which you want to change the font for a configured language.
2. On the **Galaxy** menu, point to **Configure**, and then click **Languages**. The **Configure Languages** dialog box appears.



3. In the list of languages, select the target language.
4. In the **Font** column, double-click on the name of the font so that a drop-down arrow appears.
5. Click the name of the new font.
6. Click **OK**.

Changing the Default Language for a Galaxy

When you change the default language for the Galaxy:

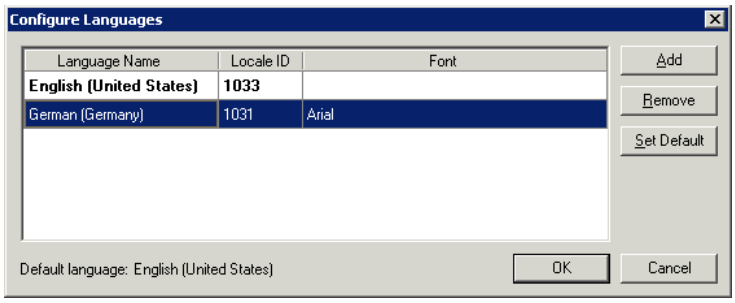
- The new default language is shown when the Symbol Editor opens.
- The base language for translation export is changed to the new default language.
- You cannot import translations for that language.
- Symbols opened in design time use the default language when showing text that does not have a specific translation in a secondary language.
- Symbols shown at run time use the default language if specific translated values are not provided for the currently viewed language.
- Thumbnails for symbols are shown using the default language.

The default language for managed InTouch applications is always the same as the InTouch installed language.

To change the default language for a Galaxy

1 Using the ArchestrA IDE, open the Galaxy for which you want to change the default language.

2 On the **Galaxy** menu, point to **Configure**, and then click **Languages**. The **Configure Languages** dialog box appears.



The **Configure Languages** dialog box shows the base language of the Galaxy.

3 In the list of languages, select the target language.

4 Click **Set Default**.

5 Click **OK**.

Note: Changing the default language does not change the alarm comment default language. Alarm comments are always stored as the Galaxy Repository default language, which is the locale of the operating system on which the Galaxy is created.

Exporting Symbol Text for Offline Translation

If your application has many strings, you typically send the text strings for your graphic symbols out for bulk translation. You can export symbol strings for translation and modify them using a text editor, an XML editor, or a spreadsheet program like Microsoft Excel.

We recommend that you do not make changes to symbols while symbol text is being translated.

If you make changes to your application after you export your dictionary files, you must export the dictionary file again. For more information, see “Exporting Symbol Text to an Existing Dictionary File” on page 384.

You can only export the text strings for one language at a time. You cannot export text strings for the default language.

Each symbol in a Galaxy is only exported one time, no matter how many times it is referenced.

When you export the text, you specify a folder for the dictionary files. Creating a new folder to export phrases for each language makes it easy to manage dictionary files. For example, ...\Galaxy1\My German Files\.

Also, all exported files have the following convention: <*GalaxyName*>_<*LanguageID*>.xml. If you will be exporting language data for different objects at different times, use separate target directories to prevent subsequent exports from overwriting the first export.

When you export the dictionary for an application, the files are .xml files that you can edit using Microsoft Excel 2003 or later.

Types of Language Dictionary Files

A language export creates the following types of dictionary files:

- Dictionary file for all Graphics Toolbox symbols, template symbols, and AutomationObject symbols. The file naming convention is: <*GalaxyName*>_<*LanguageID*>.xml. For example, if the Galaxy name is TestSample and the language being exported is French (Language ID = 1036) then the file name is TestSample_1036.xml.
- Dictionary file for each managed InTouch application.
- Dictionary file for each SmartSymbol in managed InTouch applications.

Note: The alarm comment export files are named and formatted differently from symbol dictionary files. For further information see “Exporting Alarm Comments for Offline Translation” on page 392.

Exporting Language Data for All Symbols in a Galaxy

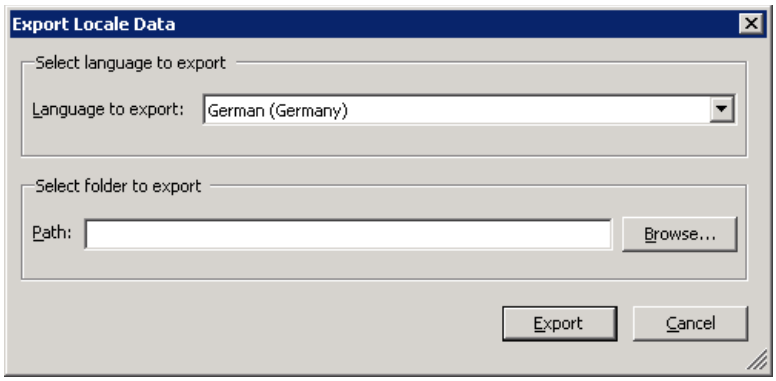
You can export language data for all symbols in a Galaxy at one time. The export contains language data for:

- Graphic Toolbox symbols.
- All symbols contained in AutomationObject templates, except for an $InTouchViewApp template.
- All symbols contained in AutomationObject instances.

The export operation only applies to symbols that are checked in.

To export language data for all symbols

1. Using the ArchestrA IDE, open the Galaxy for which you want to export symbol text.
2. On the **Galaxy** menu, point to **Export**, and then click **All Symbols**. The **Export Locale Data** dialog box appears.



3. Configure the export settings.
   - In the **Languages to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click **Browse** to select an existing folder or create a new folder.
4. Click **Export**. The export progress is shown.
5. Click **Close**.

Exporting Language Data for Specific Objects

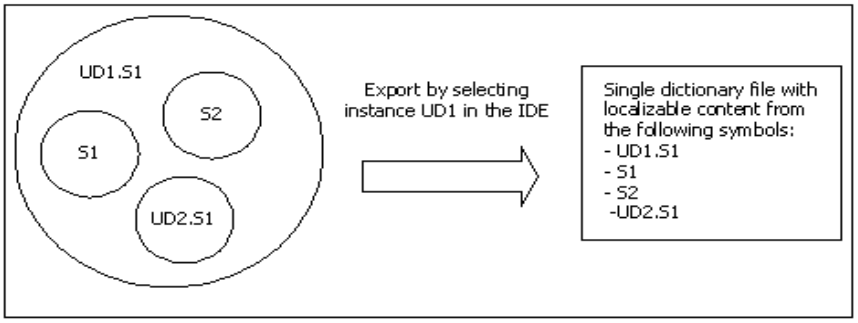
You can export language data for these types of Galaxy objects:

- Graphic ToolBox symbols
- AutomationObject templates and instances
- $InTouchViewApp templates and instances.

All symbols in an instance or template are exported.

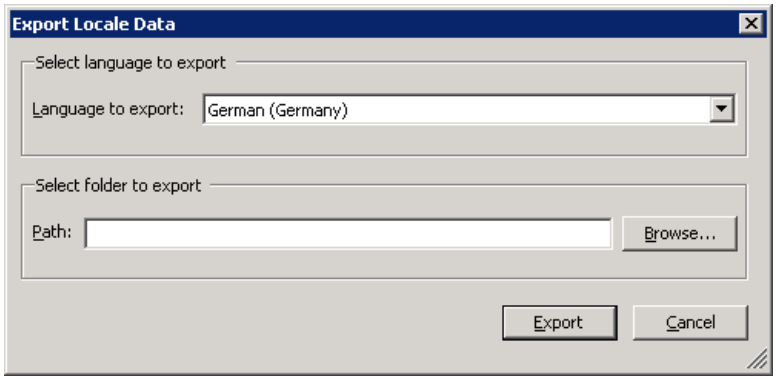
When you export the language data for a symbol, the language data for all symbols referenced by the symbol is also exported. The exported symbols can be referenced through direct embedding or through a show symbol animation.

For example, a Galaxy has Symbol1 and Symbol2 defined in the Graphic Toolbox. There are two instances called UserDefined1 and UserDefined2. UserDefined1 includes Symbol1. UserDefined2 includes Symbol1 and Symbol2. The instance symbol UserDefined1.Symbol1 embeds Symbol1 and Symbol2 from the Graphics Toolbox and one instance symbol UserDefined2.Symbol1. If you select the instance UserDefined1 and export the language data, then the language data would also be exported for the symbols Symbol1, Symbol2, and UserDefined2.Symbol1.



To export language data for specific objects

1. Using the ArchestrA IDE, open the Galaxy for which you want to export symbol text.
2. Select one or more objects to export.
3. On the **Galaxy** menu, point to **Export**, and then click **Localization(s)**. The **Export Locale Data** dialog box appears.



4. Configure the export settings.
   - In the **Language to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click **Browse** to select an existing folder or create a new folder.
5. Click **Export**. The export progress is shown.
6. Click **Close**.

Exporting Symbol Language Data for a Managed InTouch Application

You export language data for a managed InTouch application using the IDE. You cannot export translations from within the InTouch HMI.

The export includes strings for:

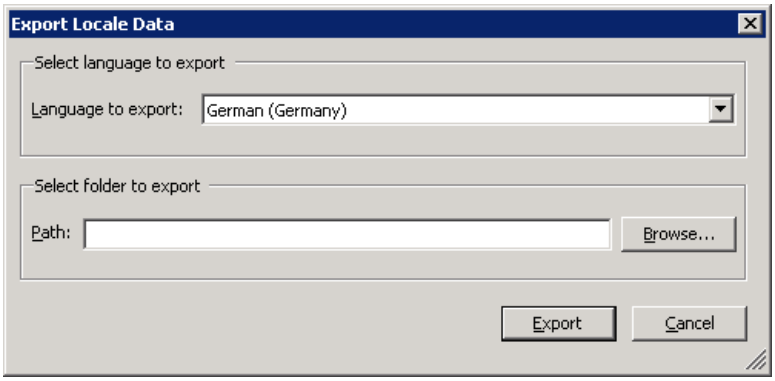
- All InTouch windows
- All SmartSymbols
- ArchestrA Symbols referenced by the $InTouchViewApp template

The export causes a cascade export of all referenced ArchestrA Symbols.

When you export language data for a managed InTouch application, the default language for the Galaxy is ignored. The InTouch default locale is always the installed InTouch locale. If the InTouch installed locale and the Galaxy default language are not the same, the export is skipped for the selected InTouchViewApp and a message is logged.

To export language data for a managed InTouch application

1. Using the ArchestrA IDE, open the Galaxy for which you want to export symbol text.
2. Select one or more managed InTouch applications.
3. On the **Galaxy** menu, point to **Export**, and then click **Localization(s)**. The **Export Locale Data** dialog box appears.



4. Configure the export settings.
   - In the **Language to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click **Browse** to select an existing folder or create a new folder.

5 Click **Export**. The export progress is shown.

6 Click **Close**.

Exporting Symbol Language Data for a Published InTouch Application

If you export languages for a published InTouch application, the following are not included:

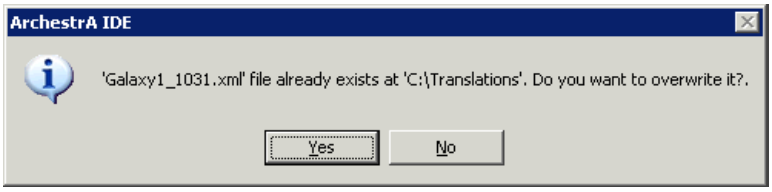
- ArchestrA embedded symbols
- Custom property overrides
- String overrides.

The export only includes native InTouch translations.

Exporting Symbol Text to an Existing Dictionary File

If you change your application after you translate the text strings, you need to export the text again. For more information, see “Exporting Symbol Text for Offline Translation” on page 379.

If you export more than one time to the same directory, a message box appears.



If you click **Yes**, the existing .xml files are deleted and the current text for symbols in the Galaxy are exported as a new .xml file.

Any existing translations for a symbol are reflected in the new .xml file.

Translating Exported Symbol Language Files

The procedures and tools for translating the exported language files are similar for both symbol text and for alarm comment text. However, there are important differences. Procedures for each are described in this section.

Translating Exported Symbol Text Dictionary Files

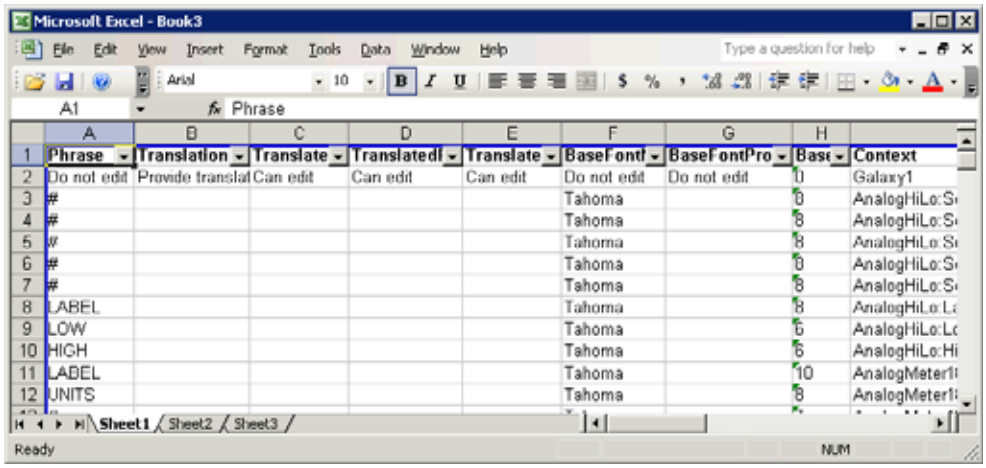
After you export the dictionary file containing your application text, use Microsoft Excel 2003 or later to edit the text or, for very large Galaxies, Excel 2007.

To translate an exported dictionary file

1. Open the XML file in Excel. The **Open XML** dialog box appears.
2. Click **As an XML** list, then click **OK**. A message may appear informing you that an XML schema will be created.
3. Click **OK**.

   The XML file opens in Excel with columns for the:

   - Phrases in your application.
   - Translated phrases from the translator.
   - Translated font name.
   - Translated font properties.
   - Translated font size.
   - Base font properties.
   - Base font size.
   - Context, phrase ID, language ID and foreign language ID.



Important: Only modify data in the **Translation**, **TranslatedFontSize**, **TranslatedFontName**, and **TranslatedFontProperty** columns. Do not change any column header. Do not insert or delete rows.

4. Type the language-specific text in the **Translation** column in the row that corresponds with the base language string in the **Phrase** column.

5 If necessary, change the font parameters for the translated strings. If you only provide a translation, the Galaxy-configured font for the language is used to render the text after the translation is imported. If you specify a font, it overrides the Galaxy-configured font.

- In **TranslatedFontName** column, type the font name.
- In the **TranslatedFontProperty** column, type the notation for the font properties:

  B = **bold**

  I = *italic*

  U = underline

  For example, if you want to the text to be bold, type **B** in the **TranslatedFontProperty** column. If you want the text to be bold and underlined, type **BU** in the **TranslatedFontProperty** column.

6 Save the file using XML Data as the file type.

Important: If you save as another file type, such as XML Spreadsheet, Excel changes the schema and the Galaxy cannot load the file. If you change the name of the XML file, the file will not import properly into the Galaxy.

Importing Translated Symbol Language Files

You can import alarm comment language files for re-export by area, to facilitate the export of new, untranslated alarm comments, and to facilitate re-export of previously translated alarm comments that have been changed.

For symbol text, you must import the translated dictionary files for each language to enable run-time language switching for those languages. All dictionary files for a given language should be placed in the same folder.

You can import files for only one language at a time. When you import, you select the desired language and specify the files to import.

Importing Translated Symbol Dictionary Files

All affected symbols and relevant objects are checked out before the import begins and are checked in when the import is done. If an affected symbol or object is already checked out, a message appears in the progress dialog box, and the import is skipped for the checked out object.

For a published InTouch application, only the native InTouch translations are imported.

You can configure how you want Galaxy and symbol/object mismatches handled during the import.

