

# Wonderware® Application Server User's Guide

Invensys Systems, Inc.

Revision A

Last Revision: 7/25/07



## Copyright

© 2007 Invensys Systems, Inc. All Rights Reserved.

All rights reserved. No part of this documentation shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Invensys Systems, Inc. No copyright or patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this documentation, the publisher and the author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

The information in this documentation is subject to change without notice and does not represent a commitment on the part of Invensys Systems, Inc. The software described in this documentation is furnished under a license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of these agreements.

Invensys Systems, Inc.  
26561 Rancho Parkway South  
Lake Forest, CA 92630 U.S.A.  
(949) 727-3200

<http://www.wonderware.com>

For comments or suggestions about the product documentation, send an e-mail message to [productdocs@wonderware.com](mailto:productdocs@wonderware.com).

## Trademarks

All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized. Invensys Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

Alarm Logger, ActiveFactory, ArcestrA, Avantis, DBDump, DBLoad, DT Analyst, FactoryFocus, FactoryOffice, FactorySuite, FactorySuite A<sup>2</sup>, InBatch, InControl, IndustrialRAD, IndustrialSQL Server, InTouch, MaintenanceSuite, MuniSuite, QI Analyst, SCADAAlarm, SCADASuite, SuiteLink, SuiteVoyager, WindowMaker, WindowViewer, Wonderware, and Wonderware Logger are trademarks of Invensys plc, its subsidiaries and affiliates. All other brands may be trademarks of their respective owners.

---

# Contents

<b>Welcome.....</b>	<b>11</b>
Documentation Conventions.....	11
Technical Support .....	12
<b>Chapter 1 Getting Started with IDE .....</b>	<b>13</b>
What's a Galaxy? .....	13
Creating a New Galaxy .....	15
Connecting to an Existing Galaxy .....	17
Getting Around the IDE.....	18
Using the Template Toolbox .....	20
Using the Graphics Toolbox.....	20
Using the Application Views.....	21
Model View .....	21
Deployment View.....	24
Derivation View .....	27
Operations View .....	28
Customizing Your Workspace.....	29
Docking Views .....	29
Floating Views .....	29
Hiding Views.....	30
Resetting the Workspace .....	30
Synchronizing the Views.....	30
Configuring User Information .....	31
Logging on and Logging off.....	33
Changing Users .....	34

**Chapter 2 Getting Started with Objects .....35**

About Templates and Instances .....	37
Instances .....	37
Templates .....	37
Propagation.....	38
About Base Templates .....	39
Application Templates .....	39
Device Integration Templates .....	39
System Templates .....	40
About Derived Templates .....	42
Viewing Object Properties .....	43

**Chapter 3 Working with Objects .....45**

Managing Toolsets .....	45
Creating Toolsets.....	46
Creating Child Toolsets .....	47
Deleting Toolsets .....	47
Creating Derived Templates.....	48
Deriving Templates from Another Derived Template..	49
Creating Contained Templates.....	50
ApplicationObject Containment .....	53
Using Contained Names .....	57
Containment Examples.....	58
Viewing Containment Relationships.....	60
Renaming Contained Objects .....	60
Editing Objects .....	61
Getting Help .....	63
Help File Structure .....	63
About the General Editor Layout.....	64
Locking and Unlocking Template Attributes .....	65
Setting Object Security .....	68
Group Locking/Security .....	69
About the Object Information Page.....	70
Customizing Help.....	71
Finding the Help Folders.....	71
About the Scripts Page.....	72
About the UDAs Page .....	74
About the Extensions Page.....	76
Referencing Objects Using the Galaxy Browser.....	78
Browsing for Attributes .....	78
Viewing Attribute Details in the Galaxy Browser ...	80
Browsing for Graphics .....	83

	Browsing for Element Properties .....	85
	Creating a Filter for the Galaxy Browser .....	86
	Changing How Information is Shown in the Galaxy Browser.....	88
<b>Chapter 4</b>	<b>Managing Objects.....</b>	<b>89</b>
	Checking Objects Out.....	89
	Checking Objects In .....	90
	Validating Objects .....	90
	Validating Scripts and Other External Components ...	91
	Validating Manually .....	92
	Creating Instances .....	93
	Renaming Objects.....	94
	Deleting Objects .....	96
	Exporting Objects.....	96
	Exporting Script Function Libraries .....	97
	Importing Objects.....	98
	Importing Scripts Function Libraries .....	100
	After You Import .....	100
<b>Chapter 5</b>	<b>Enhancing Objects .....</b>	<b>101</b>
	Creating and Working with UDAs .....	102
	UDAs and Scripting .....	103
	UDA Naming Conventions.....	104
	Writing and Editing Scripts.....	105
	About Scripts .....	106
	Script Execution .....	106
	Locking Scripts .....	111
	Creating and Working with Extensions.....	113
	About Extension Inheritance.....	113
	Using the InputOutput Extension.....	116
	When Objects Are On Scan.....	117
	Using InputOutput Extensions in Scripts .....	118
	Using the Input Extension.....	119
	Using the Output Extension.....	120
	Working with Outputs .....	121
	Quality of Input, InputOutput and Output Extensions .....	123
	Using the Alarm Extension .....	124
	Using the History Extension .....	125
	Using the Boolean Label Extension .....	126

---

Creating and Working with Graphics .....	128
Adding Graphics .....	129
Modifying Graphics .....	129
Renaming Graphics .....	129
Deleting Graphics .....	130
<b>Chapter 6 Deploying your Galaxy .....</b>	<b>131</b>
Planning for Deployment .....	131
Determining Galaxy Status .....	133
Deploying Objects .....	134
Deployment Error Messages .....	138
Redeploying Objects .....	139
Undeploying Objects .....	139
Uploading Run-time Configuration .....	140
Undeployment Situations .....	142
<b>Chapter 7 Working with History .....</b>	<b>143</b>
Configuring History .....	144
About the Wonderware Historian .....	146
About Manual Data Acquisition Service (MDAS) ..	146
Configuring Objects to Store History .....	147
During Run Time .....	148
Deploying and Undeploying .....	149
Store Forward Mode .....	149
Configuring WinPlatforms and AppEngines for History .....	150
<b>Chapter 8 Working with Alarms and Events .....</b>	<b>153</b>
About Events and Alarms .....	153
Event Examples .....	154
Alarm Examples .....	154
Items That Are Not Events or Alarms .....	154
Configuring Alarms .....	155
Setting Alarms on the Extension Page .....	157
About Alarm Event Distribution .....	158
Area AutomationObject .....	159
Alarm and Event Subscription .....	159

Enabling and Disabling Alarms .....	160
Enabling Alarms.....	160
Disabling Alarms.....	161
During Run Time.....	161
Using the InTouch HMI as the Alarm and Event Client .....	162
Alarms and Events in the InTouch HMI and in Application Server.....	164
<b>Chapter 9 Working with References .....</b>	<b>167</b>
Using Message Exchange and Attributes .....	168
Reference Strings .....	168
Relative References .....	169
Property References .....	170
Arrays.....	170
Formatting Reference Strings .....	171
Using Literals .....	171
Viewing Attributes in Objects.....	174
Viewing References and Cross References.....	175
Finding Objects.....	177
Using Galaxy References in InTouch .....	178
<b>Chapter 10 Working with Security .....</b>	<b>183</b>
About Security .....	184
About Authentication Modes .....	186
Multiple Accounts Per User.....	187
Changing Security Settings .....	187
About Security Groups.....	188
About Roles .....	189
About Users .....	190
Configuring Security .....	190
Assigning Users to Roles.....	195
Deleting Security Groups.....	197
Deleting Roles.....	197
Deleting Users .....	197
About OS Group-based Security.....	198
Connecting to a Remote Node for the First Time .....	198
Cached Data at Log In .....	198
Mixed or Native Domains .....	199
Using InTouch Access Levels Security.....	199

<b>Chapter 11</b>	<b>Managing Galaxies .....</b>	<b>201</b>
	Backing Up and Restoring Galaxies .....	201
	Changing Galaxies .....	203
	Deleting a Galaxy .....	204
	Exporting a Galaxy Dump File.....	204
	Looking at the Galaxy Text Dump File Structure.....	206
	Host Attributes.....	207
	About Quotation Marks and Carriage Returns .....	207
	Time Formats in Excel.....	208
	Importing a Galaxy Load File.....	208
	Synchronizing Time across a Galaxy .....	210
	Using Time Synchronization in Windows Domains ...	210
	Synchronization Schedule.....	210
	Required Software.....	211
	Hosting Multiple Galaxies in One Galaxy Repository ..	212
	Managing Licensing Issues.....	212
	Viewing License and End-User License Agreement Information .....	212
	Updating a License.....	217
	Disk Space Requirements .....	218
	Managing Communication between Galaxy Nodes.....	218
	About ArcestrA User Accounts.....	219
	Using Multiple Network Interface Cards .....	220
	Defining the Order of the NIC.....	220
	Configuring the IP Address and DNS Settings .....	221
<b>Chapter 12</b>	<b>Working with Redundancy .....</b>	<b>223</b>
	About Redundancy .....	223
	Configuring AppEngine Redundancy.....	224
	Redundancy during Run Time.....	225
	Working with AppEngine Redundancy.....	226
	Configuring the Redundancy Message Channel.....	227
	Configuring Redundancy .....	228
	Configuring Redundancy in Templates.....	229
	Deleting Redundant AppEngines .....	230
	Deploying AppEngine Objects .....	230
	Configuration Requirements .....	231
	Undeploying AppEngine Objects.....	232
	During Deployment.....	233
	AutomationObjects at Run Time .....	233
	During Run Time .....	233
	AppEngine Redundancy States .....	234



---

Troubleshooting .....	236
Generating Alarms .....	238
Generating History .....	239
Working with Data Acquisition Redundancy .....	240
Configuring Data Acquisition Redundancy .....	240
Deploying Redundant DIOObjects .....	241
What Happens in Run Time .....	241
RedundantDIOObject and PLC Connectivity .....	241
<b>Index .....</b>	<b>253</b>



---

# Welcome

This documentation describes how to use the ArchestrA Integrated Development Environment (IDE) to develop and customize your 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.
<b>Bold</b>	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.

---

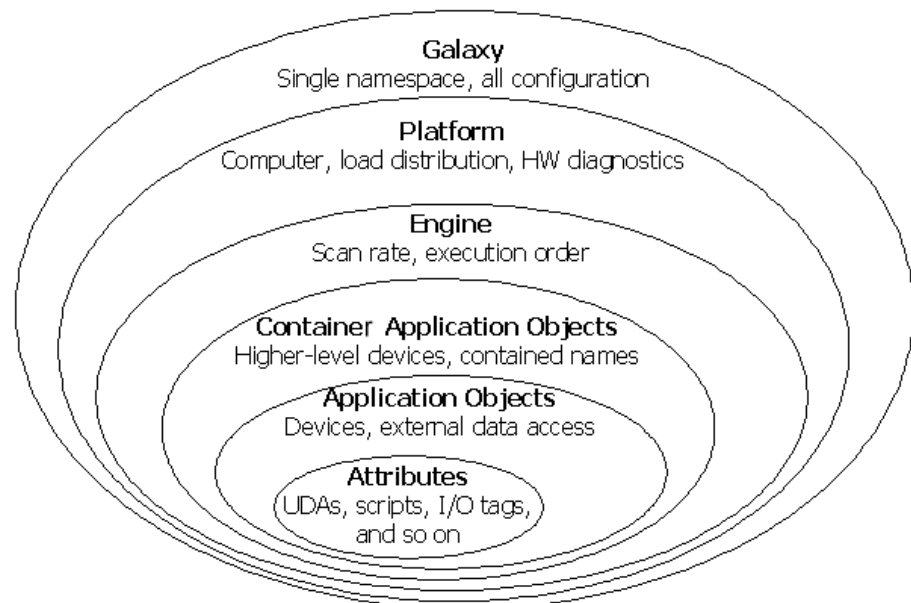
# Chapter 1

## Getting Started with IDE

This section shows you how to create or open a Galaxy. It also briefly describes the IDE and how to navigate your Galaxy in the workspace.

### What's a Galaxy?

A Galaxy is your specific production environment, including all computers and components. It is a collection of platforms, engines, templates, instances, and attributes you define as the parts of your specific application. This collection is stored in a Galaxy database.



A Galaxy database resides on a single network computer. 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.

A Galaxy database can reside on any computer on your network with the SQL server, Bootstrap, and Galaxy Repository software installed.

You can deploy Galaxy components, such as platforms and engines, to multiple computers to share the working load during run time. For more information, see *Deploying your Galaxy* on page 131.

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, animation links, etc, from any machine in the system 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 183.

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.

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 212.

## Creating a New Galaxy

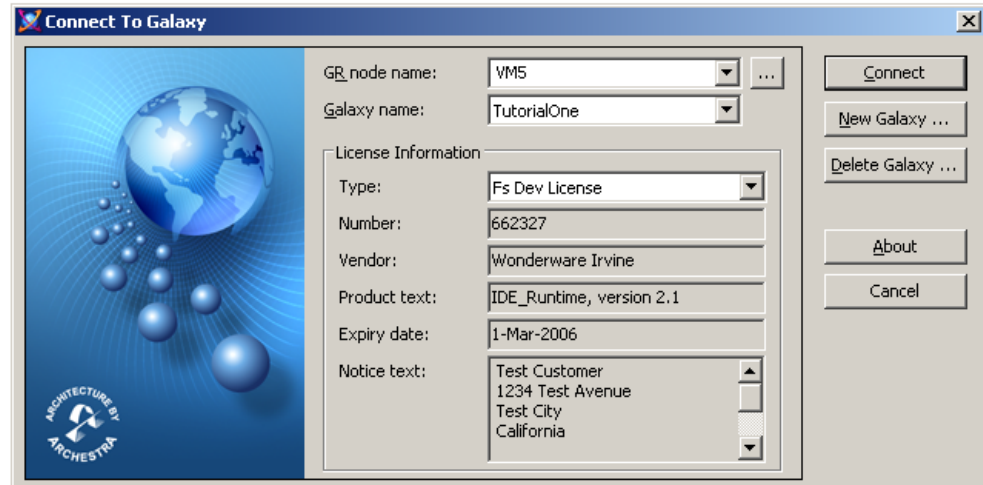
Creating a new Galaxy requires you to specify a Galaxy Repository (GR) node name and the name of the Galaxy. The Galaxy database is created and is ready for you to connect to and use.

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

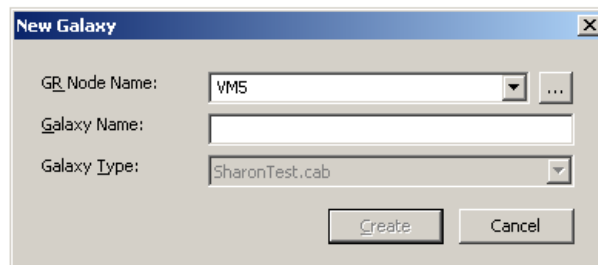
New Galaxies are created without any security restrictions. To learn more about security restrictions, see *Working with Security* on page 183.

### 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** Do the following:

- In the **GR Node Name** list, type, or select the name of a computer that has the Galaxy Repository software installed. Click the **Browse** button to browse the available domains and nodes on your network.
- In the **Galaxy Name** box, type the name of the Galaxy you want to create within that Galaxy Repository. Galaxy names can be up to 32 alphanumeric characters, including \_ (underscore), \$, and #. The first character must be a letter. Galaxy names cannot contain spaces.
- In the **Galaxy Type** list, select the name of the already created backup galaxy files which are located at BackupGalaxies folder. The selected file will be used as template to create new galaxy. System will restore the selected backup galaxy and rename to the galaxy name which user has provided in Galaxy Name box.

---

**Note** You cannot use the following reserved names as Galaxy names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

---

- Click **Create**. The **Create Galaxy** dialog box opens, showing the Galaxy database being created.
- 4** When the Galaxy database is created, click **Close**. You are ready to open the Galaxy and start working with it. For more information about opening an existing Galaxy, see Connecting to an Existing Galaxy on page 17.



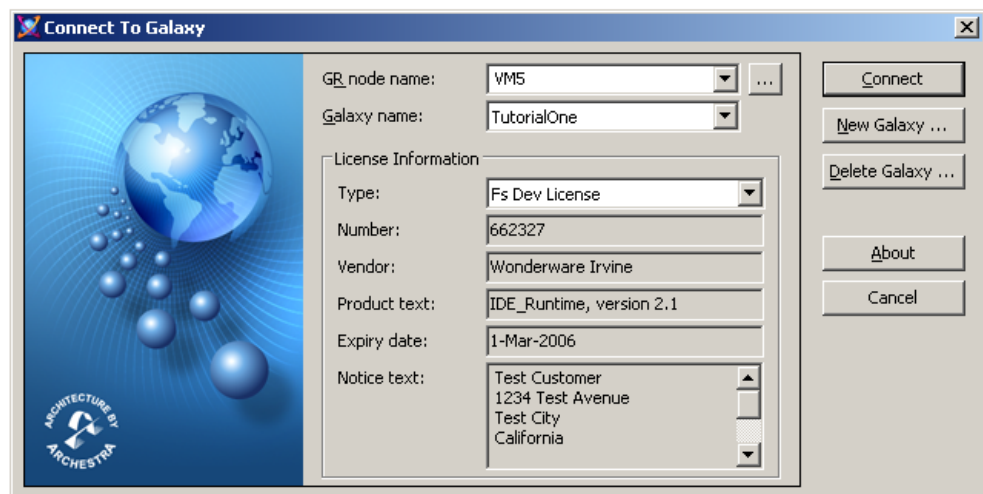
## Connecting to an Existing Galaxy

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 183.

### 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:



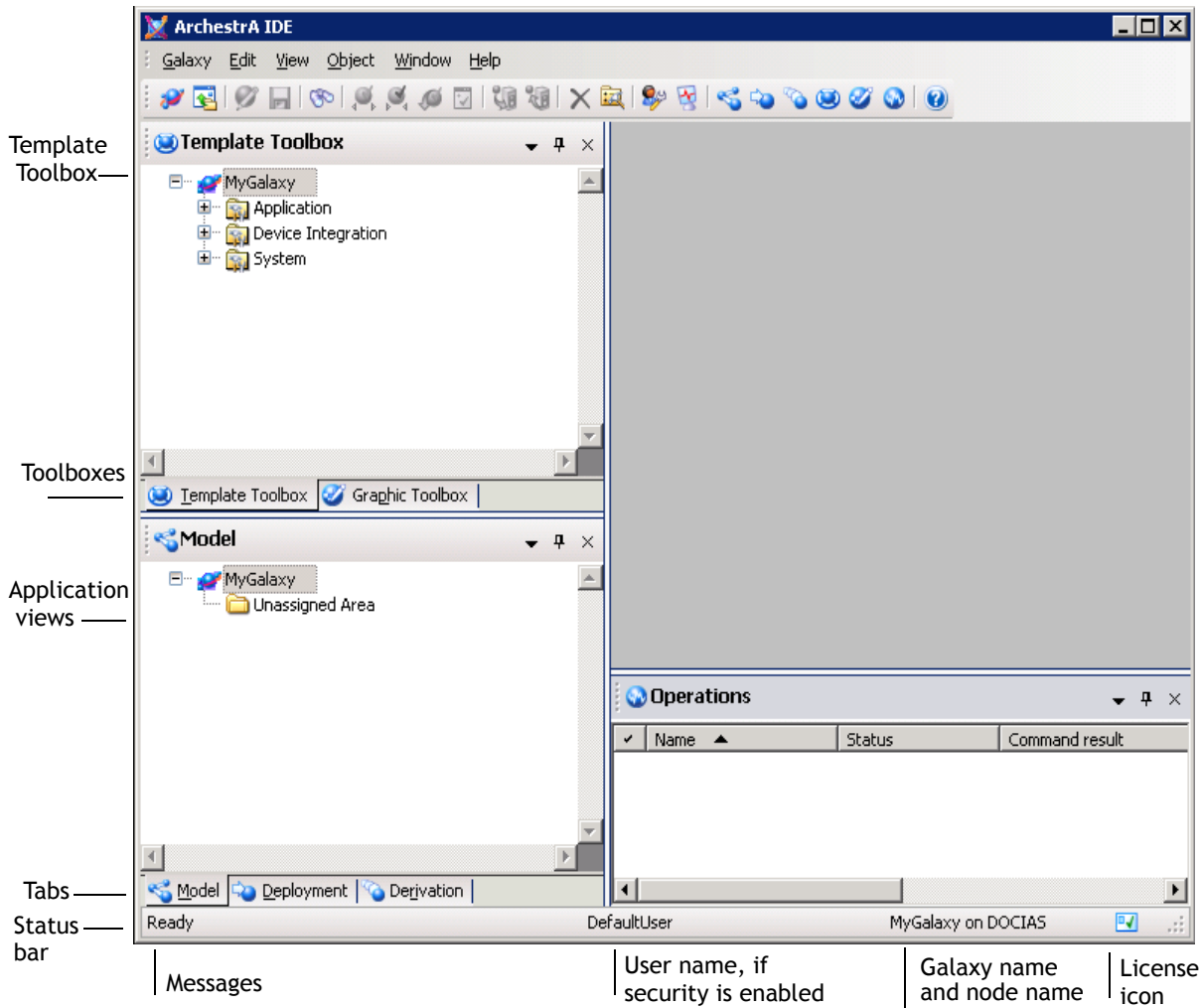
- In the **GR node name** list, select the name of a computer you previously connected to. Click the **Browse** button to browse and select from the available domains and nodes on your network.
- In the **Galaxy name list**, select the name of the Galaxy on that GR node.
- Click **Connect**.

If the selected Galaxy 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

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



For a complete discussion of items like templates and instances, see About Templates and Instances on page 37.

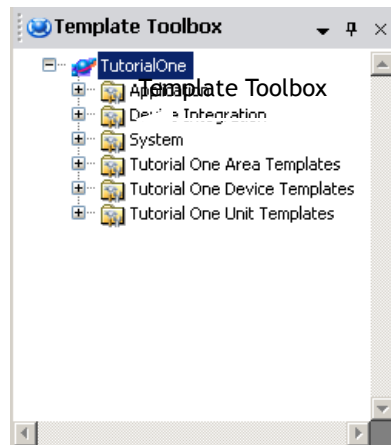
Views in the IDE include:

Template Toolbox	Expand the top level folders to see the different templates in the toolboxes.
Graphic Toolbox	Contains the global ArcestrA graphics that can be used in the Galaxy. For information, see the <i>Creating and Managing ArcestrA Graphic's User Guide</i> .
Application views	Click the tabs at the bottom to open: <ul style="list-style-type: none"><li>• <b>Model view</b> - the object relationship to the automation scheme layout. The objects are organized into Areas that typically represent the physical plant layout.</li><li>• <b>Deployment view</b> - the object relationship to the computers that comprise the deployed system that the objects run on.</li><li>• <b>Derivation view</b> - 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.</li></ul>
Status bar	Shows messages, user name, Galaxy name and node, and license information. Turn off the Status bar by clicking <b>Status</b> bar on the <b>View</b> menu.

## Using the Template Toolbox

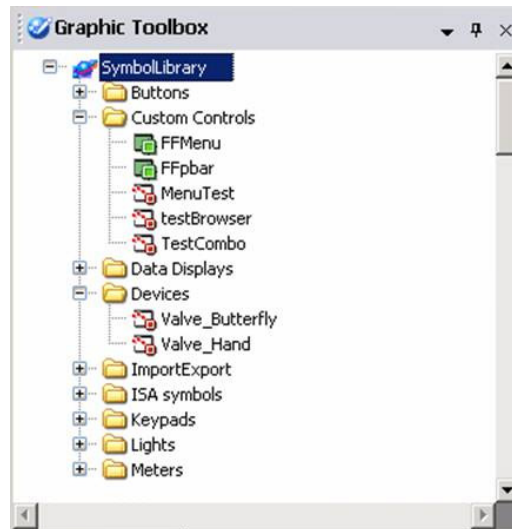
The Template Toolbox lists template toolsets, which contain object templates. The Template Toolbox shows a tree view of template categories in the Galaxy. Double-click a category to open that toolset and see the templates it contains.

A new Galaxy is automatically populated with base templates.



## Using the Graphics Toolbox

The Graphic Toolbox shows a treeview of toolsets which 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 the Application Views

Base templates and non-base templates are included with Application Server. Non-base templates include: \$Boolean, \$Double, \$Float, \$Integer, and \$String. They are derived from \$FieldReference. 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 in 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 and Model view.

### Model View

The Model view shows objects and their containment relationships, organized in a folder structure. For more information about containment, see [Creating Derived Templates](#) on page 48.

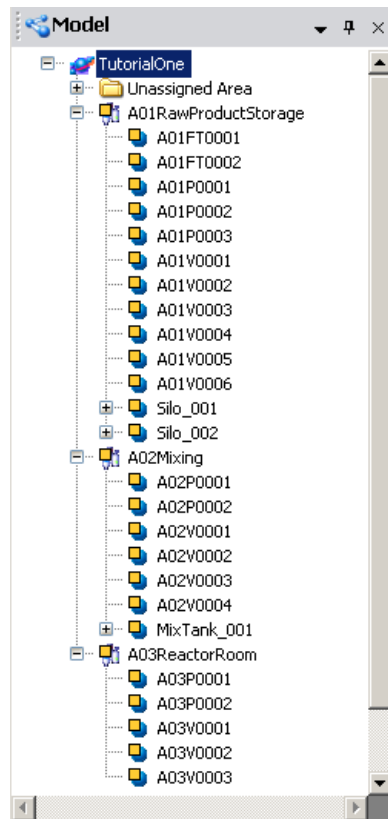
This view most accurately represents an application perspective of your processes. For example, specific process areas, tanks, valves, pumps and their relationships are listed based on containment.

---

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

---

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 only be nested 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 Derived Templates](#) on page 48.

---

**Note** 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 57.

- 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.



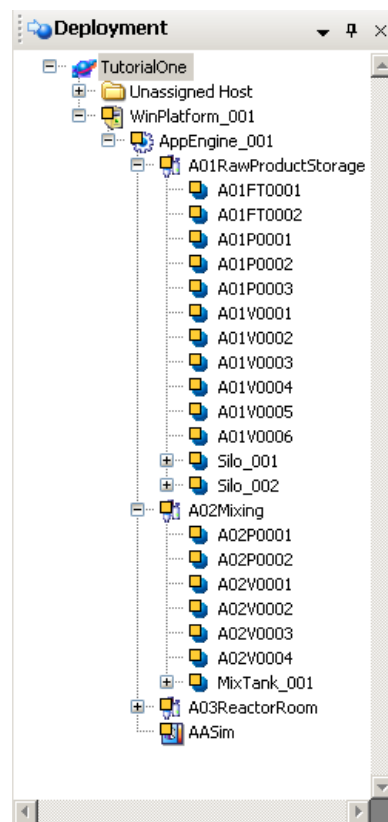
To assign an object to another, drag it onto the host object. If that object is an inappropriate assignment match, the international **Not** symbol appears. To unassign an object, drag it to the **Unassigned Host** folder.

## 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 ArchedrA 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 DI Objects, 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 DI Device 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

[No icon]

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



Warning



Error. The object is 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 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.

---

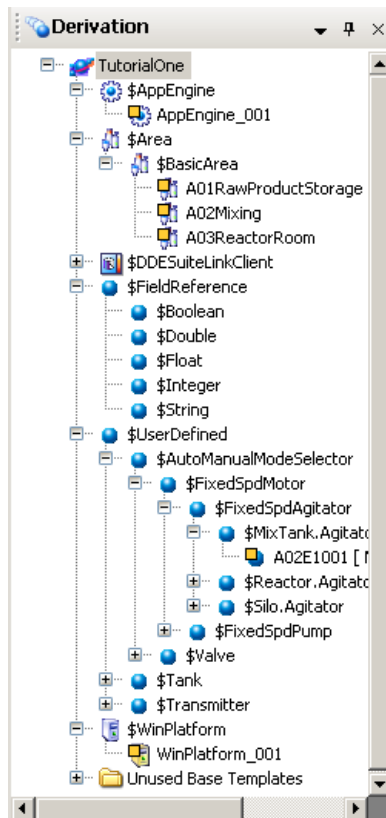
**Note** You must undeploy an object that is currently deployed before reassigning it from one object to another.

---

## 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:

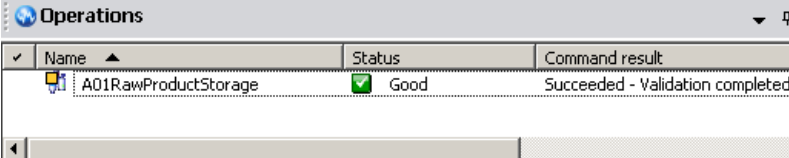
- The top of the tree is the Galaxy.
- Base templates with associated child objects, either derived templates or instances, are shown under the Galaxy.
- Under each base template, derived templates and instances created from the base template are listed. Multiple levels are allowed. Instances created 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.

## 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**.



Name	Status	Command result
A01RawProductStorage	Good	Succeeded - Validation completed

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 90.

---

**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.

---

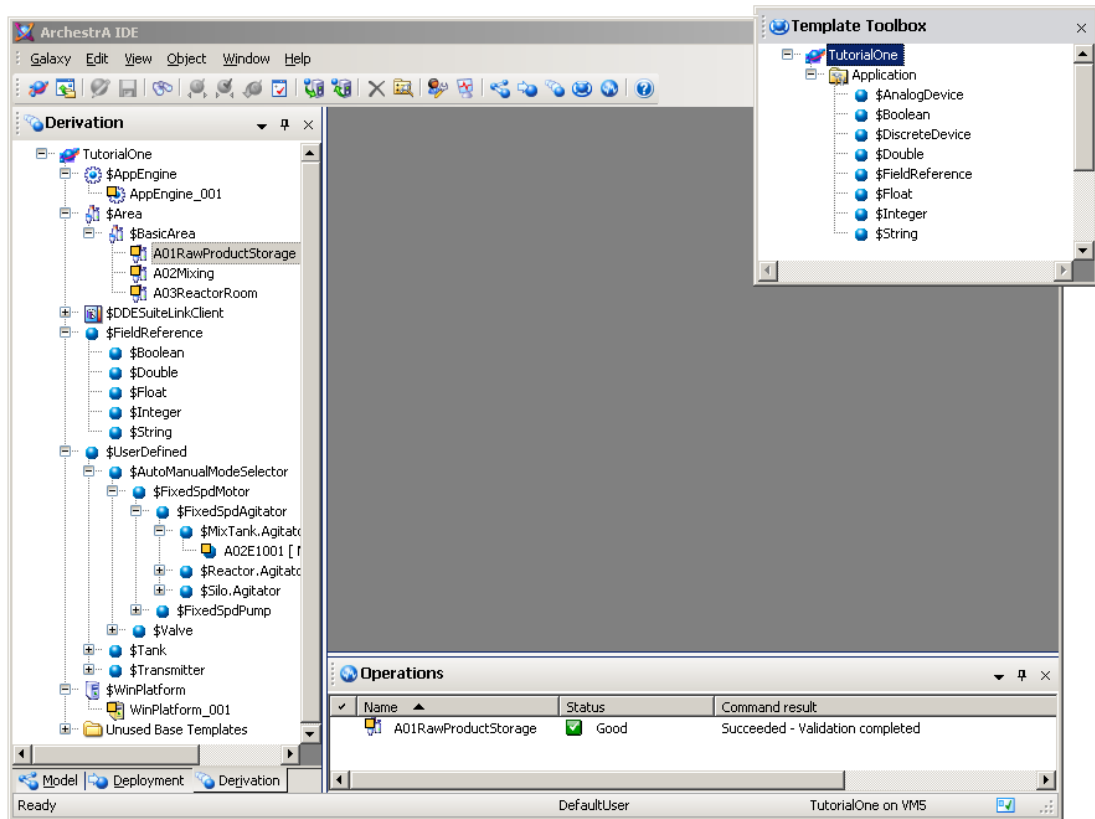
The **Operations view**, like the **Template Toolbox** and **Applications views**, is also updated as the status and conditions of objects in the Galaxy change.

