

Training Manual

Revision A
August 2013
Part Number 11-HM-10012

InTouch 2012 R2 - Part 2



INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE.

© 2013 by Invensys Systems, Inc. All rights reserved. No part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, without the express written permission of Invensys Systems, Inc. Except where noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.

Invensys and the author(s) assume no responsibility for errors or omissions and no liability is assumed for damages resulting from the use of the information contained herein. Use of the Invensys software described in this document is subject to the terms of the applicable Invensys Systems, Inc./Wonderware license. These terms include provisions that limit your rights such as use restrictions, disclaimers of warranties and limitations of Wonderware/Invensys liability. By installing or using the Invensys software, you agree to accept all of the terms of the applicable Invensys Systems, Inc./Wonderware license. A copy of the applicable license will be displayed upon initial installation of the software. If a copy of the license is not displayed or you require an additional copy of the license, you may obtain one from Invensys' Wonderware business unit by calling 1.949.727.3200 or by sending an e-mail to support@wonderware.com.

Invensys, Wonderware, ArchestrA, Factelligence, IntelaTrac, InBatch, InControl, IndustrialSQL Server, InSQL, InTouch, InTrack, QI Analyst, SCADAAlarm, SuiteLink, SuiteVoyager, WindowMaker, WindowViewer are trademarks and registered trademarks of Invensys plc, its subsidiaries and affiliated companies. All other brands and product names may be the trademarks or service marks of their respective owners.

Table of Contents

Module 1	Introduction	1-1
	Section 1 – Course Introduction.....	1-3
	Section 2 – InTouch HMI Overview	1-17
	Section 3 – System Requirements, Licensing, and Support.....	1-21
Module 2	Application Setup.....	2-1
	Section 1 – Managing InTouch Applications Using ArchestraA IDE.....	2-3
	Lab 1 – Galaxy and Application Setup.....	2-7
	Section 2 – Tag Functionality.....	2-15
	Lab 2 – Tag Viewer.....	2-21
Module 3	Application Navigation Tools.....	3-1
	Section 1 – Visualization and Indirect Addressing	3-3
	Lab 3 – Indirect Addressing	3-7
	Section 2 – The Show Symbol Animation Link	3-39
	Lab 4 – Show Symbol Animation Link	3-41
	Section 3 – The ShowGraphic Script Function	3-51
	Lab 5 – Basics of the ShowGraphic Function	3-53
	Section 4 – Panel Display	3-71
	Lab 6 – Panel Navigation.....	3-73
Module 4	Enhanced Application Development.....	4-1
	Section 1 – Create Graphic Effects with Animation and Scripting	4-3
	Lab 7 – Drawing a Pulsating Light	4-7
	Section 2 – Grow/Shrink Functionality	4-23
	Lab 8 – Creating Re-Sizeable Symbols	4-25
	Section 3 – Modify and Enhance Existing Symbols.....	4-39
	Lab 9 – Modifying and Enhancing Existing Symbols	4-41
	Section 4 – ActiveX Controls	4-57
	Lab 10 – Using an ActiveX Control as a Universal Document Viewer.....	4-61
	Section 5 – .NET Controls	4-71
	Lab 11 – Using .Net Controls.....	4-75
Module 5	I/O Troubleshooting	5-1
	Section 1 – Graphical Overrides	5-3
	Lab 12 – Working with Graphical Overrides	5-5
	Section 2 – Visualization and Indirect Addressing	5-9
	Lab 13 – Working with Failover.....	5-11
Module 6	Alarm Tools	6-1
	Section 1 – Basic Review of Alarming and Controls	6-3
	Lab 14 – Working with the Alarm Viewer	6-7
	Lab 15 – Working with the Alarm Tree Viewer.....	6-25
	Section 2 – Alarm Databasing	6-31
	Lab 16 – Working with the Alarm DB Manager.....	6-35
	Lab 17 – Working with the Alarm Pareto Control.....	6-55
	Section 3 – Alarm Client	6-63
	Lab 18 – Working with the Alarm Client.....	6-65
	Lab 19 – Creating Pop-Up Alarm Displays	6-75