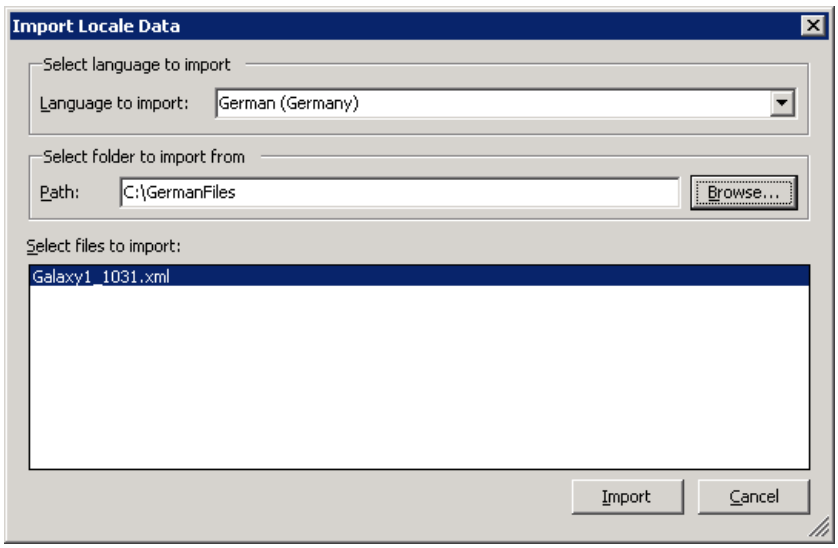
The import is skipped if:

- If default locale specified in the .xml translation file does not match the current default locale of the Galaxy.
- An InTouchViewApp's installed locale does not match the current default locale of the Galaxy.

Important: You cannot cancel the import after it starts.

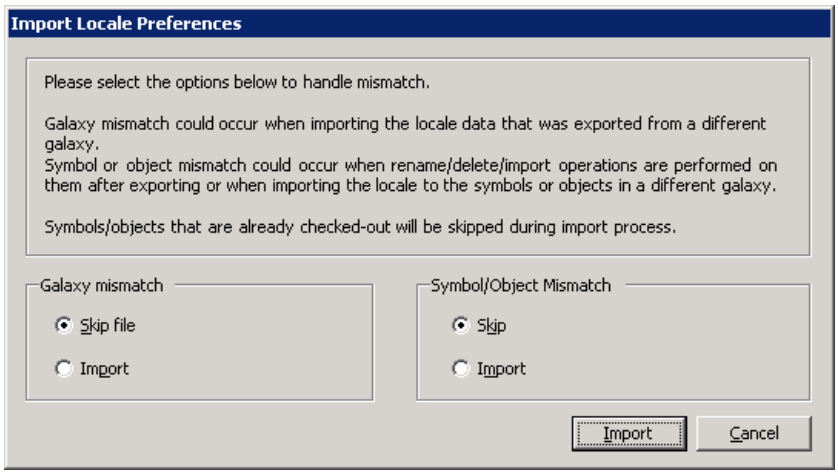
To import a translated dictionary file

1. Using the ArchestrA IDE, open the Galaxy for which you want to import symbol text.
2. Close all editors and check in all Galaxy objects.
3. On the **Galaxy** menu, point to **Import**, and then click **Localization**. When a message appears, click **OK**. The **Import Locale Data** dialog box appears.



4. Configure the import settings.
   - In the **Language to import** list, select the language dictionary to import.
   - In the **Path** box, specify the folder that includes the dictionary file to import.
   - In the **Select files to Import** box, select the .xml files to import. Only files that include the current Galaxy name and the locale ID for the selected language are shown.

5 Click **Import**. The **Import Locale Preferences** dialog box appears.



6 In the **Galaxy mismatch** area, configure how you want Galaxy mismatches handled. A Galaxy mismatch occurs when you try to import a translation file that was exported from a different Galaxy. The Galaxy name in the translation .xml file is used to match the current name of the Galaxy.

- Click **Skip file** to skip all the files that do not contain the current Galaxy name in the file name.
- Click **Import** to import all the selected files, regardless of what the Galaxy name is in the .xml filename.

- In the **Symbol/Object Mismatch** area, configure how you want symbol and object mismatches handled. A symbol or object mismatch occurs when the name of the symbol and the internal ID (GObjectID) of the symbol do not match what is within the .xml file. Objects include AutomationObjects and InTouchViewApp objects.
  - Click **Skip** to skip the symbols and objects that have mismatch names or mismatch IDs in the .xml file
  - Click **Import** to import a symbol or object only if it has a matching name or matching ID. If the name resolves to one object, and the ID resolves to another object, then the import is skipped.

  For examples, see “Examples of Symbol or Object Mismatch Handling during Language Imports” on page 389.

7 Click **Import**. The import progress is shown.

8 Click **Close**. The **Import Language Dictionary Files** dialog box appears.

9 Click **Check In**. The check in progress is shown.

10 Click **Close**. A summary of the import is shown.

11 Click **OK**.

Examples of Symbol or Object Mismatch Handling during Language Imports

The following table shows an example of handling mismatch conditions for a toolbox symbol. The bold name/ID is the matching name/ID in the .xml file and current Galaxy during import.

	Symbol Name and ID in the Galaxy while Exporting	Symbol Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
1	S1, 100	**S1, 100**	No change	Import to S1	Import to S1
2	S1, 100	S2, **100**	Rename S1 to S2	Skip	Import to S2
3	S1, 100	**S1**, 200	Delete S1, Create/Import S1	Skip	Import to S1
4	S1, 100	**S1**, 200 S2, **100**	Rename S1 to S2 Create/Import S1	Skip	Ambiguous, skip import
5	S1, 100	No S1 and No 100 in the Galaxy	Deleted S1	Skip	Skip import

The following table shows an example of handling mismatch conditions for an AutomationObject symbol. The bold name/ID is the matching name/ID in the .xml file and current Galaxy during import.

	Symbol Name and ID in the Galaxy while Exporting	Symbol Name and ID in the Galaxy while Importing	Object Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
1	Pump1, 10	S1, 100	**Pump1, 10**	No change	Import to S1 only if Pump1 has S1 with an ID of 100	Import to S1 only if Pump1 has S1 with an id of 100
2	Pump1, 10	S1, 100	Pump2, **10**	Rename Pump1 to Pump2	Skip	Skip import if S1 and 100 pointing to two different symbols in Pump2 (ambiguous) Import to S1 only if Pump2 has S1 or a symbol with id of 100.
3	Pump1, 10	S1, 100	**Pump1**, 20	Delete Pump1 Create and Import Pump1	Skip	Skip import if S1 and 100 pointing to two different symbols in Pump1 (ambiguous) Import to S1 only if Pump1 has S1 or a symbol with id of 100.

	Symbol Name and ID in the Galaxy while Exporting	Symbol Name and ID in the Galaxy while Importing	Object Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
4	Pump1, 10	S1, 100	**Pump1**, 20 Pump2, **10**	Rename Pump1 to Pump2 Create and Import Pump1	Skip	Ambiguous, skip import
5	Pump1, 10	S1, 100	No Pump1 and No 10 in the galaxy	Deleted Pump1	Skip	Skip import S1

Language Data Handling for Galaxy Operations

If you import or export objects, all language data within the associated symbols is imported or exported.

You can export all objects in the Galaxy, selected instances/templates, or selected symbols from the Graphics Toolbox. All language data is exported as part of the graphics definition, regardless of the languages configured in the Galaxy.

You can import Galaxy objects that contain language data from the same Galaxy or a different Galaxy. All language data in the ArchestrA graphics is imported, regardless of what languages are configured for the target Galaxy.

No ArchestrA Symbol or InTouchViewApp language data is read or modified during the import operation.

If you import an unmanaged InTouch application, the configured locales in the Galaxy are applied to the unmanaged InTouch application. No element translations in the InTouch application are removed during import, even though languages may no longer be visible in the InTouch HMI or available at run time in WindowViewer.

Exporting Alarm Comments for Offline Translation

As with symbol text export for translation, you typically send the alarm comment text out for bulk translation. You can export alarm comments and modify them using a text editor or a spreadsheet program such as Microsoft Excel. Once translated the files can be reimported to Application Server and can be imported by InTouch for run-time alarm comment language switching.

Guidelines and Recommendations

The following guidelines and recommendations will help you make best use of the alarm comment language switching feature:

Important: The exported file name is generated automatically and must not be changed.

Organizing Your Export

- You can only export the alarm comment text for one language at a time.
- When you export text, you specify a folder for the language files. We recommend that you create a separate folder for each language. For example, ...\Galaxy01\ChineseFiles\.
- For large Galaxies, exporting files for each area, after being translated and re-imported in the Galaxy, will perform faster when the alarm clients access those files.
- For small Galaxies or for InTouch applications, we recommend that you export the entire Galaxy for translation.

Translation File Formatting and Editing

Microsoft Excel 2007 is the recommended spreadsheet program. Excel 2007 provides enhanced formatting capabilities to convert the text file into tabular format. Also, Excel 2007 supports larger worksheets. See “Exporting Alarm Comments from Very Large Galaxies” on page 394.

Reimporting

- We highly recommend reimporting the translated files to the Galaxy after translation. Although reimport is not mandatory, you can use the reimport to optimize the export files to be used in the InTouch application.
- If you make changes to your alarm comments after you export comment text, you must export the alarm comments again. For this reason, we further recommend using the import functionality in Application Server to reimport the language file after each translation. This practice will avoid the need to retranslate alarm comments you have already translated.

For further information and procedures, see the following sections in this chapter:

"Exporting Alarm Comments from Very Large Galaxies" on page 394

"Exporting Alarm Comments by Area" on page 396

"Importing Translated Alarm Comment Language Files" on page 400

About the Alarm Comments Language File

The alarm comment language switching feature exports alarm comments to a .txt file. That file can be edited directly, but working in a text editor typically is more difficult and prone to errors.

- All files export as Unicode. We recommend that you save the exported files as Unicode.
- We recommend that you open the .txt file in Microsoft Excel 2003 or later for editing and save it as a .txt file.

The language file name is not user-configurable. The file naming convention for alarm comment export is:

Galaxy_<GalaxyName>_<localeID>_Alarm_Comments.txt. or

Galaxy_<GalaxyName>_<AreaName>_<localeID>_Alarm_Comments.txt, where the locale ID is the selected language decimal identification number.

For example, if the Galaxy name is "TestSample" and the exported language is Chinese, the language file name would be Galaxy_TestSample_2052_Alarm_Comments.txt.

For example, if the Galaxy name is “TestSample” the Area name is “Area001”, and the exported language is German, the language file name would be Galaxy_TestSample_Area001_1031_Alarm_Comments.txt

See “Translating Exported Alarm Comment Language Files” on page 398 for further information about alarm comment language files.

Exporting Alarm Comments from Very Large Galaxies

You can export language data in one operation for all alarm comments in a Galaxy. However, very large Galaxies with numerous alarm comments require longer export processing time and can result in a very large .txt language file. We recommend the following best practices for handling large Galaxies and a large number of alarm comments:

- Export alarm comments by Area rather than for the entire Galaxy to a single language file. See “Exporting Alarm Comments by Area” on page 396.
- If you prefer to export all Galaxy alarm comments to a single file, we recommend using Microsoft Excel 2007 rather than Excel 2003. MS Excel 2007 supports 1-million-line worksheets whereas Excel 2003 supports 65-thousand-line worksheets.

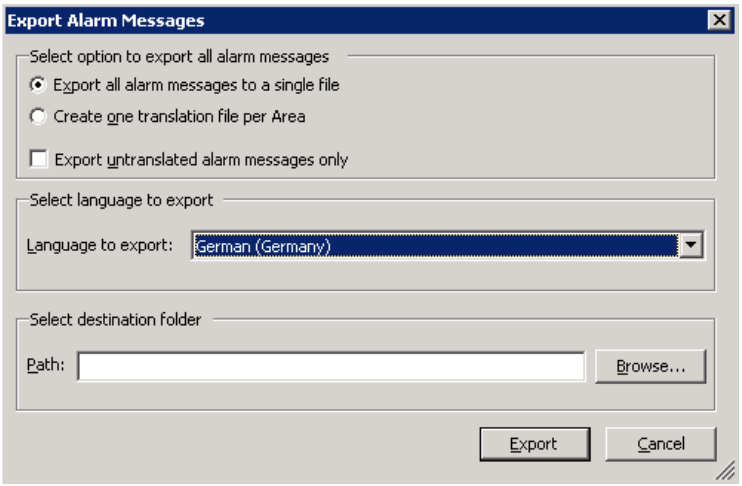
Exporting All Galaxy Alarm Comments

You can export language data for all alarm comments in a Galaxy at one time. The export operation only applies to objects that are checked in.

To export all Galaxy alarm comments to a single file

1. Using the ArchestrA IDE, open the Galaxy for which you want to export alarm comments.

2 On the **Galaxy** menu, select **Export**, then select the **Localization** menu option, and then select the **All Alarm Messages** menu option. The **Export Alarm Messages** dialog box appears.



3 Configure the export settings:

a In the **Select option to export all alarm messages** area, select the **Export all alarm messages to a single file** radio button.

b Check the **Export untranslated alarm messages only** check box if you have already exported, translated and re-imported your language file and you want to export only newer, untranslated messages. Otherwise, leave the check box unchecked.

c In the **Language to export** pull-down list, select the language to export. You can export only one language at a time.

d In the **Path** box, type the folder to which you want to export the language file. Click **Browse** to select an existing folder or create a new folder.

4 Click **Export**. The export progress box appears.

5 Click **Close** when the export completes.

Exporting Alarm Comments by Area

You can export all alarm comments by Area or you can export Alarm comments for a specific Area.

Using File Names

The *Galaxy_<GalaxyName>_<AreaName>_<localeID>_Alarm_Comments*.txt file name convention applies to specific Area alarm comment exports. Area names will be included the file name. For example, given a Galaxy name “TestSystem”, an Area name “Area001”, and an export to Chinese (language designation 2052), the language file name would be Galaxy_TestSystem_Area001_2052_Alarm_Comments.txt.

Exporting Objects Not Assigned to an Area

System Objects typically are not assigned to an Area. These are:

- Platform Objects
- Device Integration Objects
- Engine Objects

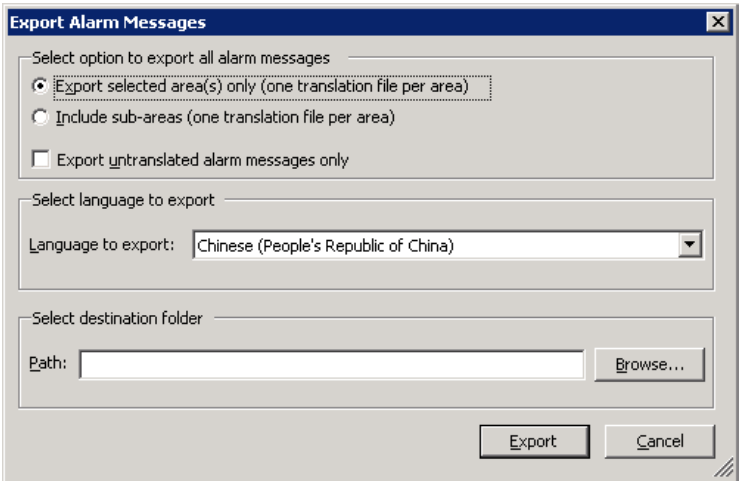
For purposes of language export, these System Objects form their own “Area” or become notification distributors for the purpose of sending alarms. Exporting System Objects follows these important conventions:

- System Object can only be exported from the Galaxy menu. They cannot be selected and exported using a context menu.
- System Objects can only be exported by exporting alarm comments for all Areas.

To export alarm comments for all Areas.

1 Using the ArchestrA IDE, open the Galaxy for which you want to export alarm comments.

2 On the **Galaxy** menu, select **Export**, then select the **Localization** menu option, and then select the **All Alarm Messages** menu option. The **Export (Area) Alarm Messages** dialog box appears.



3 Configure the export settings:

   a In the **Select option to export all alarm messages** area, select the **Export selected area(s) only** radio button, or select the **Include sub-areas** radio button to include all sub areas for each selected Area.

   b Check the **Export untranslated alarm messages only** check box if you have already exported, translated and re-imported your language files by Area and you want to export only newer, untranslated messages. Otherwise, leave the check box unchecked.

   c In the **Language to export** pull-down list, select the language to export. You can export only one language at a time.

   d In the **Path** box, type the folder to which you want to export the language file. Click **Browse** to select an existing folder or create a new folder.

4 Click **Export**. The export progress box appears.

5 Click **Close** when the export completes.

To export alarm comments for a selected Area or Areas

1 Using the ArchestrA IDE, open the Galaxy for which you want to export alarm comments.

2 Expand the **Application View** tree and select the Area(s) you want to export.

3 Right click to view the context menu. Select **Export** on the context menu, then select **Localization**, and then select **Alarm message(s) of selected Area(s)**.

As an alternate method to using the context menu, you can select the Area you wish to export, then use the Galaxy menu as in the preceding procedures.