## 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.

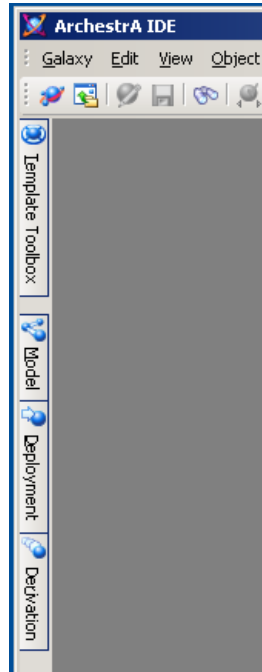


### Floating Views

- ▼ 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 workspace. This moves everything back to the 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 prompting for check-in comments and specifying user defaults. These options are for the currently open Galaxy and do not change any options for other Galaxies you use.

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 190.

### 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:
  - To be prompted to type comments when checking in templates and objects, select the **Ask for 'Check In' Comments** check box. This lets you provide text about the changes you made.
  - To see a prompt that tells you if you are opening an instance or template as read-only, select the **Warn before launching an editor for a read-only object** check box. This lets you know 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.

- To see a prompt that tells you if you have permission to create or modify InTouchView Applications, select the **Warn for insufficient permissions** check box. These permissions authorize or prevent you from creating and modifying InTouch View Applications.
  - To see a prompt each time you attempt to edit an InTouchView application instance, select the **Warn before launching an InTouchViewApp editor** check box. You can edit the associated template or cancel the operation. If you don't 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 your Galaxy* on page 131.
  - 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 **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.
- 7 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.

---

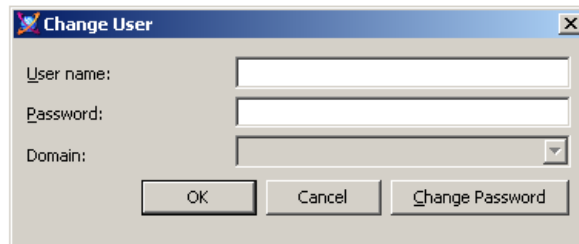
**Note** If you do not have logon rights, you cannot open a Galaxy.

---

For information about setting up security in the ArchestrA environment, see *Working with Security* on page 183.

### 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 machine. 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

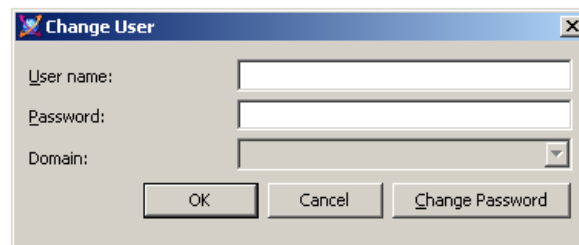
You can change users in a Galaxy at any time. Any security restrictions associated with a user change when the user logs on or logs off from the Galaxy. For more information about users and security, see *Configuring Security* on page 190.

If the Galaxy has not be 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 contains common configuration parameters for objects instances that you use multiple times in your application.

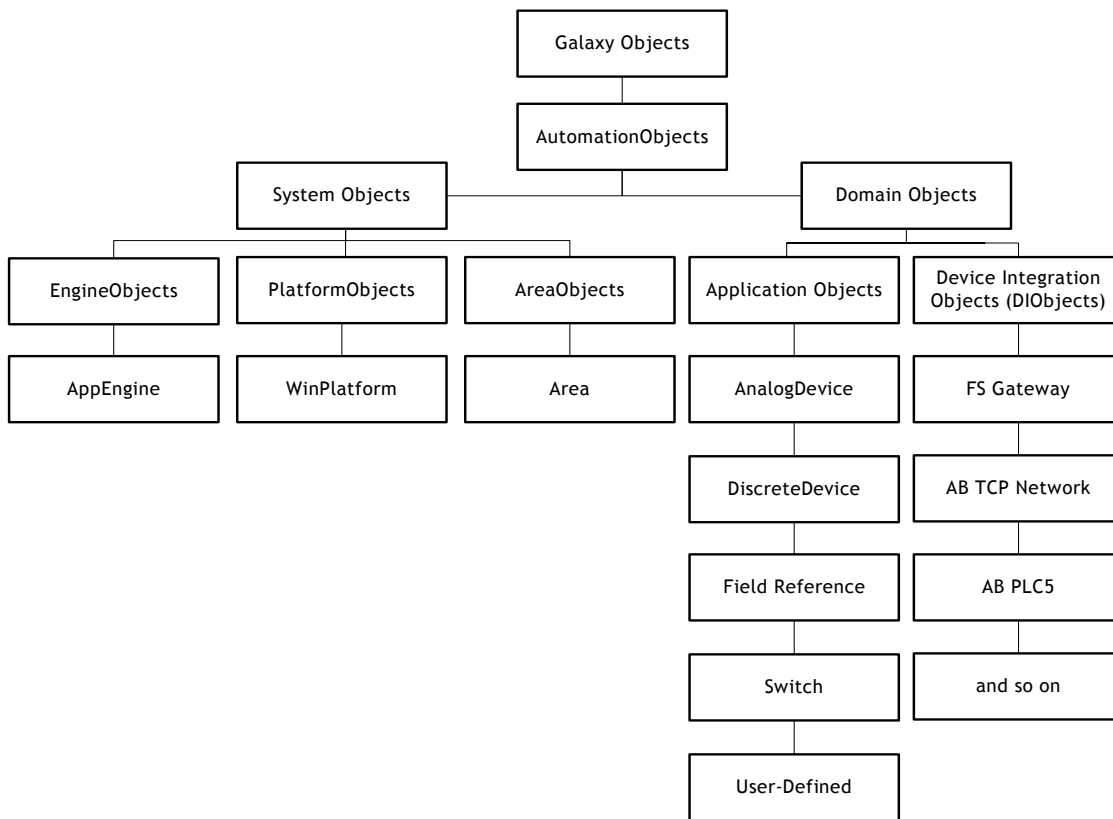
For example, you might create a template for valves. You configure the template with all the unique attributes for valves. You use that template to make object instances of valves. You can further configure and customize each object instance to represent a specific valve.

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.

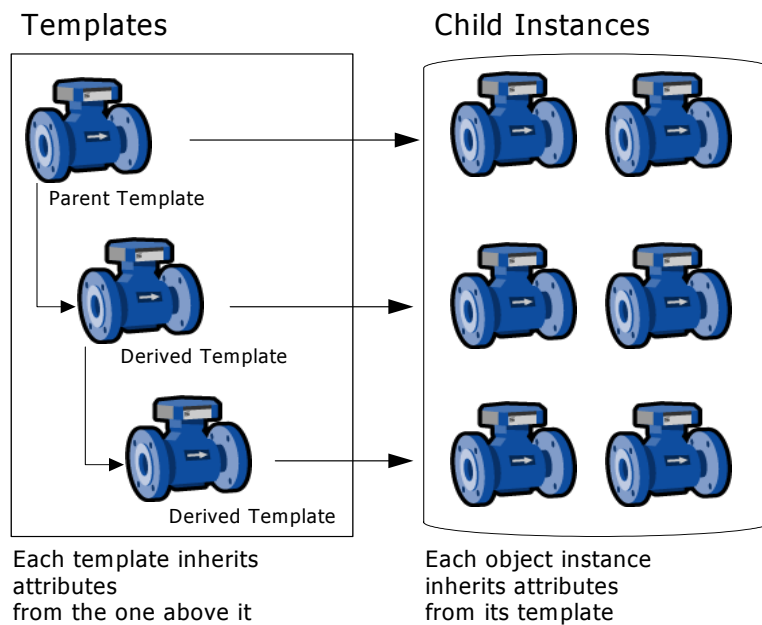
A template can also define an area of your environment. You can extend and customize a template by adding User Defined Attributes (UDAs), scripts, or extensions to meet the specific needs of your environment. Objects inherit attributes from their parents.

Application Server comes with predefined templates, called base templates. You cannot change these 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 an 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** area. The base templates are provided to build your own derived templates from. You cannot change the base templates. You create and modify your own derived templates and objects from the derived templates you create.

The template classes are as follows:

- **Application Templates** Use these templates to create 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 `DIOObject` 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, using UDAs, scripts, and extension, you can extend and customize any device you select.

### 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 an IAS system that represent the system itself 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 execute.



### Area Object

All ApplicationObjects 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.

This object allows the value of three attributes to be historized:

- 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 execute.

### 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 instances from.

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 so they cannot be changed in the child template.

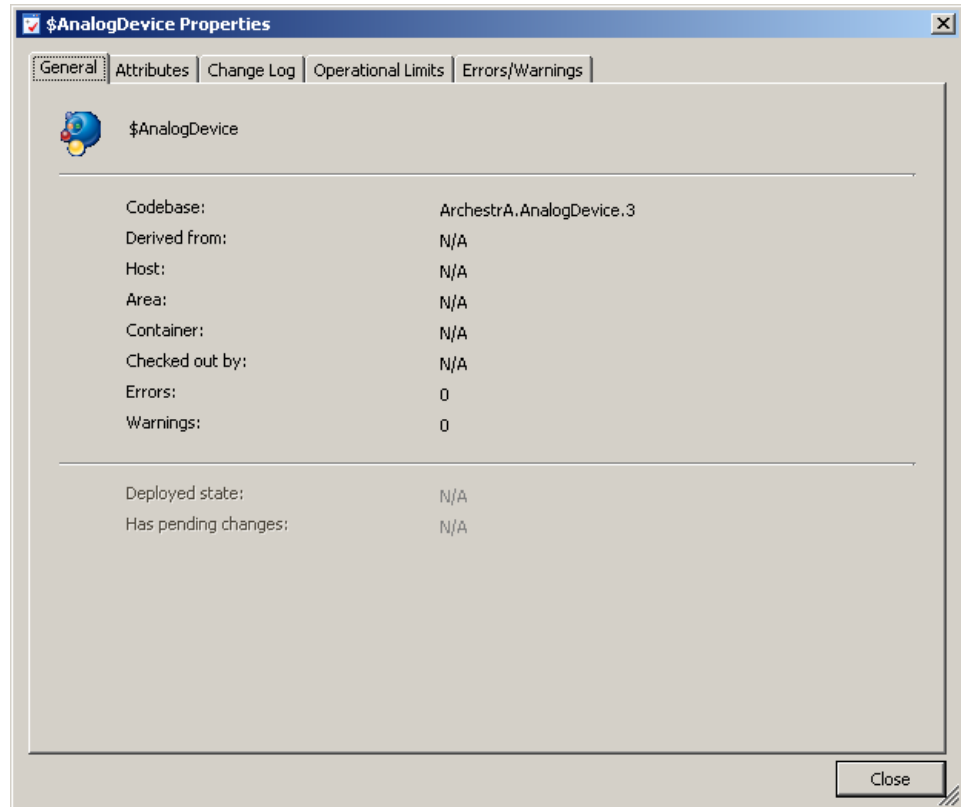
After you create a new derived template, you can customize it. For more information about customizing and extending templates, see [Creating Derived Templates](#) on page 48.

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 45.

## 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.



For more information about specifying the properties of objects, see [Working with Objects](#) on page 45.



# Chapter 3

## Working with Objects

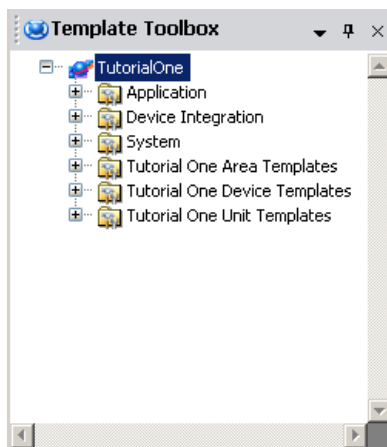
You can work with objects using the Application Server development environment.

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 37.

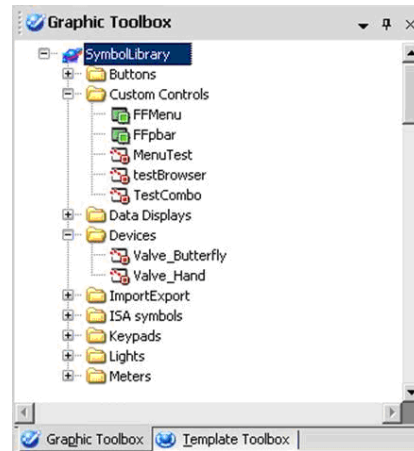
### Managing Toolsets

Toolsets are the high-level folders shown in the Application Toolbox. Toolsets can be created for templates and graphics for the purpose of organizing each. 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`. You can use up to 64 alphanumeric and special characters, including spaces and punctuation, 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.

### 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 and is in focus. Now, you can drag templates into the new 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 be deleted.

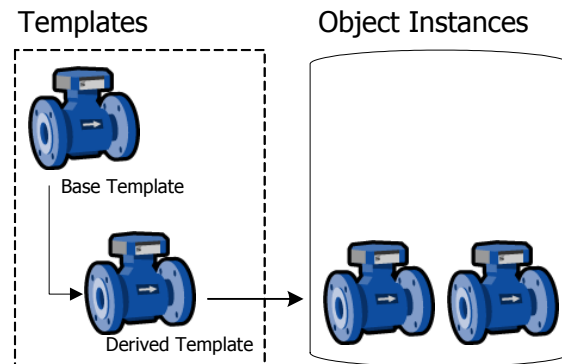
### To delete a toolset

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

## 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 the derived template, you can customize and modify the attributes in the template you created. If you change locked attributes in the parent template, the changes propagate to the derived template.



After you create derived templates, 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.

After you create a derived template, you can customize it.

### 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** You cannot use the following reserved names as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

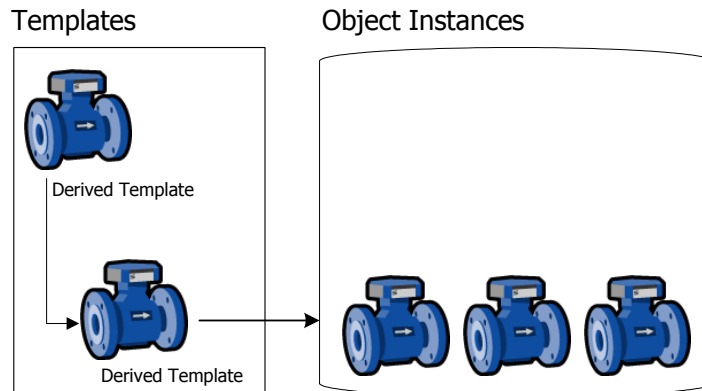
---

- 4 You are ready to customize your new template. For more information, see [Editing Objects](#) on page 61.

## 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** You cannot use the following reserved names as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

---

- 3 You can create another derived template by repeating the steps above, or you can customize your new derived template. For more information about customizing your template, see *Editing Objects* on page 61.

## 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.

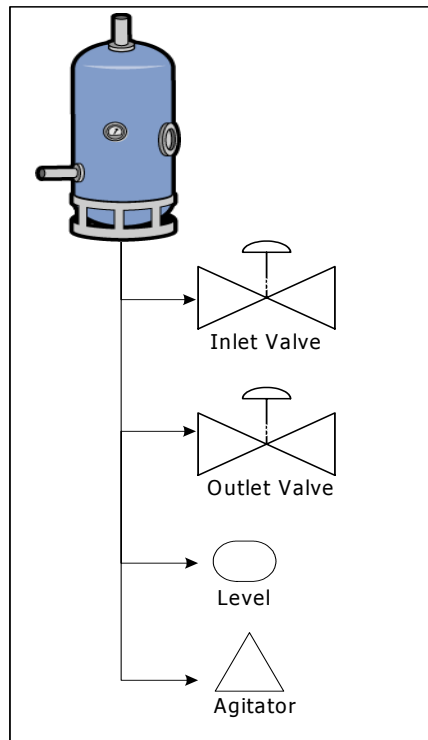
In scripts, these objects can be referred to by the name that derives from the containment relationship. This name is called a hierarchical name.

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

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

**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.

---

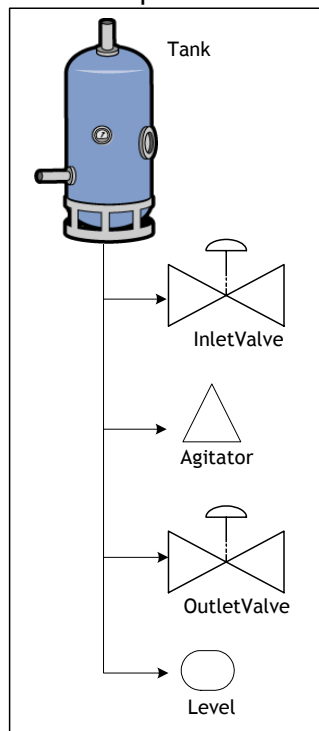
**Note** Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

---

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

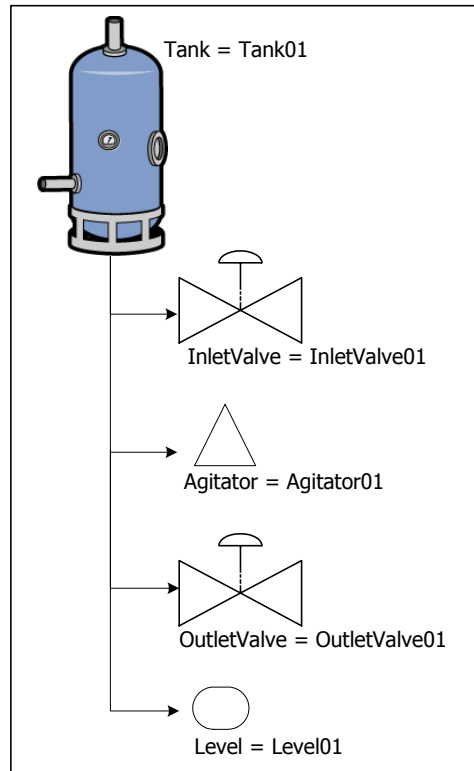
- Inlet Valve
- Agitator
- Outlet Valve
- Level

Tank Template



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

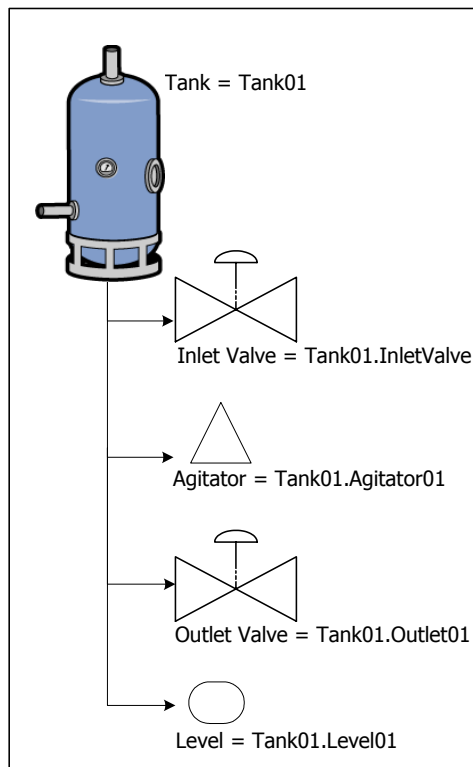
- Inlet Valve = InletValve01
- Agitator = Agitator01
- Outlet Valve = OutletValve01
- Level = Level01



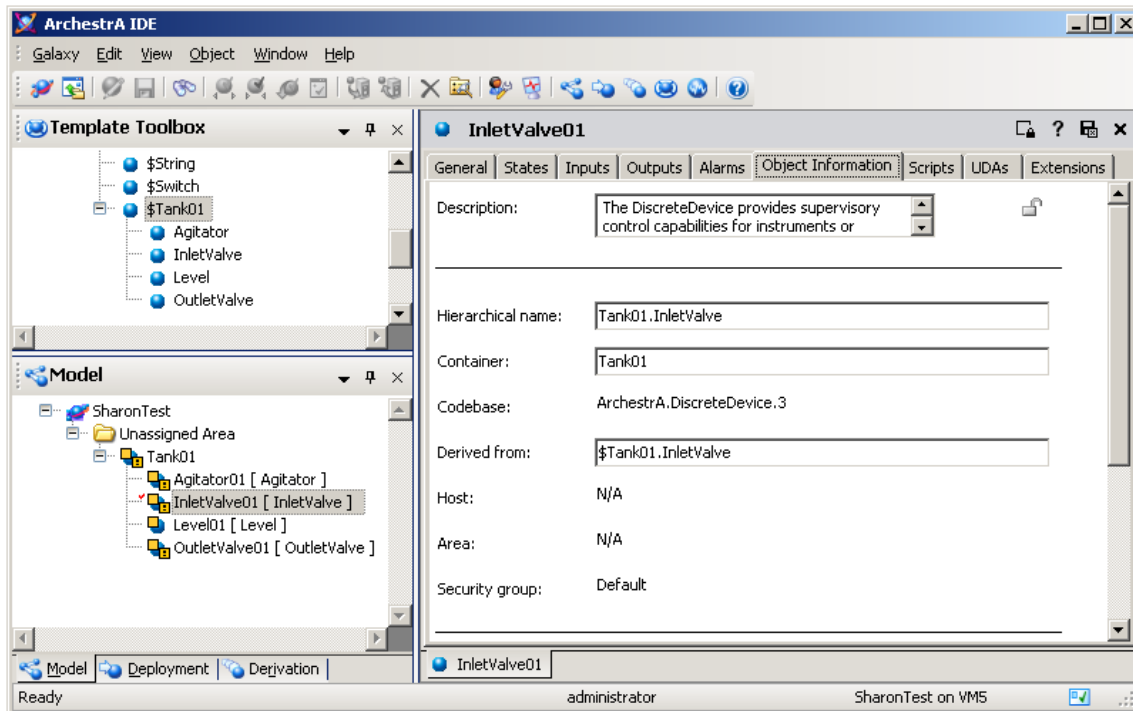
Within the context of each hierarchy, the contained names are unique, in that the names only refer to this tank system and the contained objects.

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

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

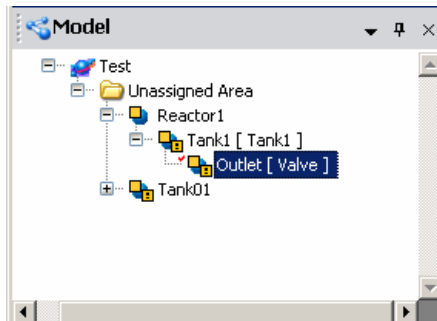
- 1 Create the following instances: Tank1 from \$UserDefined and Valve from \$DiscreteDevice. Valve has only one name, Valve.
- 2 In the **Model** or **Deployment view**, drag Valve on to Tank.

---

**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.

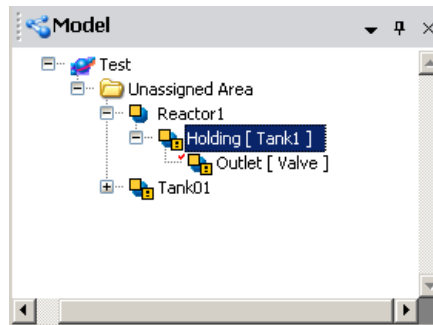
---

- 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.

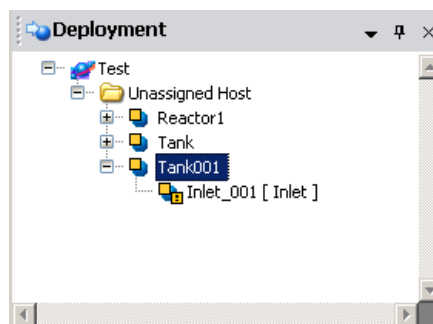
For the three objects in this example (Reactor1 containing Tank1 containing Valve1), the following naming hierarchy exists:



### 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.

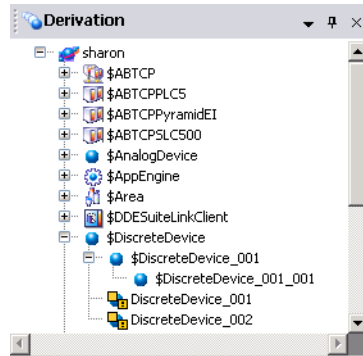
- 1 Create the following derived templates: \$Tank from \$UserDefined and \$Valve from \$DiscreteDevice.
- 2 Derive \$Inlet from \$Valve.
- 3 In the **Template Toolbox**, drag \$Inlet on to \$Tank. If \$Tank already contains a template named Inlet, the Galaxy generates a unique tagname for the new contained template, such as Inlet\_001. The contained template now has a hierarchical name \$Tank.Inlet.
- 4 Create an instance (Tank1) of \$Tank.
- 5 The **Model** and **Deployment** views show an instance Tank1 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

Using the Object Editor, you define attributes specific to the object.

The Object Editor shows object extension pages that are common to all objects and may also show you pages that are unique to the object. See *Enhancing Objects* on page 101 for more information about the **Scripts**, **UDAs**, and **Extensions** pages. Click the tab of each page to open that page.

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 it, you can open it to view but you cannot make changes.

When editing an object, you may see attribute text boxes showing a --- (dashdashdash). 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 ---.--- (dashdashdash dot dashdashdash). 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.

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.
- 3 Make your changes. For more information about locking attributes, see About the UDAs Page on page 74. For more information about setting security, see Setting Object Security on page 68.
- 4 When you are done 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

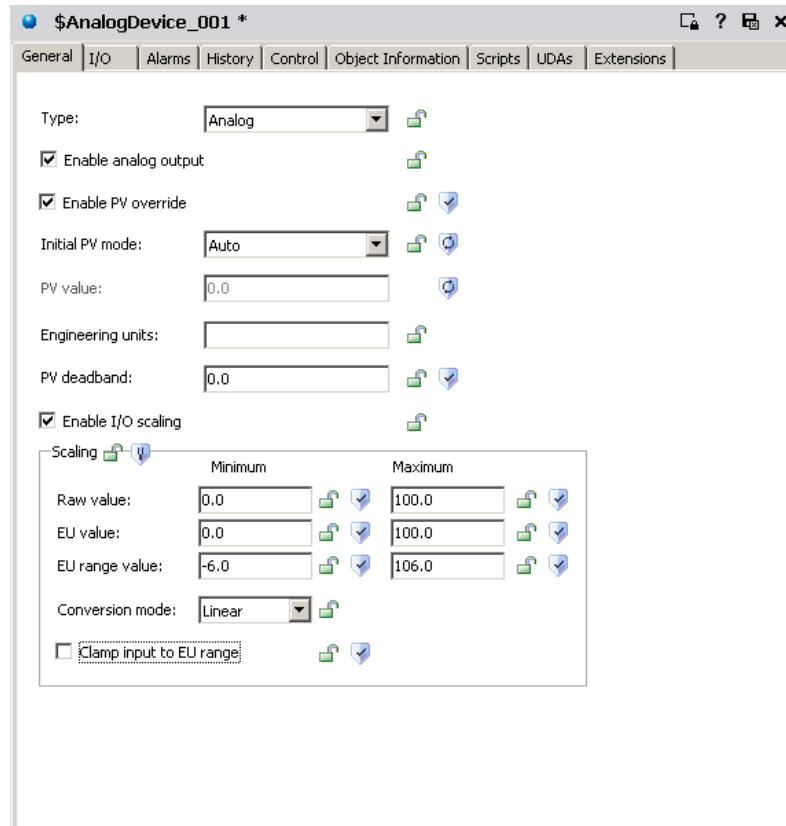
The header part of the Help file contains the following information:

- **Tagname**            The object's name.
- **Contained Name**   The object's contained name. For more information, see [Creating Contained Templates](#) on page 50.
- **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 is executed 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 screen below shows you 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, 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. Locking an attribute prevents the attribute 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.





You can reference the attributes:

- 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.”

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.





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)	<p>The associated attribute is locked (in me) and enabled. Only templates can have this kind of lock. The attribute value is read/write.</p> <p>Derived templates and instances do not have a unique copy of this attribute. Child objects share the locked attribute of the parent.</p> <p>Changing the value of a locked attribute in the parent template updates the value of that attribute in all derived templates and instances.</p>




Icon	Name	Description
	Locked (in parent)	<p>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.</p> <p>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.</p>
	Unlocked	<p>The associated attribute is unlocked and enabled. Both templates and instances can have this kind of lock. The attribute is read/write.</p> <p>The object has its own copy of the attribute value and the value is not shared by derived objects.</p>
	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 a UDA during configuration makes its value a constant. You cannot write to locked UDAs during run time.

**To lock an attribute example**

- 1 Create a derived template from the \$Discrete Device base template. Name the derived template \$Valve.
  -  2 Edit the \$Valve template and set an attribute value. Lock the attribute by clicking the **Lock** icon for the attribute.
  - 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 runtime and can have an associated security control, which is used to modify its run-time security classification.

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. For example, 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 before writing to the attribute. You also need to have Operate permissions 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 quickly set those conditions for all options in the group.

The group control typically reflects the setting for all options in the group. 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.