Module 7	Tag Servers	7-1
	Section 1 – Tag Server and Tag Provider	7-3
	Lab 20 – Working with a Tag Server	7-7
	Section 2 – Alarm Providers	7-17
	Lab 21 – Distributed Alarming	7-19
	Section 3 – Access Tag Server History	7-29
	Lab 22 – Distributed History	7-31
 Module 8	 Deployment	 8-1
	Section 1 – Deploying Applications	8-3
	Lab 23 – Application Deployment.....	8-5
	Section 2 – Terminal Services.....	8-21



invenSYS[®]
Learning Services



Module 1 – Introduction

Section 1 – Course Introduction	1-3
Section 2 – InTouch HMI Overview	1-17
Section 3 – System Requirements, Licensing, and Support	1-21

Module Objectives

- Introduce the InTouch 2012 R2 – Part 2 course
- Provide an overview of InTouch HMI
- Discuss system requirements, licensing, and support

Section 1 – Course Introduction

This section describes the fundamental concepts and architecture of InTouch 2012 R2 and discusses the software solutions Wonderware offers.

Course Description

The InTouch 2012 R2 - Part 2 course is a 4 day instructor led course that provides lectures and hands-on labs designed to further expand knowledge of InTouch functionality. There will be in-depth discussions of ArchestrA Graphics, Script usage, Alarming, and I/O Functionality of an InTouch application.

Objectives

Upon completion of this course, you will be able to:

- Continue the learning of InTouch Software
- Design and modify ArchestrA graphics
- Integrate Indirect Tags into an InTouch application
- Implement the functionality of ShowSymbol and ShowGraphic features
- Configure applications to work with ActiveX and .Net controls
- Configure applications to work with advanced alarm controls
- Understand I/O monitoring and failover functionality
- Configure applications to work in conjunction with a Tag Server application
- Access alarms and history from a central source
- Deploy and update InTouch applications

Audience

- Engineers
- System Integrators
- Consultants
- Other individuals who need to use InTouch in their manufacturing processes

Prerequisites

- Completion of one of the following training courses:
 - InTouch 2012 R2 - Part 1
 - InTouch 2012 - Part 1
- Manufacturing experience

Course Outline

Module 1 – Introduction

Section 1 – Course Introduction

This section describes the fundamental concepts and architecture of InTouch 2012 R2 and discusses the software solutions Wonderware offers.

Section 2 – InTouch HMI Overview

This section introduces the InTouch Human Machine Interface (HMI) and provides an overview of the benefits of using InTouch, its key capabilities, and features.

Section 3 – System Requirements, Licensing, and Support

This section provides a detailed explanation of the system requirements necessary for System Platform, discusses Licensing details, and covers Support services.

Module 2 – Application Setup

Section 1 – Managing InTouch Applications Using ArchestrA IDE

This section describes how InTouch is integrated and used within the ArchestrA IDE. It also discusses creating a new Galaxy and importing existing applications.

Section 2 – Tag Functionality

This section covers the concepts of application and configuration of InTouch tags; including SuperTags.

Module 3 – Application Navigation Tools

Section 1 – Visualization and Indirect Addressing

This section discusses how InTouch is used to visualize data and how that data can be displayed through the use of Indirect Addressing.

Section 2 – The Show Symbol Animation Link

This section explains the use and configuration of the Show Symbol animation, which allows one symbol to reference and control another.

Section 3 – The ShowGraphic Script Function

This section discusses ShowGraphic script function, as another method of calling a symbol into InTouch from within another symbol.

Section 4 – Panel Display

This section builds upon the concepts of the ShowGraphic script function from the previous section, but uses scripting to alter the size and location of the symbol to be displayed in a panel display setup.

Module 4 – Enhanced Application Development

Section 1 – Create Graphic Effects with Animation and Scripting

This section discusses the drawing tools and scripting necessary to create a pulsation effect in an ArchestrA symbol.

Section 2 – Grow/Shrink Functionality

This section discusses the utilization of growing and shrinking functionality as another method of enhancing symbols, while making the most of limited space in an InTouch display.

Section 3 – Modify and Enhance Existing Symbols

This section describes the modification and enhancement of existing graphics and the issues associated with the modifications.

Section 4 – ActiveX Controls

This section describes the installation, configuration, and application of ActiveX controls in InTouch applications.

Section 5 – .NET Controls

This section describes how to import, configure, and use .NET Client Controls in InTouch.

Module 5 – I/O Troubleshooting**Section 1 – Graphical Overrides**

This section presents the methods for monitoring and graphically displaying the Quality and Status of data in an InTouch application.