4 Configure the export settings:

a In the **Select option to export all alarm messages** area, select the **Export selected area(s) only** radio button, or select the **Include sub-areas** radio button to include all sub areas for each selected Area.

b Check the **Export untranslated alarm messages only** check box if you have already exported, translated and re-imported your language files by Area and you want to export only newer, untranslated messages. Otherwise, leave the check box unchecked.

c In the **Language to export** pull-down list, select the language to export. You can export only one language at a time.

d In the **Path** box, type the folder to which you want to export the language file. Click **Browse** to select an existing folder or create a new folder.

5 Click **Export**. The export progress box appears.

6 Click **Close** when the export completes.

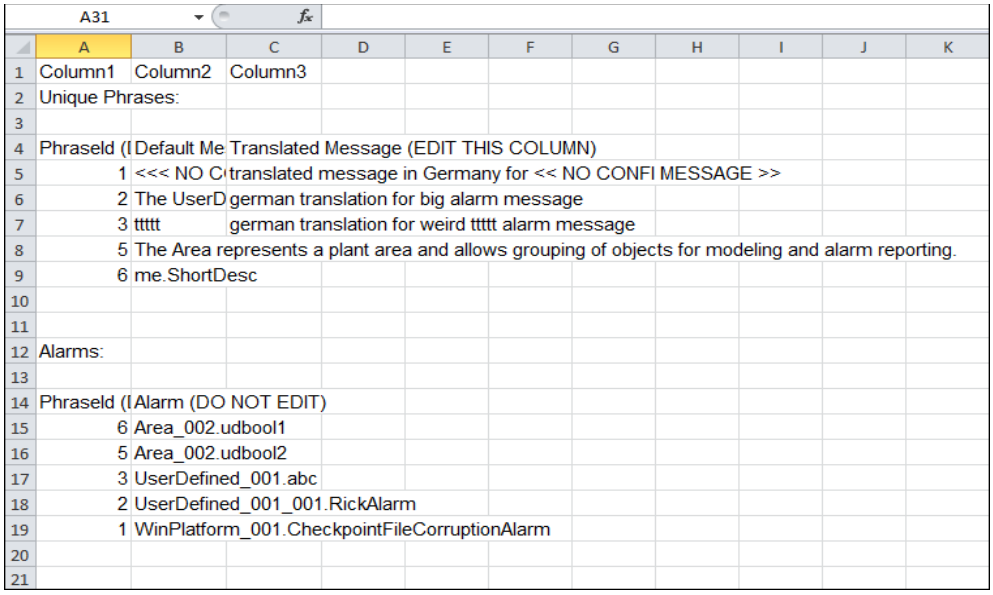
Translating Exported Alarm Comment Language Files

Alarm comments are exported to a .txt file, located in the selected directory. You can edit the files with a text editor, but we recommend opening and editing them in Microsoft Excel 2003 or Excel 2007.

Using the format-as-table feature in Excel greatly simplifies editing the specific text to be translated

To translate an exported alarm comment file

1 Open the .txt file in Excel. The Excel **Text Import Wizard** appears. Follow the wizard instructions to import the text file in tab delimited, general column data format. The file appears unformatted in Excel.



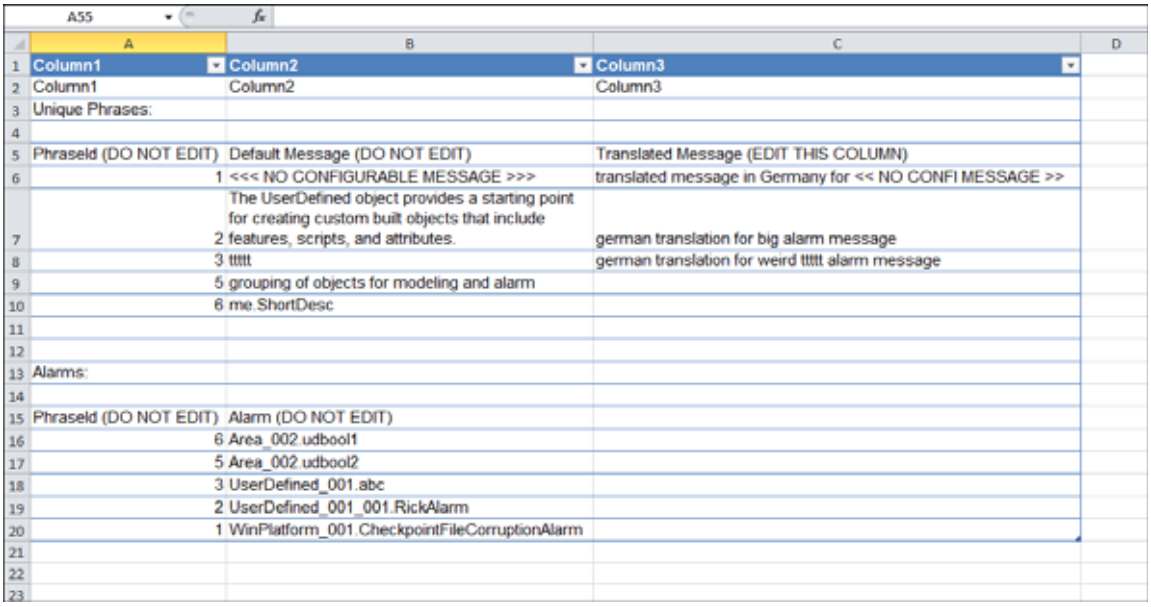
	A	B	C	D	E	F	G	H	I	J	K
1	Column1	Column2	Column3								
2	Unique Phrases:										
3											
4	PhraseId (I	Default Me	Translated Message (EDIT THIS COLUMN)								
5	1	<<< NO C	translated message in Germany for << NO CONFI MESSAGE >>								
6	2	The UserD	german translation for big alarm message								
7	3	ttttt	german translation for weird ttttt alarm message								
8	5	The Area represents a plant area and allows grouping of objects for modeling and alarm reporting.									
9	6	me.ShortDesc									
10											
11											
12	Alarms:										
13											
14	PhraseId (I	Alarm (DO NOT EDIT)									
15	6	Area_002.udbool1									
16	5	Area_002.udbool2									
17	3	UserDefined_001.abc									
18	2	UserDefined_001_001.RickAlarm									
19	1	WinPlatform_001.CheckpointFileCorruptionAlarm									
20											
21											

2 Format the text as a table in Excel:

a Select all of the rows containing data in Column A through Column C.

b Select the **Home tab**, then select **Format as Table**.

c Select the color pattern you want and click **OK** in the **Format as Table** dialog box. Excel will format the region to be edited.



	A	B	C	D
1	Column1	Column2	Column3	
2	Column1	Column2	Column3	
3	Unique Phrases:			
4				
5	PhraseId (DO NOT EDIT)	Default Message (DO NOT EDIT)	Translated Message (EDIT THIS COLUMN)	
6	1	<<< NO CONFIGURABLE MESSAGE >>>	translated message in Germany for << NO CONFI MESSAGE >>	
7	2	The UserDefined object provides a starting point for creating custom built objects that include features, scripts, and attributes.	german translation for big alarm message	
8	3	ttttt	german translation for weird ttttt alarm message	
9	5	grouping of objects for modeling and alarm		
10	6	me.ShortDesc		
11				
12				
13	Alarms:			
14				
15	PhraseId (DO NOT EDIT)	Alarm (DO NOT EDIT)		
16	6	Area_002.udbool1		
17	5	Area_002.udbool2		
18	3	UserDefined_001.abc		
19	2	UserDefined_001_001.RickAlarm		
20	1	WinPlatform_001.CheckpointFileCorruptionAlarm		
21				
22				
23				

The Excel display is divided into four regions:

- Column 1 contains Phrase ID numbers used internally in the language file. Do not edit this column.
- Column 2 contains the exported alarm messages. Do not edit this column.
- Column 3 contains the alarm message translations. Edit this column with your translation text.
- Below the alarm comment text columns are the alarms and phrase IDs, which map to column 1 and are used internally in the language file. Do not edit this area.

3 Translate the alarm comment text. Type the alarm comment translations into the appropriate rows in column 3.

Important: Only edit the text in column 3, translations. No other text in the file may be edited. You can edit the original alarm comment text only within the ArchestrA IDE.

4 Save the file as a .txt file in the directory you originally selected for export.

Important: Do not change the file name or the file type, otherwise it will not import correctly into the Galaxy.

Importing Translated Alarm Comment Language Files

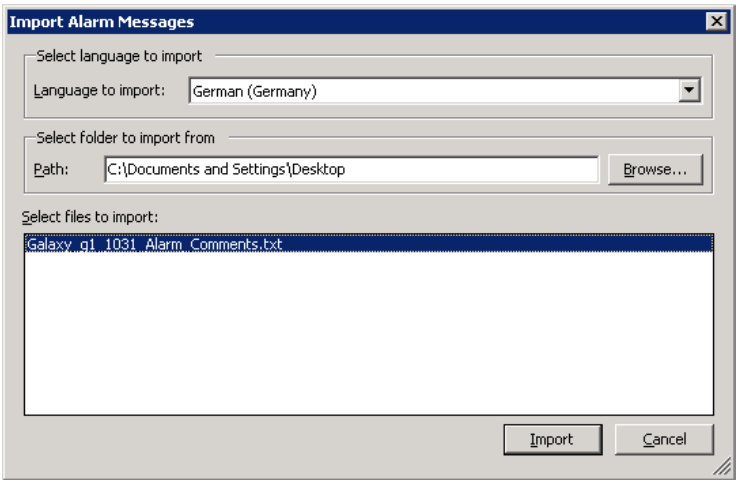
You can import translated alarm comment language files only from the Galaxy menu, not from the context menu for specific Areas.

To import alarm comment language files, you must first have exported alarm comments for translation of some or all of the alarm comments.

To import a translated alarm comment language file

1 Using the ArchestrA IDE, open the Galaxy for which you want to import translated alarm comments.

2 On the **Galaxy** menu, select **Import**, then select the **Localization** menu option, and then select the **Alarm Message(s)** menu option. The **Import Alarm Messages** dialog box appears.



3 Configure the import settings:

   a In the **Language to import** pull-down list, select the language to import. You can import only one language at a time.

   b In the **Path** box, type or browse to the folder where you previously exported your language file. The available language files appear in the **Select files to import** list.

   c Select a file or files to import.

4 Click **Import**. The import progress box appears.

5 Click **Close** when the import completes.

Note: If an alarm name or message do not match, the alarm comment import will display a message for each failed alarm message import.

To import a translated alarm comment language file into InTouch

1 Open the application in WindowViewer.

2 On the **Special** menu, select **Language**, then select the name of the language you want to switch to. Information from the corresponding translated dictionary file (if one exists and has been imported into InTouch) loads and displays.

3 Click **Close** when the import completes.

Re-exporting Alarm Comments

After exporting and translating alarm comments, you can re-import them to ArchestrA to help maintain up-to-date alarm comment language files.

After adding new alarm comments that require translation or after modifying existing, already translated alarm comments, you must re-export the language file to update the translations.

Exporting New Untranslated Alarm Comments

In the **Export Alarm Messages** dialog box, select the **Export untranslated alarm messages only** check box. This creates a file named _untranslated.txt. For example, exporting untranslated alarm comments in Chinese for a Galaxy name “TestSystem” would create a file named Galaxy_TestSystem_ 2052_Alarm_Comment_Untranslated.txt.

Exporting Modified Existing Alarm Comments

You can make changes to alarm comments that you previously exported, translated and imported back into ArchestrA. You must re-export the alarm comment language file(s) to update the translations.

In the **Export Alarm Messages** dialog box, select the same language option and folder you selected for the previous export. When asked, confirm that you want to overwrite the existing file.

Testing the Language Switching Functionality at Run Time

We recommend that you test the run-time language switching functionality. Follow the procedures outlined in this chapter to enable run-time language switching in your application. Then import the appropriate language file into InTouch. The import step applies to both stand-alone and managed InTouch applications.

To test the language switching functionality

1. Open the application in WindowViewer.
2. On the **Special** menu, point to **Language**, and then click the name of the language to switch to.

   The information from the corresponding translated dictionary file (if one exists and has been imported into InTouch) loads and appears.
3. When you are done, click **Close**.

Chapter 14

Managing Galaxies

You can back up and restore Galaxies, change the Galaxy you are working with, delete a Galaxy, and export and import all or part of a Galaxy.

If you want to create a Galaxy, see “Creating a New Galaxy” on page 21.

Backing Up and Restoring Galaxies

Periodically, you should back up your Galaxy. Backing up your Galaxy helps if you have a computer failure or other problem. If there is a failure, you can restore your Galaxy from the backup.

Use the Galaxy Database Manager to back up and restore your Galaxy. The Galaxy Database Manager is part of the suite of ArchestrA System Management Console utilities.

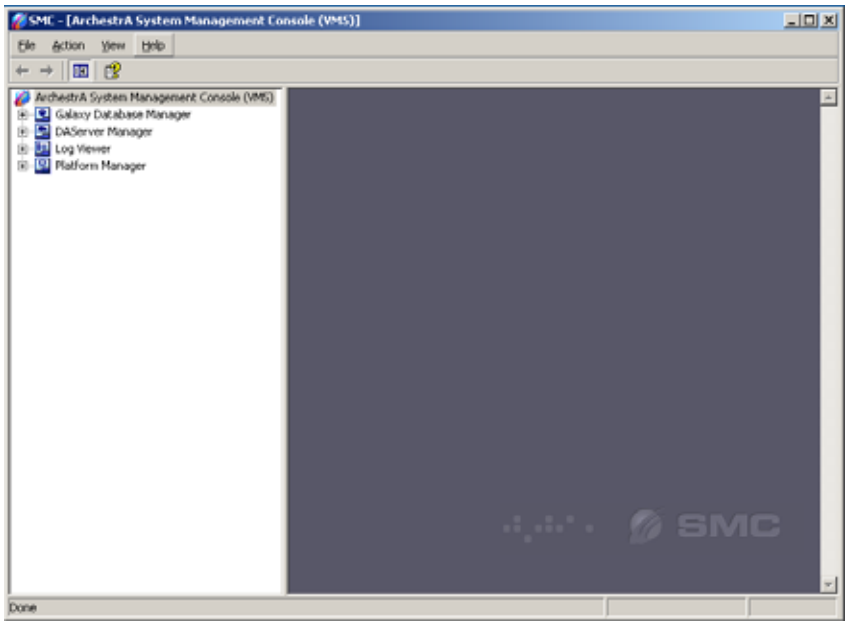
The aaConfigSQL utility is another utility that you may need to use when restoring Galaxies. aaConfigSQL is used to set ArchestrA user permissions for SQL Server. In most cases, the default level of permissions is adequate for restoring Galaxies.

However, if you are restoring a Galaxy created with an older version of Application Server, you may need to change the permissions level.

See the section about SQL Server Rights Requirements in the *Wonderware System Platform Installation Guide* for additional information about the aaConfigSQL utility.

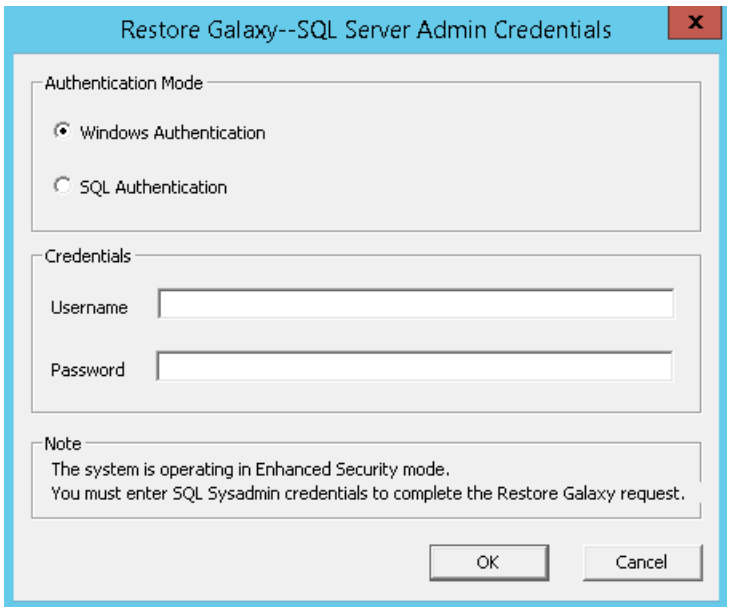
To open the Galaxy Database Manager

- Click **Start**, point to **Programs** and then to **Wonderware**, and then click **System Management Console**.