## About the Object Information Page

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

The screenshot shows a web-based configuration interface for an object named '\$AnalogDevice\_001'. The interface has a tabbed menu at the top with the following tabs: General, I/O, Alarms, History, Control, Object Information (selected), Scripts, UDAs, and Extensions. The main content area is divided into several sections:

- Description:** A text field containing 'The AnalogDevice provides supervisory control capabilities for instruments or' with a lock icon on the right.
- Hierarchical name:** A text field containing '\$AnalogDevice\_001'.
- Container:** A text field containing 'N/A'.
- Codebase:** A text field containing 'ArchestraA.AnalogDevice.3'.
- Derived from:** A text field containing '\$AnalogDevice'.
- Host:** A text field containing 'N/A'.
- Area:** A text field containing 'N/A'.
- Security group:** A text field containing 'Default'.

Below these fields is an 'Execution order' section with a lock icon:

- Process order:** A dropdown menu set to 'None' with a lock icon on the right.
- Relative object:** A text field with a search icon (three dots) and a lock icon on the right.

At the bottom of the page, there is a text label 'Click the button to add help for this object.' and a button labeled 'Add Object Help'.

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 execute 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 183.
- **Execution Order:** If you want this object to be executed 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 78.
- **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 editor like Microsoft FrontPage.

If clicking **Add Object Help** opens Word on your computer, change the program associated with editing HTML 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:

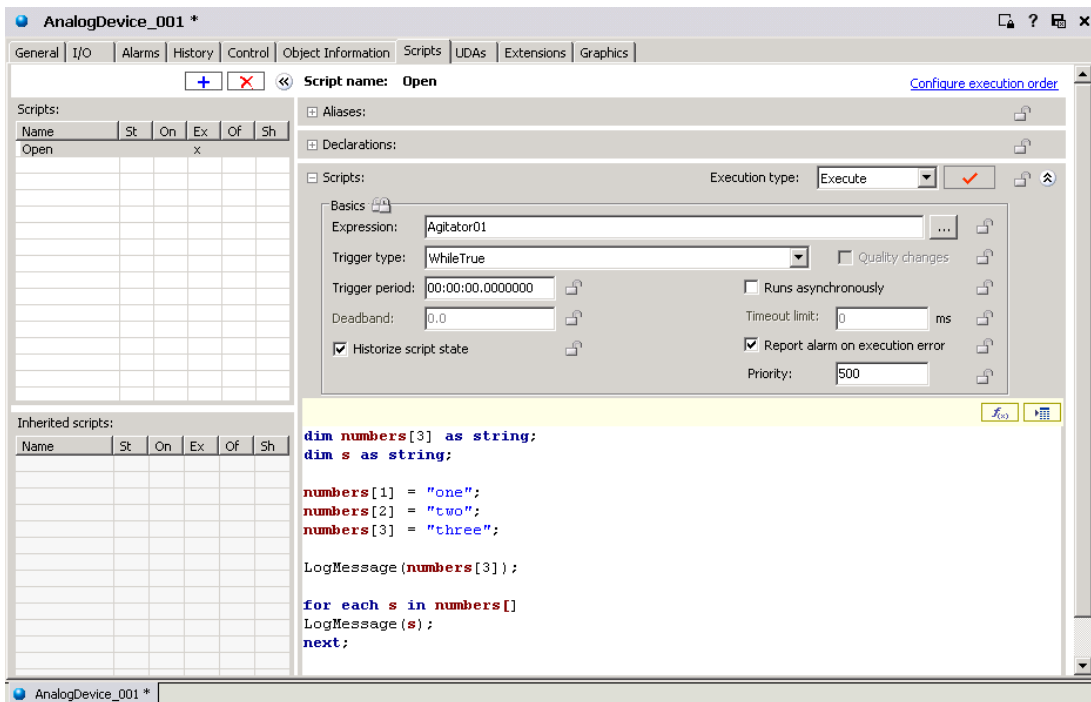
```
<Installation Path> is \<Program Files\Archestra\.
```

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 example above, place images in the \1033 folder or create an images folder under it.

## About the Scripts Page

The **Scripts** page has five areas. To learn more about using scripts, see [Writing and Editing Scripts](#) on page 105.



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.

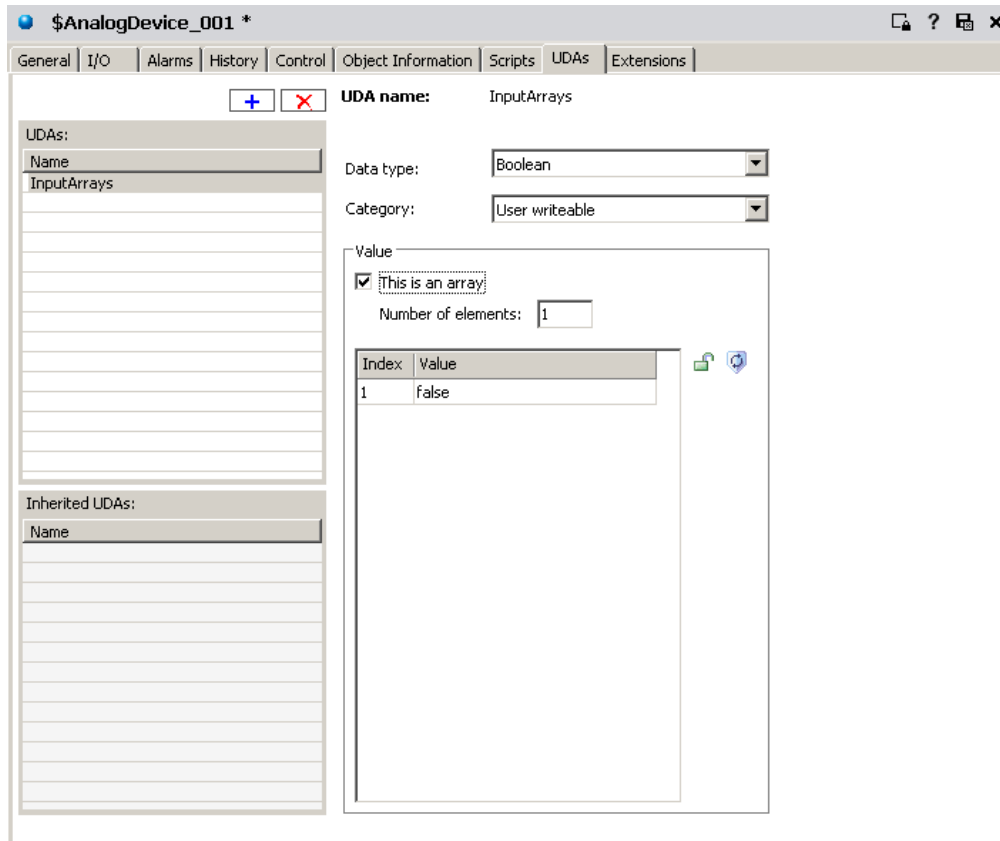




- **Declarations area:** Provides a place to add variable declaration statements, such as `DIM MyArray[1] as FLOAT;`. These declared variables live from the startup to the shutdown of the object and can be used to hold values that persist from one execution of the script to the next. They apply only 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 105 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 UDAs Page

The **UDAs** page has four areas. To learn more about creating UDAs, see [Creating and Working with UDAs](#) on page 102.



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

- +
**UDAs list:** Lists all UDAs currently associated with the object. Click the **Add** button to add a new UDA.
- Inherited UDAs list:** Lists all UDAs associated with the object's parent. The object automatically includes these UDAs. They can only be edited by modifying the parent template.
- Data type list:** Shows the data type options for configuring the selected UDA.

Select from the data types **Boolean**, **Integer**, **Float**, **Double**, **String**, **Time**, **ElapsedTime** or **InternationalizedString**. For more information about each data type and category, see the help file.

- Category list:** Shows the category options for configuring the selected UDA.

Allowed categories are:

**Calculated:** Permits only scripts within the same object to write to the attribute. Calculated attributes are not saved across restarts.

**Calculated retentive:** Permits only scripts within the same object to write to the attribute. Calculated retentive attributes are saved across restarts.

**Object writable:** 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, and they are Writeable\_S. 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, and they are Writeable\_USC\_Lockable and they can be locked at configuration time. This category is not user writeable.

---

**Note** You can lock writable attributes. If you select **Calculated** for an attribute, only scripts running on the same object can write to the attribute.

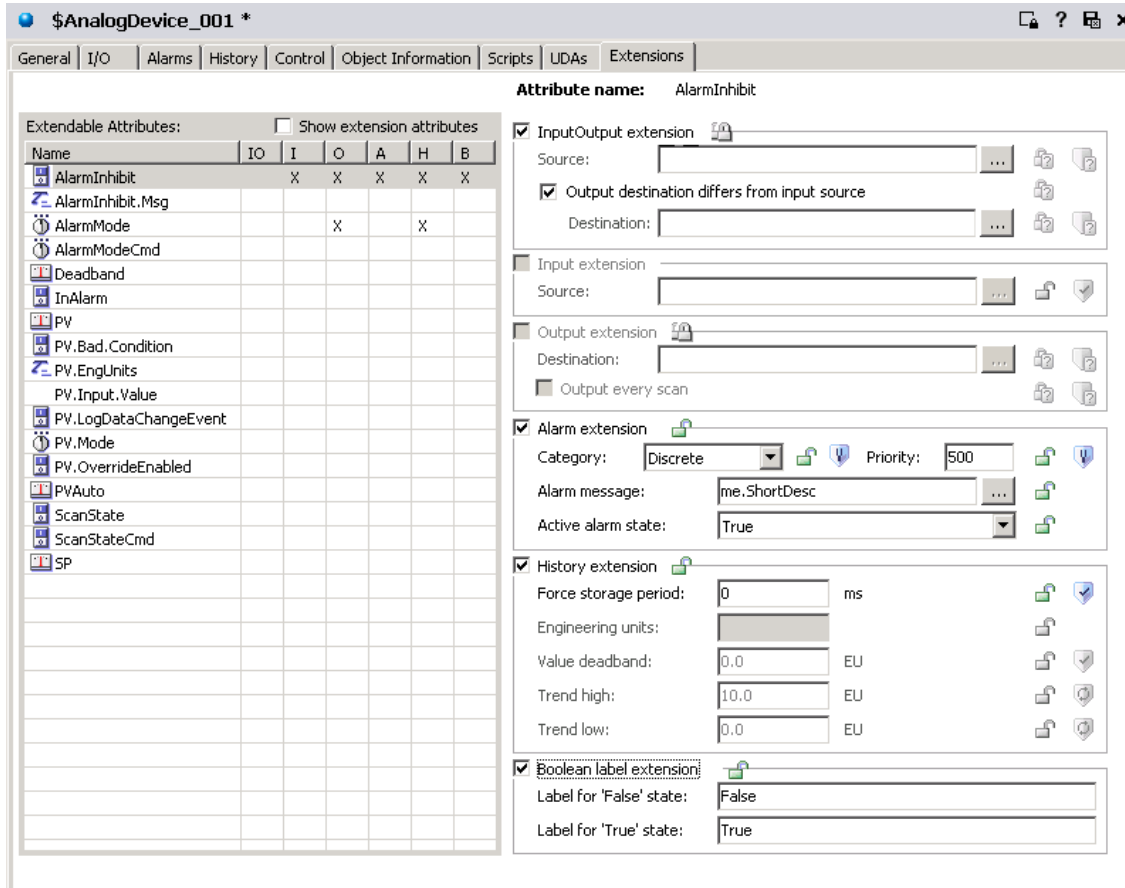
---

Select **This is an array** and specify the array's length in the **Number of elements** box. You can create an array for each data type except **InternationalizedString**.

The **Value** parameter specifies the initial setting for the attribute when the object is deployed. Enter value data for each data type. In the case of a non-arrayed **Boolean**, select the **True/False** check box to use a True value. Clear the check box to use a False value. For an arrayed Boolean, select the desired element and provide a default value by typing either `true` or `false`.

## About the Extensions Page

The **Extensions** page consists of seven areas. To learn more about creating extensions, see *Creating and Working with Extensions* on page 113.



The areas include:

- **Extendable Attributes list:** Lists all attributes currently associated with the object that can be extended. The list can include those added through the UDA tab. Select the **Show Extension Attributes** check box to include attributes added on the UDAs page.
- **InputOutput extension group:** Configure an attribute so that its value is both read from an external-reference source and written to an external-reference destination. The source and destination might not be the same. The extension reads the **Source** attribute's value and quality and updates the extended attribute's value and quality every scan. Changes read from the source are not written back to the **Destination** attribute.

- **Input extension** group: Configure the attribute to be readable for an external object. The extended attribute gets the update value of the **Source** attribute.

If you have a large I/O count, use InputOutput Extension instead of using Input Extension and Output Extension separately. Boolean attributes/UDAs that are extended as an InputOutput can handle momentary changes such as false-true-false transitions within a scan.

- **Output extension** group: Configure an attribute to be writeable to an external object. When the value or quality (from Bad or Initializing to Good or Uncertain) of the extended attribute is modified, the value of the **Destination** attribute is updated. The behavior of Boolean attributes/UDAs that are an extended as InputOutput can handle momentary changes such as false-true-false transitions within a scan.

- **Alarm extension** group: An alarm is triggered depending on the state defined in the **Active alarm state**.

When the alarm name contains more than 294 characters (<object name>, <attribute name>), InTouch will not raise an alarm even though the pv.limit and description are minimum length. For more information, see Working with References on page 167.

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 can be seen in the InTouch HMI.

If you specify custom labels for the **False** and **True** Boolean states in the **Boolean label extension** area, these custom strings appear in the **Alarm active state** list. These strings can also be used in the InTouch HMI.

Select the **Category** for this alarm. Specify a **Priority** for this alarm. Valid values are 0 to 999.

- **History extension** group: Historize the value of an attribute that does not already have history capabilities. These values are stored in the Wonderware historian.
- **Boolean label extension:** Specify custom text strings for the False state and the True state. These custom text strings appear in the **Active alarm state** list in the **Alarm extension** area for you to select. If you are using the InTouch HMI, you can see these custom text strings in InTouch.

## Referencing Objects Using the Galaxy Browser

Use the Galaxy Browser to browse for:

- Attributes of AutomationObjects. You can quickly find an object attribute or attribute property and add a reference to it when you are configuring an object.
- ArcestrA 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 AutomationObjects, 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 ArcestrA 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.

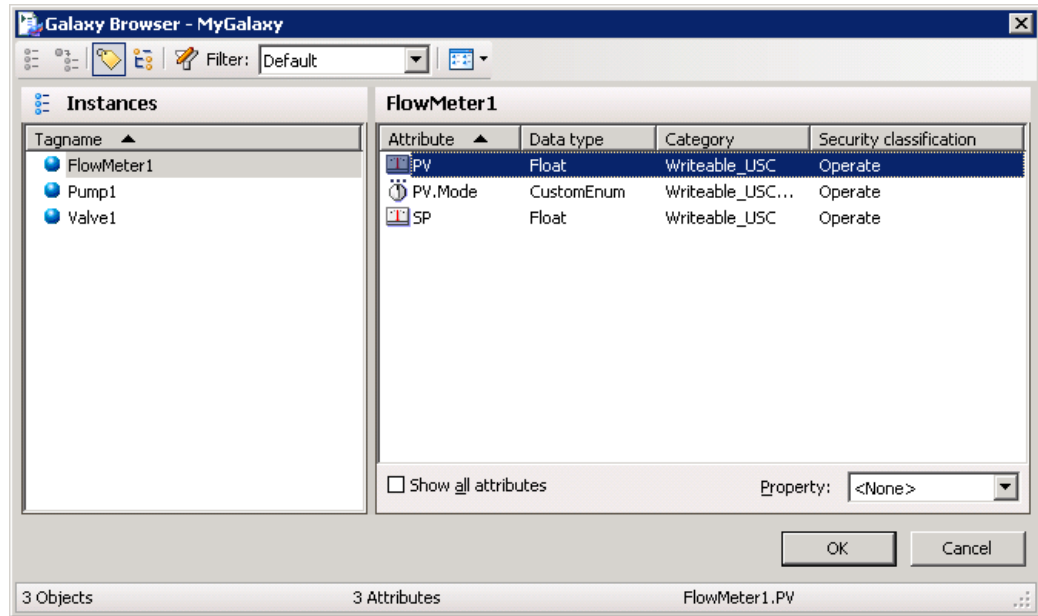
The Galaxy Browser shows objects in the left pane and the attributes associated with the current selection on the right pane. Only attributes that can be referenced at run time are shown. You can browse "Me." references for alarm messages only.

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 (Tagname 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.

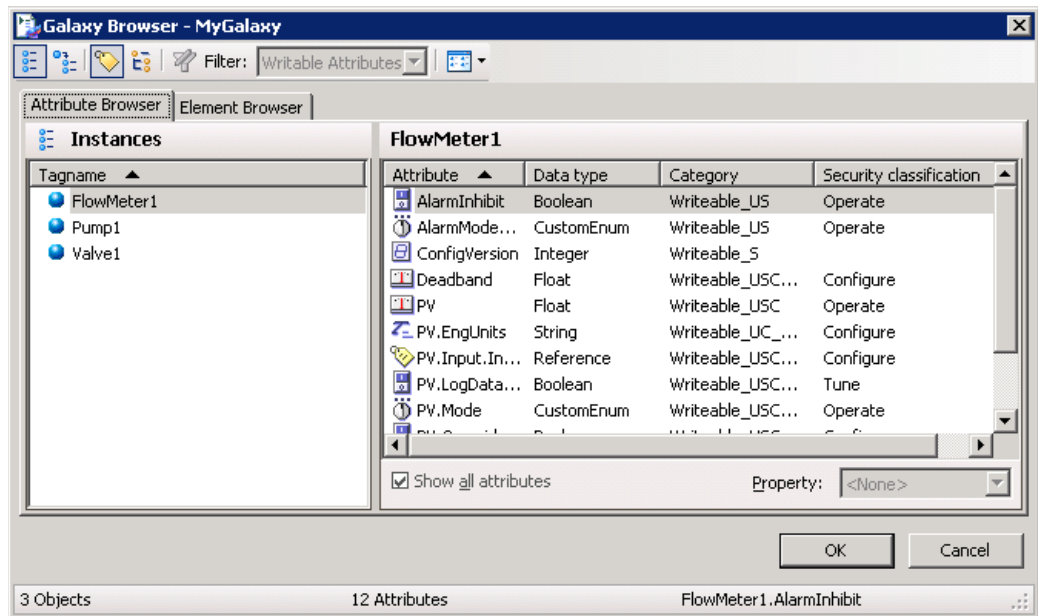
The status bar displays the attribute property name, and the it displays the graphic element attribute name and description.

### 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 86. 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.

### 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 223.

---



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. ArcestraA 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 ArcestraA 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 183.

Data Type	<p>The data type of the attribute:</p> <p>Integer          Boolean          Float          Double          String          Internationalized String          Time          ElapsedTime          ReferenceType          CategorizedStatusType          DataTypeEnum          SecurityClassificationEnum          DataQualityType          CustomEnum          CustomStruct</p> <p>For information about each of these, see the help for the object.</p>
Value	<p>The primary value of the attribute.</p> <p>Sometimes, a list of numbers is included in the <b>Property</b> list. Those numbers map to single bits in an integer attribute's Value property. Valid bit field specifiers are:</p> <p>.00 (least significant bit)</p> <p>.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</p> <p>.31 (most significant bit)</p>

---

**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 ArcestrA Symbol Editor, when editing a graphic from the Galaxy, being either a graphic from an AutomationObject (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.



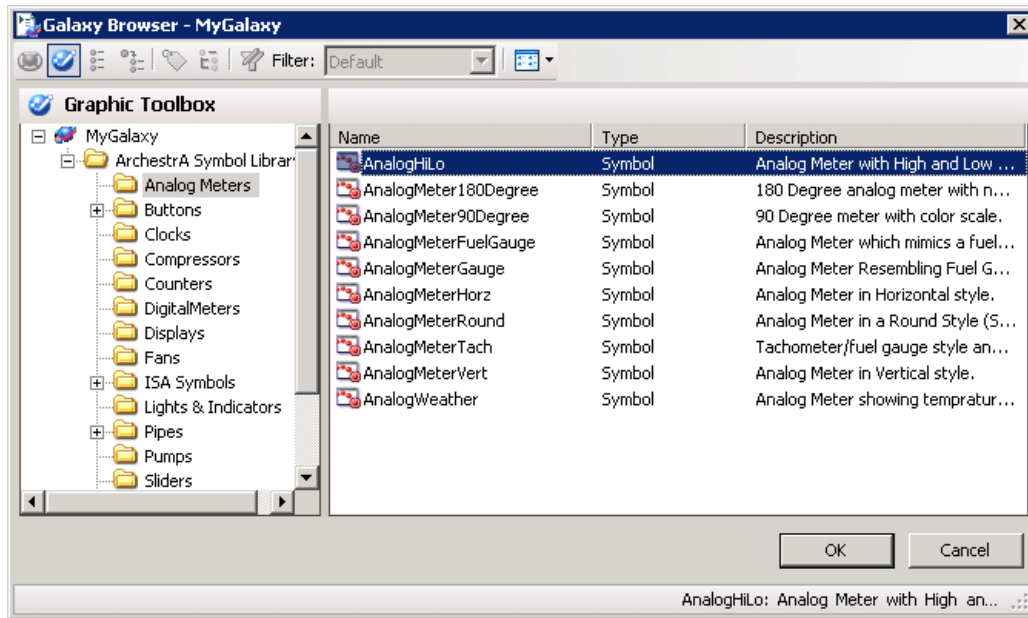
- AutomationObject instances. The browser shows a flat list of existing instances of AutomationObjects in the left pane. The right pane shows the graphics associated with the currently selected instance. You create or apply filters to reduce the scope of the instances shown in the left pane.



- Relative references. This is possible only when you edit a graphic belonging to an AutomationObject 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.

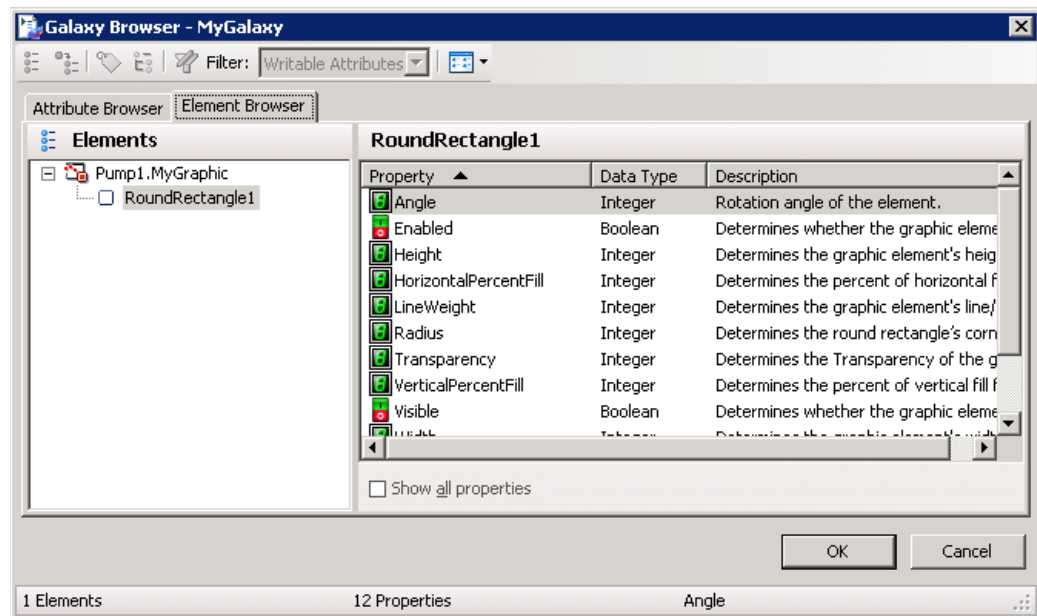
## Browsing for Element Properties

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 the *Creating and Using 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.

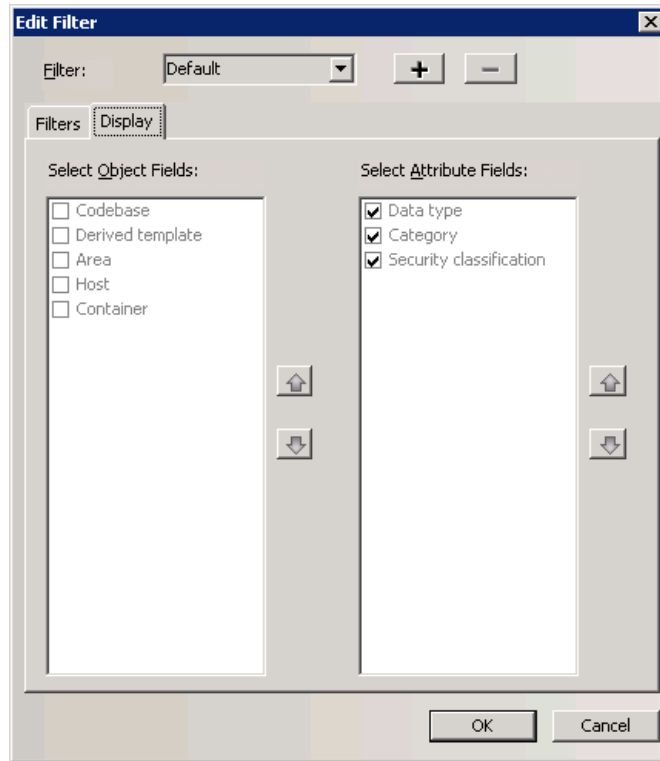
### To create a filter

- 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.

The screenshot shows the 'Edit Filter' dialog box. At the top, there is a 'Filter:' dropdown menu with 'NewFilter1' selected, and '+' and '-' buttons. Below this are two tabs: 'Filters' (selected) and 'Display'. The 'Filters' tab contains several rows of input fields and dropdown menus. The rows are: 'Tagname:', 'Hierarchical name:', 'Area:', 'Codebase:', 'Derived template:', 'Host:', 'Container:', 'Attribute:', 'Data type:', 'Security classification:', and 'Category:'. Each of the first eight rows has a text input field and a dropdown menu set to 'Contains'. The 'Data type:', 'Security classification:', and 'Category:' rows have only dropdown menus. At the bottom right are 'OK' and 'Cancel' buttons.

- 4 Configure the filter details.

- 5 Click the **Display** tab.



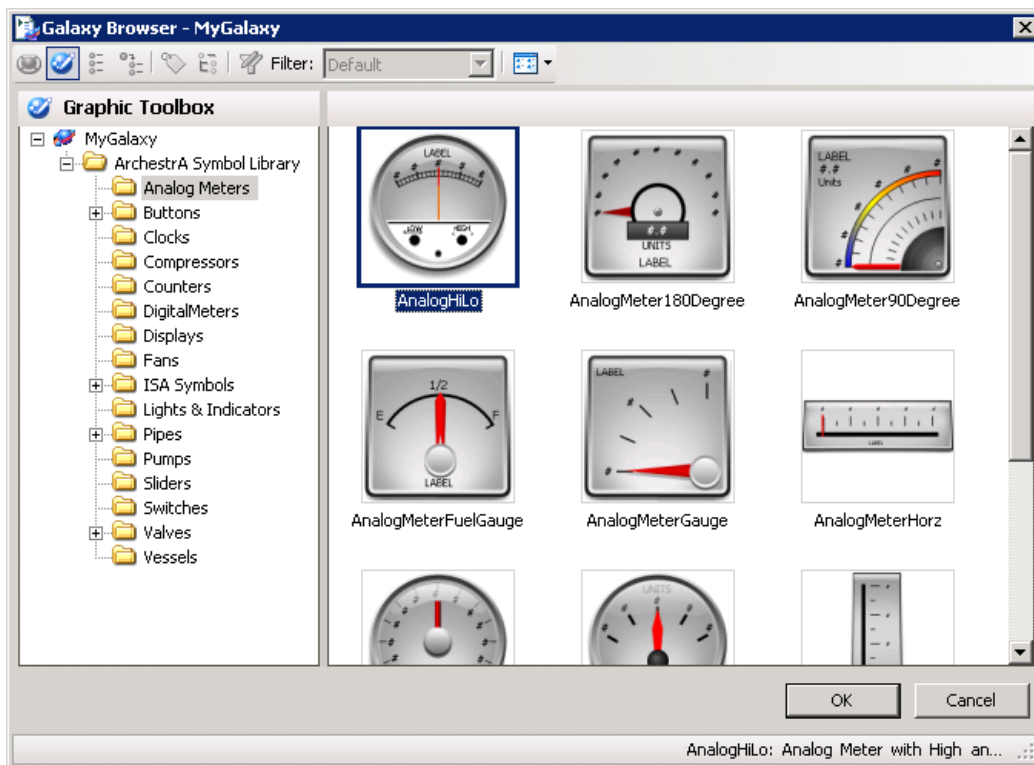
- 6 Configure the columns to show in the right pane of the **Galaxy Browser**. Use the up and down arrows to set the order for the columns.
- 7 Click **OK**. The new filter appears in the **Filter** list.

## 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 the object out 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 check objects out

- 1 In the **Template Toolbox** or **Application views**, select the object(s) 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 Objects In

When you are finished making your changes, you can 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 changes. For more information, see *Customizing Your Workspace* on page 29.

---

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

---

### To check an object in to the Galaxy database

- 1 In an **Application view** or the **Template Toolbox**, select the object after you are finished working with.
- 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 its configuration either when you are configuring it, 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 non-existing UDAs are not 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 execute, such as script function libraries and references to other objects' attributes. The status of these external components can change, perhaps rendering some capability of the object inoperative.

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 92.

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.

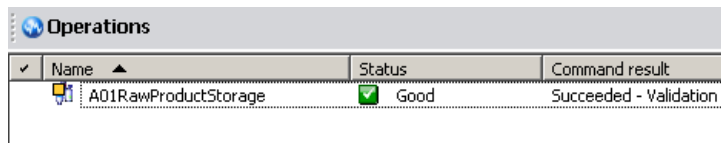
---

**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 object(s) you want to manually validate.
- 2 On the **Object** menu, click **Validate**. The **Operations** view in the IDE opens.



Name	Status	Command result
A01RawProductStorage	Good	Succeeded - Validation

- 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.

---

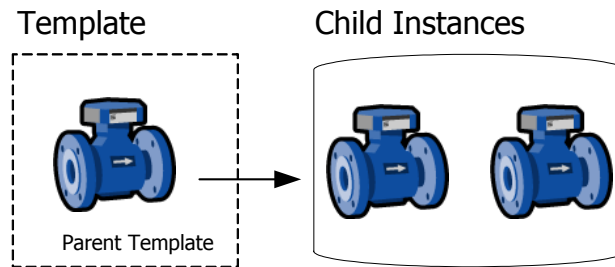
**Note** You can also see the errors or warnings that led to an object having a status that is not Good by looking at the Error/Warnings tab within the object's Properties.