Section 2 – Visualization and Indirect Addressing

This section discusses the configuration and application of I/O Failover, which is a process that preserves the integrity of data even when there is a network communication error.

Module 6 – Alarm Tools**Section 1 – Basic Review of Alarming and Controls**

This section provides a basic summary of alarm concepts and discusses how to work with the ActiveX alarm control and alarm tree viewer to navigate alarms.

Section 2 – Alarm Databasing

This section discusses how InTouch is used to configure alarm databasing options, and work with the alarm database view and the alarm pareto.

Section 3 – Alarm Client

This section discusses the ArchestraA Graphics Alarm Client.

Module 7 – Tag Servers**Section 1 – Tag Server and Tag Provider**

This section discusses how InTouch is used to configure the local application to browse the tag server's tagname dictionary and retrieve data from the tag server application.

Section 2 – Alarm Providers

This section discusses how to receive alarms from a remote application.

Section 3 – Access Tag Server History

This section discusses using InTouch history from a remote application.

Module 8 – Deployment**Section 1 – Deploying Applications**

This section discusses how InTouch is used to create a Platform, ViewEngine, and instance of the InTouch application for deployment. Graphical issues regarding deployed applications using show symbol scripting will be discussed.

Section 2 – Terminal Services

This section discusses how InTouch is used in a Terminal Services environment and in a Virtualization environment.

Wonderware Software Solutions

Wonderware is the market leader in real-time operations management software, including Supervisory Human Machine Interface (HMI), GeoSCADA, Mobile Operations, Production Management, Manufacturing Execution System (MES), Performance Management, and Enterprise Manufacturing Intelligence (EMI) workflow. It is also the leader in integration with asset management, supply and demand chain, and Enterprise Resource Planning (ERP) applications. Wonderware is a brand offering of the Invensys Operations Management Division.

Wonderware software delivers significant cost reductions associated with designing, building, deploying, and maintaining secure and standardized applications for manufacturing and infrastructure operations. Wonderware solutions enable companies to synchronize their production and industrial operations with business objectives, obtaining the speed and flexibility to attain sustained profitability.

More than one-third of the world's plants and facilities run Wonderware software solutions in dozens of industries worldwide, such as:

- Automotive
- Chemical & Pharmaceutical
- Consumer Packaged Goods
- Discrete Manufacturing
- Electrical Power
- Facilities Management
- Food & Beverage
- Mining and Metals
- Oil and Gas
- Process Manufacturing
- Water and Wastewater

Wonderware software solutions deliver manufacturing and operational performance improvements that help reduce the amount of project-specific work that is required to develop integrated information and automation applications across entire operational enterprises. Wonderware software solutions can be implemented in the context of existing systems, at a company's own pace, and to the extent that the company chooses.

These solutions leverage a powerful, layered software architecture that enables a variety of features and capabilities, such as visualization, optimization and control of plant floor data collection, data storage, and analysis.

Wonderware offers the following software solutions:

- **Manufacturing Execution Systems** – Wonderware MES software helps customers track the transformation of products from raw materials through finished goods. The software gathers in-process data, such as yields, throughput, equipment efficiencies, material consumption, and product quality. It also provides functionality, such as inventory management, work order/Bill of Material management, real-time reporting, and analysis, for a full view into operational efficiencies.

Leveraging the ArchestrA software architecture (see [page 1-9](#)), Wonderware MES solutions are completely scalable and configurable. This enables a unique, incremental approach to operational improvements, where low-risk deployment of increased application functionality can be realized one step at a time.

Wonderware MES solutions help to substantially reduce lead time and manufacturing costs, increase production throughput and product quality, and reduce efforts involved in compliance and governance.

- **Enterprise Manufacturing Intelligence** – EMI software solutions empower companies to analyze their overall operational Key Performance Indicators (KPIs), using simple, yet powerful, data analysis, reporting, and visualization tools.

KPIs such as production, costs, process capability, equipment downtime, and quality and variance data can be collected, aggregated, and displayed using Wonderware EMI software solutions. A powerful, yet secure, web interface, with intuitive, drill-down dashboards, helps deliver this information to the full range of plant workers, tailored to their specific information requirements.

- **HMI/Supervisory Control and Data Acquisition (SCADA)** – HMI/SCADA solutions often impose complex demands on software architecture. Wonderware InTouch HMI visualization software, coupled with the award-winning ArchestrA technology-based ArchestrA System Platform, is uniquely positioned to meet these challenges.