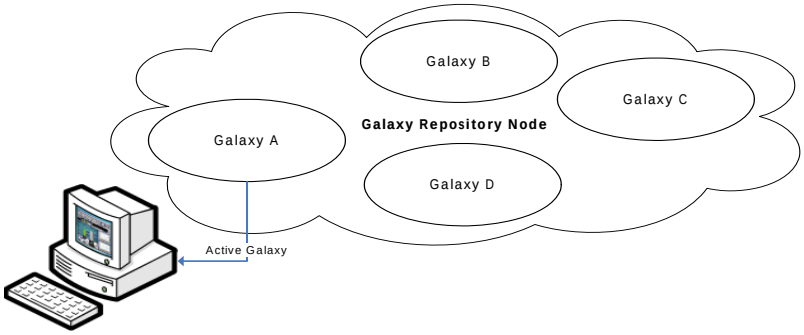
See the Galaxy Database Manager documentation for more information about backing up or restoring your Galaxy.

If Enhanced Security mode is in effect, you will be prompted to enter SQL SysAdmin credentials before you can restore a Galaxy created in an older version of Application Server.



Changing Galaxies

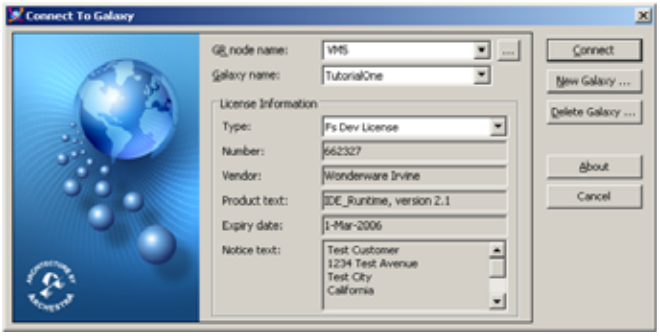
You can have many Galaxies in the Galaxy Repository. If you are a systems integrator, you have at least one Galaxy for every client.



For a Galaxy to deploy objects to other computers in the Galaxy, the Galaxy Repository (GR) must host a platform defined in that Galaxy. Because of this, you can only deploy one Galaxy from the Galaxy Repository at a time to the computers on your network. For more information about deploying, see “Deploying Objects” on page 196.

To change from one Galaxy to another

1 On the **Galaxy** menu, click **Change Galaxy**. The **Change Galaxy** dialog box appears.



2 Do one of the following:

- Select another Galaxy from the **Galaxy Name** list.
- Select another Galaxy Repository node from the **GR Node Name** list.

3 Click **Connect**.

Deleting a Galaxy

You can delete a Galaxy. Deleting a Galaxy removes all of the Galaxy information from your computer.

Before you delete a Galaxy, you must undeploy all objects within it. For more information about undeploying objects, see “Undeploying Objects” on page 202.

Make sure you select the right Galaxy to delete. After you delete a Galaxy, you cannot undelete it. You can only recreate it by restoring from a backup. For more information, see “Backing Up and Restoring Galaxies” on page 403.

To delete a Galaxy

1. Undeploy all objects from the Galaxy you want to delete.
2. Close any Galaxy and connected IDE you have open.
3. On the **File** menu, click **Change Galaxy**. The **Connect to a Galaxy** dialog box appears.
4. Select the Galaxy you want to delete, except the connected Galaxy.
5. Click **Delete Galaxy**.
6. At the prompt, click **Yes**.
7. When the Galaxy is deleted, click **Close**.

Galaxy Object Components Synchronization

The Galaxy Object Components Synchronization feature provides a detection and recovery mechanism when the client is out-of-sync with the server. Synchronization means synchronizing the installed object components on the client node with respect to the connected GR.

Synchronization of object components as defined in this section is fundamental to the subsequent functionality. For that reason, it is unnecessary to synchronize script libraries, graphics or InTouchViewApp-related files, client controls, IDE extensions, run-time components, security, or history data.

In the case of a run time out-of-synchronization condition, recovery is accomplished by redeploying the run-time platform.

Definitions

The following terms are important in understanding the Galaxy Object Components Synchronization feature:

- **Object Component:** Base template-dependent files

- **Server:** The GR node, which maintains the object component information
- **Client:** The IDE

Typical Out-of-Sync Scenarios

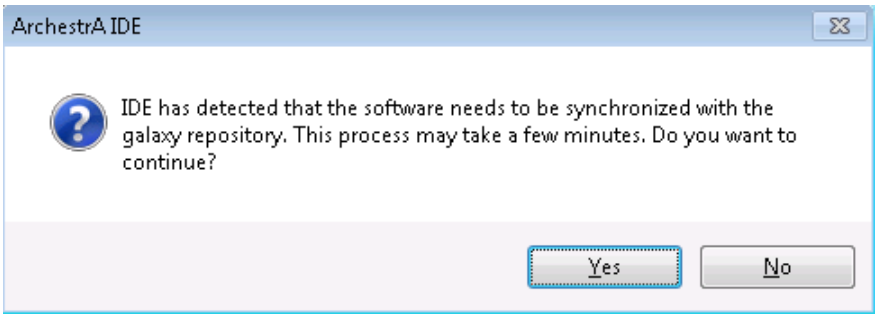
The GR and the client nodes can become out-of-sync in a number of ways. The following are the typical causes:

- Re-imaging the client or server node or both
- Restoring a Galaxy from a backup
- Deleting and creating a Galaxy on the server node

Using the Synchronization Feature

The synchronization detection and resolution process is the same for all scenarios. Out-of-sync conditions between a client and a Galaxy are detected and resolved at connection time.

If the system recognizes that the minimum connectivity requirements are met, and there are object versioning conflicts, then you are prompted about the out-of-sync state with the option either to synchronize or to reject the synchronization operation.



Select the option you prefer. If you select to synchronize, the synchronization operation will run without further prompts.

A warning message is placed in the Logger for any out-of-sync detection and synchronization issues.

Exporting a Galaxy Dump File

You can export instances and their configuration data in a Galaxy to a comma-separated values file with a .csv file name extension. After you export the Galaxy to a .csv file, you can edit it with Microsoft Excel 2000 or later. This makes it very easy to do large editing changes, such as search and replace.

Exporting only exports instances. Templates cannot be exported in .csv file format.

The .csv file contains the configuration for the checked-in versions of the selected instances and the checked-out instances of the user who does the Galaxy dump. If an instance is checked out by another user when you export the Galaxy, the checked in version is exported.

The file contains only those attributes that are unlocked and configuration time-writeable, one column per attribute. The following are not exported:

- Scripts libraries are not exported. Scripts within an object are exported.
- Attributes that are not text-based are not exported. For example, type QualifiedStruct is not exported.
- Custom object online help files are not exported.

Note: I/O Auto Assignment information is not included in the Galaxy dump file. For information about manually adding I/O Auto Assignment information to a Galaxy dump file for inclusion in the Galaxy load process, see "Enabling I/O Auto Assignment in the Galaxy Dump File" on page 411.

See "About the Galaxy Dump File Structure" on page 409 for specific information about the structure of the .csv file. Before you start, make sure you know what instances you want to export to a file.

To export objects to a Galaxy dump file

1. In the **Application views** area, select at least one instance. Shift+click to select multiple instances. You can export all instances derived from a template by selecting the template.
2. On the **Galaxy** menu, click **Export** and then click **Galaxy Dump**. The **Galaxy Dump** dialog box appears.
3. Browse to the location of the `.csv` file to which you want to dump the selected instances. Type the name of the file. Click **Save**.
4. When the Galaxy export process is done, click **Close**. A `.csv` file is created containing the selected objects and configuration data.

Editing the Galaxy Dump File

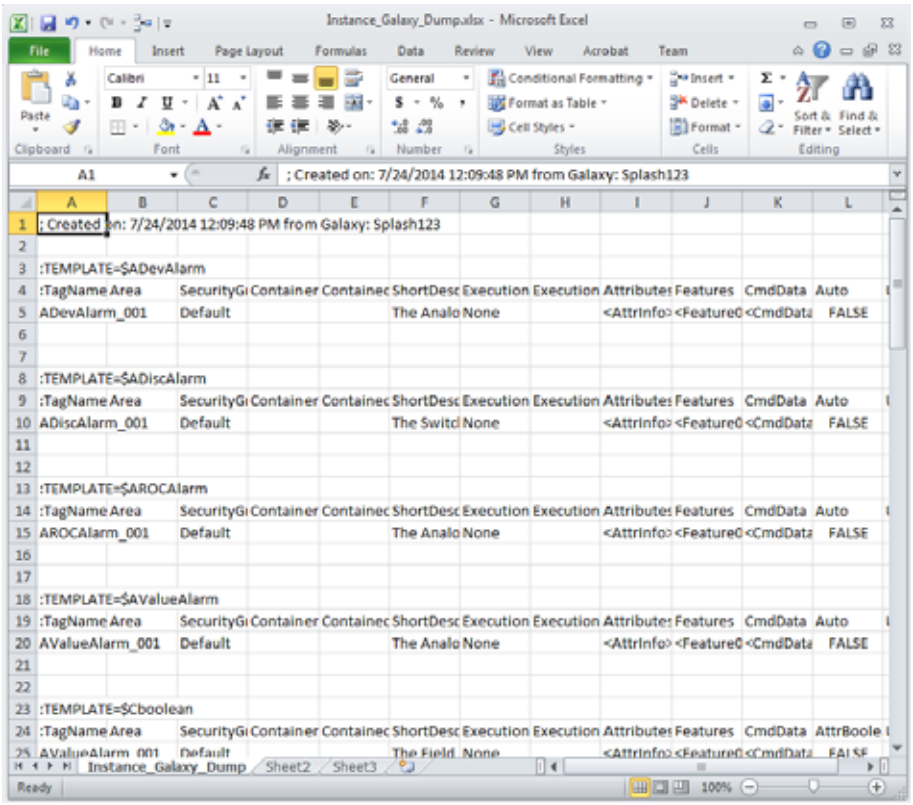
You can open the .csv file created by the Galaxy dump process in a text editor such as Notepad or in Microsoft Excel 2000 or later. When you have finished editing the file, save it as plain text .csv (comma delimited) file to import back into the Galaxy.

About the Galaxy Dump File Structure

The Galaxy .csv file has a specific structure. This section describes that structure.

Objects are organized in the `.csv` file based on the template each is derived from. A header row per template indicates the instance's columns' reference.

Other information is organized in columns. This makes it easy for you to read the information and carefully make any changes you need. You can easily import the `.csv` file into a text editor or into Microsoft Excel.



Add comments by adding a line with a semi-colon as the first character in the comment.

Host Attributes

Galaxy dump files contain a column for the Host attribute of the objects being dumped. In the case of Platform objects, Host is always the name of the Galaxy from which the object is being dumped.

This data is ignored in subsequent Galaxy Load operations because the Host for Platform objects is automatically the name of the Galaxy into which it is being loaded, regardless of the name of the Galaxy from which it was dumped.

Using a text editor, you can delete the Host attribute column, like any other data in the Galaxy dump file. This has no effect on Platform objects in subsequent Galaxy Load operations because they take the Galaxy name as their Host.

About Quotation Marks and Carriage Returns

Carriage returns in scripts associated with dumped objects are replaced with `\n` in the `.csv` file. If you edit the dump file, **do not** delete the `\n` characters. If you edit scripts in the dump file, use `\n` for a carriage return. This character set is interpreted as a carriage return when the dump file is used in a Galaxy Load operation.

When editing a script in a dump file, use `\\n` if you want to include the backslash character (`\`) followed by the letter n in a script. This character set is not converted to a carriage return in a Galaxy Load function.

Be careful when adding or editing quotation marks in the `.csv` file. Type all single quotation marks as two single quotation marks and surround the entire string with opening and closing quotation marks.

Make sure the string contains an even number of quotation marks. When the object is loaded in a Galaxy Load operation, the extra quotation marks are stripped from the string.

For example, if you want to enter `3"Pipe` as a Short Description, add a second quotation mark (`3""Pipe`) and then surrounding quotation marks (`"3""Pipe"`).

Time Formats in Excel

If you edit a Galaxy dump file in Microsoft Excel, be careful typing time entries. Excel can change the time format and the resulting entries do not work when you reload the changed Galaxy.

Galaxy Load accepts two formats:

- `DAYS HH:MM:SS:SSSSSS` - the number of `DAYS` is followed by a space.
- `HH:MM:SS:SSSSSS` - Excel automatically changes the entry to an incompatible format.

When you type time entries in Excel, use the following format: DAYS HH:MM:SS:SSSSSS.

For example:

```
0000 01:02:12.123:1
04:03:06.12:120
22:66:88:123456
```

Enabling I/O Auto Assignment in the Galaxy Dump File

You can use Galaxy dump to convert a Galaxy that contains traditional I/O hard coding to use I/O auto assignment instead. If you are planning to expand an existing Galaxy by adding many objects, or if you are upgrading PLCs, converting hard coded I/O references to auto assignment can save you time. To convert the references, edit the Galaxy dump file as described below. This bulk edit is faster than changing each object individually in the IDE object editor.

The Galaxy dump file displays a flag, “---Auto---”, for input and output attributes that use I/O auto assignment. This flag indicates that I/O auto assignment is enabled for the attribute. The actual I/O auto assignment to a DI Object and scan group is not preserved in the dump file. If you want to preserve existing I/O auto assignment settings, use the export objects function instead of Galaxy dump. See “Exporting Objects” on page 110 for additional information.

To convert hard coded I/O references, edit the Galaxy dump file to replace the hard coded reference with “---Auto---” for each <attribute>.InputSource and/or <attribute>.Output.Dest. Import the edited objects back into the Galaxy, and then assign the objects in the **IO Devices** view of the IDE to a DI Object and scan group.

For more information about I/O auto assignment, see “Using I/O Auto Assignment” on page 156.

To convert hard coded I/O references to I/O auto assignment

1. Use Galaxy dump to export the objects with hard coded I/O references that you want to convert.
2. Open the exported .CSV file in a text editor or Excel.
3. In the data row of the **Attribute.InputSource** column, delete the hard coded I/O assignment. Enter "---Auto---" in the data row.
4. Repeat for each object that you want to enable for I/O auto assignment.
5. In the data row of the **Attribute.OutputDest** column, delete the hard coded I/O assignment. Enter "---Auto---" in the data row.

6 Repeat for each object that you want to enable for I/O auto assignment.

7 Save your changes.

To complete the process of activating I/O auto assignment:

1 Use Galaxy load to import the objects in the edited Galaxy dump file. See “Importing a Galaxy Load File” on page 412 for additional information.

Once imported, the objects will appear in the **IO Devices** view, under the **Unassigned IO Device** folder.

2 Drag and drop objects in the **IO Devices** view to link the objects to the applicable DI Object and scan group. See “Using I/O Auto Assignment” on page 156 for additional information.

Importing a Galaxy Load File

After you are done editing a .csv file, you can import it back into your Galaxy.

A load file contains only instances. Templates cannot be dumped or loaded. See for more information about the contents of the `.csv` file.

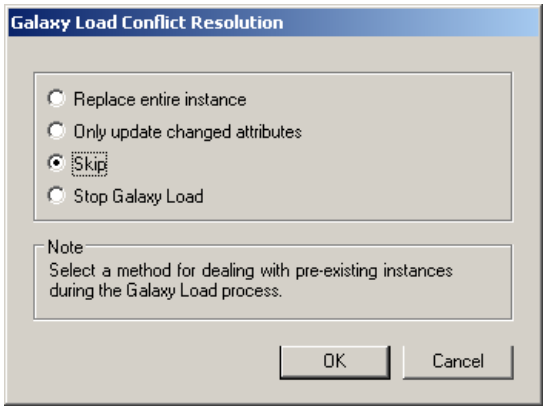
Note: A comment line in a `.csv` file created in Microsoft Excel can create an unintended default-value object. To avoid this, open the `.csv` file in Notepad to make sure the comment line does not contain quotation marks.

To import a .csv file

1 On the **Galaxy** menu, click **Galaxy Load**. The **Galaxy Load** dialog box appears.

2 Browse to find the .csv file that contains the objects and configuration data you want to import. Select the file and click **Open**.

3 The **Galaxy Load Conflict Resolution** dialog box appears. Use it to resolve conflicts that can occur if objects you want to load already exist in the Galaxy.



4 Select one of the following:

- **Replace entire instance** if an instance of an object with the same name already exists and you want to replace it entirely with the object in the import file.
- **Only update changed attributes** if an instance with the same name already exists and you want to replace only the attributes of the object where the values are different.
- **Skip** if an instance with the same name already exists and you want to keep the version already in the Galaxy.
- **Stop Galaxy Load** if an instance with the same name already exists and you want to cancel the Galaxy Load.