---

## 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.

Once 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. You cannot use 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 61](#).

## 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 changes automatically. See Renaming Contained Objects on page 60 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.
- Instances 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, depending 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".

Name	Description
Hierarchical Name	<p>Hierarchical names that are fully-qualified names of a contained object include the name of the objects that contain it.</p> <p>Since the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object.</p> <p>For example, if:</p> <p>"Reactor1" contains Tank1 (also known within Reactor1 by its contained name "SurgeTank").</p> <p>"Tank1" contains Valve1 (also known within Tank1 by its contained name "Outlet").</p> <p>Valve1 could be referred to as:</p> <p>"Valve1"</p> <p>"Tank1.Outlet"</p> <p>"Reactor1.SurgeTank.Outlet".</p>

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 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 97. 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 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 Automation Objects**.
- 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 a .aaPKG extension.
- 5 When the export is complete, click **Close**. Now you can import the .aaPKG file into another existing Galaxy.

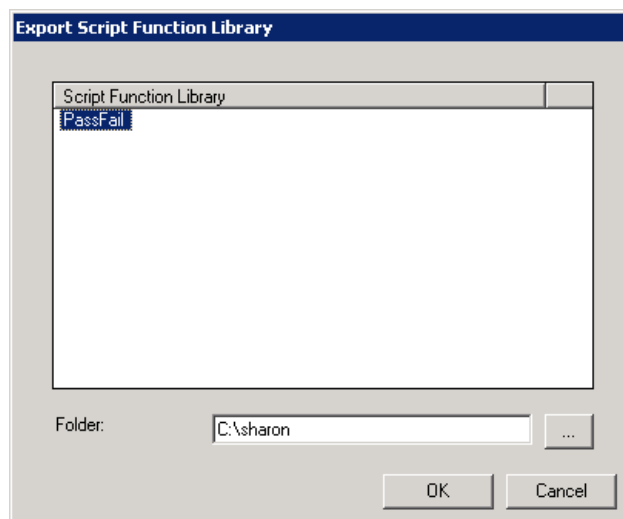
## Exporting Script Function Libraries

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 a `.aaSLIB` 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.

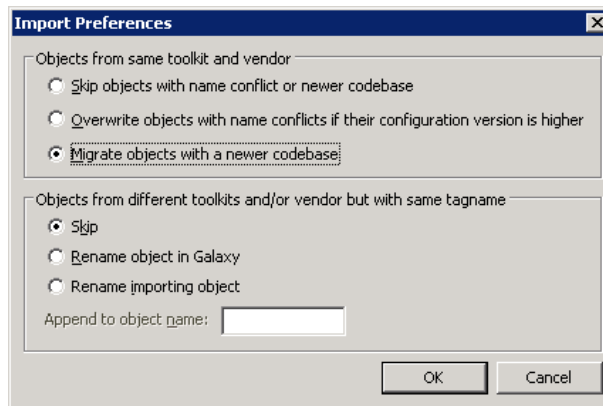
Importing instances previously exported from a Galaxy retains previous associations, when possible, such as assignment, containment, derivation, and area.

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's created by a developer using the Archestra Object Toolkit.

Before you start, 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, these conflicts are identified and you can manage them.

### 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 a `.aaPKG` or a `.aaPDF` extension. You can select more than one file. Click **Open**. The **Import Preferences** dialog box appears.



- 3 In the **Objects from same toolkit and vendor** area, select one of the following:

**Skip objects with same conflict or newer codebase**

leaves the existing object unchanged.

**Overwrite objects with name conflicts if their configuration version is higher** replaces the existing object with the object being imported if the codebases are the same and the imported object has been edited more often than the existing object.

**Migrate objects with a newer codebase** also overwrites existing objects if the codebases (versions) are the same. This option also migrates the state of existing objects to the newly imported version if the object supports it.

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 run time to the Galaxy database. For more information, see Uploading Run-time Configuration on page 140.

For more information about migrating, see the Application Server Help.

- 4 In the **Objects from different toolkits and/or vendor but with same tagname** area, select one of the following:

**Skip** does not import the object when a name match exists in the Galaxy.

**Rename Object in Galaxy** renames the existing object by appending to its current name the string (up to four characters) you type in the **Append to Object Name** box. The default value is `_old` but you can change it to any four-character string.

**Rename Importing Object** renames the object being imported by appending to its current name the string (up to four characters) you type in the **Append to Object Name** box. The default value is `_new`, but you can change it to any four-character string.

---

**Note** Object name conflict resolution only applies to templates and instances derived from different base templates.

---

- 5 Click **OK**. The import process starts.
- 6 When the import process is complete, you can start using the objects you imported.

## Importing Scripts 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 90.

## 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 objects with a newer codebase** 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.

---

**Note** 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.

---

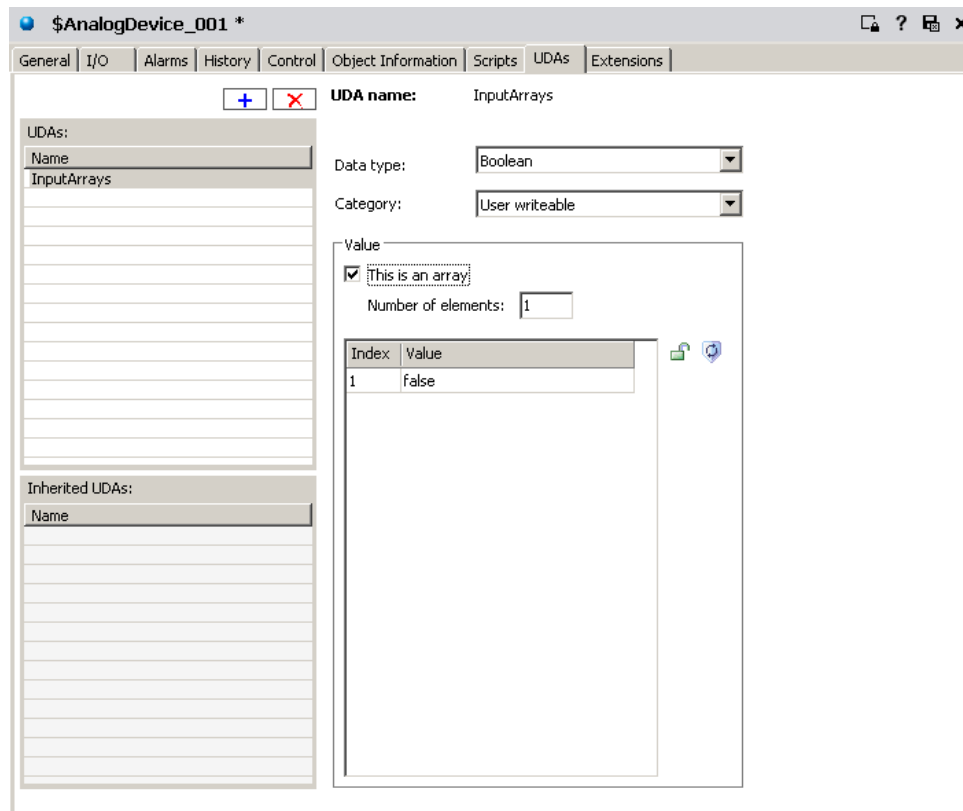
# Chapter 5

## Enhancing Objects

After you create an object, you can enhance and extend the object by using User Defined Attributes (UDAs), scripts, and graphics extensions.

## Creating and Working with UDAs

You can add UDAs to a template or an instance. When you add a UDA to a template, the UDA, its data type, and category are automatically locked in the child instances. For an overview of the UDAs page, see About the UDAs Page on page 74.



If UDA 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 extension functions. For more information about using the Attribute Browser, see Referencing Objects Using the Galaxy Browser on page 78.

In the **UDAs** page, you can:

- Add a new attribute to an object.
- Configure its data type.
- Specify the attribute category.
- Set initial values and locks on the new attribute.
- Set whether the new attribute is an array and how many elements are in the array.

## UDAs and Scripting

When using UDAs in scripting, keep the following list in mind.

- If you use **Calculated** and **Calculated retentive** UDAs as counters, they must be manually initialized. For example, if you use `me.UDA=me.UDA+1` as a counter in a script, you must also initialize the UDA with something like `me.UDA=1` or `me.UDA=<some attribute value>`.
- **Calculated** UDAs 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** UDAs in Startup scripts and you can initialize these UDAs in On Scan and Execute scripts. A **Calculated retentive** UDA 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 UDA. If the value is `FALSE`, initialize the UDA. For more information about `StartingFromCheckpoint`, see the help for the AppEngine object.

## UDA Naming Conventions

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


A UDA 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**.

---

**Important** After creating a UDA, it is available, like all attributes, when extending the object further on the **Extensions** page. If you extend a user defined attribute and then delete or rename the user defined attribute, all object extensions added to the object on the **Extensions** page are lost.

---

### To create and associate a UDA with an object

-  1 On the **UDAs** page of the Object Editor, click the **Add** button. A UDA is added to the **UDAs** list.
- 2 Type the new UDA name.
- 3 In the **Data type** list, select the **Data type** for the new attribute. The available options in the **Data type** list 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 **UDAs** page, see *About the UDAs Page* on page 74.

---

- 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 68.
- 7 Save and close the Object Editor when you are done.



## Writing and Editing Scripts

Scripts let you extend and customize your objects. A script lets you runs 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 executes 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 startup/shutdown state

A script is typically executed based on attributes of the object that contains it, but it can be executed by another script based on changing values of attributes of more than one object.

When the script condition is true, the script will execute at least one time immediately. The trigger period has a maximum of 49-days. If the trigger period is set to greater than 49-days, the script will never execute.

For specific information about writing scripts, including the scripting language, syntax, commands, and using .NET, see the *Wonderware® Application Server Scripting Guide*.

The screenshot shows the configuration window for a script named 'Open' on the 'AnalogDevice\_001' object. The 'Scripts' tab is selected, and the 'Open' script is highlighted in the table. The configuration panel for this script is visible, showing the following settings:

- Expression: Agitator01
- Trigger type: WhileTrue
- Trigger period: 00:00:00.0000000
- Deadband: 0.0
- Execution type: Execute
- Quality changes:
- Runs asynchronously:
- Timeout limit: 0 ms
- Report alarm on execution error:
- Priority: 500
- Historize script state:

The script code is displayed in a text area below the configuration panel:

```

dim numbers[3] as string;
dim s as string;

numbers[1] = "one";
numbers[2] = "two";
numbers[3] = "three";

LogMessage (numbers [3] );

for each s in numbers[]
LogMessage (s) ;
next;

```

For more information about the scripts page, see About the Scripts Page on page 72.

---

**Important** You cannot pass UDAs as parameters for system objects. Instead, use a local variable as an intermediary or explicitly convert the UDA 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 the errors are resolved.
- Scripts can be validated before you use 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 executed 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 can be executed in one of two ways:

- 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 72.

---

- 3 Select a trigger for executing 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 up, 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 is executed. 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	Script is executed based on a time interval specified in the <b>Trigger Period</b> box. A time interval of zero (0) executes the script during every scan. This trigger does not require an expression.
While True	<p>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 executes if true and then periodically thereafter at the trigger interval.</p> <p>The script is executed as long as the <b>Expression</b> 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.</p>
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 is executed 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 is executed at the transition between the expression going from true to false.

Trigger Type	Description
Data Change	<p>Script is executed 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 <b>Quality changes</b> check box.</p> <p>Deadband can be specified for all 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.</p> <p>After the object is put on scan, Data Change-triggered scripts are executed on the AppEngine’s next scan period and then on each subsequent scan period in which the value or quality changes.</p>
While False	<p>When the object containing the script is going On Scan, a While False script evaluates its expression at the next scheduled scan period, executes if false and then periodically thereafter at the trigger interval.</p> <p>The script is executed as long as the <b>Expression</b> value evaluates to be false. A <b>Trigger Period</b> 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.</p>

- 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 up 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 the *Application Server Scripting Guide*.



Click the **Validate Script** button to validate your script syntax.

- 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 doesn't impact the 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 that 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.

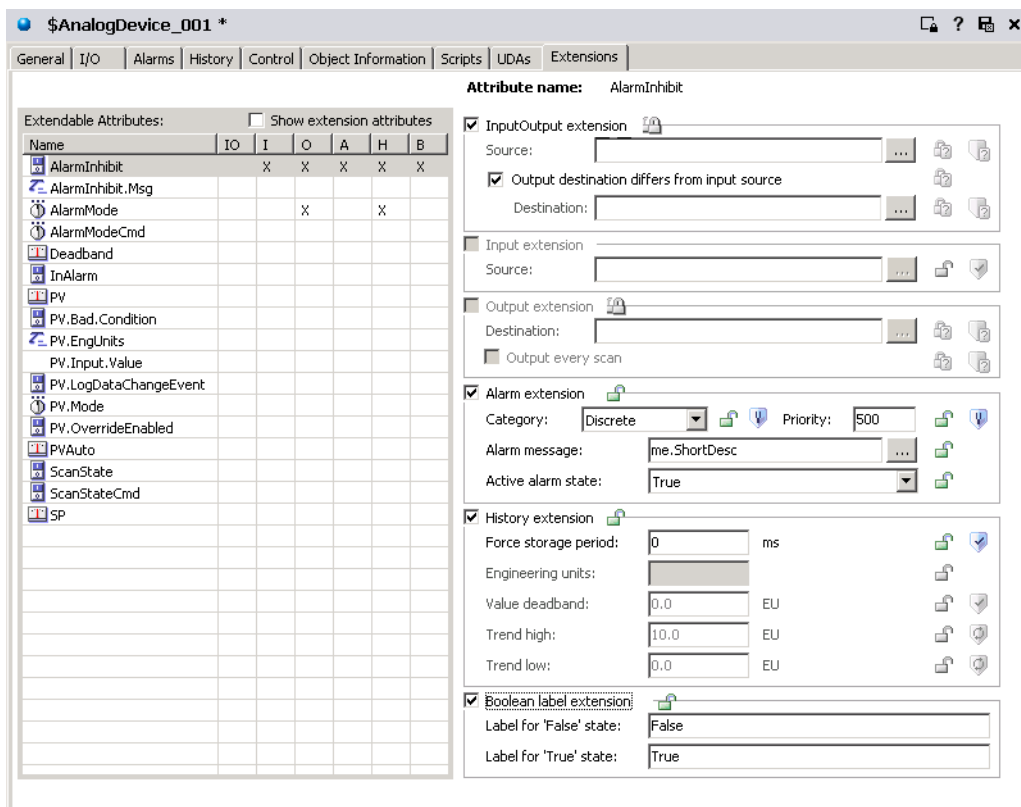
- If an attribute is not locked in a template, then all templates and instances derived from that template receive their own copy of the value of that locked attribute. A change to that unlocked value is allowed in derived templates and instances because they own their own copy. Any change to the unlocked 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 Extensions

The **Extensions** page allows you to configure an existing attribute for input, output, alarm, and history functionality not embedded in the original object.



## About Extension Inheritance

You can add object extensions to either derived templates or instances. Base templates cannot be extended. The following parent-child object characteristics also apply to object extensions:

- If you add an extension to a derived template that has objects derived from it, all child objects inherit the extension.
- You cannot add an extension to derived objects that duplicate parent object extensions in name and type.
- You cannot add an extension with the same name as an existing attribute extension.
- Renaming an extension 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 a new extension with the same name as an existing attribute in a derived object. The template definition of the extension overrides the extension in the derived object.
- If you remove an extension from a template, that extension is removed from any child object. You see the change when you check the template in.

#### To create and associate an extension with an object

- 1 On the **Extensions** page, select an attribute from the **Extendable Attributes List**. The extension groups dynamically change to allowed extension rules for the selected attribute type.
- 2 Select the check box for the kind of extension you want to apply to the selected attribute. The associated parameters for each kind of extension become available. For detailed information about each item on the **Extensions** page, see *About the Extensions Page* on page 76.
- 3 Select the parameters you want. Do the following:
  - For **InputOutput extension**, enter a **Source** attribute by either typing in the reference string or by using the **Attribute Browser** to search for the reference string in an object. For specific information about using this extension, see *Using the InputOutput Extension* on page 116.

If **Destination** is different from **Source**, click **Output destination differs from input source**. Enter a **Destination** attribute by either typing in the reference string or clicking the **Attribute Browser** button. An **X** appears in the **IO** column of the selected attribute.

---

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

---

- For **Input extension**, enter a **Source** attribute by either typing in the reference string or clicking the attribute browser button at the right. Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. An **X** appears in the **I** column of the selected attribute. For more information about using the Attribute Browser, see Referencing Objects Using the Galaxy Browser on page 78.
- For **Output extension**, enter a **Destination** attribute by either typing in the reference string or clicking the attribute browser button at the right. Use the **Attribute Browser** to search for a reference string in an object. For specific information about using this extension, see Using the Output Extension on page 120.

Select the **Output Every Scan** check box if you want the extended 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. An **X** appears in the **O** column of the selected attribute.

- For **Alarm extension**, select a **Category** from the list: **Discrete, Value LoLo, Value Lo, Value Hi, Value HiHi, DeviationMinor, DeviationMajor, ROC Lo, ROC Hi, SPC, Process, System, Batch** or **Software**. For specific information about using this extension, see Using the Alarm Extension on page 124.

Type a **Priority** level for the alarm (default is 500). An **X** appears in the **A** column of the selected attribute.

From the **Active alarms state** list, select the value that triggers that alarm. The items in this list can be customized in the **Boolean label extension** area.

- For **History extension**, enter values for the remaining parameters: **Force Storage Period, Engineering Units, Value Deadband, Trend High and Trend Low**, if available (depends on the data type of the selected attribute). An **X** appears in the **H** column of the selected attribute. For specific information about using this extension, see Using the History Extension on page 125.
- For **Boolean label extension**, specify different text strings for the **Label for 'False' state** and the **Label for 'True' state**, if needed. These text strings appear in the **Active Alarm State** list for you to select.

- 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 68.
- 6 Save and close the Object Editor to include the new attribute extensions in the configured object.

## Using the InputOutput Extension

InputOutput extensions allow 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 InputOutput extension monitors the value/quality of an input and sends outputs on state change.

The screenshot shows the Object Editor for the attribute `AlarmInhibit`. The **Extensions** tab is active, displaying a list of extendable attributes on the left and configuration options for the selected attribute on the right. The **InputOutput extension** is highlighted with a red box.

Extendable Attributes:							
Name	IO	I	O	A	H	B	
AlarmInhibit		X	X	X	X	X	X
AlarmInhibit.Msg							
AlarmMode			X		X		
AlarmModeCmd							
Deadband							
InAlarm							
PV							
PV.Bad.Condition							
PV.EngUnits							
PV.Input.Value							
PV.LogDataChangeEvent							
PV.Mode							
PV.OverrideEnabled							
PVAuto							
ScanState							
ScanStateCmd							
SP							

**Attribute name:** AlarmInhibit

**InputOutput extension**

Source:  ...

**Output destination differs from input source**

Destination:  ...

**Input extension**

Source:  ...

**Output extension**

Destination:  ...

**Output every scan**

**Alarm extension**

Category: **Discrete** Priority: **500**

Alarm message: **me.ShortDesc**

Active alarm state: **True**

**History extension**

Force storage period: **0** ms

Engineering units:

Value deadband: **0.0** EU

Trend high: **10.0** EU

Trend low: **0.0** EU

**Boolean label extension**

Label for 'False' state: **False**

Label for 'True' state: **True**

The output destination can be the same or different from the input source. The references are always to another acceptable attribute type in the Galaxy.

You can add multiple InputOutput extensions to an AutomationObject. However, you cannot add an InputOutput extension to an attribute that already has an input or output extension.

---

**Note** You can extend lockable attributes with an InputOutput extension, but they only function correctly during run time if the extended attribute is unlocked.

---

### When Objects Are On Scan

When an object is On Scan, the value and quality of the InputOutput-extended attribute mirrors the quality of the externally referenced attribute during a successful read. The data quality of the extended attribute is set to Bad when reads fail. Reads can fail because of communication errors or datatype conversion failures.

While the extended object is On Scan, the data can change quality. If an external set (for example, from a user) to the extended attribute changes either the value or quality, then a write of the extended 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 extended object is Off Scan, quality is always Bad and user sets are accepted.

### Using InputOutput Extensions in Scripts

Two common types of scripts can be written on InputOutput-extended attributes: 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 extended attribute in its expressions. For example, if the extended attribute is “me.udal”, the script refers directly to “me.udal” for data change conditions and for expressions within the script.

The output side script can manipulate an output or validate a new requested output value. This script refers to the “WriteValue” attribute that extends the extended attribute: “me.udal.WriteValue”. So, to validate a new requested value to the udal, for example, a data change condition expression is written on “me.udal.WriteValue”. In addition, if the script wants to do clamping or validation, it can manipulate the “me.udal.WriteValue” directly to clamp the output value. For example:

```
If (me.udal.WriteValue > 100.0 ) then
    Me.udal.WriteValue = 100.0;
Endif;
```

The data change expression for this script is “me.udal.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 121.

## Using the Input Extension

You can add multiple Input extensions to an AutomationObject. However, you cannot add an InputOutput extension to an attribute that already has either an input or output extension. Arrays are not supported.

**Note** Lockable attributes can be extended with an Input extension, but they only function correctly during run time if the extended attribute is unlocked.

The screenshot shows the configuration window for the \$AnalogDevice\_001 \* object, specifically the Extensions tab for the AlarmInhibit attribute. The window is divided into two main sections: a table of extendable attributes and a configuration area for the selected attribute.

**Extendable Attributes Table:**

Name	IO	I	O	A	H	B
AlarmInhibit		X	X	X	X	X
AlarmInhibit.Msg						
AlarmMode						
AlarmModeCmd						
Deadband						
InAlarm						
PV						
PV.Bad.Condition						
PV.EngUnits						
PV.Input.Value						
PV.LogDataChangeEvent						
PV.Mode						
PV.OverrideEnabled						
PVAuto						
ScanState						
ScanStateCmd						
SP						

**Attribute name:** AlarmInhibit

**Input extension configuration (highlighted in red):**

- Input extension
- Source: [Empty text field]

**Other extension configurations:**

- Output extension
  - Destination: [Empty text field]
  - Output every scan
- Alarm extension
  - Category: Discrete
  - Priority: 500
  - Alarm message: me.ShortDesc
  - Active alarm state: True
- History extension
  - Force storage period: 0 ms
  - Engineering units: [Empty text field]
  - Value deadband: 0.0 EU
  - Trend high: 10.0 EU
  - Trend low: 0.0 EU
- Boolean label extension
  - Label for 'False' state: False
  - Label for 'True' state: True

If the data types of the extended and **Source** attributes are the same, they are set to equal values according to the extended 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 extended attributes range, quality for the extended attribute is set to Bad. Otherwise, the extended attribute's quality matches the **Source** attribute. When the extended object is Off Scan, quality is always Bad and user sets are accepted.

Attributes extended by an input extension are not protected by their security classification. The only enforced security specifies if an IDE user can edit, or extend, the object. An input extension can be added to a template or instance. If added to a template, the existence of the input extension is automatically locked in derived objects.

## Using the Output Extension

Writeable and Calculated attributes can be extended with an output extension. Arrays are not supported.

The screenshot shows the configuration window for the \$AnalogDevice\_001 \* object, specifically the Extensions tab for the AlarmInhibit attribute. The window is divided into two main sections: a table of extendable attributes and a list of extension types.

**Extendable Attributes:**

Name	IO	I	O	A	H	B
AlarmInhibit		X	X	X	X	X
AlarmInhibit.Msg						
AlarmMode						
AlarmModeCmd						
Deadband						
InAlarm						
PV						
PV.Bad.Condition						
PV.EngUnits						
PV.Input.Value						
PV.LogDataChangeEvent						
PV.Mode						
PV.OverrideEnabled						
PVAuto						
ScanState						
ScanStateCmd						
SP						

**Attribute name:** AlarmInhibit

**Extensions:**

- Input/Output extension
  - Source: [ ]
  - Output destination differs from input source
    - Destination: [ ]
- Input extension
  - Source: [ ]
- Output extension (highlighted with a red box)
  - Destination: [ ]
  - Output every scan
- Alarm extension
  - Category: Discrete
  - Priority: 500
  - Alarm message: me.ShortDesc
  - Active alarm state: True
- History extension
  - Force storage period: 0 ms
  - Engineering units: [ ]
  - Value deadband: 0.0 EU
  - Trend high: 10.0 EU
  - Trend low: 0.0 EU
- Boolean label extension
  - Label for 'False' state: False
  - Label for 'True' state: True

An output extension can be added to a template or an instance. If added to a template, the existence of the output extension is automatically locked in derived objects. The output **Destination** attribute in the extension is separately lockable in templates.



If the data types of the extended 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 extended 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 an Output Extension 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 InputOutput and Output extensions as well as 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 above. 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. 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.

Boolean data types are the exception to this rule. Boolean data types are used in User sets from InTouch or FactorySuite A<sup>2</sup> 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 above 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 Arcestra) 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 Input, InputOutput and Output Extensions

When the object is On Scan, the value and quality of the Input-extended attribute mirrors the quality of the externally referenced attribute in the case of successful reads. The data quality of the extended attribute is set to Bad when reads fail because of communication errors or datatype conversion failures.

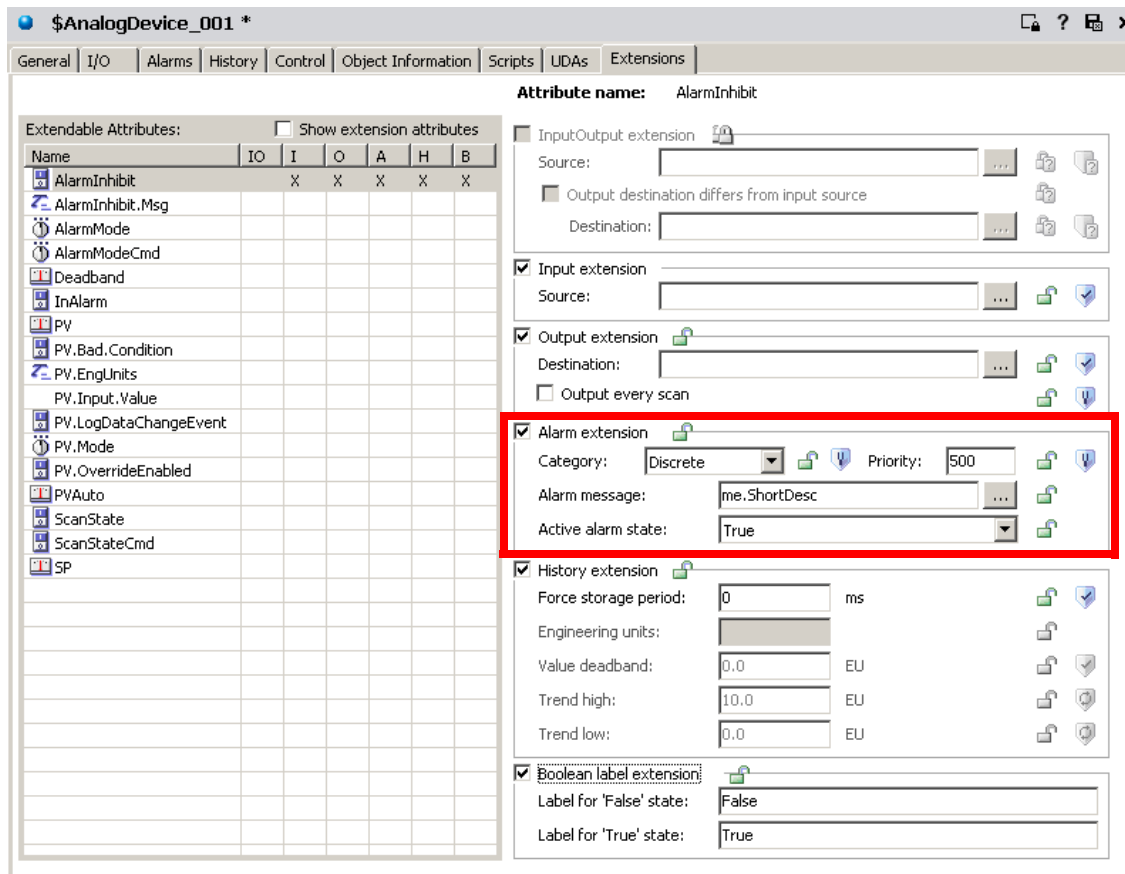
While the extended object is On Scan, it behaves as follows: If an external set (for example, from a user) to the extended attribute causes either the value or quality to change, then a write of the extended attribute's value to the destination occurs during the next execute phase.

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 extended object is Off Scan, quality is always Bad and user sets are accepted.

## Using the Alarm Extension

An alarm extension can be added to a template or instance Boolean attribute. If added to a template attribute, the alarm extension is automatically locked in derived objects. Attribute arrays cannot be extended.



Select the **Category** and specify an **Priority** for this alarm. Valid values are 0 to 999.

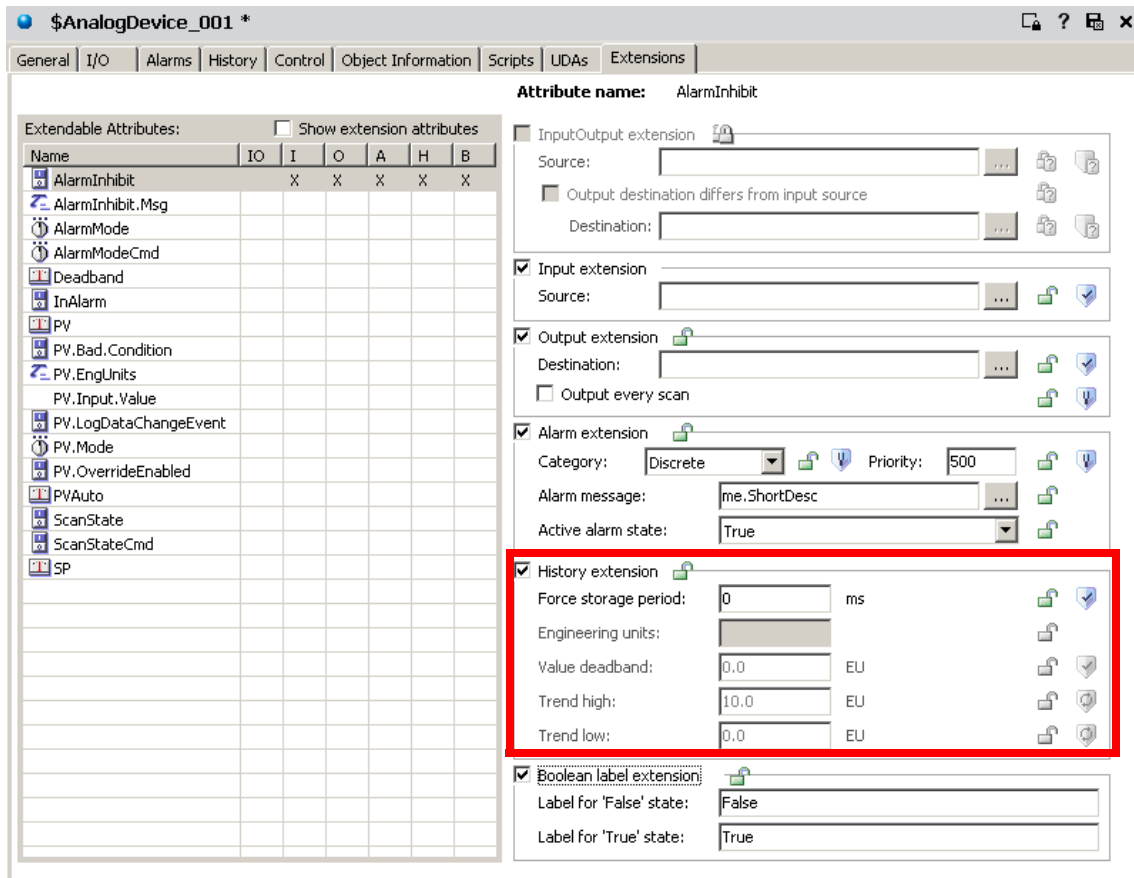
In the **Alarm message** box, you can browse and select an existing attribute or you can type a text string as an alarm message. This text string appears in the InTouch alarm view.