The HMI/SCADA software solutions are easy to use, implement, and configure, and offer simplified maintenance, high security and availability, and virtually unlimited scalability.

- **Data Historian** – Wonderware Historian Server software leverages the state-of-the-art ArchestrA System Platform, industry-leading historian technology, web-based reporting capabilities, and renowned open data source connectivity from Wonderware. The resulting Historian solution is unlike any other data archiving and reporting solution found in the market today. With blazing speed, broad scalability, highly efficient data storage and retrieval, high availability, and simple one-click historization setup, the Wonderware Historian Server software has an industry reputation for low total cost of ownership.

Preconfigured web-based reports and data analysis capabilities derive immediate value from data captured by the Wonderware Historian Server.

- **Batch Management** – Wonderware batch management solutions perform repeatable and consistent execution of batching processes across all hybrid industries, whether it is Electronic Batch Records (EBR) systems in regulated industries, Paper-On-Glass capabilities in paperless production environments, or automated recipe management for supervisory systems.

From simple batch processes, where only the formula changes for different products, to the most complex batch processes requiring dynamic allocation of shared equipment, Wonderware has a solution. Each of these solutions ensures reduced lifecycle costs and investment protection by leveraging the ArchestrA architecture.

- **Product Quality Management and Statistical Process Control (SPC)** – Delivering products with high “quality”—defined as “meeting specifications at the lowest possible cost”—is a top priority for manufacturers and industrial operations. Quality has many aspects, and Wonderware applications provide valued, integrated Quality Management functionality to tens of thousands of companies worldwide.

Starting with Wonderware MES, which provides the most sophisticated, integrated quality management and enterprise-wide SPC, to Historian Server, which provides incremental solutions to store volumes of process data for quality analysis, Wonderware applications consistently meet quality needs. The Wonderware InTouch HMI software offers real-time data monitoring and alarming. Historian Client trends data. MES Software provides specification management, genealogy, BOM enforcement, OEE, downtime monitoring, and quality data documentation and monitoring. System Platform monitors data levels, and application templates can deliver nearly any quality capability. InBatch software collects information on batch quality and recipe settings. The capability list goes on.

- **Mobile Solutions** – Wonderware Mobile Solutions enable manufacturers and producers to manage the processes and procedures used to ensure all field tasks required to achieve reliable operations are executed consistently.

Software running on rugged handheld computers enables field operators to collect data on noninstrumented machinery and to receive contextual information and guidance based on conditions encountered in the field. This helps ensure the consistent execution of best operating practices.

Collected data can also be used for process analysis and production reporting, as well as integrated into existing back-end systems and plant data historians.

- **Workflow** – Workflow solutions allow customers to take their standard operating procedures, in-house knowledge base, and standard practices and make them part of everyday operations. Workflow can overlay any Wonderware product to provide visibility, collaboration, enforcement of procedures, and documentation of results.

For more information on Wonderware software solutions and products, visit the Wonderware website at <http://www.wonderware.com>.

ArchestrA Technology

ArchestrA is a comprehensive automation and information software architecture designed to integrate and extend the life of legacy systems by leveraging the latest, open industry standards and software technologies. It also unifies the Invensys Operations Management products that make up the InFusion ECS. ArchestrA “industrializes” Microsoft .NET and other Microsoft technologies to provide an even more productive toolset for building critical operations management software solutions for manufacturing, production, and facilities operations. The result exposes services needed by manufacturing and industrial infrastructure, such as common name space, object management, industrial security, high availability and redundancy, plant connection, enterprise connection, client interface, web portal, and systems management.

Using ArchestrA technology, applications can be rapidly assembled using software objects rather than being “programmed.” Templates can be created for almost any purpose, and then used to build new applications, simply through reassembly and slight modifications, saving time and lowering development costs. Offerings built on ArchestrA empower decision makers to achieve their business goals without abandoning prior investments in systems or intellectual property.

ArchestrA technology helps reduce application engineering effort and deployment, increase efficiency, provide optimization and standardization, and enable integration of distributed automation systems and applications from virtually any vendor. Geographically dispersed applications (from a few hundred to one million I/O, and from a single node to hundreds of stations) can be rapidly and securely implemented.