5 Click **OK**. A progress box appears showing the Galaxy load process.

When the load is done, all objects changed or created during the Galaxy Load process are checked in.

Synchronizing Time Across a Galaxy

Some of the Application Server functions like scripting, alarming, and historizing depend on all member computers of a Galaxy synchronized to the same time. A time master is a Network-Time-Protocol Server that provides a time that other nodes on your network can synchronize with.

The time master can be a non-ArchestrA node or one within the Galaxy. The ArchestrA nodes in the Galaxy periodically synchronize their clocks to the time master.

Using Time Synchronization in Windows Domains

The system administrator of the Windows 2000 domain may have time synchronization configured already. If this is the case, configuring a Galaxy Time Master is unnecessary and can conflict with the existing time synchronization.

Time synchronization in your Galaxy is critical if one or more nodes run the Windows XP operating system. Windows XP supports a network authentication protocol that requires time synchronization between two nodes to enable communication between them. This protocol fails the communication between the nodes if the time on the two nodes differs by a predetermined amount (for example, five minutes).

If a node in your Galaxy runs Windows XP operating system and its system time is beyond the allowed variance, ArchestrA operations, such as deployment, may fail.

Synchronization Schedule

The designated node clock serves as the master clock for all timestamping functions. Time synchronization is based on Microsoft's Windows Time Service. ArchestrA does not implement its own time synchronization algorithm.

The default synchronization period is one time every 45 minutes until three successful synchronizations occur. After three successful synchronizations, synchronization runs one time every eight hours.

All WinPlatforms begin synchronizing the time on their node when they are deployed.

You can specify a time master node in another time zone. The time on each Application Server node is set to the time specified on the node in the other time zone.

Required Software

If a time master or a time client node runs either Windows 2003 Server or Windows XP software, you must install a Microsoft hotfix on that computer before you can synchronize to a time master. Install the appropriate hotfix according to the following table.

Operating System	Hotfix
Windows XP	`WindowsXP-KB823456-x86-ENU.exe`
Windows 2003 Server	`WindowsServer2003-KB823456-x86-ENU.exe`

Contact Microsoft through its Product Support Services to get these hotfixes.

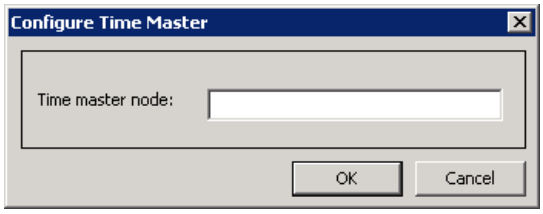
If your time master computer is a non-Galaxy node, regardless of operating system, you must also change certain Registry keys. Contact Wonderware Technical Support to obtain the files that can set those Registry keys.

Important: You must complete the software and Registry updates before configuring a node as a time master.

Before you start, you need the fully qualified node name in the format of: `nodename.domain.organization`.

To configure a time master node

1. Apply all hotfixes and make the Registry changes as explained previously.
2. On the **Galaxy** menu, point to **Configure** and then click **Time Master**. The **Configure Time Master** dialog box appears.



3. Type the node name in the **Time Master Node** box.
4. Click **OK**.

Hosting Multiple Galaxies in One Galaxy Repository

You can create and configure multiple Galaxies in a single Galaxy Repository on the same computer. You can configure one Galaxy from an IDE and then change Galaxies and configure the second one using the same IDE.

Note: If you try to deploy objects from the second Galaxy to a computer that hosts deployed objects from the first Galaxy, the deploy fails.

Managing Licensing Issues

Your license controls access to the Galaxy Repository. If a license-related message appears when you open the IDE, there is a problem with your license. The message appears as a result of one of the following conditions:

- A license is not installed.
- Your license expired.
- You exceeded the licensed I/O count or number of licensed WinPlatforms.
- The number of I/O or WinPlatforms specified in your IDE development license is more than the I/O or WinPlatforms specified in your Galaxy license.

Note: If a license expires while you are using the IDE, you cannot connect to the Galaxy the next time you open the IDE.

Viewing License and End-User License Agreement Information

Use the License Utility to view information about your license and End-User License Agreement. Until any problems are resolved, you cannot:

- Open the IDE
- Connect to existing Galaxies
- Create new Galaxies

After you update your license, you can connect to your Galaxy and open the IDE with no further problems. For more information about updating your license, see “Updating a License” on page 420.

To view ArchestrA IDE version information

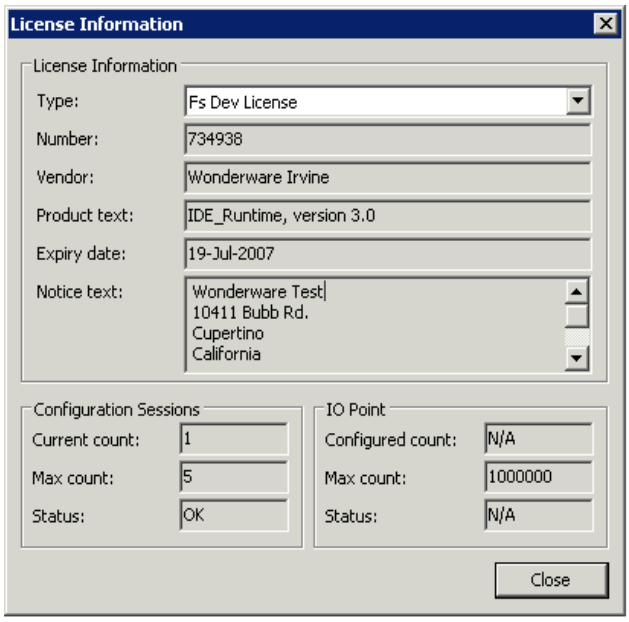
1 On the **ArchestrA IDE** menu, click **Help**.

2 Click **About ArchestrA IDE**. The **About ArchestrA IDE** dialog box appears, showing the copyright notice, version, and build date for your installed ArchestrA IDE application.

3 Click **OK**.

To view your license information

1 On the **About ArchestrA IDE** screen, click the **View License** icon. The **License Information** dialog box appears.



2 In the **Type** list, select the license you want to view.

3 In the **License Information** area, check your **Expiry date**.

4 Select **Fs Dev License** to view:

- In the **Configuration Sessions** area:

  Current count — The number of IDE sessions currently open.

  Max count — Maximum number of configured sessions allowed by your license.

  Status — Relationship between the configured and maximum I/O point count values.

  You see one of three states:

  - OK: Everything is fine.
  - Exceeded: Configured I/O points count exceed the maximum allowed by your license.
  - DEV: Your license has no I/O and Platform Count feature line.

- In the **IO Point** area

  Configured count — Number of configured I/O points in your Galaxy.

  Max count — Maximum number of configured I/O points allowed by your license. If your license has no I/O and Platform Count feature line, N/A appears.

  Status — Relationship between the configured and maximum I/O point count values.

  You see one of three states:

  - OK: Everything is fine.
  - Exceeded: Configured I/O points count exceed the maximum allowed by your license.
  - DEV: Your license has no I/O and Platform Count feature line.

5 Select **App Server License** to view:

- In the **Platform** area:

  Current Count — Number of deployed WinPlatforms in your Galaxy.

Max Count	Maximum number of deployed WinPlatforms allowed by your license. If your license has no I/O and Platform Count feature line, N/A appears.
Status	Relationship between the current and maximum WinPlatform count values. You see one of three states: • OK: Everything is fine. • Exceeded: Deployed WinPlatform count exceeds the maximum allowed by your license. • DEV: Your license has no I/O and Platform Count feature line.

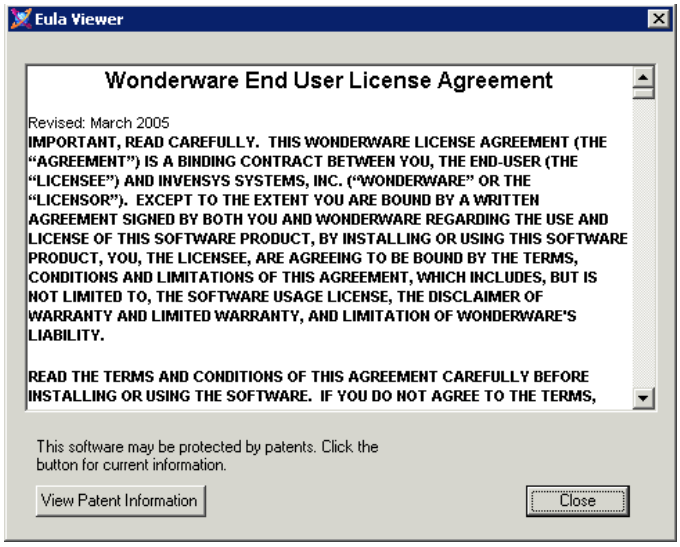
- In the **IO Point** area:

Configured Count	Number of configured I/O points in your Galaxy.
Max Count	Maximum number of configured I/O points allowed by your license. If your license has no I/O and Platform Count feature line, N/A appears.
Status	Relationship between the configured and maximum I/O point count values. You see one of three states: • OK: Everything is fine. • Exceeded: Configured I/O points count exceed the maximum allowed by your license. • DEV: Your license has no I/O and Platform Count feature line.

6 When you are done, click **Close**.

To view the end-user license agreement

1 On the **About ArchestrA IDE** dialog box, click **View License Agreement**. The **Eula Viewer** dialog box appears. Scroll down to read the terms and conditions of the agreement.



2 Click **View Patent Information** to read the patent information. You must have an Internet connection to view the patent information.

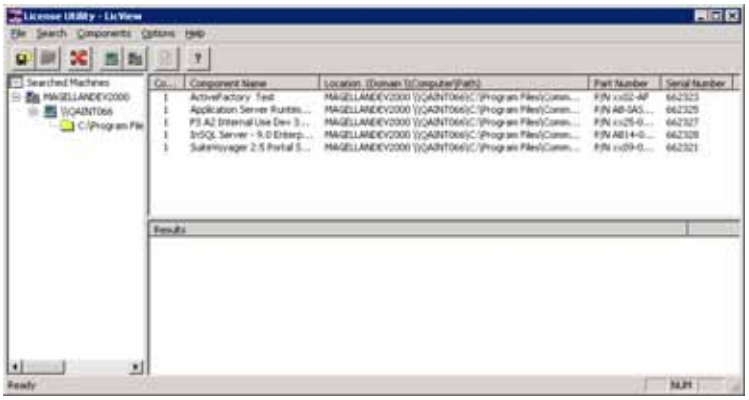
3 When you are done, click **Close**.

Updating a License

After you get your new license from Wonderware, you must install it to update your Galaxy Repository license. For updating your license, do the following steps:

To update your license

1 Start the **License Utility**. On the Windows **Start** menu, point to **Programs, Wonderware**, then to **Common**, and then click **License**. The **License Utility** appears.



2 On the **File** menu, click **Install License File**. Browse to the folder where you saved the `.lic` file.

3 Select the `.lic` file and click **Open**. The **Destination Computer for Installation** dialog box opens.

4 Do the following:

- In the **Domain** box, type the name of the domain in which the computer resides.
- In the **Computer** box, type the name of the computer on which you want to install the license file.
- Click **OK**.

If a license file exists on that domain, the **Installing a License File** dialog box appears.

- To overwrite the license file, click **Overwrite**.
- To add the new license file information to the existing license, click **Append**.
- When you are done updating the license file, you can see the results of the installation in the **Results** pane.

Disk Space Requirements

After Application Server is installed, certain operations require at least 100 MB of available disk space. These operations include

- Creating a Galaxy
- Deploying objects
- Importing and exporting objects
- Loading and dumping a Galaxy
- Restoring and backing up a Galaxy.

This minimum requirement applies to the Galaxy node as well as any remote IDE nodes.

If your computer has less than 100 MB of available hard disk space, running any of these operations can result in erratic behavior.

Managing Communication between Galaxy Nodes

You can use Application Server in a distributed environment. All computers with ArchestrA-enabled software installed must be able to communicate with each other.

Two items must be considered to ensure communication between nodes:

- ArchestrA user accounts
- Multiple network interface cards in computers

All Platforms communicate with several attributes on the GR Platform to detect configuration changes. This communication sends NMX heartbeats from each Platform to the GR Platform. For example, the attributes include “time of last deploy,” and “time of last configuration change.”

In addition to NMX heartbeats, all Bootstraps on each Platform in the Galaxy send each other heartbeats. These heartbeats are for Platform status information, occur every two seconds, and are configurable.

Mapping Network Drives with User Account Control Enabled

Mapping network drives with Windows User Account Control (UAC) enabled can create issues with subsequent access to those drives. Windows considers mapping a network drive and later access to that drive, to import objects from a network location, for example, as separate logon contexts, and issues separate credentials. Without correct credentials, the mapped network drives can become inaccessible.

You can ensure that no issues occur with mapped drives by using one of two methods to map the drives:

1. Map drives from the command prompt as an administrator.

   From Windows **Start**/**All Programs**, click **Accessories**, then right-click **Command Prompt** and select **Run as administrator** on the context menu.

   At the command prompt, type the "net use" command, using the following format:

   ```
   net use \\<computername>\<sharename> /user:<username>
   ```

   Example:

   ```
   net use z: \\MainComputer\ObjectFiles /user:DRoberts
   ```

   where the drive being mapped is z:, the computer name is MainComputer, the share name is ObjectFiles, and the user name is DRoberts.

2. Map drives from the IDE using elevated privileges.

   For example, you can map drives from the **Import Object(s)** dialog.

About ArchestrA User Accounts

Communication occurs with an ArchestrA-specific user account set up during the initial installation of each ArchestrA component on each computer, including the IDE.

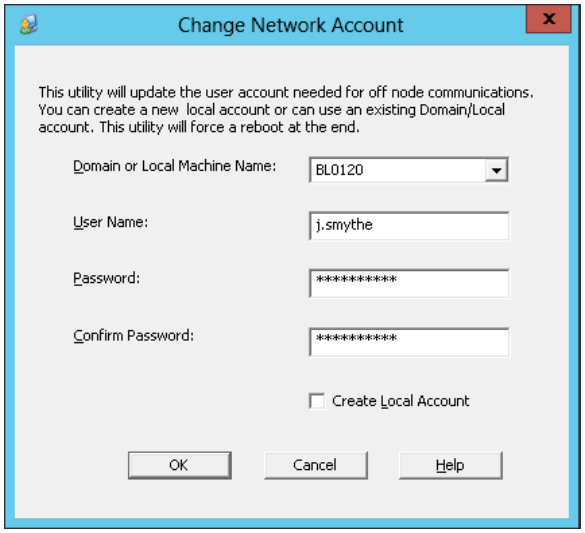
WARNING! The user account that is used for ArchestrA communication is a standard Windows operating system account located on the local computer or on a domain. Do not delete this account with operating system account management tools. If you do, the IDE stops functioning.

If you delete the ArchestrA user account on an IDE node, you must recreate it with the Change Network Account utility. You must have Administrator privileges on the computer to make changes with the Change Network Account utility.

Before you start, find out the user name and password that is created on all computers with ArchestrA-enabled software installed.

To recreate an ArchestrA user account

1 Start the **Change Network Account** program by clicking **Start**, point to **Programs**, and click **Wonderware**. Click **Common** and then click **Change Network Account**.



2 Do one of the following:

- In a single-node ArchestrA system, create any account.
- In a multi-node ArchestrA system, create the same user account with the same user name and password on all other ArchestrA computers.

3 Click **OK**. After you recreate the user account, the Microsoft Windows security component on your computer can take several minutes to update this information on the ArchestrA Galaxy node. Until that happens, your IDE might not function properly. Restarting the Galaxy node applies this update immediately.

Using Multiple Network Interface Cards

You can use nodes with more than one network interface card (NIC). These nodes must be configured properly so they can communicate with other ArchestrA nodes.

If a node contains multiple NICs for redundancy reasons, see “Working with Redundancy” on page 429 for more information.

If a multiple NIC computer in your Galaxy uses only one NIC, you need to disable all cards except the supervisory network.

Defining the Order of the NIC

For any multiple NIC ArchestrA node to communicate with all other Galaxy nodes, you must define the correct order of network connections in the network services of the computer.

Before you start configuring multiple network cards, you need the IP address for the second and subsequent nodes.