The **Active alarms state** list can show the customized items you specified in the **Boolean label extension** area. It lets you select the state that triggers the alarm. If you don't specify a value in the **Boolean label extension** area, you see **True** and **False**.

For more information about using Alarms, see Working with Alarms and Events on page 153.

## Using the History Extension

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



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

You can extend Writeable and Calculated attributes of the following data types with a history extension:

- 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

History Extension group parameters include:

- **Force Storage Period:** Period after which the value must be historized even if the value has not changed.
- **Engineering Units:** Engineering units of the attribute to be historized.
- **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.
- **Trend Low:** The default bottom of a trend scale.

For more information about using History, see *Working with History* on page 143.

## Using the Boolean Label Extension

In the **Boolean label extension** area, you can specify a explicit name for the True and False states of Boolean attributes. For example, if the Boolean attribute is associated with the operating condition of a motor, you can specify the states as **Stopped** and **Running**.

The labels defined in this area are available in the **Active alarm state** box in the **Alarm** extension area.

These labels are also shown in the **Value** and **Limit** columns of the Alarm and Event database and InTouch AlarmView control.

\$AnalogDevice\_001 \*

General | I/O | Alarms | History | Control | Object Information | Scripts | UDAs | Extensions

Attribute name: AlarmInhibit

Extendable Attributes:  Show extension attributes

Name	IO	I	O	A	H	B
AlarmInhibit		X	X	X	X	X
AlarmInhibit.Msg						
AlarmMode						
AlarmModeCmd						
Deadband						
InAlarm						
PV						
PV.Bad.Condition						
PV.EngUnits						
PV.Input.Value						
PV.LogDataChangeEvent						
PV.Mode						
PV.OverrideEnabled						
PVAuto						
ScanState						
ScanStateCmd						
SP						

InputOutput extension

Source:

Output destination differs from input source

Destination:

Input extension

Source:

Output extension

Destination:

Output every scan

Alarm extension

Category:  Priority:

Alarm message:

Active alarm state:

History extension

Force storage period:  ms

Engineering units:

Value deadband:  EU

Trend high:  EU

Trend low:  EU

Boolean label extension

Label for 'False' state:

Label for 'True' state:

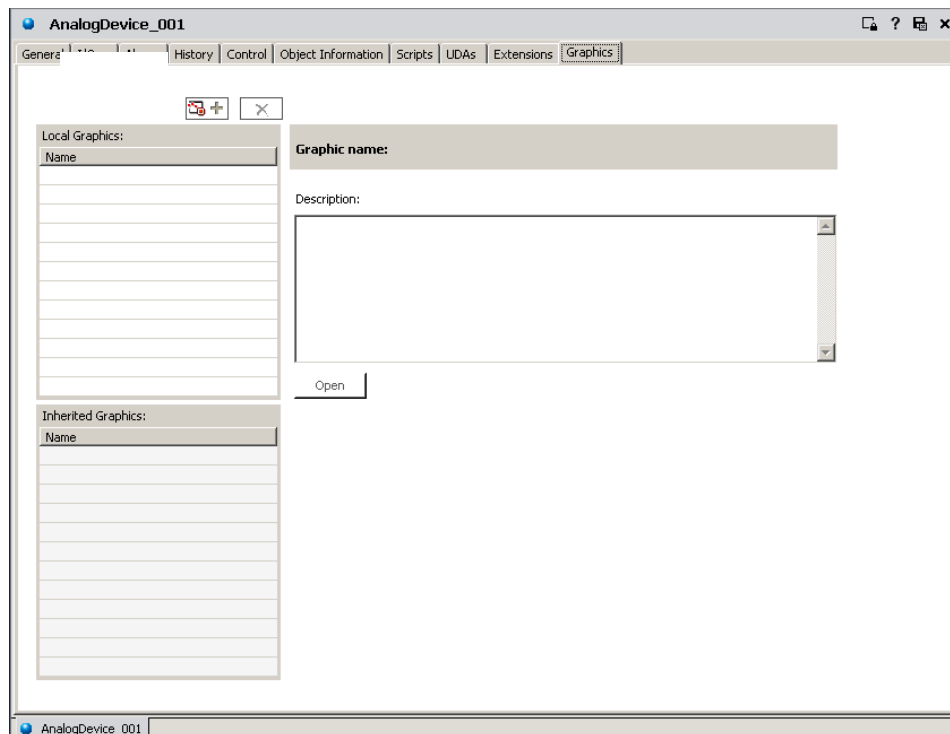
## Creating and Working with Graphics

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 *Creating and Managing ArcestrA 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. For instructions on using the Symbol Editor, see *Creating and Managing ArcestrA 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.
- 3 Click **Yes**.

# Chapter 6

## Deploying your Galaxy

You can deploy and test your objects at any time during development. When you're ready to test or move the application into production, it's time to 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 the Application Views* on page 21.

### 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*.

Objects deploy from the configuration environment to the run-time environment as follows:

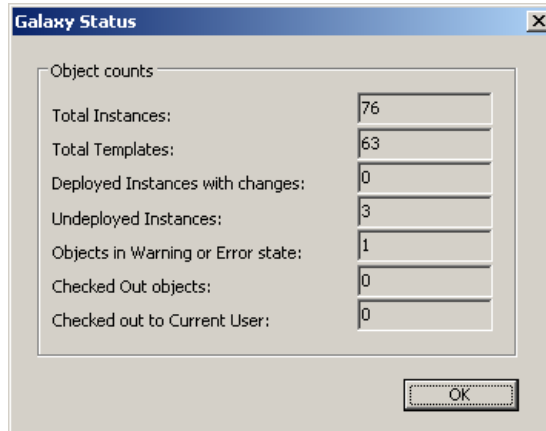
<b>IDE Configuration Environment</b>	<b>deploys to</b>	<b>Object Viewer Run-time environment</b>
Galaxy database	‡	[Does not exist in run-time environment]
Templates	‡	[Does not exist in run-time environment]
Instance objects	‡	Instance objects <i>[Run-time configuration and behavior]</i>
Security: General 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 <i>[Wonderware Historian]</i>

## 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.



You see information about 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**.

## Deploying Objects

You deploy object instances for three reasons:

- Testing.
- Place the application into production to process field data.
- Update 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 computer(s).
- 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.

---

**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

[No  
icon]

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



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 be visible for only a few moments at the end of an InTouchViewApp deployment operation, unless the object is deployed on a slow network.

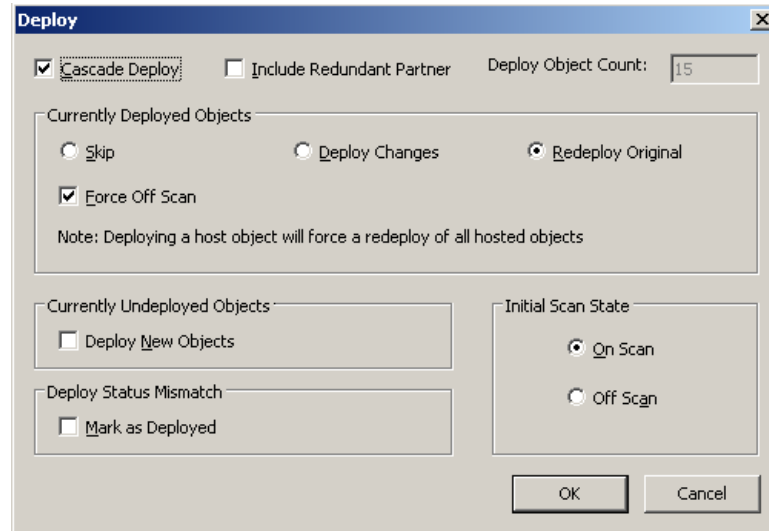
---

**Note** 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 will be 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 runtime state from the runtime 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 object(s) 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 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 only execute 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 can not execute.

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 object(s) you are deploying. If you deploy objects off scan, you must use the ArcestrA 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 on 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 31.

- 8 Click **OK** to deploy the object(s). The **Deploy** progress box appears. If you see error messages, see *Deployment Error Messages* on page 138. 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 timeout, a communication error occurs and the object is shown as not deployed. You may need to adjust the message timeout for the WinPlatform object to accommodate the slow network speed.

## 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 whose deployment state is Pending Update. That means the object changed since its 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 134.

## Undeploying Objects

You may need to undeploy one or more objects. Undeploying removes one or more objects from the run-time environment.

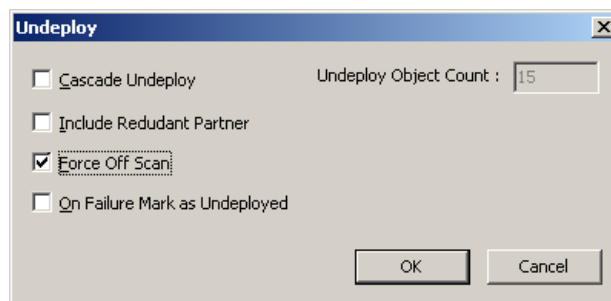
Before you start, you need to select the object or objects you want to undeploy in the IDE.

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** Upload runtime changes will not be permitted from old runtime node to the galaxy

---

The values of certain attributes can be set in the configuration environment, but they can also be changed by the user at run time. As a result, the values of these attributes can differ between the run-time and configuration environments. These attribute types are:

- Writeable\_UC
- Writeable\_UC\_Lockable
- Writeable\_USC
- Writeable\_USC\_Lockable

For example, you create an object with a UDA `myInteger`. In the Object Editor, you specify an initial value of 30.

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.

If you want to upload run-time changes to the Galaxy, make sure the selected objects are:

- Not edited and checked in since last deployment or upload
- Not in Pending Update state
- Checked in

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.

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.

---

#### 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.

## Undeployment Situations

The following situations occur in these specified undeployment scenarios:

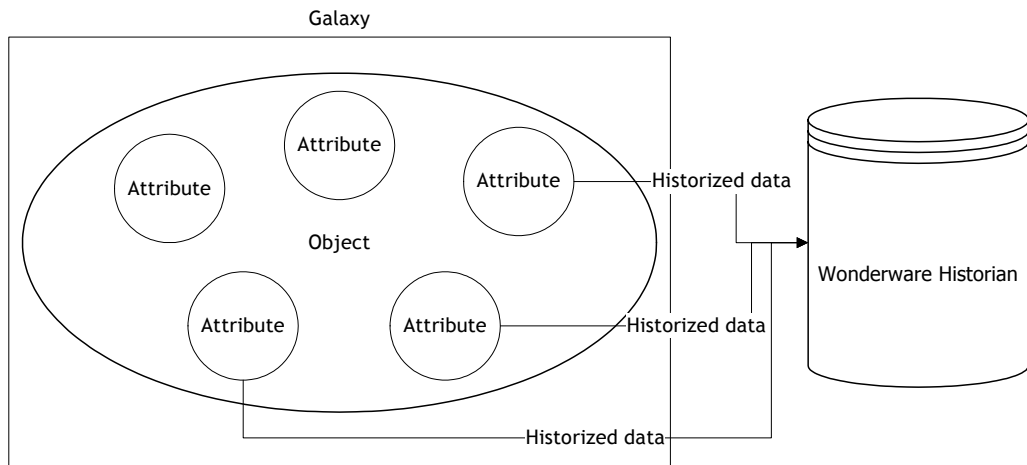
- After undeploying a WinPlatform, only the Bootstrap software remains on the target computer. All other WinPlatforms are notified of the undeployment of the WinPlatform and stop trying to communicate with it over the network.
- Alarm Clients know immediately that an area is undeployed and is no longer 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.

# Chapter 7

## Working with History

You can configure your objects to store data in the Wonderware® Historian (formally called IndustrialSQL Server). Storing data in the historian is useful because it allows you to analyze stored data.

This section describes how to configure objects to store data in the historian.

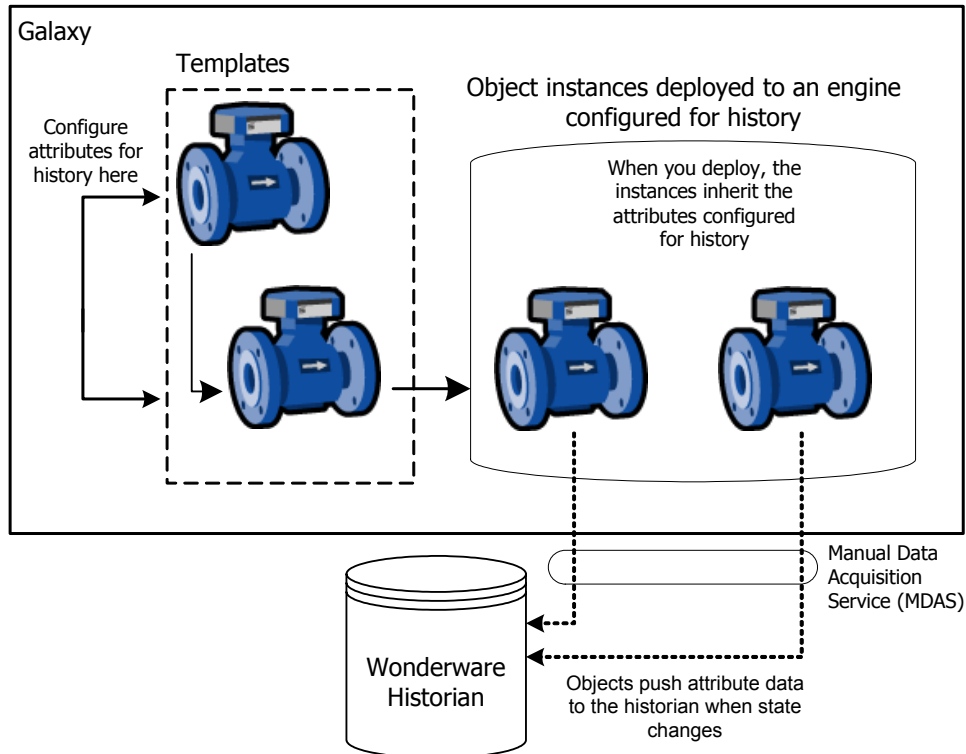


For more information about historizing data, see the *FactorySuite A<sup>2</sup> Deployment Guide*.

## Configuring History

Some attributes can be historized when values change. Data quality is also recorded along with attribute values. The historical records of this data can then be retrieved, trended and used for other purposes later.

For attribute data to be stored in the historian, a host AppEngine must be configured to send all history data to a Wonderware Historian node and deployed in the Galaxy.




---

**Important** A change in quality always causes a new record to be stored, regardless of whether the value changed.

---

For information about configuring specific history attributes for specific objects, see the online help in each object.



For each object, you can configure attributes of the following data types to be historized.

- Float (numerical)
- Double (numerical) maps to a Wonderware Historian float. 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, so truncation can occur. If so, quality is set to Uncertain.
- CustomEnum (non-numerical) maps to a Wonderware Historian integer
- ElapsedTime (numerical) maps to a Wonderware Historian float and converted to seconds
- Arrays or parts of arrays are not supported.
- Enum type attributes are historized as integer ordinal values.
- NaN values for float and double data types are converted to null values.
- All numerical attributes are sent to the Wonderware Historian in the engineering units exposed by the attribute. The Wonderware Historian does not scale the value.

## About the Wonderware Historian

To historize your process data, you must install the Wonderware Historian. When you install the Wonderware Historian, follow these guidelines:

- Install the Wonderware Historian on a computer outside the Galaxy but on the local network.
- A single the Wonderware Historian installation can receive historical data from a single Galaxy.

The Wonderware Historian creates additional VTQ records that modify quality when certain situations happen, such as network disconnects suggesting down times.

The Wonderware Historian stores two additional data quality fields: quality and quality detail. These are not OPC compliant for each record received from Application Server components.

For more information, see the Wonderware Historian documentation.

### About Manual Data Acquisition Service (MDAS)

Application Server communicates with the Wonderware Historian node through an interface called Manual Data Acquisition Service (MDAS). MDAS uses Distributed COM (DCOM). DCOM requires TCP/UDP port 135 to be available on the Wonderware Historian node, as well as in any Galaxy node pushing data to the Wonderware Historian and any other network device such as firewalls.

If port 135 is not available, Application Server fails to configure and store history to the Wonderware Historian. One possible reason that port 135 is not available can be that a router between Application Server node and the the Wonderware Historian node is blocking the port. Another reason might be that DCOM is disabled on either node.

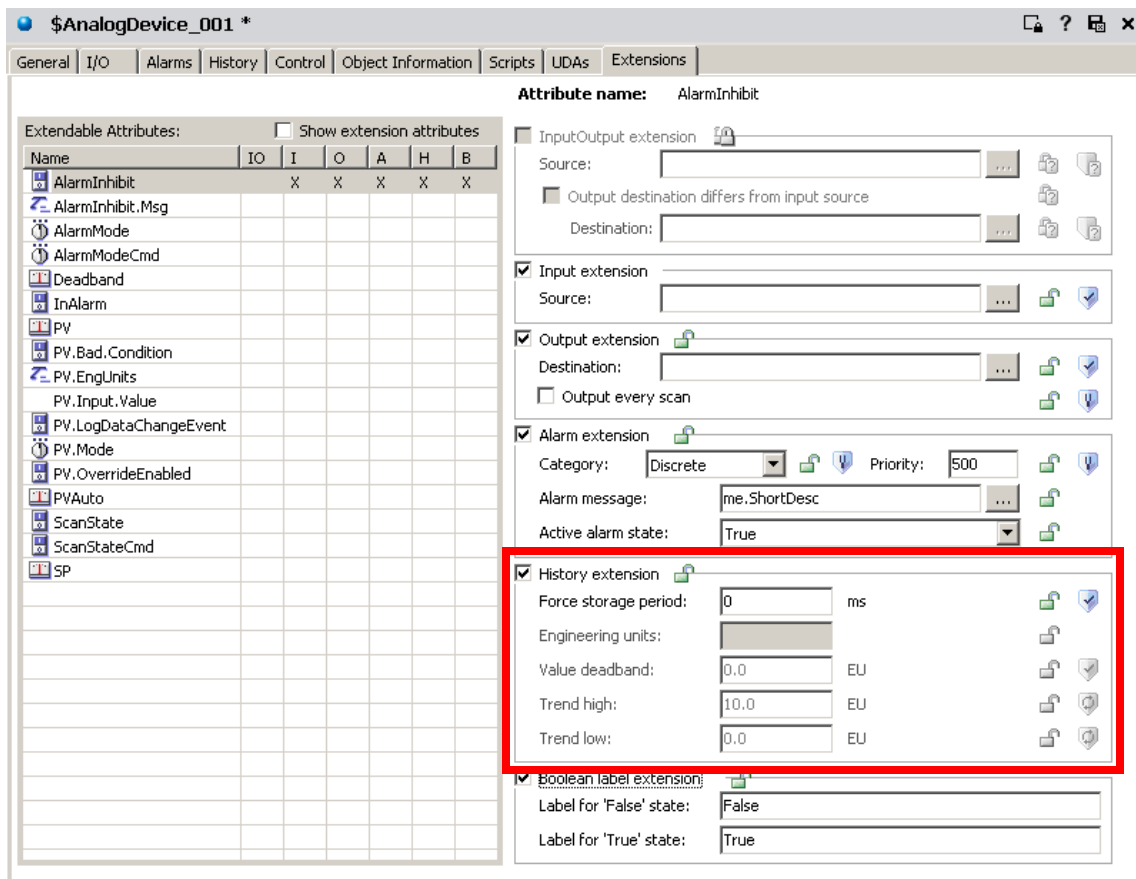
To make sure that the Wonderware Historian and Application Server work together properly, make sure that DCOM is enabled and not blocked.

## Configuring Objects to Store History

If you want to historize an attribute in run-time, you must specify this setting in the object.

### To configure an object to store history

- 1 Check out the object and open the Extensions page in the Object Editor.
- 2 For each attribute you want to historize, select the **History** check box. Optionally, provide a **Force Storage Period** value.



**Force Storage Period** is the time interval in seconds, which the attribute value must be stored, regardless of the value deadband setting. In addition to the **Value Deadband** setting, the value is stored continuously at this interval. A value of zero (0) disables this feature.

**Important** If further and identical value/quality pairs are received by the Wonderware Historian, they are not actually stored to disk.

- 3 For non-array, numerical attributes that you want to historize, such as integer, floats and doubles, configure the **Value Deadband**, **TrendHi** and **TrendLo** options.
  - **Value Deadband:** the threshold value in engineering units difference between the new and last-stored values before storing the new value to history. For example, a deadband value of 5 means that a new value is not stored until it changes more than 5 units from the previous value. Zero (0) is the default. Zero means that any change stores the changed value.
  - **Trend Hi** specifies the initial maximum trend value for clients.
  - **Trend Lo** specifies the initial minimum trend value for clients.
- 4 Save and close the Object Editor. Check in the object to the Galaxy.
- 5 Deploy the object in an On scan state to a host AppEngine that is configured for History. For more information, see Configuring WinPlatforms and AppEngines for History on page 150.

## During Run Time

During run time execution, data is historized as follows:

- If no previously-historized value exists, then the first value is always historized.
- 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 historized.
- If the attribute type is qualitative (enumeration, string or Boolean) and the attribute's value or quality changes.
- If the current time exceeds the time the last Forced Store occurred for the attribute by the time period that was set as the Force Storage Period. If no last Forced Store occurred since startup, a new store occurs immediately.

---

**Note** The Value Deadband mechanism, if enabled, resets itself based on this new value stored.

---

- The new attribute value, timestamp and quality are sent to the Wonderware Historian for storage.
- Wonderware Historian logs the data to disk.

### Deploying and Undeploying

When an AutomationObject configured for history is redeployed, changes to the attribute configuration of the object makes the historian reconfigure storage. For example, if the engineering units string for the tag change from “Deg F” to “Deg C” when the object is redeployed, the Wonderware Historian configuration database shows the change.

When an AutomationObject configured for history is undeployed, all history remains in the Wonderware Historian historian. The history data can be examined in the future even if the AutomationObject is no longer deployed.

### Store Forward Mode

If the Wonderware Historian node shuts down or the network connection with the Wonderware Historian node is lost, data storage continues locally. When the Wonderware Historian node recovers, the data is forwarded to the node at a low priority. For more information, see the Wonderware Historian documentation.

If an AppEngine loses connectivity with the Wonderware Historian node, the Wonderware Historian reports Bad data quality to clients. When you undeploy an object with attributes configured for historization, the Wonderware Historian stores the final data points with Bad quality.

## Configuring WinPlatforms and AppEngines for History

If an AppEngine is deployed before Wonderware Historian is started, history is not stored until the objects successfully register with the Wonderware Historian.

When an AutomationObject is deployed on scan, some data points for attributes are not stored until they are registered and committed to the Wonderware Historian.

See the online help for WinPlatform and AppEngine objects for specifics about the Object Editor pages of these objects.

### To configure WinPlatforms and AppEngines for history

- 1 Check out the object and open the Object Editor.
- 2 Do the following:
  - To configure a WinPlatform, go to the **Engine** page.
  - To configure an AppEngine, go to the **General** page.

The **History** area on each page looks the same.

The screenshot shows the configuration window for 'WinPlatform\_001'. The 'History' section is highlighted with a red border and contains the following settings:

- Enable storage to historian
- Enable Tag Hierarchy
- Historian: [ ]
- Store forward deletion threshold: 100 MB
- Store forward minimum duration: 0 s
- Forwarding chunk size: 1024 Bytes
- Forwarding delay: 250 ms

Other settings visible in the window include:

- Engine startup type: Auto
- Restart the engine when it fails:
- Scan period: 1000 ms
- Scripts:
  - Maximum time for scripts to execute: 1000 ms
  - Maximum asynchronous thread count: 5 threads
- Checkpoint period: 0 ms
- Checkpoint directory location: [ ]
- Alarm throttle limit: 2000 alarms/s
- Statistics average period: 10000 ms
- Maximum input queue size: 16 MB
- Engine failure timeout: 10000 ms
- Maximum number of consecutive data notification failures allowed: 0

- 3 Do the following:
  - To use historization, select **Enable storage to historian**. If you want the portion of the model view hierarchy hosted by the engine to get replicated into the public group namespace when the historian engine starts up (either deployment or restart), also select **Enable Tag Hierarchy**.
  - In the **Historian** box, type or select the Wonderware Historian node name.
  - Specify the **Store forward deletion threshold**, in megabytes.
  - Specify the **Store forward minimum duration**, in seconds.
  - Specify the **Forwarding chunk size**, in bytes.
  - Specify the **Forwarding delay**, in milliseconds.
- 4 Save and close the Object Editor. Check in the object to the Galaxy.
- 5 Deploy the object to its target computer in an on scan state.





---

# Chapter 8

## Working with Alarms and Events

By enhancing and extending a template, you can automate detection, notification, and viewing application (process) events and alarms or system and software events and alarms using the Application Server.

### About Events and Alarms

Events and alarms are different.

- An event is a condition at a point in time. Application Server can detect events, store them historically, and report them to various clients.
- An alarm is a condition that is considered abnormal and requires immediate attention. Alarms are a special type of event that have a state and must be acknowledged. ArcestrA handles the real-time reporting of alarms in a special way and provides special clients for viewing them.

After an Area object is deployed, alarm clients can monitor and control alarms generated by Automation Objects belonging to the Area.

ApplicationObjects 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.

## Event Examples

Examples of events include:

- A plant process starts. For example, a new batch starts.
- The operator changes a plant operator parameter. For example, a setpoint on a temperature controller.
- The system engineer changes the run-time system configuration. For example, deployment of a new AutomationObject.
- A system engineer starts or stops a system component. For example, stopping an AppEngine.
- A WinPlatform comes back online after a failure or shutdown.
- A user logs into the system.
- A severe software problem is detected. For example, a ApplicationObject component fails.

## Alarm Examples

Examples of alarms include:

- A process measurement exceeded a predefined limit. For example, a high temperature alarm.
- A process device is not in the desired state. For example, a pump that should be running has stopped.
- The system hardware is not operating within desired limits. For example, the CPU performance on a WinPlatform exceeds 99% for a specified time.

## Items That Are Not Events or Alarms

The following items are not considered events or alarms:

- Configuration actions within the Galaxy Repository. For example, import or check out of an object.
- Installing Bootstrap on a computer.
- A message sent to the ArcestrA Logger. Sometimes, certain software events log a message in addition to generating an event. Logger messages are not events.
- Viewing a window in InTouch.

## Configuring Alarms

Alarming capabilities are a part of object templates, but they are not implemented until the object is configured in the IDE. After they are configured, you can view the Application Server alarms in InTouch.

Configuring an AutomationObject to be an alarm provider includes:

- Deciding whether alarm notification is needed for each possible alarm condition in the object. For example, a command timeout alarm for a valve if the output command fails to move the valve.
- Editing the object and configuring an attribute that specifically commands alarm notification.
- Configuring the alarm configuration attributes. Typically, the fields that require configuration are Category, Priority and Description.
- Configuring any limit fields for triggering alarm detection. For example, the feedback timeout time limit.

You can add alarms detection and reporting capabilities to AutomationObjects that were not originally developed to detect alarms. You do this by using alarm extensions.

When the alarm name contains more than 294 characters (<object name>, <attribute name>), InTouch will not raise an alarm even though the pv.limit and description are minimum length. For more information, see Working with References on page 167.

### To configure a WinPlatform to be an InTouch alarm provider

- 1 Check out the WinPlatform object.
- 2 Open the Object Editor.