The ArchestrA architecture leverages advanced software technologies to fill the gap between ERP systems and control systems. This architecture provides the following:

- **Framework**, which supports common services and a core set of system objects
- **Domain Objects**, which are industry-specific objects
- **Object Development Toolkit**, which enables third parties to create new domain objects customized for specific needs

The supervisory control and manufacturing information environment is served by a variety of systems, including HMI, Distributed Control Systems (DCS), SCADA, Process Information Management systems (PIM), MES, batch and recipe management systems, and advanced control/simulation systems. The ArchestrA Framework supports core services that are required by most of these different types of supervisory control and manufacturing information systems.

These core services include the following:

- Integrated Development Environment (IDE)
- Version management

- License management and centralized deployment
- System diagnostics and system administration
- Internationalization
- Data visualization and monitoring
- Event-based processing, scripting, and calculation capabilities
- Alarm and event management, historization, and security
- Data acquisition and field device integration
- Interobject communications and name service
- Reporting and ad-hoc query capability
- Support for industry standards, such as OPC and SQL

The ArchestrA architecture consists of the following:

- **Configuration and Deployment-Related Components** that are required for centralized deployment of the runtime components. These components are installed like any Windows application and include the following:
 - Centralized object repository (called Galaxy Repository)
 - Integrated Development Environment (IDE)
 - Object deployment services (called Bootstrap)
- **Runtime Components** that are centrally deployed and administered. These components include the following:
 - PCs with core infrastructure (called Platforms)
 - Key software applications (Engines)
 - Objects (Framework Objects) that expose framework-related functionality

Wonderware Individual Software Products

Wonderware software solutions offer robust, best-of-breed software components that empower customers to effectively develop and manage their automation and information applications in continuous, discrete, process, hybrid, and batch manufacturing environments. All the latest Wonderware software offerings leverage the latest ArchestrA technology and offer increased functionality and flexibility, as well as extensive connectivity.

ArchestrA System Platform

ArchestrA System Platform provides a single and scalable platform for all the SCADA, Supervisory HMI, MES, and EMI software-solutions needs of industrial automation and information personnel.

At the center of the ArchestrA System Platform is the “plant model,” which is the logical representation of the physical processes being controlled and supervised. ArchestrA object technology makes configuration, logging, delivery, and maintenance of real-time and historical information point-and-click simple.

Within the System Platform is a high-performance process historian with production history archiving, efficient data compression, and autoconfiguration of historical archiving that eliminates duplicate effort. The platform also contains an industrial web information server that dramatically simplifies the organization and delivery of operations information for use across all functions in an organization.

Wonderware InTouch HMI

InTouch software provides graphic visualization that takes operations management, control, and optimization to a completely new level. The InTouch HMI reputation stands above all the rest. What the industry now knows as HMI all began with InTouch software more than 20 years ago. No other HMI can match InTouch software for industry-leading innovation, architectural integrity, unequalled device integration and connectivity, uninterrupted software version migration path, and truly legendary ease of use.

All this leads to well-designed standards-driven systems that maximize productivity, optimize user effectiveness, increase quality, and lower development, maintenance, and operational costs, helping to make a company the best it can be.

Wonderware Development Studio

Wonderware Development Studio consists of a suite of cooperative tools designed to enable the rapid construction and maintenance of Wonderware applications.

The heart of the Development Studio is the ArchestrA IDE. Within the ArchestrA IDE, engineers can design, develop, test, and maintain any industrial application.

Without leaving the ArchestrA IDE, the user can develop vibrant and sophisticated graphics, and incorporate those into effective HMI, SCADA, MES, or Operations Intelligence applications.

Once development is ready for delivery, a single click is all it takes to deploy applications, objects, and associated logic to anywhere in the enterprise namespace.

ArchestrA Workflow Software

ArchestrA Workflow software is an advanced workflow application that enables companies to digitize manual and automated processes that include people or systems, or both. This sophisticated Business Process Management (BPM) application enables companies to model, execute, analyze, and improve processes inside and outside their organization to drive higher levels of collaboration, productivity, and innovation.

With ArchestrA Workflow, companies can institutionalize work processes that manage normal, unscheduled, or disruptive events within their operations environment, providing the *Right People* with the *Right Information* at the *Right Time*.

Wonderware Dream Reports

Wonderware Dream Reports provide the most intuitive and easy-to-use reporting solution available on the market today, giving the power to quickly and easily create reports from Wonderware InTouch HMI and many other data sources. It requires no IT or programming skills to design, schedule, and produce appealing and informative dynamic reports. It is easy to configure Dream Reports through simple drag-and-drop operations within the Report Studio. The design Studio also provides ease-of-use features, such as intelligent report objects and dialog boxes.