To define the correct order

1. Open **Network Connections** through the **Control Panel** (access **Network Connections** through **Control Panel > Network and Sharing Center**, then select **Change adapter settings**).
2. Rename each network card with a clearly identifiable function, for example, `Supervisory Net` and `PLC net`.
3. Select **Advanced** from the menu bar (press the **Alt** key if the menu bar is not visible), and click **Advanced Settings**.
4. In the **Advanced Settings** dialog box, click the up and down arrows to define the correct order of Connections.
   - The first connection in the list must be the supervisory network card.
   - If a computer contains more than two network cards, for example, a supervisory connection, a PLC connection, and a Redundancy Message Channel (RMC) connection for ArchestrA redundancy, the supervisory network must be listed first. The others can be listed in any other position.
5. Click **OK** to accept the changes.

Configuring the IP Address and DNS Settings

After you specify the order of the network cards, you are ready to configure several other parameters to ensure successful node-to-node communications in the ArchestrA environment.

You must configure the IP address and DNS settings as follows for each network card to function properly.

To configure the IP address and DNS settings

1. In the **Network Connections** dialog box, right-click the network connection and click **Properties** in the shortcut menu. The **Properties** dialog box for this connection appears.
2. In the list of components used by this connection, select **Internet Protocol (TCP/IP)** and click **Properties**. The **Internet Protocol (TCP/IP) Properties** dialog box appears.
3. For the supervisory network, select **Obtain an IP address automatically**.

For the other network connections, select **Use the following IP address**. Type the correct IP address for the secondary and further cards. See your network administrator for the proper settings for the remainder of the parameters in this group.

4 Click **Advanced**. The **Advanced TCP/IP Settings** dialog box appears. Click the **DNS** tab.

5 For the supervisory network, select **Register this connection's addresses in DNS**.

For the other network connections, clear this check box.

6 Click **OK**.

Configuring Multiple NICs

Network profiles are identified and assigned during computer startup and each time a connection changes. The three profiles currently in use in the supported Windows operating systems are:

- **Domain profile:** Active only when the computer can authenticate with a domain controller on all active interfaces (for example, LAN, wireless, and VPN). The domain profile may be more or less restrictive than the other two profiles depending on network security policies.
- **Private profile:** Active whenever the network type for all active network connections on the computer are identified as private networks. The private profile typically is used in a more trusted environment and is less restrictive than the public profile to allow for network discovery.
- **Public profile:** Active in all other circumstances. The public profile typically is more restrictive than the private profile because the computer often is connected to the Internet in an insecure location. Network discovery and remote access are disabled rather than explicitly blocking specific traffic.

All programs allowed to communicate through the firewall are blocked if one or more network interface cards (NICs) are configured as "Public Network." When the IP address of a network card changes, the NIC is automatically updated to "Public Network." If even one NIC in a computer with multiple NICs is configured as "Public Network", the firewall exceptions are disabled.

This is particularly important for computers configured to run as a redundant pair. If the TCP/IP properties for a network card are set to obtain an IP address automatically, there is a possibility that the network address will change when the computer restarts. The computer may suddenly change from having a "Private Network" to having a "Public Network," and potentially block the programs that were previously allowed to communicate through the firewall.

Note: An exception to this would occur if the NICs are configured and set to the Public Profile before Application Server is installed. The Application Server installation would allow the necessary programs to communicate through the firewall in the current (Public) profile.

The following describes the key NIC settings for a redundant pair. You configure these settings for both the primary and backup computers. These settings apply if more than two NICs are installed for redundancy (RMC) or other networking purposes, such as networking PLCs.

For detailed instructions on how to configure multiple network interface card binding order settings, see “Defining the Order of the NIC” on page 425 and “Configuring the IP Address and DNS Settings” on page 425.

To configure multiple NICs

1. Under **Internet Protocol Version 4 (TCP/IPv4) Properties**, select the **Use the following IP address option** and provide an IP address, subnet mask, and default gateway.

   a. For the primary computer, the default gateway should be the IP address that you configure for the backup computer. For example:

      IP address: 100.100.100.94

      Subnet mask: 255.255.255.0

      Default gateway: 100.100.100.95

   b. For the backup computer the default gateway should be the IP address that you configure for the primary computer. For example:

      IP address: 100.100.100.95

      Subnet mask: 255.255.255.0

      Default gateway: 100.100.100.94

   c. A NIC acting as Redundancy Message Control (RMC) does not require a default gateway. The IP address and Subnet mask can be configured as described for the primary and backup computers. Windows will identify this NIC and assign it a private profile. For example:

      IP address 100.100.100.96

      Subnet mask: 255.255.255.0

2. Under **Advanced Settings**, check that the **Register this connection's address in DNS option** is not selected.

3. Make sure that the primary network adapter is first in the binding order. This places it above the RMC network in the binding order.

4 Make sure that ALL network cards are configured as “Private Network.”

5 Configure the RMC NIC to have a persistent (across reboots) Private profile.

   a Open a command window and enter the command `secpol.msc`. The **Security Policy** window appears.

   b Select **Network List Manager Policies**.

   c Select **Unidentified Network**.

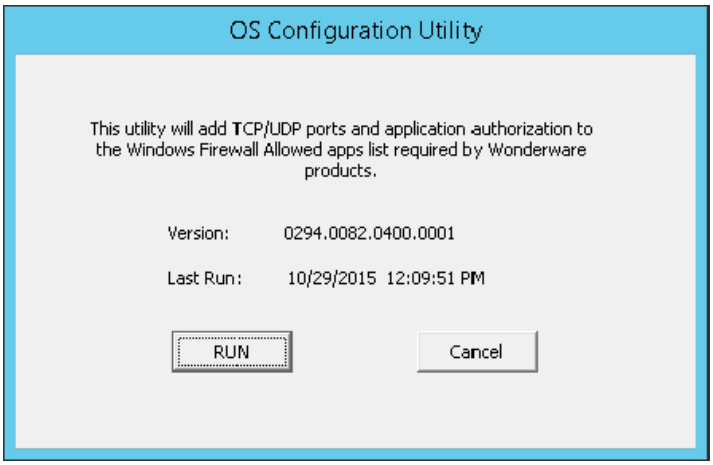
   d Right click, then select **Properties**, and then change the **Location** type from **Not configured** to **Private**.

   e Close the command window.

It may be necessary to restart the computer for the changes in steps 4 and 5 to take effect.

If the two computers are set up properly, you can install the Wonderware software and the OSConfigurationUtility sets up the firewall exceptions appropriately.

If you install the Wonderware software before configuring the NICs properly, configure the NICs as described in this section and then run the OSConfigurationUtility again. To do this, run the OSConfigurationUtility.exe, located in the “C:\Program Files (x86)\Common Files\ArchestrA” folder.



To verify that the firewall exceptions are set, open **Control Panel** and then open the **Windows Firewall** application. Verify that the executables for the Wonderware products you have installed are allowed to communicate through the firewall.

Chapter 15

Working with Redundancy

You can set up and run Application Server in a redundant environment. In Application Server, two types of redundancy ensure continued run-time operation. You can configure redundant:

- AppEngine object pairings for computer or software failures.
- Data acquisition communications to one or more PLCs.

A failover is the condition during which run-time operations are moved from one critical component to another. Failover can occur due to failure conditions or it can be forced manually, called a forced failover.

For more information about redundancy, see the Schneider Electric Software Global Customer Support website.

About Redundancy

Application Server provides redundancy in two critical functions:

- AppEngine

  You must configure both an AppEngine and two WinPlatforms.

- Data acquisition

  You must configure two DIObjects (data sources) and a RedundantDIObject.

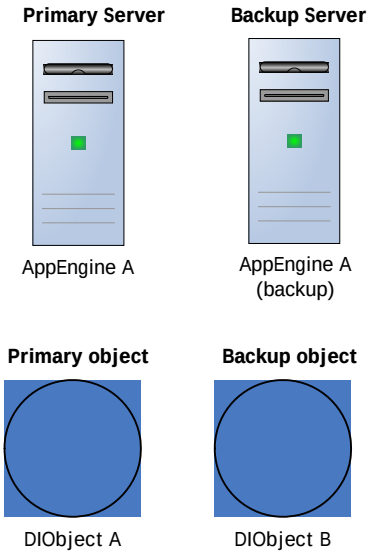
Configuring AppEngine Redundancy

The Primary/Backup AppEngines form a redundant pair. The ArchestrA infrastructure will generate the backup AppEngine automatically when redundancy is enabled for an AppEngine. Hierarchy of ApplicationObjects can only be assigned to the primary AppEngine. The primary and backup engines need to be assigned to redundancy enabled platforms, and they can be deployed separately.

For data acquisition, the Primary/Backup DIObjects (the data sources) must be separately created, configured, and deployed. Also, you must create, configure, and deploy a RedundantDIObject to control failovers between the two data source objects.

In a redundant system, install and configure the primary OPC server on the backup engine node.

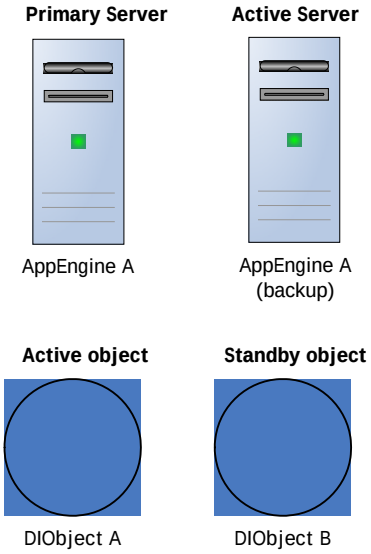
When you configure redundancy, you configure the Primary object and the Backup object.



- **Primary object**: The main or central object that provides the functionality during run time. For AppEngines, this is the object you enable for redundancy. For data acquisition, this is the DIObject you intend to use first as your data source in run time.
- **Backup object**: The object that provides the functionality of the Primary object when the Primary object fails. For AppEngines, this is the object created by the ArchestrA infrastructure when the Primary object is enabled for redundancy. For data acquisition, this is the DIObject you do not intend to use first as your data source in the run-time.

Redundancy during Run Time

When you deploy these objects to a run-time environment, the configuration objects become the Active object and the Standby object.



- **Active object**: The object that is currently executing functions. For AppEngines, it is the object that is hosting and executing ApplicationObjects. For data acquisition, it is the object that is providing field device data through the RedundantDIObject.
- **Standby object**: The object that is waiting for a failure in the Active object or for a force-failover. For AppEngines, it is the object that monitors the status of the Active AppEngine. For data acquisition, it is the object that is not providing field device data through the RedundantDIObject.

In the AppEngine redundancy environment, the Active and Standby objects monitor each other's statuses and switch during a failover situation.

During a failover, when an Active engine fails, it restarts and the Standby engine becomes active. The Active engine then turns to the standby mode. In the data acquisition environment, the RedundantDIObject monitors the status of the two DIObject data sources, and handles the switching from Active to Standby objects.

The relationship between the configuration time (Primary/Backup) and run-time (Active/Standby) object pairs is not static. In the run time, either the Primary or Backup object can be the Active object at any particular time. Whenever one becomes the Active object, the other automatically becomes the Standby.

Engine Restart After Failure

An AppEngine will restart automatically after an engine failure such as a crash or hang. If the engine is redundant, it will synchronize with its redundant partner, and running applications will resume.

Engine restart on failure is automatic and cannot be disabled.

CPU Load Balancing

CPU load balancing for redundant AppEngines can be used to maintain system stability by limiting high CPU usage during AppEngine startup after failover. When CPU usage by a core is close to 100%, other processes can be starved and startup times can extend as a result. When CPU load balancing is enabled, objects are loaded in discrete chunks, thus improving CPU utilization.

Application Server includes a registry key that you can use to configure CPU load balancing for redundant engines. CPU load balancing is disabled by default, and the configuration of each node is independent. That is, you can enable CPU load balancing on some nodes and not on others, and have different settings for each node.

Note: CPU load balancing will only work on run-time nodes with redundant engines.

- Registry Subtree – 64-bit:
  `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ArchestrA\Framework`
- Registry Subtree – 32-bit:
  `HKEY_LOCAL_MACHINE\SOFTWARE\ArchestrA\Framework`
- Registry Key:
  `EngineFailoverStartup`

Subkey entries used for enabling and tuning CPU load balancing are:

Subkey Name	Default Value	Description
EnableCPUBalancing	0	Turns CPU load balancing on or off (0 = OFF, 1 = ON)
LoadObjectsNumberPer Group	50	Specifies the number of objects loaded at a time during startup. If set to 0, this function is disabled.

Subkey Name	Default Value	Description
LoadObjectsTimeToSleep	200	Specifies in milliseconds, the time between loading groups of objects (end of load to start of next load). If LoadObjectsNumberPerGroup is 0, sleep is disabled.
FirstScanCycleNumberPer Group	25	Specifies the number of objects to be initialized during the first scan cycle after failover. If set to 0, this function is disabled.
FirstScanCycleTimeToSleep	100	Specifies in milliseconds, the time between initializing groups of objects (end of first cycle to start of second). If FirstScanCycleNumberPerGroup is 0, sleep is disabled.

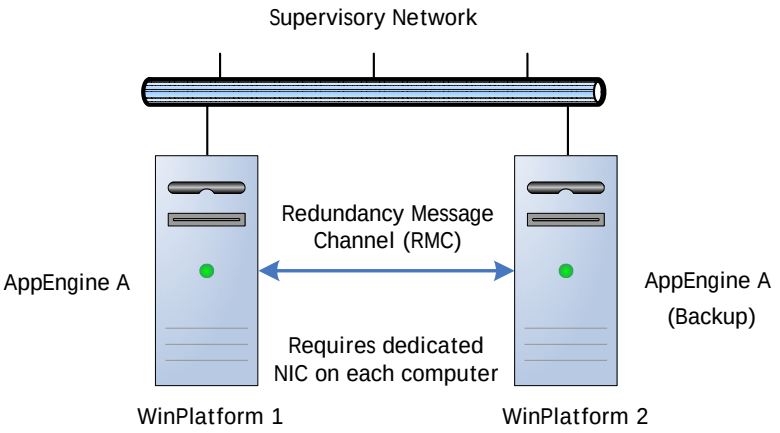
Working with AppEngine Redundancy

You enable AppEngine redundancy in the Primary AppEngine. You must also configure two WinPlatforms for redundancy, one to host the Primary AppEngine and one to host the Backup AppEngine.

Important: WinPlatforms hosting redundancy-enabled AppEngines must be deployed to computers running the same operating system.

The configuration of both WinPlatforms should be the same. At a minimum, the store-forward directory configurations must be common to both WinPlatforms:

During configuration, you can assign Primary and Backup AppEngines to the same WinPlatform. To deploy, you must assign the Primary and Backup AppEngines to different WinPlatforms.



Each production computer hosting a redundancy-enabled AppEngine must have a minimum of two network cards. One NIC is for the supervisory network and PLC network, if the computer has only two network cards. The other must be for a dedicated Ethernet connection between computers for the redundancy message channel (RMC).

For information about distributed networks, see “Using Multiple Network Interface Cards” on page 424.

The RMC handles redundancy monitoring, message handling and data synchronization between redundant pairs.

Configuring the Redundancy Message Channel

For the redundant pair of AppEngines to successfully communicate with each other, you must define the correct order of network connections in the network services of each computer.

We recommend that you name each card with a clearly identifiable function (for example, “Supervisory Net” and “Redundant Message Channel”).

You must configure at least the following parameters:

- Redundancy Message Channel IP Address
- Redundancy Message Channel Port
- Primary Message Channel

To configure network cards

1. Open the **Network Connections** dialog box in Windows. The specifics about opening this dialog box varies, depending on the version of Windows you are working on.
2. Click **Advanced Settings**. If the computer name is used in the platform's “node name” box, the first connection in the list must be the supervisory network card. Use the up and down arrows to define the correct order.
3. **Connections**.

   If a computer contains more than two network cards, the supervisory net must be listed first and the RMC connection can be listed in any other position. For example, your computer might have a supervisory connection, a field device connection, and a RMC connection.
4. Configure the DNS settings for the supervisory network card to function properly.
   - On the **DNS** page of the **Advanced TCP/IP Settings** dialog box, select the **Register this connection’s addresses in DNS** check box.
   - For the RMC network card to function properly, clear the **Register this connection's addresses in DNS** check box. For more details on these settings, see “Using Multiple Network Interface Cards” on page 424.
5. In Application Server, open the WinPlatform in the Object Editor for the computer you just configured.
6. Change the **Redundancy Message Channel IP Address** to the IP address of the RMC connection. See the WinPlatform Help for more information about these parameters.
7. Save and close.