The screenshot shows the 'WinPlatform\_001' Object Editor window. The 'Alarms' tab is selected. The configuration is as follows:

- Network address: VM5
- History store forward directory: (empty)
- Minimum RAM: 512 MB
- InTouch alarm provider
- Alarm areas (blank for all): (empty)
- Statistics average period: 10000 ms
- Redundancy section:
  - Redundancy message channel IP address: (empty)
  - Redundancy message channel port: 30001
  - Redundancy primary channel port: 30000
- Message Exchange section:
  - Message timeout: 30000 ms
  - NMX heartbeat period: 2000 ms
  - Consecutive number of missed NMX heartbeats allowed: 3
  - Message exchange port: 5026

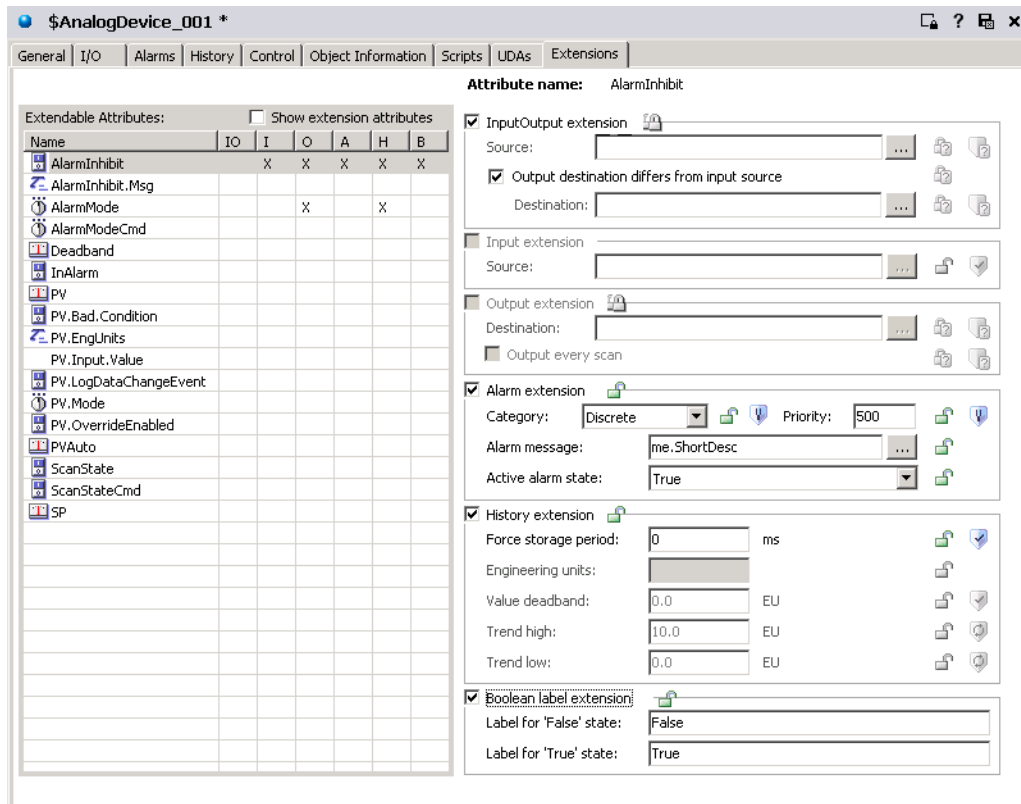
- 3 Select the **InTouch alarm provider** check box. Enter specific Areas in the **Alarm Areas** box, or leave the box blank to include all Areas.
- 4 Save and close the Object Editor.
- 5 Check the object in to the Galaxy.
- 6 Deploy the object in an on scan state.

## Setting Alarms on the Extension Page

You set alarms in the Extensions page in the Object Editor.

### To specify alarms

- 1 On the **Extensions** page of the Object Editor, select an attribute from the **Extendable Attributes List**. The four extension groups dynamically change to allowed extension rules for the selected attribute type.



- 2 Select the **Alarms** check box. For **Alarm Extension**, select a **Category** from the list:

Batch	DeviationMajor	DeviationMinor
Discrete	Process	ROC Hi
ROC Lo	Software	SPC
System	Value Hi	Value HiHi
Value Lo	Value LoLo	

- Type a **Priority** level for the alarm (default is 500).
- Select to use either the **Object Description** for **Alarm Message** or type another alarm message in the **Message** box. An **X** appears in the **A** column of the selected attribute.

- 3 For **Boolean Label Extension**, specify text strings for the **False** state and the **True** state, if needed. These text strings appear in the **Active Alarm State** list for you to select.
- 4 Lock the values, if needed. The lock symbol is available only when you are extending a template. Otherwise, it indicates the lock condition of the value in the parent object.

## About Alarm Event Distribution

After you configure `AutomationObject` instances for alarm detection, deploy the instances and put them `On scan`. The instances begin checking for alarm conditions right away.

When an alarm is detected, or an event occurs, the condition is reported to its alarm and event distributor, which is running on the same `AppEngine`.

These alarm and event distributors include:

- `Area AutomationObjects` All `AutomationObjects` and `Area` objects report detected alarms through the `Area`, which distributes them to alarm and event clients.
- `WinPlatform AutomationObjects` Report their own alarms and events.
- `AppEngine AutomationObjects` Report their own alarms and events.
- `Device IntegrationObjects` Report their own alarms and events.

`WinPlatforms`, `AppEngines` and `Device Integration` objects do not report their alarms and events to `Area AutomationObjects` even though they belong to `Areas`. This allows alarm clients to receive alarm notifications without any dependencies on `Area AutomationObjects`. 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.

## Area AutomationObject

The Area AutomationObject plays a key role in alarm and event distribution. All AutomationObjects 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.

## Alarm and Event Subscription

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.

## Enabling and Disabling Alarms

Alarms can be enabled or disabled in the run-time environment. This action can be specified at the Area level, at a container object level, or at an individual object level. You cannot selectively enable or disable individual alarms in a single object.

Run-time alarm actions include:

- **Enabled:** All alarms for the object are reported to clients and historized normally.
- **Disabled:** No alarms for the object are detected. The alarm is return-to-normal until the alarm is reenabled.

---

**Important** Silenced mode is reserved for future use. It currently functions in the same way as Disabled.

---

The table below shows the resulting Enable and Disable state given the state of other objects in a hierarchy. The most restrictive state controls.

Object's Area Alarm Mode	Container Alarm Mode (if any)	Object Commanded Alarm Mode	Resulting State for Object
Enabled	Enabled	Enabled	Enabled
Disabled	Any	Any	Disabled
Any	Disabled	Any	Disabled
Any	Any	Disabled	Disabled

### Enabling Alarms

To enable an AutomationObject's alarms, you must ensure that the AlarmModeCmd and AlarmInhibit attributes are enabled for the AutomationObject, its container, and its Area.

An event, including the user's name, is generated indicating the object's alarms are enabled.



## Disabling Alarms

A user with the correct permissions can disable an AutomationObject's alarms through an InTouch window. This is done by requesting that the enable and disable state of the AlarmModeCmd attribute of any one of the following objects be disabled:

<b>If you disable alarms in:</b>	<b>Alarms are disabled for:</b>
an AutomationObject	the AutomationObject
the object's Area	all contained Areas and AutomationObjects belonging to those Areas
a container AutomationObject	all contained AutomationObjects

An event, including the user's name, is generated showing the object's alarms are disabled. Active alarms for the disabled object are removed from the InTouch Alarm display.

## During Run Time

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.

## Using the InTouch HMI as the Alarm and Event Client

InTouch run-time clients subscribe to event reports from the Galaxy. The InTouch operator can view alarms, acknowledge alarms, disable alarms, and enable alarms from InTouch.

Application Server reports the alarm to the InTouch Distributed Alarm System, which reports it to InTouch.

In the InTouch alarm client display, the alarm information for the new, unacknowledged alarm, including all required fields, appears. The new alarm is in the unacknowledged state.

To view alarms for objects in an Area, you must point InTouch to the Area that contains the objects. To view alarms for a platform, DI Object, or an AppEngine, you can create either assign these objects to an area and reference the area in your query, or reference the objects directly in your query.

The syntax for the query is:

```
\\ITProviderNode\Galaxy!Area or Object name
```

where *ITProviderNode* is the name of the node that the InTouch Alarm Provider platform is deployed to and *Area or ObjectName* is the name of the platform, AppEngine, or DI Object.

For more information, see the Application Server *Deployment Guide*.

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 AutomationObject is on scan.
- The source AutomationObject's Area is on scan.
- Alarming must be enabled for the target AutomationObject.
- An InTouch alarm provider on any WinPlatform in the Galaxy.

In InTouch:

- The InTouch client (WindowViewer) is running.
- The InTouch alarm client is placed on a window and configured as an alarm consumer for the Galaxy.
- The user is logged into InTouch, using ArcestrA security. This user is authorized to acknowledge alarms for the AutomationObject that is in the alarm state, if the user should be able to acknowledge alarms.

Application Server validates that the user has sufficient security privileges to acknowledge the alarm.

If the user does not have privileges to acknowledge alarms, the user can acknowledge the alarm but the Galaxy rejects the acknowledgement 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.

## Alarms and Events in the InTouch HMI and in Application Server

The InTouch HMI and Application Server both implement alarms and events. The table below shows the similarities and differences between the two models.

Item	InTouch	Application Server
Alarm configured or detected by	Within a Tag	Within an AutomationObject
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 Primitive name is the closest concept. For example, ".PVHiAlarm". Mapped from Category.
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)	Comment = short description.	AutomationObject 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.

---

<b>Item</b>	<b>InTouch</b>	<b>Application Server</b>
State	UNACK, ACK, RTN	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.

---



# Chapter 9

## Working with References

References allow identification and communication between objects in the ArcestrA 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 ArcestrA and the Wonderware® Application Server.

---

**Note** ArcestrA 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 ArcestrA 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. For more information, see the *FactorySuite A<sup>2</sup> Deployment Guide*.

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.

AutomationObject Reference + Attribute Reference

A reference string is the object name plus the attribute name: `ObjectName.AttributeName`.

In `TIC101.PV`, `TIC101` is the `AutomationObject` 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 giving 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 a UDA Float named B2. A1.B2 refers to the UDA 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 UDA 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, see [ApplicationObject Containment](#) on page 53.

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`.

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:

<code>Tank1</code>	Cannot reference at this level because this is not contained
<code>Inlet Valve (InletValve)</code>	Can reference at this level because this object is contained
<code>Outlet Valve (OutletValve)</code>	Can reference at this level because this object is contained

## Property References

Certain property names are reserved for ArcestrA. If a string has a reserved property name in the ArcestrA 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`.

### Example:

```
obj.int.PROPERTY(quality)
```

where:

`obj` = object specifier

`int` = attribute

`PROPERTY` = keyword

`(quality)` = property specifier

---

**Note** If you have an object named “obj” with two UDAs named “int” and “int.quality,” the reference string `obj.int.quality` is ambiguous. Using the reference string `obj.int.PROPERTY(quality)` states that you want a reference to the quality property of “int.”

---

## 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, primitive 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 ::= [.<primitive>][.<attribute>] | [.<primitive>][.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 runtime)`
- `attribute ::= <static_attribute>|<dynamic_attribute>`
- `static_attribute ::= [<static_attribute>].<identifier>`
- `<dynamic_attribute> ::= <any_char_but>{<any_char_but>}`
- `primitive ::= [<primitive>].<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 <primitive> 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 <primitive> 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 primitives or attributes when enhancing an object's functionality on the **Scripts**, **UDAs** and **Extensions** 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.

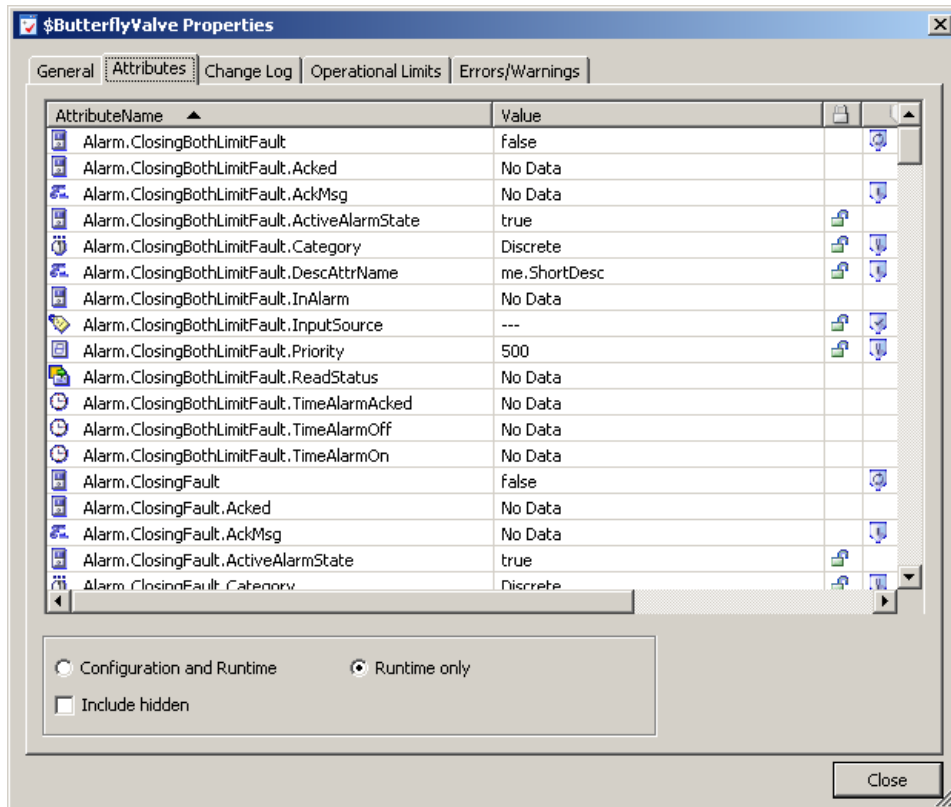
InTouch Pseudo-Property Names include: #VString, #VString1, #VString2, #VString3, #VString4, #EnumOrdinal, #ReadSts, #WriteSts and #QString. For more information on InTouch Pseudo-Property Names, see the *InTouch® HMI Data Management Guide*.

## Viewing Attributes in Objects

Within the ArchedrA 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:

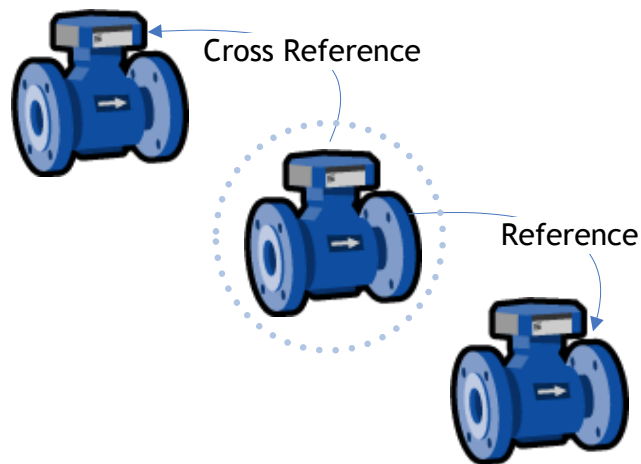
- **AttributeName** list    The selected attributes for the object, the current value, and locked and security status. The information shown depends on whether **Configuration and Runtime** or **Runtime Only** is selected.
- Value    Shows the value of the attribute.
- Locked/Unlocked    Shows if the 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 the:
  - **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.
  - **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**.

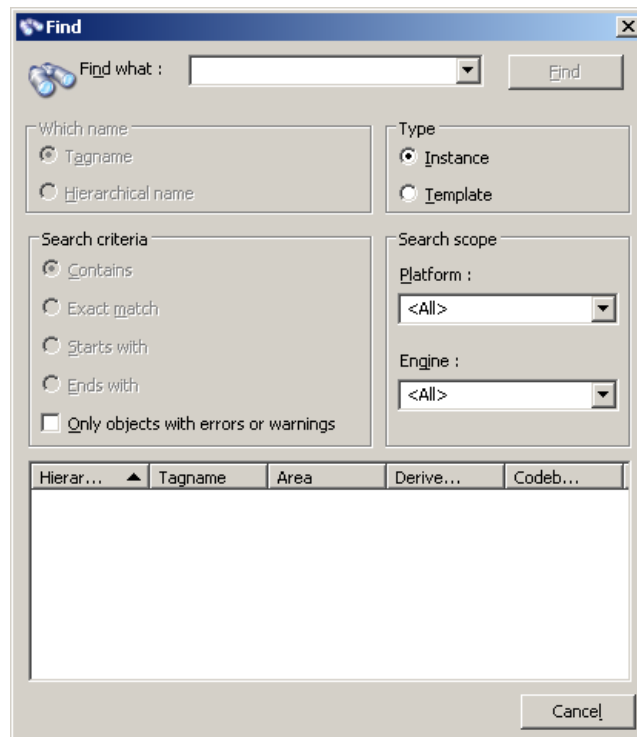


## Finding Objects

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. Also, you can search by part or all of a tag name or hierarchical name.

### 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**.
  - 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 226 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.

---

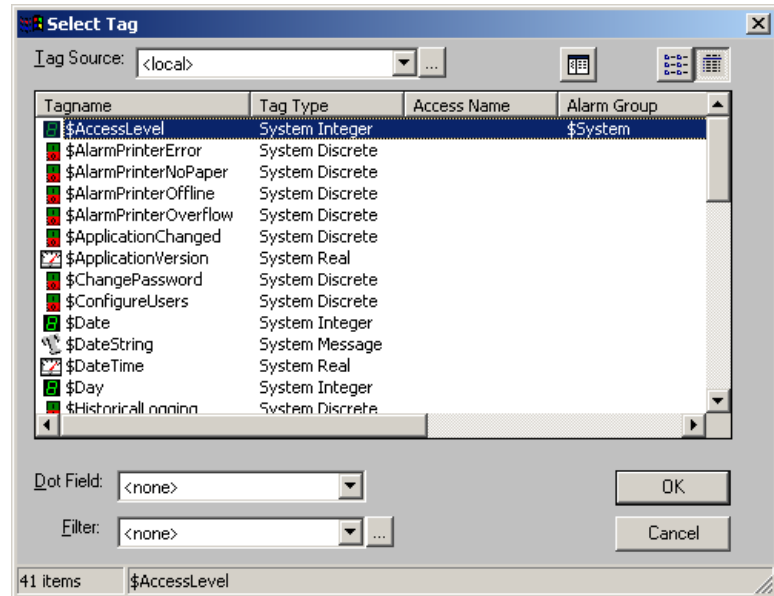
**Note** You must install the Bootstrap and IDE on the InTouch node to create the TagSource to browse Galaxy objects.

---

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.

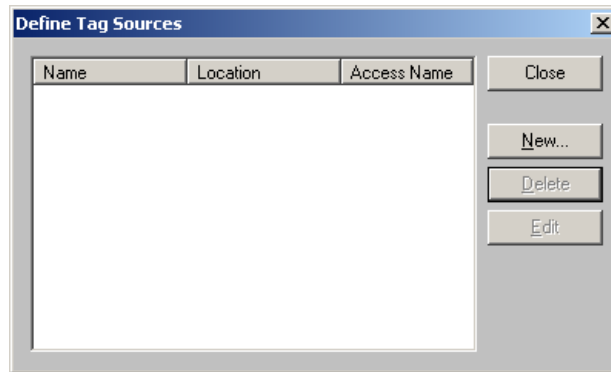


For complete information about using the InTouch Tag Browser, see the InTouch documentation.

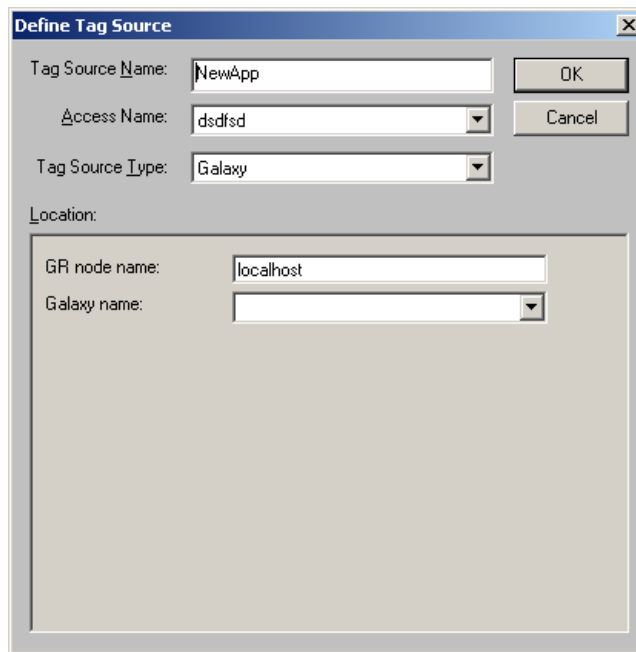
Before you can browse Galaxy references, you must define a new tag source.

### 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.

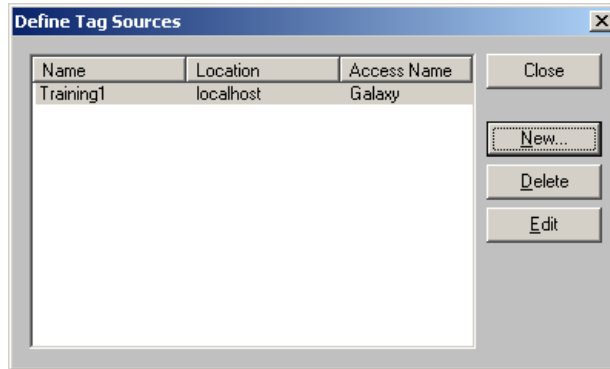


- 2 Click **New** to open the **Define Tag Source** dialog box.



- 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 78.



# Chapter 10

## Working with Security

Galaxies are created without security restrictions. After a Galaxy is created, you can assign restrictions to manage access. Using security lets you manage access to:

- IDE for configuring and managing objects.
- ArcestrA System Management Console (SMC) for performing maintenance and system administration functions.
- Any run-time operations.

This section describes the architecture of ArcestrA security and how to use it to manage access to configuration and run-time aspects of your Application Server application.

For more information on ArcestrA security, see your *FactorySuite A<sup>2</sup> Deployment Guide*.

## About Security

Security not only controls access to user interfaces in the ArcestrA environment but also controls access to object attributes and the data they represent.

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:

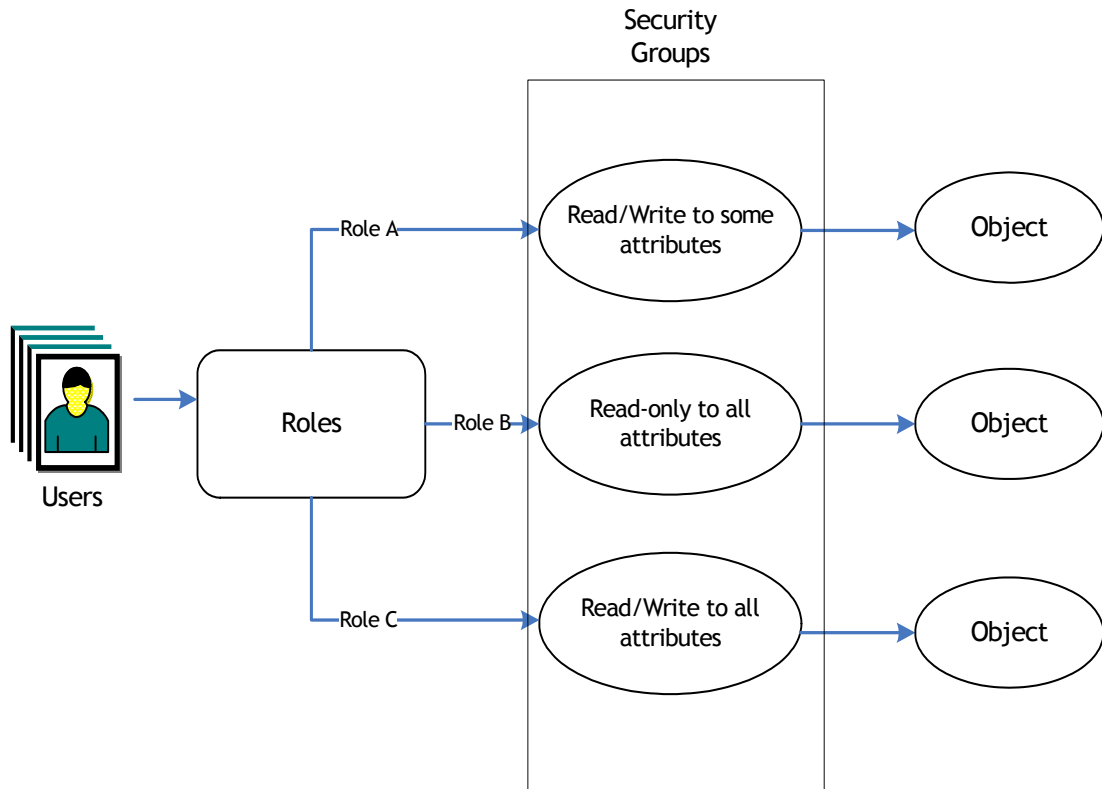
- Users associated with specific roles
- User roles associated with specific system administration, configuration and run-time (operational) permissions, which map to security groups
- Security groups associated with specific objects in the Galaxy

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 ArcestraA security model.



## About Authentication Modes

When you assign security, you can select one of three authentication modes:

- **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 are decided 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 198.

## Multiple Accounts Per User

Regardless of the security system you select, as with most security systems, 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 ArcestrA 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 four 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

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

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.

## About Users

If you select either OS based authentication mode, users with local machine accounts are added to the **Authorized Users Available** list in the following format: `.\<username>`.

If you select **OS Group Based** authentication mode, the local machine 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.

## 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.

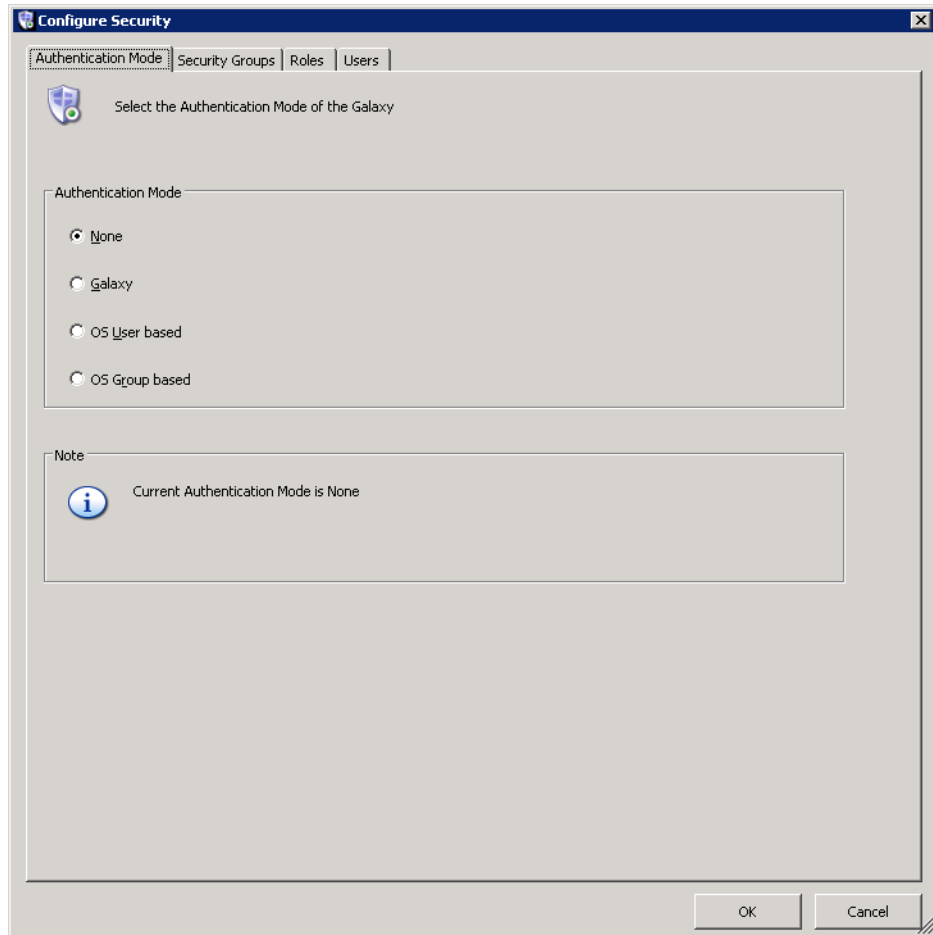
---

**Caution** You can only change the ArcestrA 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 ArcestrA User Accounts on page 219.

---

### To configure Galaxy security

- 1 On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears.

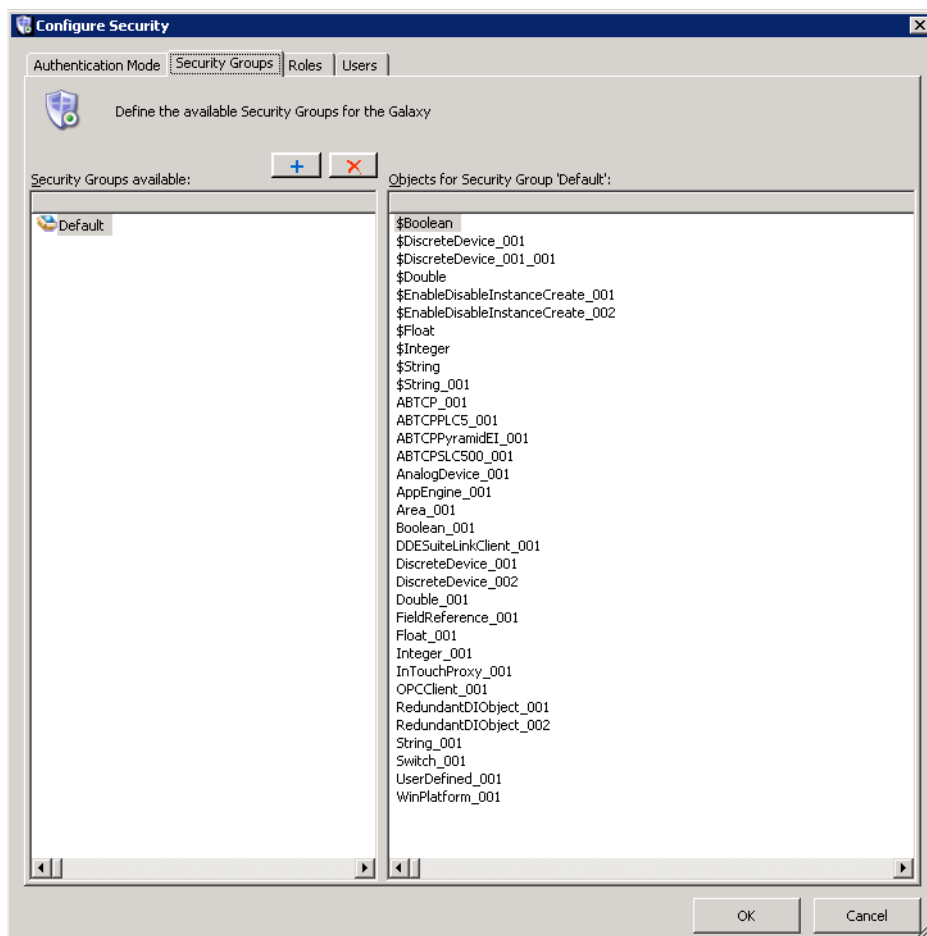


- 2 On the **Authentication Mode** tab, do the following:
  - Select the security type you want. Depending on what you select, more options become available.
  - If you select **OS Group-based** and you are working on a slow or intermittent network, you can specify the intervals in milliseconds:

**Login Time:** The timeout period (measured in milli-seconds) during which the system validates the user's membership against the OS groups selected as ArcestrA 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 ArcestrA. 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 *FactorySuite A<sup>2</sup> Deployment Guide*.

- Click **OK** or click the **Security Groups** tab.