Configuring Redundancy

You configure redundancy in AppEngine/WinPlatforms objects using their Object Editors.

The redundancy-related parameters in the AppEngine Object Editor are located on the **Redundancy** and **General** tabs.

An AppEngine that is part of a redundancy pair has a deployment status indicating its own status and that of its partner object. These statuses are visually indicated in the **Application** views.

You can set the status of a redundancy pair to one of the following states:

Pair Deployed	Both Primary and Backup AppEngines are deployed.
Pair Undeployed	Both Primary and Backup AppEngines are undeployed.
Partial Deployed	Either the Primary or Backup AppEngine is deployed and its partner is not deployed. If an AppEngine has a Partial Deployed status, its partner has a Partial Undeployed status.
Partial Undeployed	Either the Primary or Backup AppEngine is undeployed and its partner is deployed. If an AppEngine has a Partial Undeployed status, its partner has a Partial Deployed status.

To create AppEngine redundancy

1. In the Primary AppEngine, select the **Enable Redundancy** check box on the **Redundancy** tab.
2. Configure the remaining redundancy parameters as needed. See the help file for the AppEngine for specific information about these parameters.
3. On the **General** tab, set the **Engine Failure Timeout** option to 2000 milliseconds. You may need to tune this parameter between 2000 ms and the default 10000 ms, depending on the requirements of your application.

   When the Active engine fails during a failover, it restarts automatically and becomes the Standby engine.

Note: The actual engine failure time-out is three times the value of this parameter. If you set the parameter to 2000 ms (2 seconds), a failover occurs if the AppEngine fails to communicate with the computer's Bootstrap for 6 seconds. A setting of 10000 ms (10 seconds) can be too long a wait period (30 seconds) for a well-functioning redundancy operation.

After you save the configuration of the object and check it into the Galaxy, the icon for the object changes. A Backup AppEngine is created with the same configuration as the Primary object.

Icon	Object
	Primary AppEngine
	Backup AppEngine

Configuring Redundancy in Templates

You can create a redundancy-enabled AppEngine template. When you create an instance of the template, both the Primary and Backup instances are created.

The Backup AppEngine is hosted by the Unassigned Host or the default WinPlatform if you specified one in the **Configure User Information** dialog box.

If you change the configuration and check in the Primary AppEngine, the Backup AppEngine:

- Is checked out in the background
- Updates the configuration
- Is checked in without notification to connected clients

Deleting Redundant AppEngines

You can disable redundancy in an AppEngine after the ArchestrA infrastructure creates the Backup object. If you disable redundancy and check in the Primary AppEngine, the Backup is deleted from the Galaxy. The exception is when the Backup is already deployed. In that case the newly configured Primary object cannot be checked in.

To delete a Backup AppEngine from the Galaxy, you must undeploy it first.

There is no redundancy-related configuration required on ApplicationObjects hosted by an AppEngine configured for redundancy. When a system failure occurs, the ApplicationObjects and their attribute values are duplicated on the Standby AppEngine, which becomes the Active AppEngine. For more about failover functionality, see “During Deployment” on page 440.

If the Platform hosting the redundant AppEngine is shutdown in SMC, the AppEngine cannot be flagged as “On failure mark as undeployed” when you undeploy the AppEngine. You see an engine communication error in the **Deploy** dialog box.

Deploying AppEngine Objects

Primary and Backup AppEngines can be deployed together or individually.

- When they are deployed together, regardless of which object is actually selected for deployment, the Primary always becomes the Active and the Backup becomes the Standby.
- When they are deployed individually, the first one deployed becomes the Active.

Hosted ApplicationObjects are always deployed to the Active AppEngine. When deploying the first of a redundant pair of AppEngines, you can cascade deploy all objects it hosts. This operation can be paired with deploying both the Primary and Backup AppEngines at the same time.

Important: If you deploy the Backup AppEngine first and then deploy hosted objects to that AppEngine, make sure the network communication to both target computers is good before deploying the Primary AppEngine. Otherwise, errors occur.

In the run-time environment, either the Primary or Backup AppEngine can become the Active or Standby depending upon failure conditions on either computer.

Configuration Requirements

Before deploying the Primary and Backup AppEngines, all configuration requirements must be met.

- Each AppEngine must be assigned to a separate WinPlatform.
- A valid redundancy message channel (RMC) must be configured for each WinPlatform.
- To deploy the Primary and Backup together, select **Include Redundant Partner** in the **Deploy** dialog box. This option is not available when doing the following operations:
  - Cascade deploy from the Galaxy
  - Multiple object selection deploy
  - Deploying the WinPlatform that hosts the Primary or Backup AppEngine

The following table shows a matrix of allowed operations based on specific conditions.

Condition	Deploy Both Primary and Backup Objects	Cascade Deploy Allowed
Backup AppEngine's host WinPlatform configured for failover and deployed	Yes	Yes
Backup AppEngine in error state	Yes	Yes
Backup AppEngine's host WinPlatform not deployed	No	Yes
Backup AppEngine's host WinPlatform not configured for failover and deployed	Yes	Yes
Deploy from Galaxy node	No	Yes
Deploy from WinPlatform hosting Primary AppEngine	No	Yes
Multiple object selection deploy	No	No
Backup AppEngine's host WinPlatform not configured for failover and not deployed	No	Yes
Deploy from Backup AppEngine	Yes	Yes
Deploy from Primary AppEngine	Yes	Yes

Undeploying AppEngine Objects

Undeploying redundant pairs of AppEngines is just like undeploying any regular object.

You can undeploy the Active and Backup AppEngines separately or as a pair.

Important: Undeploying any objects, including redundant AppEngine pairs, does not uninstall code modules for that object from the hosting computer. Code modules are uninstalled only when the WinPlatform is undeployed.

To undeploy as a pair

1. Select one of the objects in an Application view.
2. On the **Object** menu, click **Undeploy**. The **Undeploy** dialog box appears.
3. Select **Include Redundant Partner** in the **Undeploy** dialog box and click **OK**.

During Deployment

During initial deployment of a redundant pair of AppEngines, files are deployed in the following order:

- Code modules and other files for the Primary AppEngine are deployed
- Those files for its assigned ApplicationObjects
- All of these files are deployed to the Standby AppEngine by the Active engine's WinPlatform using the redundancy message channel (RMC)

Note: If some or all of these files already exist on the Standby AppEngine's WinPlatform, perhaps, assigned to another AppEngine on that platform, only the delta files are deployed to the Standby AppEngine.

Objects at Run Time

Objects are always assigned to the Primary AppEngine in the configuration environment.

During run time, objects are always deployed to the Active AppEngine whether or not it was initially configured as the Primary object.

All files are deployed by the Active AppEngine's WinPlatform to the Backup AppEngine as described in the previous section.

During Run Time

In the run-time environment, the Active and Standby AppEngines first attempt to establish communication across the RMC. This occurs when an AppEngine belonging to a redundant pair first starts. Therefore, if one AppEngine is relocated later to a different WinPlatform, this communication between AppEngines can be reestablished.

During run time, the Active and Standby engines communicate with each other and monitor each other's status.

In the case of a hardware or software failure on the Active computer, the Standby AppEngine becomes the Active one. To move the new Standby AppEngine from its hosting computer, undeploy this AppEngine by selecting the **On failure mark as undeployed** option on the **Undeploy** dialog box. Reassign and redeploy it to a WinPlatform that is configured for redundancy on another computer.

AppEngine Redundancy States

Redundant pairs of AppEngines can have one of the following states at a time:

- **Active**: The state of an AppEngine when it has communication with its partner object, its partner is in Standby-Not Ready, Standby-Sync'ing with Active, or Standby-Ready state. A Standby AppEngine transitions into this state when a failover condition is detected. In this state, an AppEngine schedules and runs deployed objects, sends checkpoint data and sends subscriber list updates to the Standby AppEngine.
- **Active - Standby not Available**: The state of an Active AppEngine when it determines it cannot achieve communications with its partner object. This could mean that checkpoint, subscription and alarm state changes are not successfully transmitted to the Standby object because the partner AppEngine is not deployed, number of heartbeats missed from standby objects exceeds the configured max consecutive number of heartbeats missed, or notification is received that the Standby AppEngine shutdown or is not running. If an AppEngine is in this state, it 1) continues normal execution of hosted objects, 2) cannot be manually switched to Standby state, and 3) while continuing to attempt communicate with the Standby, does not attempt to send data to the Standby object.

- **Determining Failover Status**: The initial state of a redundancy-enabled AppEngine when it is first started. It has not determined yet whether it is the Active or Standby AppEngine. Communication between the two AppEngines is attempted first over the RMC and then over the primary network to make this determination. If communication cannot be made after a certain time-out period, an AppEngine assumes the Active role if it has all of the code modules and checkpoint file data to do so. Continued attempts are made at communicating with its partner.
- **Standby - Missed Heartbeats**: The state of an AppEngine when 1) the heartbeat pings are not received from its Active partner through the RMC, but the number of missed beats haven't reached the maximum consecutive number of heartbeats missed, or 2) the heartbeats from active engine have been missed through the primary channel. When in this state, the Standby object attempts to determine whether or not the Active object failed. If a manual failover is started by using the ForceFailoverCmd attribute, it is processed only if the heartbeats were missed over the primary network and not missed over the RMC.
- **Standby - Not Ready**: The state of an AppEngine when one of several conditions occurs: 1) its lost communications with its partner object or it maintains communications with its partner but missed checkpoint updates or alarm state changes from the Active AppEngine, 2) new objects are deployed to the Active AppEngine and necessary files are not installed on the Standby AppEngine yet, or 3) the Standby AppEngine lost communications over the RMC before it completed synchronizing data. Typically, the AppEngine's partner is in one of the following states: Active-Standby not Available, or Active.
- **Standby - Ready**: The state of an AppEngine when it completed synchronizing code modules and checkpoint data with the Active AppEngine. In this state, the AppEngine monitors for Active AppEngine failure by verifying heartbeat pings received from the Active engine and checks that all files required for execution are in sync with the Active engine. It receives the following from the Active AppEngine: checkpoint change data, subscription-related notifications, alarm state changes, and history blocks.
- **Standby - Sync'ng with Active**: The state of an AppEngine when it is synchronizing code modules with the Active object. If code modules exist on the Standby computer that do not exist on the Active node, they are uninstalled, and likewise, any code modules that exist on the Active node but not on the Standby node are installed. After all code modules are synchronized, the AppEngine transitions to Standby-Sync'd Code state.
- **Standby - Syncing Code**: The state of a Standby AppEngine that successfully synchronized all code modules with the Active object.

- **Standby - Syncing Data**: The state of a Standby AppEngine when all object-related data, including checkpoint and subscriber information, are synchronized with the Active object. An object in this state typically transitions to Standby-Ready state.
- **Switching to Active**: A transitional state when a Standby AppEngine is commanded to become Active.
- **Switching to Standby**: A transitional state when an Active AppEngine is commanded to become Standby. When the active engine is switched to standby, the engine will be restarted. This transition state can be switched off by the user changing the attribute of the engine.
- **Failed:** The state of a redundant partner when its process crashes or is terminated by the user. The AppEngine process can be restarted using System Management Console/PlatformManager.
- **Unknown**: The state of a redundant partner when a communication loss occurs between AppEngines or when the partner AppEngine is stopped, shutdown, or undeployed.

For examples on redundant configuration, see the Schneider Electric Software Global Customer Support website.

Troubleshooting

Most troubleshooting happens in the System Management Console. For more information about using the System Management Console, see the *System Management Console User's Guide.*

Certain requirements are validated by the system infrastructure. For example, the order in which you configure an object pair for redundancy is validated.

The following problems can occur when you are using redundancy:

- You can configure an AppEngine for redundancy before configuring its associated WinPlatform. If you do this, you see an error message that the Platform (specifically, the RMC) is not configured yet.
- If the **RMC IP Address** parameter is not configured in both hosting WinPlatforms, then the configuration state of both Primary and Backup AppEngines changes to Error. You also see a message indicating that the host WinPlatform is not configured with the network adapter required for redundant communications. When the RMC IP Address is configured and the WinPlatforms are checked in, the hosted AppEngines are automatically revalidated and the Error state is resolved. If hosted AppEngines are checked out, they are not revalidated.

- If both Primary and Backup AppEngines are assigned to the same WinPlatform and you try to deploy both engines, both the Primary and Backup fail to deploy. You see a message that the Primary and Backup objects must be hosted by different WinPlatforms. Reassign the Backup object to another WinPlatform and deploy it separately.
- If both the **Network Address** and **RMC IP Address** parameters in the WinPlatform's editor refer to the same network card, you get a warning message when you save the configuration. These parameters must refer to different network cards.
- Before restarting a computer that hosts one of a redundant pair of AppEngines (either the Active or Backup), ensure that the Primary Network is connected. A restart while the Primary Network is disconnected makes the Primary Network bind to the RMC's IP address. An incorrect redundancy state occurs, indicating that redundancy functionality is good. Any time you restart a redundancy-enabled computer, check for proper network connections afterwards. For more information, see "Using Multiple Network Interface Cards" on page 424.
- When scripting is used to activate attributes, attributes displayed in the watch window may appear to remain in an initializing state after deployment or engine failover. This happens because the script is capturing the attributes while they are still initializing on one of the redundant engines, before the engines are able to synchronize. The attributes will move to active state, but unless there is retry logic in the script, that information will not be returned. This situation can be avoided by ensuring that your scripts contain retry logic to capture the updated state of the attributes.

Generating Alarms

When failover conditions occur, the ArchestrA system reports alarms to subscribed alarm clients like InTouch.

These alarms contain the following information:

- The name of the AppEngine reporting the alarm.
- The node name of the AppEngine reporting the alarm.
- The state of the AppEngine.
- The node name of the AppEngine's partner object.

Depending on what caused the failover, the Standby AppEngine can become the Active AppEngine in an Off scan state and alarms might not be generated.

If the Active AppEngine is shutdown off scan, the checkpointer can transfer that state to the Standby. When the Standby becomes the Active, it starts off scan. When the AppEngine is put on scan, alarms then are generated.

Reported alarms include the following:

Alarm	Previous State	Current State	Alarm Raised When	Alarm Cleared When	Alarm Reported By
Standby Not Ready *	Active	Standby Not Ready	Standby Not Ready	Entering Standby Ready	Active Engine
Standby Not Available	Active	Active Standby Not Available	Active Standby Not Available	Entering Active	Active Engine
Failover Occurred			Standby becomes Active	During the next scan of the Active engine	Active Engine

* The Active AppEngine monitors the status of the Standby through the RMC to determine when to raise this alarm. Also, if the Active AppEngine is in Active-Standby not Available state, this alarm is not generated.

When a failover occurs, the Standby AppEngine that becomes active carries alarms outstanding from the old Active AppEngine. The state of those old alarms, though, will change to reflect the new partner's status.

Timestamps are preserved for alarms, including when:

- The alarm was acknowledged
- The alarm condition went true
- The alarm condition went false

Finally, the following information is preserved for alarms:

- An alarm was acknowledged
- The message input by the operator when the alarm was acknowledged

All alarm state information is collected and sent to the Standby AppEngine at the end of a scan cycle and before being sent to alarm clients.

The sequence of reporting alarms ensures that alarm clients do not report alarms in states that are different from those reported by the Standby AppEngine if the Active AppEngine fails.

Generating History

All active objects (AppEngine and hosted objects) report history data as they normally do in the run-time environment.

Historical data is reported to the historian only from the Active AppEngine.

Losing connectivity with the historian does not cause a failover. The Active AppEngine goes into store-forward mode and caches data every 30 seconds. Store-forward data is synchronized with the Standby AppEngine.

When failover conditions occur, no more than 30 seconds of history data is lost in the transition from Standby to Active status. For more information, see the Schneider Electric Software Global Customer Support website.

Working with Data Acquisition Redundancy

The RedundantDIObject monitors and controls the redundant DIObject data sources at the object level. Unlike redundant AppEngines, individual DIObject data sources do not have redundancy-related states. For all practical purposes, they function as standalone objects.

Only one DIObject data source provides field device data through the RedundantDIObject at a time. Both data sources must have commonly-configured DAGroups. These must be reflected in and channeled through the RedundantDIObject, which monitors the two DIObject data sources. It also determines which one is Active at any given time. Both data sources must also have the same item address space.

After deployment or engine failover, attributes may appear to remain in an initializing state when scripting is used to activate the attributes. The script may capture the attributes while still initializing, before the engines are able to synchronize with each other. When synchronization occurs, the attributes will move to active state, but unless there is retry logic in the script, that information may not be returned. This situation can be avoided by ensuring that your scripts contain retry logic to capture the updated state of the attributes.