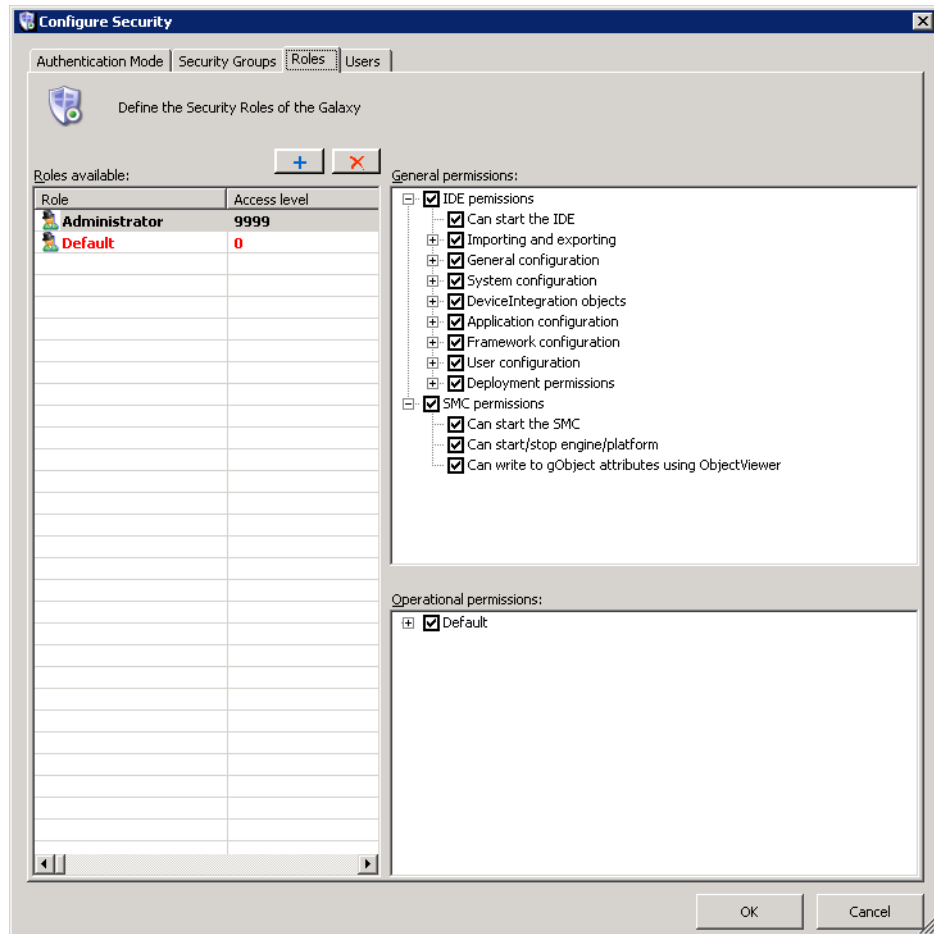




- 3 On the **Security Groups** page, do the following:
- Create a new security group by clicking the **Add** button. Type a unique name for the new group in the **Security Groups Available** pane. Security group names can be up to 32 alphanumeric characters, including a period. The name must include 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 object(s) to the new security group.
- Click **OK** or click the **Roles** tab.



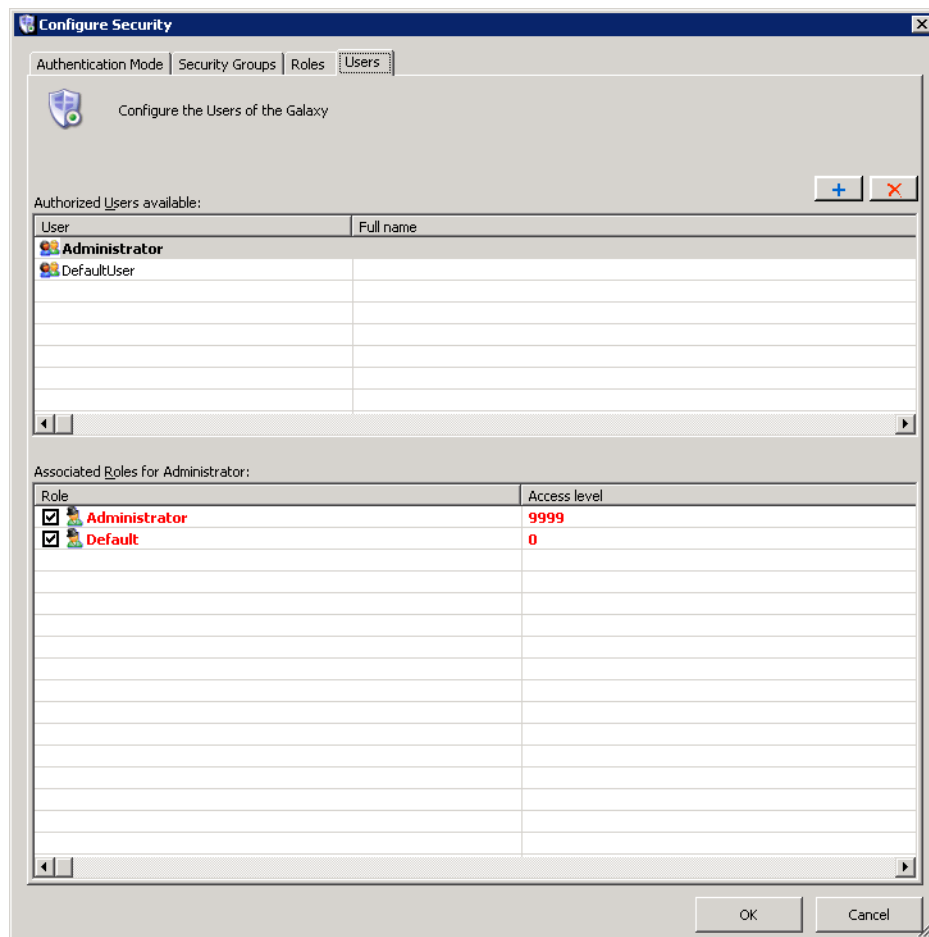
- 4 On the **Roles** page, do the following:
  - Create a new role by clicking the **Add** button. Type a name for the new role in the **Roles Available** pane. Role names can be up to 512 alphanumeric characters, including a period.
  - Select the **General** and **Operational Permissions** for the new role.

---

**Important** 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.

---

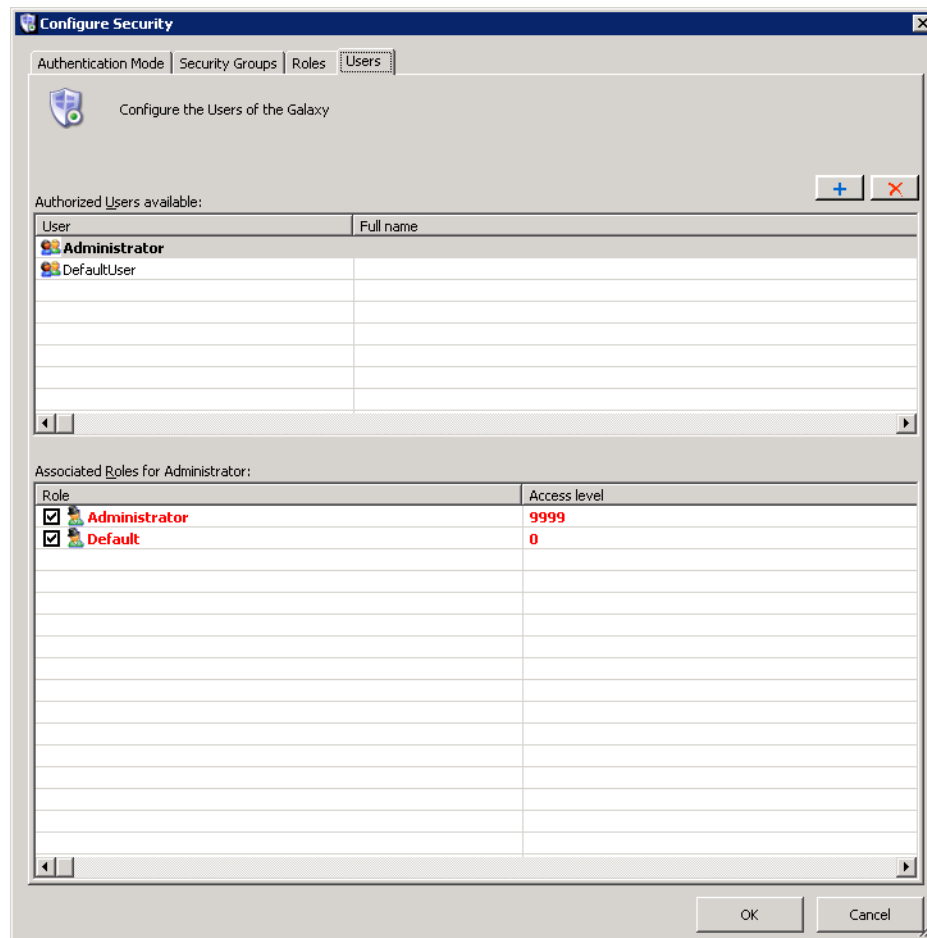
The Administrator user can log on any authentication mode except when security is disabled. When logged on as Administrator on the Galaxy Repository node, you can change the password of any Galaxy user without providing the old password.

- 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.

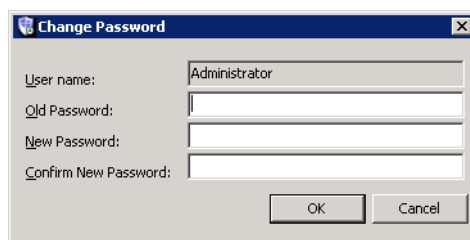
You can assign users to more than one role.

### To assign a role to a user

- 1 On the **Galaxy** menu, click **Configure** and then click **Security**. The **Configure Security** dialog box appears. Click the **Users** tab.



- 2 Select the user in the **Authorized Users available** area. Select a role in the **Associated Roles for <user name>** area.
- 3 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 ArcestraA password may be set to a maximum of 127 characters.

- 4 Enter the correct information. This information is used in the configuration, administration and run-time environment to authenticate users.
- 5 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 AutomationObjects are associated with it.

You cannot delete the Default security group.

### 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.

## 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. ArcestrA 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.

## 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.

## Cached Data at Log In

If you log on to ArcestrA 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 ArcestrA for the first time.

If the user previously logged on to ArcestrA 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 ArcestrA.

## Mixed or Native Domains

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.

Your unique listing maps to the list of domains and user groups you see when you use the Windows tool for managing your domain. A domain group configured as “Distribution” instead of “Security” 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 ArcestrA 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.





# Chapter 11

## 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 15.

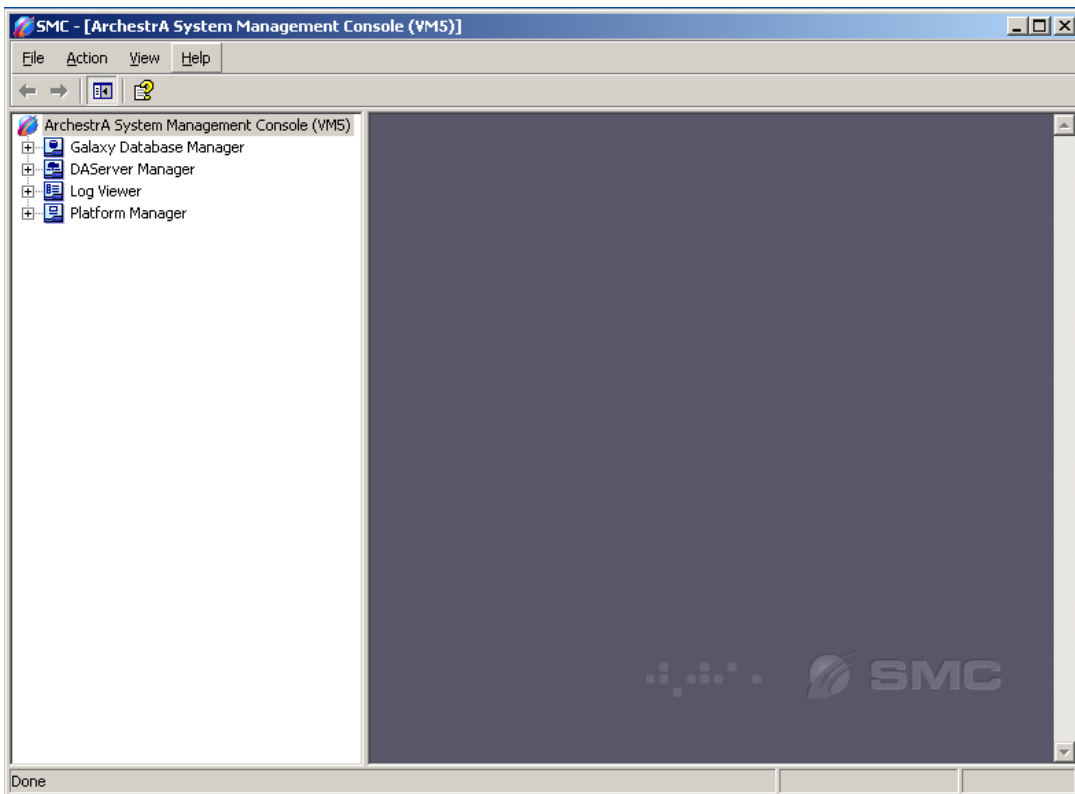
### Backing Up and Restoring Galaxies

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 ArcestrA System Management Console utilities.

**To open the Galaxy Database Manager**

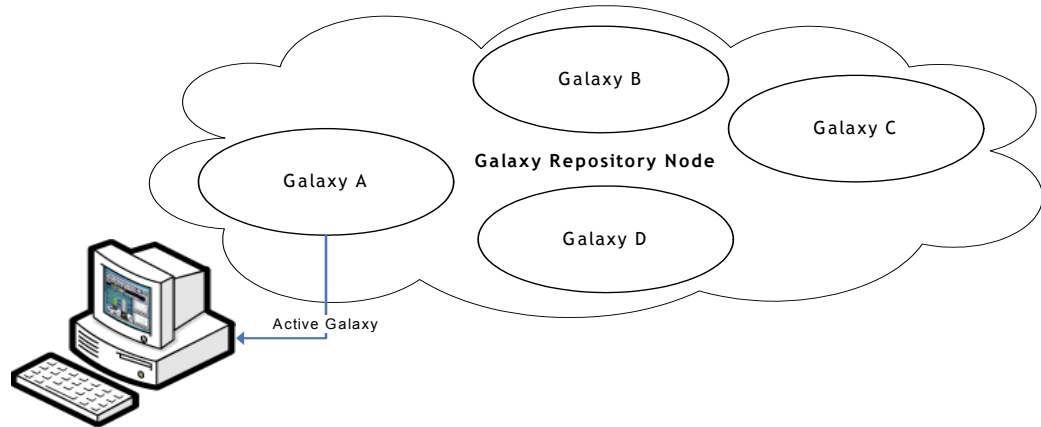
- ◆ Click **Start**, point to **Programs** and then to **Wonderware**, and then click **System Management Console**.



See the Galaxy Database Manager documentation for more information about backing up or restoring your Galaxy.

## Changing Galaxies

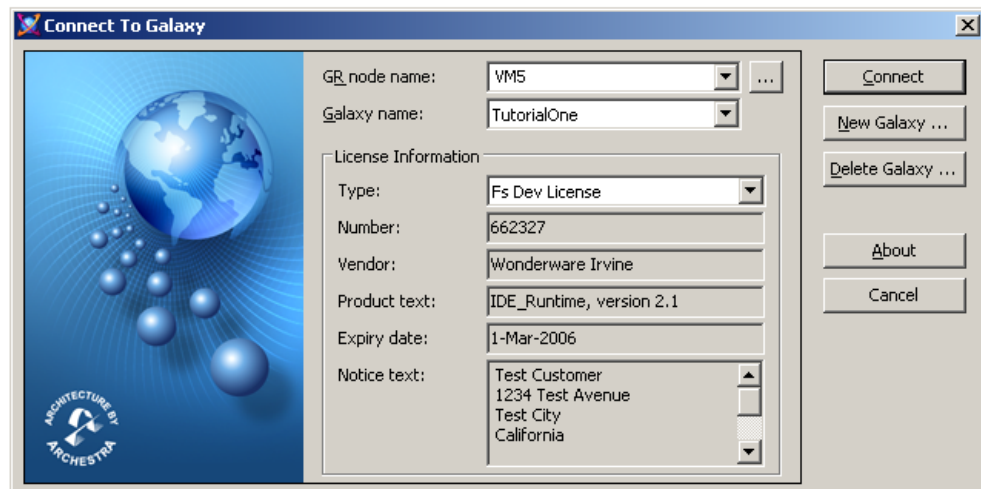
You can have many Galaxies in the Galaxy Repository. If you are a systems integrator, you have a Galaxy for every client, at least.



For a Galaxy to deploy objects to other computers in the Galaxy, the 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 134.

### 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 fully undeploy it. For more information about undeploying, see [Undeploying Objects](#) on page 139.

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 201.

### To delete a Galaxy

- 1 Undeploy 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**.

## Exporting a Galaxy Dump File

You can export instances and their configuration data in a Galaxy to a text file with a .CSV extension. After you export the Galaxy to a text file, you can open it in Microsoft Excel 2000 or later and edit it. 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.

See [Looking at the Galaxy Text Dump File Structure](#) on page 206 for specific information about the structure of the `.CSV` file.

Before you start, make sure you know what instances you want to export to the `.CSV` 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.

You can open this file in a text editor like 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.

## Looking at the Galaxy Text 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.

	A	B	C	D	E	F	G	H	I	J	K	L
1	; Created on: 7/4/2005 12:09:48 PM from Galaxy: Splash123											
2												
3	:TEMPLATE=\$ADevAlarm											
4	:TagName	Area	SecurityGr	Container	Contained	ShortDesc	Executionf	Executionf	UDAs	Extension	CmdData	Auto
5	ADevAlarm_001		Default			The Analog	None		<UDAInfo>	<Extension	<CmdData	FALSE
6												
7												
8	:TEMPLATE=\$ADiscAlarm											
9	:TagName	Area	SecurityGr	Container	Contained	ShortDesc	Executionf	Executionf	UDAs	Extension	CmdData	Auto
10	ADiscAlarm_001		Default			The Switch	None		<UDAInfo>	<Extension	<CmdData	FALSE
11												
12												
13	:TEMPLATE=\$AROCAAlarm											
14	:TagName	Area	SecurityGr	Container	Contained	ShortDesc	Executionf	Executionf	UDAs	Extension	CmdData	Auto
15	AROCAAlarm_001		Default			The Analog	None		<UDAInfo>	<Extension	<CmdData	FALSE
16												
17												
18	:TEMPLATE=\$AValueAlarm											
19	:TagName	Area	SecurityGr	Container	Contained	ShortDesc	Executionf	Executionf	UDAs	Extension	CmdData	Auto
20	AValueAlarm_001		Default			The Analog	None		<UDAInfo>	<Extension	<CmdData	FALSE
21												
22												
23	:TEMPLATE=\$CBoolean											
24	:TagName	Area	SecurityGr	Container	Contained	ShortDesc	Executionf	Executionf	UDAs	Extension	CmdData	UDABool
25	CBoolean_001		Default			The FieldR	None		<UDAInfo>	<Extension	<CmdData	FALSE

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
```

## 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 *Looking at the Galaxy Text Dump File Structure* on page 206 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.

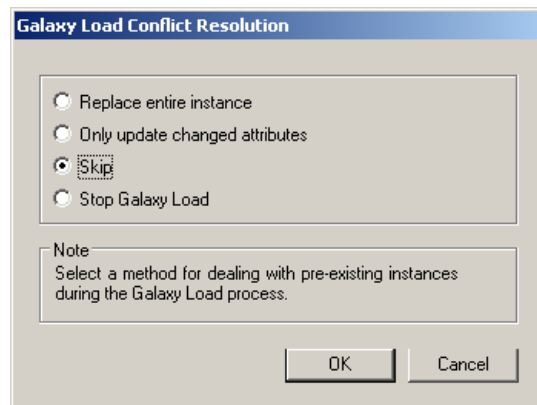
---

### 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 in a Galaxy being set 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.

Version	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.

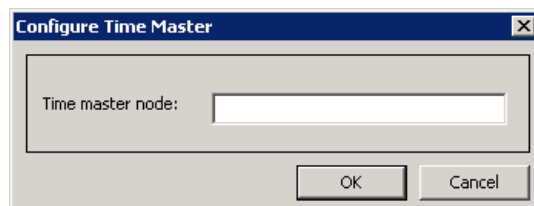
If your time master computer is a non-Galaxy node, regardless of operating system, you must also change certain Registry keys. Contact Wonderware Technical Support to obtain the files that can set those Registry keys.

**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. This message means 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

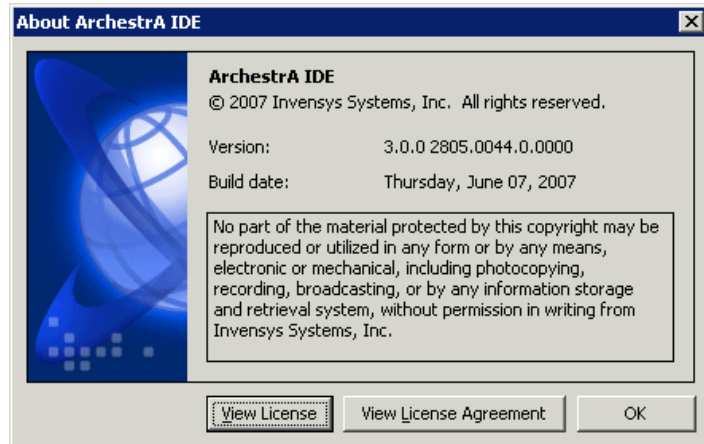
Use the License Utility to view information about your license and End-User License Agreement. Until any problems are resolved, you cannot:

- 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 217.

### To view ArchestrA IDE version information

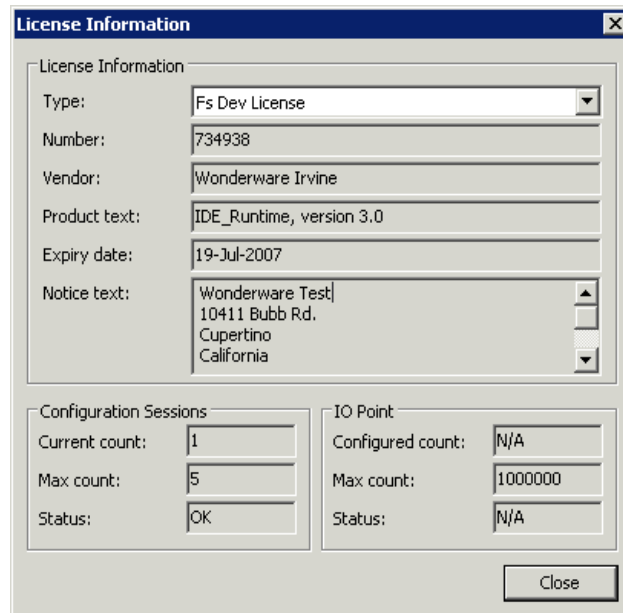
- 1 On the **ArchestrA IDE** menu, click **Help**.
- 2 Click **About ArchestrA IDE**. The **About ArchestrA IDE** dialog box appears, showing the copyright notice, version, and build date for your installed ArchestrA IDE application.



- 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 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 exceeds 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.

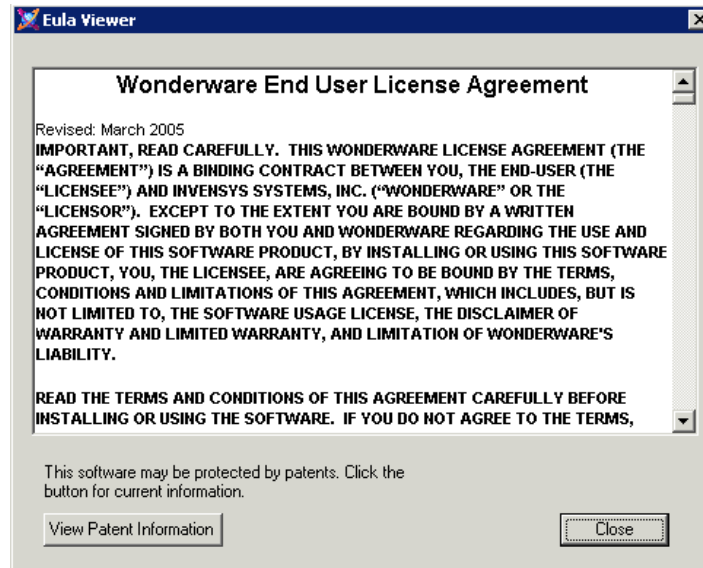
You see one of three states:

- OK: Everything is fine.
- Exceeded: Configured I/O points count exceeds 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**.

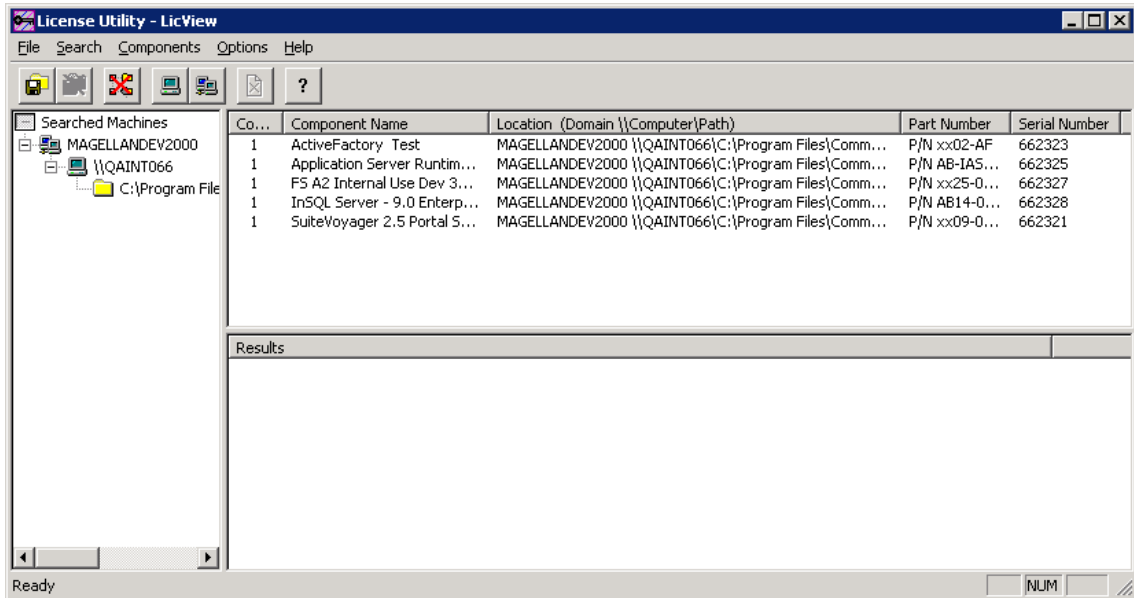


## Updating a License

After you get your new license from Wonderware, you must install it to update your Galaxy Repository license. Follow the steps below.

### 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 on your system, 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 ArcestrA-enabled software installed must be able to communicate with each other.

Two items must be considered to ensure communication between nodes:

- ArcestrA user accounts
- Multiple network interface cards in computers

All Platforms communicate with several attributes on the Galaxy Repository (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.

## About ArcestrA User Accounts

Communication occurs with an ArcestrA-specific user account set up during the initial installation of each ArcestrA component on each computer, including the IDE.

---

**WARNING!** The user account that is used for ArcestrA 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 properly.

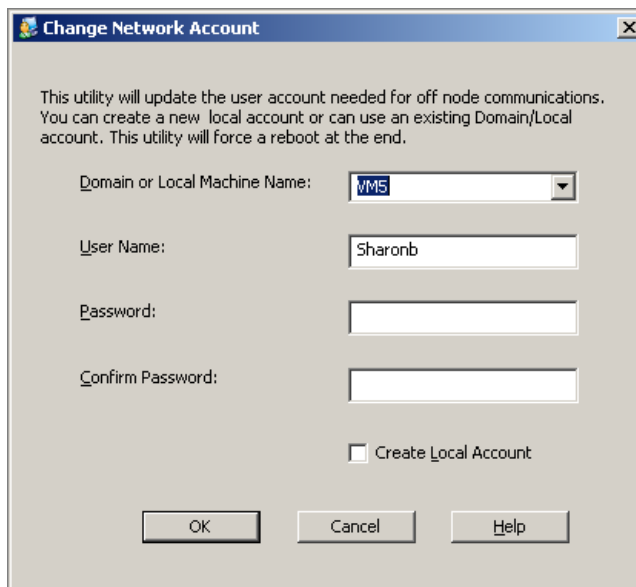
---

If you delete the ArcestrA 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 ArcestrA-enabled software installed.

### To recreate an ArcestrA 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 ArcestrA system, create any account.
  - In a multi-node ArcestrA system, create the same user account with the same user name and password on all other ArcestrA 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 ArcestrA 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 ArcestrA nodes.

If a node contains multiple NICs for redundancy reasons, see *Working with Redundancy* on page 223 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 ArcestrA 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 further nodes.

### To define the correct order

- 1 Open the **Network and Dial-up Connections** dialog box and rename each card with a clearly identifiable function, for example, `Supervisory Net` and `PLC net`.

Opening the **Network and Dial-up Connections** dialog box varies, depending on the version of Windows you are using.

- 2 Order the network cards properly. On the **Advanced** menu, click **Advanced Settings**. 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 an RMC connection for ArcestrA redundancy, the supervisory net must be listed first and the others can be listed in any other position.

- 3 Click **OK**.

## 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 ArcestrA 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 and Dial-up 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**.



---

# Chapter 12

## Working with Redundancy

You can set up and run Application Server in a redundant environment. In the Wonderware® 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 your *FactorySuite A<sup>2</sup> Deployment Guide*.

### 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 redundancy pair. The ArcestrA 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 engine 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.

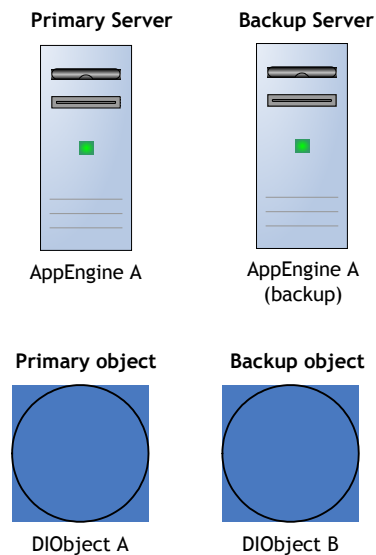
In a redundant system, install and configure the primary OPC server on the backup engine node.

---

**Important** For AppEngine redundancy, ArcestrA supports a one-to-one topology in which the computers hosting the Primary and Backup object sets must be connected by crossover cable and have fixed IP addresses.

---

When you configure redundancy, you configure the Primary object and the Backup object.



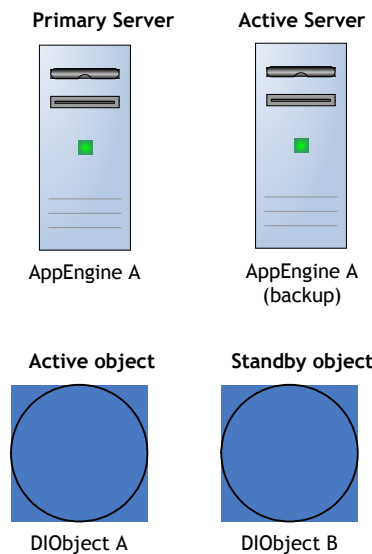
- **Primary object:** The main or central object that provides the functionality during run time. For AppEngines, this is the object you enable for redundancy. For data acquisition, this is the DIObject you intend to use first as your data source in run time.



- **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 DIO object 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 RedundantDIO object.
- **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 RedundantDIO object.

In the AppEngine redundancy environment, the Active and Standby objects monitor each other's status and switch when failure conditions occur.

In the data acquisition environment, the RedundantDIO object monitors the status of the two DIO object 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.

## 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.

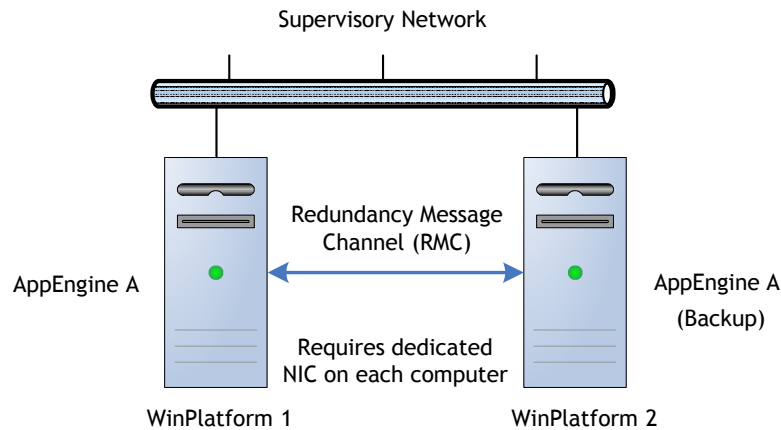
---

**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 220.

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. For more information, see the Windows Help.

**2** Click **Advanced Settings**. If the machine 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 of **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.

**3** 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 220.

**4** In Application Server, open the WinPlatform in the Object Editor for the computer you just configured.

**5** 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.

**6** 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.

There are four deployment statuses:

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 **Enable Redundancy** 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.

---

**Note** The actual engine failure timeout 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.

---

- 4 On the **General** tab, clear the **Restart the engine when it fails** check box.



---

**Important** If you enable redundancy in an AppEngine, do not select the **Restart the engine when it fails** option on the General page of the AppEngine's editor.

---

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 ArcestrA 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 233.

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

Condition	Deploy Both Primary and Backup Objects	Cascade Deploy Allowed
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 AutomationObjects, 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.

---

### AutomationObjects at Run Time

AutomationObjects are always assigned to the Primary AppEngine in the configuration environment.

During run time, AutomationObjects 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 above.

## 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 up. 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 executes 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 timeout 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 *FactorySuite A<sup>2</sup> Deployment Guide*.

## 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 220.

## Generating Alarms

When failover conditions occur, the ArcestrA system reports alarms to the 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 up 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 *FactorySuite A<sup>2</sup> Deployment Guide*.

## 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.

### 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 DIOjects

Deployment for data acquisition redundancy is the same as individually deploying unrelated objects. No special conditions apply to each DIOject data source and the RedundantDIOject.

See *Deploying your Galaxy* on page 131 for more information about deploying objects.

## What Happens in Run Time

The three objects in the data acquisition redundancy scheme (RedundantDIOject and its two DIOject data sources) work like any other ArcestrA object when deploying, alarming, and historizing. They have no special redundancy-related states or restrictions.

During run time, the RedundantDIOject monitors the status of the DIOject data sources, and handles the switching from Active to Standby object if failure conditions occur.

### RedundantDIOject and PLC Connectivity

For the RedundantDIOject, 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 RedundantDIOject to communicate with DIOjects, you should configure the PingItem attribute for each scan group.



---

# Glossary

<b>application</b>	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.
<b>Application Engine (AppEngine)</b>	A scan-based engine that hosts and executes the runtime logic contained within AutomationObjects.
<b>ApplicationObject</b>	An AutomationObject that represents some element of your production environment. This can include things like <ul style="list-style-type: none"><li>• an automation process component. For example, a thermocouple, pump, motor, valve, reactor, or tank</li><li>• or associated application component. For example, function block, PID loop, Sequential Function Chart, Ladder Logic program, batch phase, or SPC data sheet</li></ul>
<b>Application Server</b>	<p>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 a Data Access Server (DAServer) for device communications.</p> <p>An Application Server can be distributed across multiple computers as part of a single Galaxy namespace.</p>
<b>Application view</b>	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.
<b>Archestra</b>	The distributed architecture for supervisory control and manufacturing information systems. It is an open and extensible technology based on a distributed, object-based design.

<b>ArchestrA Object Toolkit</b>	A programmer's tool to create new ApplicationObjects and Device Integration Object (DIOjects) templates, including configuration and runtime implementations. Includes a tool to build DI Objects and create unique Domain Objects that interact with DIOjects in the ArchestrA environment.
<b>ArchestrA Symbol</b>	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.
<b>area</b>	A logical grouping of AutomationObjects that represents an area or unit of a plant. It is used to group related AutomationObjects for alarm, history, and security purposes. It is represented by an area AutomationObject.
<b>area object</b>	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.
<b>assignment</b>	The designation of a host for an AutomationObject. For example, an AppEngine AutomationObject is assigned to a WinPlatform AutomationObject.
<b>attribute</b>	An externally accessible data item of an AutomationObject.
<b>attribute reference string</b>	A text string that references an attribute of an AutomationObject.
<b>AutomationObject</b>	An object type that represents permanent things in your plant, such as ApplicationObject or Device Integration Object (DIOjects), with user-defined, unique names within the Galaxy. It provides a standard way to create, name, download, execute, and monitor the represented component.
<b>Backup Application Engine</b>	The object created by the ArchestrA infrastructure when the Primary object is enabled for redundancy. See redundancy for further details.
<b>base template</b>	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 templates names start with a dollar sign (\$).
<b>Block Read Group</b>	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.

---

<b>Block Write Group</b>	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.
<b>Bootstrap</b>	The base ArcestrA service which is required on all ArcestrA computers. It provides the base software environment to enable a platform and allows a computer to be included in the Galaxy Namespace.
<b>Change log</b>	The revision history that tracks the life cycle activities of ArcestrA Objects, such as object creation, check in/check out, deployment, and import/export.
<b>change propagation</b>	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.
<b>check in</b>	IDE operation for making a configured object available for other users to check out and use.
<b>check out</b>	IDE operation for the purpose of editing an object. It makes the item unavailable for other users to check out.
<b>Checkpoint</b>	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.
<b>compound object</b>	An ApplicationObject that contains at least one other ApplicationObject.
<b>contained name</b>	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.
<b>containment</b>	A hierarchical grouping that allows one or more AutomationObject to exist within the name space of a parent AutomationObject and be treated like parts of the parent AutomationObject. Allows for relative referencing to be defined at the template and instance level.
<b>DAGroup</b>	A data access group associated with Device Integration Object (DIOjects). It defines how communications are achieved with external data sources. It can be a Scan Group, Block Read Group or Block Write Group.

<b>DAServer Manager (DAS Manager)</b>	The System Management Console (SMC) snap-in supplied by the Data Access Server (DAServer) that provides the required interface for activation, configuration, and diagnosis of the DAServer.
<b>Data Access Server (DAServer)</b>	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.
<b>Data Access Server Toolkit (DAS Toolkit)</b>	A developer tool that can build a Data Access Server (DAServer).
<b>Deployment</b>	The operation which instantiates an AutomationObject instance in the ArcestrA runtime. 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.
<b>Deployment view</b>	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.
<b>derivation</b>	The creation of a new template based on an existing Template.
<b>Derivation view</b>	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.
<b>derived template</b>	Any template with a parent template. Derived templates inherit the attributes of the parent template. You can changes these attributes in the derived template.
<b>Device Integration Object (DIObjects)</b>	An AutomationObject that represents the communication with external devices or software. DI Objects run on an Application Engine (AppEngine), and include DINetwork Objects and DIDevice Objects.
<b>DIDevice Object</b>	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.
<b>DINetwork Object</b>	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.

---

<b>element</b>	Basic shapes, such as rectangles, lines, and text elements, and controls you can use to create an ArcestrA Symbol to your specifications.
<b>Engine Object</b>	An ArcestrA system-enabled object that contains Local Message Exchange and provides a host for ApplicationObjects, Device Integration Object (DIOObjects) and area objects.
<b>event record</b>	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.
<b>export</b>	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.
<b>FactorySuite Gateway</b>	A Microsoft Windows application program that acts as a communications protocol converter. Built with the ArcestrA DAS Toolkit, FS Gateway links clients and data sources that communicate using different data access protocols.
<b>Galaxy</b>	The entire application. The complete ArcestrA 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 PCs that constitute an automation system. This is referred to as the Galaxy Namespace.
<b>Galaxy database</b>	The relational database containing all persistent configuration information like templates, instances, security, and so on in a Galaxy Repository.
<b>Galaxy Database Manager</b>	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).
<b>GalaxyObject</b>	The object that represents a Galaxy.
<b>Galaxy Repository</b>	The software sub-system consisting of one or more Galaxy databases.
<b>Graphic Toolbox</b>	The part of the IDE main window that shows a hierarchy of graphic toolsets, which contain ArcestrA Symbols and client controls.
<b>hierarchical name</b>	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.

<b>Historical Storage System (Historian)</b>	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.
<b>host</b>	The parent of a child instance in the deployment view. Example: a Platform instance is a Host for an Application Engine (AppEngine) instance.
<b>import</b>	The act of reading a package file (.aaPKG) and using it to create AutomationObject instances and templates in the Galaxy Repository.
<b>instance</b>	An object derived from a template. You deploy instances to the runtime environment.
<b>instantiation</b>	The creation of a new instance based on a corresponding template.
<b>Integrated Development Environment (IDE)</b>	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.
<b>InTouch View</b>	InTouch View clients are InTouch runtime clients that solely use of the Application Server for its data source. In addition, standard InTouch runtimes can leverage Application Server with the addition of a Platform license.
<b>InTouchViewApp object</b>	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.
<b>I/O count</b>	Number of I/O points being accessed into the 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 PCs are in the system.
<b>Message Exchange</b>	The object to object communications protocol used by Application Server. Message Exchange includes  LMX            communication between objects NMX            communication between IAS nodes.



---

<b>Model view</b>	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.
<b>object</b>	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.
<b>object extensions</b>	The capability to add additional functions to an AutomationObject while not changing the object's original behavior. Can be added to derived templates and object instances. They include Scripts, User Defined Attributes (UDAs) and Attribute Extensions.
<b>Object Viewer</b>	A utility in which you can view the attribute values of the selected object in runtime. 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.
<b>OffScan</b>	The state of an object that indicates it is idle and not ready to execute its normal runtime processing.
<b>OnScan</b>	The state of an object in which it is performing its normal runtime processing based on a configured schedule.
<b>Operations view</b>	The area in the IDE that shows the results of validating the configuration of objects.
<b>package definition file (.aaPDF)</b>	The standard description file that contains the configuration data and implementation code for a base template. File extension is .aaPDF.
<b>package file (.aaPKG)</b>	The standard description file that contains the configuration data and implementation code for one or more objects or templates. File extension is .aaPKG.

<b>Platform Count</b>	<p>Number of PCs in the Galaxy. Each Workstation and/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. Each InTouch Terminal Services Session needs a Application Server Terminal Services Session License.</p> <p>A Platform License includes a per seat FSCAL2000 with Microsoft SQL Server CAL. Stand-alone computers only hosting InSQL Servers or a remote Data Access Server (DAServer) do not need a platform license.</p>
<b>Platform Manager</b>	<p>This utility is an extension snap-in to the ArcestrA System Management Console (SMC). Provides Galaxy application diagnostics by allowing you to view the runtime 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.</p>
<b>platform object</b>	<p>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.</p>
<b>Primary Application Engine</b>	<p>The object created by the ArcestrA infrastructure when the Backup object is created through redundancy. See redundancy for further details.</p>
<b>properties</b>	<p>Data common to all attributes of objects, such as name, value, quality, and data type.</p>
<b>proxy object</b>	<p>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.</p>
<b>redundancy</b>	<p>Two computers: One executes objects. The other is a stand by.</p>
<b>RedundantDIObject</b>	<p>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.</p>
<b>Redundant Message Channel</b>	<p>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.</p>

---

<b>reference</b>	A string that refers to an object or to data within one of its attributes.
<b>relative reference</b>	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”, and “MyPlatform”.
<b>remote reference</b>	The ability to redirect ArchestrA object references or references to remote InTouch tags. The new script function that redirects remote references at runtime is <code>IOSetRemoteReferences</code> .
<b>Scan Group</b>	A <code>DAGroup</code> that requires only the update interval be defined. The data is retrieved at the requested rate.
<b>Scan State</b>	The Scan State of an object in runtime. This can be either <code>OffScan</code> or <code>OnScan</code> .
<b>security</b>	Application Server security is applied to IDE, System Management Console (SMC), and the runtime data level. At the runtime data level which centralizes the definition of all permissions to the <code>ApplicationObjects</code> . These <code>ApplicationObjects</code> 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 <code>ApplicationObjects</code> at runtime are mapped to the objects by user-defined roles. These roles can be mapped directly to existing groups in a Microsoft Domain or workgroup.
<b>System Management Console (SMC)</b>	The central run-time system administration/management product where you perform all required runtime administration functions.
<b>system object</b>	An object that represents an area, platform or engine.
<b>TagName</b>	The unique name given to an object. For example, for a given object, its <code>TagName</code> = <code>V1101</code> and its <code>HierarchicalName</code> = <code>Line1.Tank1.InletValve</code> .
<b>template</b>	An object containing configuration information and software templates used to create a derived template and/or instance.
<b>Template Toolbox</b>	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.
<b>Toolset</b>	A named collection of templates shown together in the IDE Template Toolbox.

<b>User Defined Attributes (UDA)</b>	Allow you to add new functionality to an object. An attribute is added to an object at configuration time.
<b>ViewEngine object</b>	Hosts InTouchViewApp objects. The ViewEngine object supports common engine features such as deployment, undeployment, startup, and shutdown.
<b>WinPlatform object</b>	An object that represents a single computer in a Galaxy, consisting of a systemwide message exchange component, a set of basic services, the operating system, and the physical hardware. This object hosts the Application Engine (AppEngine).

# Index

## A

- active object 225
- active object, redundancy 225
- adding custom help to objects 71
- alarm
  - categories, setting 157
  - clients and undeploying 142
  - distributors 158
  - examples 154
  - extension group, using text strings 77
  - extensions 115, 124, 157
  - subscription 159
- alarms
  - Area AutomationObjects 159
  - configuring 155
  - definition 153
  - disabling 160
  - enabling 160
  - examples 154
  - grouping 159
  - InTouch syntax in IAS 162
  - specifying 157
  - subscribing 159
- aliases
  - rules for locking in scripts 111
  - using in scripts 110
- allowed IO, viewing license information 213
- analog device objects 121
- AppEngine
  - historized data 150
  - history configuration 150
  - object 40
  - object, definition 40
  - redundancy 226
  - redundancy states 234
- AppEngine states, redundancy 234
- application
  - architecture, ArcestrA 167
  - templates 39
- Application views
  - using 21
- ApplicationObject containment 53
- ArcestrA
  - messaging system 167
  - user accounts 219
- area
  - AutomationObject 159
  - object 41
  - objects, definition 41
- Area AutomationObjects, alarms 159
- assigning users to security roles 195
- attributes
  - dynamic 175
  - hardware register 175
  - hidden in UDAs 104

- historizing 144
  - in runtime, writing to UDAs 66
  - keeping runtime values 140
  - locking in instances 65
  - locking in templates 65
  - locking in UDAs 66
  - property, finding 78
  - runtime 175
  - security 68
  - security locked in parent 69
  - unlocking in instances 65
  - unlocking in templates 65
  - automatically initializing scripts 103
  - AutomationObjects at runtime 233
- B**
- backing up Galaxies 201
  - backup object 225
  - backup object, redundancy 225
  - backup server, redundancy 224
  - base templates 37, 39
    - importing 98
    - modifying 21
  - bit field access 82
  - boolean attributes, alarm extensions 124
  - boolean data types, output extensions 122
  - boolean label extension 77
    - using a text string 77
  - boolean label extensions 126
  - Bootstrap, hosting Galaxies 13
- C**
- cascade
    - deploy 134
    - deploy, selecting 136
  - changing
    - Galaxies 203
    - passwords 33
    - users 34
  - check-in comments, setting 31
  - checking objects
    - in 90
    - out 89
  - clients and alarms 159
  - communication at runtime 233
  - components in Galaxies 13
  - configuration, Object Editor Extension page 157
  - configure history, finding object help 144
  - configure user information 31
  - configuring
    - alarm providers 155
    - data acquisition redundancy 240
    - historize data 146
    - history 144
    - multiple network cards 220
    - network cards 227
    - objects to store history 147
    - redundancy 228
    - security 190
    - user information 31
    - WinPlatforms and AppEngines for history 150
    - Wonderware Historian 146
  - configuring bit field access 82
  - configuring computers for redundancy 224
  - configuring redundancy
    - AppEngine 228
    - ordering network connections 227
  - connecting
    - existing Galaxy 17
    - remote node for the first time 198
  - contained
    - names 51, 57, 94
    - templates, creating new 49
  - contained objects
    - naming conventions 57, 60
    - viewing 60
  - containment
    - definition 50
    - examples 58
    - names, scripting 56
    - naming conventions 57, 60
    - relationships, viewing 23, 57
    - templates 50
  - creating
    - contained templates 50
    - derived templates 48
    - extensions 113
    - Galaxy, security restrictions 15
    - instances 93
    - new Galaxy 15

- object extensions 114
  - scripts 107
  - toolsets 46, 47
  - UDAs 102
  - creating user accounts 219
  - crossover cable, wiring redundancy 224
  - .csv files
    - characters in 207
    - editing 204
    - importing 208
    - structure 206
    - time formats 208
  - custom
    - help, locating folders 71
    - toolsets 45
  - customizing
    - derived templates 48
    - information for a Galaxy 31
    - object help 71
    - your workspace 29
- D**
- data
    - acquisition redundancy 240
    - types and bit field access 82
  - data types, configure history 145
  - default, templates 37, 39
  - deleting
    - extensions 114
    - Galaxies 204
    - objects 96
    - redundant AppEngines 230
    - roles 197
    - security groups 197
    - toolsets 47
    - users 197
  - deleting Galaxies, before you start 204
  - deploy host objects first 134
  - deploying
    - AppEngine objects 230
    - DINetwork objects 134
    - history 149
    - imported objects 100
    - instances 35
    - objects 134
    - redundant diobjects 241
    - templates 35
  - deployment
    - error messages 138
    - redundancy 241
  - deployment status
    - pair deployed 228
    - pair undeployed 228
    - partial deployed 228
    - partial undeployed 228
    - redundancy objects 240
    - viewing icons 135
  - Deployment view
    - assignment relationships 24
    - using 24
  - Derivation view
    - parental relationships 27
    - using 27
  - derivation, viewing objects 27
  - derived templates 39, 42
    - contained names 57
    - creating 48
    - customizing 48
    - nesting 49
  - deriving templates from derived templates 49
  - determining Galaxy status 133
  - device intergration templates 39
  - devived locked scripts 112
  - DIDevice objects, definition 39
  - differences between alarms and events 153
  - DINetwork objects
    - configuration limits 25
    - definition 39
    - deploying 134
  - DIObject data sources 240
  - disabling
    - alarms 160, 161
    - redundancy 230
  - disk space requirements 218
  - DNS settings, configuring 221
  - documentation conventions 11
  - dynamic attributes 175
- E**
- editing
    - base templates 21
    - objects 61
    - scripts 105

- editing .csv files 206
- editing .csv files, characters in 207
- enabling alarms 160
- event
  - distributors 158
  - subscription 159
- events
  - definition 153
  - examples 154
- execution order of objects, setting 71
- expanding tabs 30
- exporting
  - Galaxy dump file 204
  - Galaxy file structure 206
  - instances 96, 204
  - object help 205
  - objects 96, 204
  - script function libraries 97
  - script libraries 205
  - scripts 96, 97
- exporting templates 204
- extension inheritance 113
  - characteristics 113
- Extensions page 76, 113
- extensions, deleting 114

## F

- failover and redundancy 223
- field reference object 121
- finding
  - help folders 71
  - object attributes 78
  - objects 177
- floating
  - areas 29
  - views 29
- formatting reference strings 171

## G

- Galaxies
  - components of 13
  - default users 184
  - definition 13
  - deleting 204
  - disk space requirements 218
  - location on the network 13
  - logging in to 33
  - logging out of 33

- managing multiple 203
- naming conventions 16
- opening with security 17
- restoring 201
- synchronizing time 210
- synchronizing time in Windows 2000 or XP 210
- validating 92

## Galaxy

- components, deploy 14
- creating 15
- database, Galaxy Repository 14
- repository, hosting Galaxies 13
- Repository, specifying 15
- reserved names 16
- status, determining 133
- text dump 206

## Galaxy Browser 78

## Galaxy dump files

- characters 207
- importing 208
- time formats 208

## general Object Editor layout 64

## generating

- alarms 238
- history 239

## group

- lock icons 69
- locking 69
- security 69

## grouping alarms 159

## H

## hardware register attributes 175

## heartbeats 218

## help header structure 63

## hidden attributes, UDAs 104

## hiding

- areas 30
- views 30

## hierarchical

- names 51, 57, 95
- names, viewing 23

## historized data

- AppEngines 150
- store forward 149
- when data is stored 148
- WinPlatforms 150



- historizing
    - attributes 144
    - data, installing Wonderware Historian 146
  - history
    - deploying 149
    - undeploying 149
  - History extension 125
  - history, data types, configuring 145
  - host attributes 207
  - hosting, multiple Galaxies in one Galaxy repository 212
  - HTML editors for customizing object help 71
- I**
- I/O licenses 212
  - icons, deployment status 135
  - IDE objects in runtime 132
  - IDE, views 18
  - importing
    - base templates 98
    - .csv files 208
    - Galaxy load files 208
    - objects 98
    - objects and deploying 100
    - objects and security groups 100
    - objects with scripts 100
    - script function libraries 100
  - inheriting attributes, parent and child relationships 35
  - Input extensions 114
  - Input extensions, data quality 123
  - InputOutput
    - attributes in scripts 118
    - extension 116
  - InputOutput extensions, data quality 123
  - installing Wonderware Historian 146
  - instances 37
    - creating 93
    - defined 37
    - deploying 35
    - exporting 96, 204
    - locking attributes 65
    - naming conventions 93
    - on other computers 24
    - reserved names 94
    - unlocking attributes 65
  - InTouch
    - alarm and event client 162
    - alarm syntax 162
    - references in a Galaxy 178
  - IO, viewing allowed 213
  - IP address, network settings 221
- L**
- .lic files, using 217
  - license information, viewing 213
  - licensed objects
    - objects
      - licenses 212
  - licensing issues 212
  - location of historian 146
  - locked scripts in child objects 112
  - locking
    - aliases in scripts 111
    - scripts 111
    - template attributes 65
    - UDAs 102
  - logging
    - in to disconnected networks 198
    - in to Galaxies 33
    - in to Galaxies, security enabled 33
    - out of Galaxies 33
- M**
- managing templates with toolsets 45
  - manually
    - initializing scripts 103
    - validating objects 92
  - MDAS
    - historizing data 146
    - using 146
  - Message Exchange and attributes 168
  - mixed or native domains 199
  - Model view
    - containment relationships 21
    - opening 21
  - moving
    - objects, undeploying 21
    - views 29

- multiple
  - accounts per user 187
  - network cards in one computer 220
- multiple Galaxies 203

## N

- naming conventions
  - containment 57, 60
  - Galaxies 16
  - instances 93
  - scripts 107
  - templates 49
  - toolsets 46
  - UDAs 104
- nesting templates 37, 52
- network
  - configuring DNS settings 221
  - configuring IP address 221
- network cards
  - multiple in one computer 220
  - redundancy 226
  - setting 220
  - specifying the order 220
- network, outages 161

## O

- object
  - derivation, viewing 27
  - properties, viewing 43
  - relationships, viewing 22
- object attributes, finding 78
- Object Editor
  - getting help 63
  - opening 61
- object help
  - adding images 71
  - exporting 205
  - finding 144
  - HTML editors for customizing 71
- Object Information page 70
- objects
  - checking in 90
  - checking out 89
  - deploying 134
  - disabling alarms 161
  - enabling alarms 160
  - exporting 96, 204

- help folders, finding 71
- importing 98
- in runtime 132
- inheriting extensions 113
- opening Object Editor 61
- redeploying 139
- specific information 121
- undeploying 139
- validating manually 92
- Objects Information page 70
- Operations view, opening 28
- ordering network cards 220
- Output
  - extension 115
  - functionality 121
- Output extensions, data quality 123

## P

- parent and child
  - relationships, viewing 27
- passwords, changing 33
- planning for deploying 131
- Primary AppEngine, viewing
  - attributes 80
- primary object, redundancy 224
- primary server, redundancy 224
- propagating changes, templates 38

## R

- reassigning objects, undeploying 21
- recreating ArcestrA user account 219
- redeploying
  - objects 139
  - objects, uploading changes first 140
- redundancy
  - alarm generation 238
  - AppEngine states 234
  - ApplicationObjects configuration 230
  - configuring templates 229
  - during deployment 233
  - history generation 239
  - pair 224
  - runtime 225
  - setting network cards 226
  - templates 229
  - WinPlatforms
    - configuration 226
  - wiring for 224

- Redundancy, page 228
  - redundant engines, undeploying 232
  - redundant partner deploy, selecting 136
  - RedundantDIObject 223
  - reference strings 168
  - references in scripts, validating 90
  - referencing objects, Galaxy Browser 78
  - relationships, viewing parent/child 27
  - renaming
    - contained objects 60
    - contained objects, updating
      - references 61
    - objects 94
  - reorganizing the workspace 30
  - required port for Wonderware Historian 146
  - required software 211
  - reserved names
    - instances 94
    - list of 16
    - templates 49
  - resetting
    - the workspace 30
    - workspace 30
  - restoring Galaxies 201
  - return views to the default locations 30
  - reusing existing objects 98
  - rules for locking
    - aliases in scripts 111
    - scripts 111
  - runtime
    - and bit field access 82
    - attributes 175
    - configuration, uploading 140
    - how objects deploy from the IDE 132
  - runtime environment, scripting 106
- S**
- script function libraries
    - exporting 97
    - importing 100
  - script libraries, exporting 205
  - scripting
    - containment names 56
    - UDAs 103
  - scripts
    - aliases, using 110
    - execution types 107
    - exporting 96
    - initializing Startup scripts 103
    - InputOutput attributes 118
    - locked in child objects 112
    - locking rules 111
    - manually initializing 103
    - naming conventions 107
    - timing 107
    - validating 90, 106
  - Scripts page 72
  - security
    - Galaxy users 184
    - groups, default 184
    - groups, importing objects 100
    - icon not available 69
    - icons 69
    - opening Galaxies 17
    - options in attributes 69
    - restricting access to attributes 68
    - restrictions, creating new Galaxies 15
    - security roles 184
  - setting network cards 220
  - setting prompts for check-in
    - comments 31
  - setting, alarms 158
  - single scan cycles 121
  - specifying
    - alarms 157
    - text strings for states 77
  - standby object, redundancy 225
  - Startup scripts, initializing 103
  - store forward, historized data 149
  - storing history, configuring 147
  - subscribing to alarms 159
  - switch objects 121
  - synchronization schedule 210
  - synchronizing
    - time across a Galaxy 210
    - views 30
  - synchronizing time across networks 210
  - synchronizing time across networks,
    - Windows 2000 or XP 210
  - system templates 40
- T**
- tagnames 51, 57, 94
  - tagnames, containment 54
  - technical support, contacting 12

Template Toolbox, location 19  
 templates 37
 

- alarm extensions 124
- application 39
- base 37, 39
- classes 39
- configuring redundancy 229
- contained names 57
- containment 50
- creating derived 48
- default 37, 39
- defined 37
- deploying 35
- derived 39, 42
- device integration 39
- exporting 204
- inheriting extensions 113
- managing 45
- naming conventions 49
- nesting 37
- nesting levels 52
- propagating changes 38
- reserved names 49
- system 40

 text strings
 

- alarm extension group 77
- boolean label extension 77

 time formats, .csv files 208
 time master
 

- specifying 210
- specifying in Windows 2000 or XP 210

 toolsets
 

- creating 46, 47
- custom 45
- definition 45
- deleting 47
- managing 45
- naming conventions 46

 troubleshooting 236

## U

UDA naming conventions 104
 UDAs
 

- adding to objects 102
- and scripting 103
- hidden attributes 104
- locking attributes 66

- locking in child objects 102
  - page 74
  - scripting 103
- undeploying 139
  - alarm clients 142
  - AppEngine objects 232
  - history 149
  - objects 139
    - objects before moving 21
- undeploying redundant engines 232
- undeployment conditions 142
- unlocking, template attributes 65
- updating your license 212, 217
- uploading runtime configuration 140
- user accounts 219
- user defaults, setting 31
- users, default 184

## V

validating
 

- Galaxies 92
- manually 92
- object manually 92
- objects 90
- objects, viewing results 28
- references in scripts 90
- scripts 90, 106

 viewing
 

- attributes in objects 174
- containment relationships 60
- cross references 175
- license information 213
- object properties 43
- object relationships 22
- objects, assignment relationships 24
- references 175

 viewing your licenses 212
 views in the IDE 18
 views, synchronizing 30

## W

Welcome 11
 Windows
 

- 2000 domains 210
- XP domains 210

 Windows user accounts 219
 WinPlatform
 

- object 40

- WinPlatforms
  - configuration in redundancy 226
  - historized data 150
  - history configuration 150
- Wonderware Historian
  - configuring for history 146
  - installing 146
  - required port 146
- working
  - extensions 113
  - UDAs 102
- workspace, resetting 30
- writing
  - scripts 105
  - to attributes in runtime, UDAs 66