Configuring Data Acquisition Redundancy

Data acquisition redundancy objects involve two DIObjects and the RedundantDIObject. In data acquisition redundancy, you must configure all three components:

- Primary DIObject data source
- Backup DIObject data source
- Redundant DIObject data source

Because data acquisition redundant components are essentially standalone objects, all valid operations that apply to any other ApplicationObjects apply to the three objects.

All IDE commands, Galaxy Dump and Load functions, and import and export operations are valid on the two DIObject data sources and the RedundantDIObject.

See the online help associated with each DIObject for help in configuring its Object Editor. Also see the help associated with the RedundantDIObject.

Before you can deploy the RedundantDIObject, you must configure at least one scan group. Also, configure only scan, block read, and block write groups shared by the Primary and Backup DIObjects in the RedundantDIObject.

To configure the Redundant DIObject

1. On the **General** tab of the Object Editor, set the **Primary DI Source** and **Backup DI Source**.
2. Set up the common scan groups.
3. Set up the common block read and block write groups on the tabs of the Object Editor.

Deploying Redundant DIObjects

Deployment for data acquisition redundancy is the same as individually deploying unrelated objects. No special conditions apply to each DIObject data source and the RedundantDIObject.

See “Deploying and Running an Application” on page 189 for more information about deploying objects.

What Happens in Run Time

The three objects in the data acquisition redundancy scheme (RedundantDIObject and its two DIObject data sources) work like any other ArchestrA object when deploying, alarming, and historizing. They have no special redundancy-related states or restrictions.

During run time, the RedundantDIObject monitors the status of the DIObject data sources, and handles the switching from Active to Standby object if failure conditions occur.

RedundantDIObject and PLC Connectivity

For the RedundantDIObject, you can use the scan group PingItem attribute to monitor the connection status of the PLC at run time. If you are using the redundancy feature of the RedundantDIObject to communicate with DIObjects, you should configure the PingItem attribute for each scan group.

Glossary

application
A collection of objects in a Galaxy Repository that performs automation tasks. Synonymous with Galaxy. There can be one or more applications within a Galaxy Repository.

Application Engine (AppEngine)
A scan-based engine that hosts and executes the run-time logic contained within AutomationObjects.

ApplicationObject
An AutomationObject that represents some element of your production environment. This can include things like

- An automation process component. For example, a thermocouple, pump, motor, valve, reactor, or tank
- An associated application component. For example, function block, PID loop, Sequential Function Chart, Ladder Logic program, batch phase, or SPC data sheet

Application Server
The supervisory control platform. Application Server uses existing Wonderware products such as InTouch for visualization, the Wonderware Historian for data storage, and the device Integration product line like an Operations Integration Server (OI Server) for device communications.

An Application Server can be distributed across multiple computers as part of a single Galaxy namespace.

Application view
The Applications view shows the object-related contents of the Galaxy in four different ways: Model view, Deployment view, Derivation view, and Operations view. The Model view appears when the IDE is opened for the first time.

ArchestrA
The distributed architecture for supervisory control and manufacturing information systems. It is an open and extensible technology based on a distributed, object-based design.

ArchestrA Object Toolkit — A programmer's tool to create new ApplicationObjects and Device Integration Object (DIObjects) templates, including configuration and run-time implementations. Includes a tool to build DI Objects and create unique Domain Objects that interact with DIObjects in the ArchestrA environment.

ArchestrA Symbol — A graphic you create and use to visualize data in an InTouch HMI system. You use the ArchestrA Symbol Editor to create ArchestrA Symbols from basic elements, such as rectangles, lines, and text elements.

area — A logical grouping of AutomationObjects that represents an area or unit of a plant. It is used to group related objects for alarm, history, and security purposes. It is represented by an area AutomationObject.

area object — The system object that represents an area of your plant within a Galaxy. The Area Object acts as an alarm concentrator, and places other Automation Objects into proper context with respect to the actual physical automation layout.

assignment — The designation of a host for an AutomationObject. For example, an AppEngine object is assigned to a WinPlatform object.

attribute — An externally accessible data item of an AutomationObject.

attribute reference string — A text string that references an attribute of an AutomationObject.

AutomationObject — An object type that represents permanent things in your plant, such as ApplicationObject or Device Integration Object (DIObjects), with user-defined, unique names within the Galaxy. It provides a standard way to create, name, download, execute, and monitor the represented component.

Backup Application Engine — The object created by the ArchestrA infrastructure when the Primary object is enabled for redundancy. See redundancy for further details.

base template — A root template at the top of a derived hierarchy. Unlike other templates, a base template is not derived from another template but developed with the ApplicationObject Toolkit and imported into a Galaxy. All base templates names start with a dollar sign ($).

Block Read Group — A DAGroup that is triggered by the user or another object. It reads a block of data from the external data source and indicates the completion status.

Block Write Group — A DAGroup that is triggered by the user or another object after all the required data items are set. The block of data is sent to the external data device. When the block write is complete, it indicates the completion status.

Bootstrap The base ArchestrA service which is required on all ArchestrA computers. It provides the base software environment to enable a platform and allows a computer to be included in the Galaxy Namespace.

Buffered Data Data captured and stored locally on a remote device for later transfer to a supervisory system for processing, analysis, and long-term storage. Can be configured within the I/O feature in the Attributes page.

change log The revision history that tracks the life cycle activities of ArchestrA Objects, such as object creation, check in/check out, deployment, and import/export.

change propagation The ability to create templates which allows each component template to support changes such that a change in one of the elements can be automatically propagated to all — or select, related — object instances.

check in IDE operation for making a configured object available for other users to check out and use.

check out IDE operation for the purpose of editing an object. It makes the item unavailable for other users to check out.

Checkpoint The act of saving to disk the configuration, state, and all associated data necessary to support automatic restart of a running AutomationObject. The restarted object has the same configuration, state, and associated data as the last checkpoint image on disk.

compound object An ApplicationObject that contains at least one other ApplicationObject.

contained name An alternate naming convention that when combined with the TagName of the root container object, results in the hierarchical name. For example, for a given object, its Hierarchical Name = `Line1.Tank1.InletValve` and its Contained Name= `InletValve`.

containment A hierarchical grouping that allows one or more AutomationObject to exist within the name space of a parent object and be treated like parts of the parent object. Allows for relative referencing to be defined at the template and instance level.

DAGroup A data access group associated with Device Integration Object (DIObjects). It defines how communications are achieved with external data sources. It can be a Scan Group, Block Read Group or Block Write Group.

DAServer Manager (DAS Manager) The System Management Console (SMC) snap-in supplied by the Operations Integration Server (OI Server) that provides the required interface for activation, configuration, and diagnosis of the OI Server.

Data Access Server (DAServer)	See Operations Integration Server (OI Server).
Data Access Server Toolkit (DAS Toolkit)	A developer tool that can build an Operations Integration Server (OI Server).
Deployment	The operation which instantiates an AutomationObject instance in the ArchestrA run time. This action involves installing all the necessary software and instantiating the object on the target platform with the object's default attribute data from Galaxy Repository.
Deployment view	The part of the Application view in the IDE that shows how objects are physically dispersed across Platforms, areas and Engines. This is a view of how the application is spread across computing resources.
derivation	The creation of a new template based on an existing Template.
Derivation view	The part of the Application view in the IDE that shows the parent-child relationship between base templates, derived templates and derived instances. A view into the genealogy of the application.
derived template	Any template with a parent template. Derived templates inherit the attributes of the parent template. You can changes these attributes in the derived template.
Device Integration Object (DIObjects)	An AutomationObject that represents the communication with external devices or software. DIObjects run on an Application Engine (AppEngine), and include DINetwork Objects and DIDevice Objects.
DHCP	Dynamic Host Configuration Protocol. A network configuration protocol for hosts on IP networks.
DIDevice Object	An object that represents the actual external device (for example, a PLC or RTU) that is associated with a DINetwork Object. It can diagnose and browse data registers of the DAGroups for that device.
DINetwork Object	An object that represents the network interface port to the device through the Data Access Server (DAServer) or the object that represents the communications path to another software application. It provides diagnostics and configuration for that specific network card.
element	Basic shapes, such as rectangles, lines, and text elements, and controls you can use to create an ArchestrA Symbol to your specifications.
Element Style	An Element Style defines one or more visual fill, line, text, blink, outline, and status properties of ArchestrA graphic elements. You can apply an Element Style to set preconfigured values to the visual properties of a graphic element.
Engine Object	An ArchestrA system-enabled object that contains Local Message Exchange and provides a host for ApplicationObjects, Device Integration Object (DIObjects) and area objects.

event record	The data that is transferred about the system and logged when a defined event changes state. For example, an analog crosses its high level limit, an acknowledgement is made, or an operator logs in to the system.
export	The act of generating a package file (.aaPKG) extension from persisted data in the Galaxy database. You can import the resulting .aaPKG file into another Galaxy.
FactorySuite Gateway	A Microsoft Windows application program that acts as a communications protocol converter. Built with the ArchestrA DAS Toolkit, FS Gateway links clients and data sources that communicate using different data access protocols.
Galaxy	The entire application. The complete ArchestrA system consisting of a single logical name space (defined by the Galaxy database) and a collection of platform objects, Engine Objects and other objects. One or more networked computers that constitute an automation system. This is referred to as the Galaxy Namespace.
Galaxy database	The relational database containing all persistent configuration information like templates, instances, and security in a Galaxy Repository.
Galaxy Database Manager	A utility to manage your Galaxy. It can back up and restore Galaxies if they become corrupt or to reproduce a Galaxy on another computer. The Galaxy Database Manager is part of the System Management Console (SMC).
GalaxyObject	The object that represents a Galaxy.
Galaxy Repository	The software sub-system consisting of one or more Galaxy databases.
Graphic Toolbox	The part of the IDE main window that shows a hierarchy of graphic toolsets, which contain ArchestrA Symbols and client controls.
hierarchical name	The name of the object in the context of its container object. For example, Tank1.OutletValve, where an object called Tank1 contains the OutletValve object.
Historical Storage System (Historian)	The time series data storage system that compresses and stores high volumes of time series data for later retrieval. The standard historian is the Wonderware Historian.
host	The parent of a child instance in the deployment view. Example: a Platform instance is a Host for an Application Engine (AppEngine) instance.
import	The act of reading a package file (.aaPKG) and using it to create AutomationObject instances and templates in the Galaxy Repository.
instance	An object derived from a template. You deploy instances to the run-time environment.

instantiation The creation of a new instance based on a corresponding template.

Integrated Development Environment (IDE) The Integrated Development Environment (IDE) is the interface for the configuration side of Application Server. In the IDE, you manage templates, create instances, deploy and un-deploy objects, and other functions associated with the development and maintenance of the system.

InTouch View InTouch View clients are InTouch run-time clients that solely use of the Application Server for its data source. In addition, standard InTouch run times can also leverage Application Server.

InTouchViewApp object Represents an InTouch application in the Wonderware Application Server environment. The InTouchViewApp object manages the check-in, check-out, and deployment of an InTouch application.

I/O count Number of I/O points being accessed within a Galaxy. I/O points are real I/O and are not equivalent to InTouch tags. I/O count is based on the number of I/O points that are configured through an OPC Server, I/O Server, Data Access Server (DAServer) or InTouch Proxy Object, over the whole Application Server namespace, regardless of how many computers are in the system.

Message Exchange The object to object communications protocol used by Application Server. Message Exchange includes

LMX	communication between objects
NMX	communication between Galaxy nodes.

Model view The area in the Application view in the IDE that shows how objects are arranged to describe the physical layout of the plant and supervisory process being controlled.

object Any template or instance in a Galaxy database. A common characteristic of all objects is they are stored as separate components in the Galaxy Repository.

Object Viewer A utility in which you can view the attribute values of the selected object in run time. This utility is only available when an object is deployed. Object Viewer shows you diagnostic information on ApplicationObjects so you can see performance parameters, resource consumption and reliability measurements. In addition to viewing an object's data value, data quality and the communication status of the object, you can also modify some of its attributes for diagnostic testing. Modifications can include adjusting timing parameters and setting objects in an execution or idle mode.

OffScan The state of an object that indicates it is idle and not ready to execute its normal run-time processing.

Operations Integration Server (OI Server)	The server executable that handles all communications between field devices of a certain type and client applications. Similar to I/O Servers but with more advanced capabilities. Also called a DAServer.
OnScan	The state of an object in which it is performing its normal run-time processing based on a configured schedule.
Operations view	The area in the IDE that shows the results of validating the configuration of objects.
package definition file (.aaPDF)	The standard description file that contains the configuration data and implementation code for a base template. File extension is .aaPDF.
package file (.aaPKG)	The standard description file that contains the configuration data and implementation code for one or more objects or templates. File extension is .aaPKG.
Platform Count	Number of computers in the Galaxy. Each Workstation or Server communicating directly with the Application Server requires a platform to be part of the Galaxy Namespace. This includes each InTouch and InTouch View client.
Platform Manager	This utility is an extension snap-in to the ArchestrA System Management Console (SMC).Provides Galaxy application diagnostics by allowing you to view the run-time status of some system objects and to perform actions upon those objects. Actions include setting platforms and engines in an executable or idle mode and starting and stopping platforms and engines.
platform object	An object that represents a single computer in a Galaxy, consisting of a system wide message exchange component and a set of basic services. This object hosts all Application Engines.
Primary Application Engine	The object created by the ArchestrA infrastructure when the Backup object is created through redundancy. See redundancy for further details.
properties	Data common to all attributes of objects, such as name, value, quality, and data type.
proxy object	An AutomationObject that represents an actual product for the purpose of device integration with the Application Server or InTouch HMI. For example, a Proxy object enables the Application Server to access an OPC server.
redundancy	Two computers: One executes objects. The other is a stand by.
RedundantDIObject	Monitors and controls the redundant Device Integration Object (DIObjects) data sources. Unlike redundant AppEngines, individual DIObject data sources do not have redundancy-related states. They function as stand-alone objects.

Redundant Message Channel — A dedicated Ethernet connection which is required between the platforms hosting redundant engines. The RMC is vital to keep both engines synchronized with alarms, history, and checkpoint items from the engine that is in the Active Role. Each engine also uses this Message Channel to provide its health and status information to the other.

reference — A string that refers to an object or to data within one of its attributes.

relative reference — Objects may refer to themselves, containers, hosts or to child objects elsewhere in the parent/child hierarchy using special reserved keywords such as “Me” or “MyContainer”. Relative references continue to work properly even if the object that is referenced is renamed. Examples of relative references are “Me”, “MyArea”, “MyContainer”, “MyHost”, and “MyPlatform”.

remote reference — The ability to redirect ArchestrA object references or references to remote InTouch tags. The new script function that redirects remote references at run time is IOSetRemoteReferences.

SCADA — Supervisory Control and Data Acquisition (SCADA) is a centralized industrial control system that controls and monitors sites or systems which are spread over large areas - an entire industrial plant or a country.

Scan Group — A DAGroup that requires only the update interval be defined. The data is retrieved at the requested rate.

Scan State — The Scan State of an object in run time. This can be either OffScan or OnScan.

security — Application Server security is applied to IDE, System Management Console (SMC), and the run-time data level. At the run-time data level which centralizes the definition of all permissions to the ApplicationObjects. These ApplicationObjects can be accessed by a variety of clients but the security is centrally defined, allowing ease of maintenance. Users that are allowed to modify these ApplicationObjects at run time are mapped to the objects by user-defined roles. These roles can be mapped directly to existing groups in a Microsoft Domain or workgroup.

System Management Console (SMC) — The central run-time system administration/management product where you perform all required run-time administration functions.

system object — An object that represents an area, platform or engine.

TagName — The unique name given to an object. For example, for a given object, its TagName = V1101 and its HierarchicalName = Line1.Tank1.InletValve.

template — An object containing configuration information and software templates used to create a derived template or instance.

Template Toolbox	The part of the IDE main window that shows Toolsets containing templates. The Template Toolbox shows a tree view of template categories in the Galaxy.
Toolset	A named collection of templates shown together in the IDE Template Toolbox.
ViewEngine object	Hosts InTouchViewApp objects. The ViewEngine object supports common engine features such as deployment, undeployment, startup, and shutdown.
WinPlatform object	An object that represents a single computer in a Galaxy. A WinPlatform object consists of a system-wide message exchange component and a set of basic ArchestrA services. The WinPlatform object hosts the Application Engine (AppEngine).

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W