Wonderware Dream Reports will help to significantly reduce report development time, simplify report modifications, and empower users to transform raw data into great looking, information-filled reports. Custom reports can be created easily, quickly, and inexpensively, with scheduling tools to help deliver them to anyone, anywhere, anytime.

The Wonderware Dream Reports solution, along with Wonderware Information server or an included web portal, extends the availability of reports throughout an enterprise network and beyond.

Wonderware Historian Server

The Wonderware Historian Server is a high-performance, real-time database for historical information. It combines the power and flexibility of a relational database with the speed and compression of a true process historian, integrating the office with the factory floor or any industrial operation.

Wonderware Historian Server is designed to collect a wide variety of plant data, at full resolution and very high data rates, ensuring that decision makers at all levels will have the historical information they need to drive vital productivity improvement initiatives. Wonderware Historian Server offers unparalleled scalability and can be configured as a single data collection and aggregation system or as part of a larger, tiered architecture offering the ability to implement sophisticated summary and replication systems.

Wonderware Historian Client

Wonderware Historian Client provides rich data analysis and reporting capabilities to maximize the value of information stored in the Wonderware Historian Server.

Wonderware Historian Client is integrated with Microsoft Office components to deliver data-trend and numerical data analysis using Microsoft Excel, comprehensive data reporting using Microsoft Word, and the capability to publish real-time and historical plant information to the web or company intranet site using Wonderware Information Server.

Plant knowledge workers using information derived from the Wonderware Historian Server can quickly troubleshoot problems, study potential process inefficiencies and eliminate the time-consuming process of locating the data. Wonderware Historian Client makes the delivery and visualization of this information easy to implement and deploy.

Wonderware Information Server

The Wonderware Information Server offers an easy solution for aggregating and presenting plant production and performance data over the web or company intranet.

Plant personnel, with a minimum of training, can install, configure, and implement a highly effective plant information website without the need for custom web programming.

Using Information Server, large amounts of process data can be aggregated into highly informative production reports tailored to the information needs of plant personnel.

Content from the Wonderware Information Server can be incorporated into other web portals, making existing corporate IT portals more informative and valuable.

Wonderware Intelligence Software

Wonderware Intelligence Software enables companies to gather, store, and report on both historical and real-time operational data, using a dashboard to present KPIs that are used to visualize, tune, and maximize their operations.

The Intelligence Data Model is the foundation for transforming data into actionable information by adding context (equipment, product, work orders, material, personnel, and so on). This data in context helps to answer operational questions, such as:

How much product is available per site today?

How many work orders contained a specific raw material ingredient last week?

Wonderware MES/Operations Software

Wonderware MES/Operations software capabilities provide a scalable and configurable MES designed to help manufacturers across a wide range of industries improve their operational efficiency, manufacturing responsiveness, and brand integrity.

The incremental, low-risk approach to building MES from Wonderware allows MES to be implemented in steps, from basic functionality, including work order management, bill of materials, specifications, data collection, and traceability (track/trace/genealogy), to enhanced capabilities, such as inventory management, certifications, labor, and production steps.

Wonderware MES software fully leverages the ArchestrA System Platform and Invensys InFusion Enterprise Control System (ECS) for integration, development, and reporting, as a result of the underlying ArchestrA technology. This approach reduces deployment and maintenance costs, while facilitating rapid development and scaling of the application throughout the enterprise.

Wonderware MES/Performance Software

Wonderware MES/Performance software capabilities provide a software solution for collecting, tracking, and communicating real-time equipment performance and efficiency information, scalable from machine/equipment level information to line/plant enterprise KPIs.

Wonderware MES/Performance software delivers critical equipment downtime and efficiency information to operators and decision makers who can then take immediate action to improve plant performance and productivity, equipped with the most up-to-date operational results.

Wonderware MES software is highly configurable, fully leveraging the ArchestrA System Platform and Invensys InFusion ECS for integration, development, and reporting, as a result of the underlying ArchestrA technology. This approach reduces deployment and maintenance costs, while also facilitating rapid development and scaling of the application throughout the enterprise.

Wonderware MES/Quality Software

Wonderware MES/Quality software capabilities provide historical documentation of quality sample data and SPC monitoring of the sample data collected.

Wonderware MES/Quality software helps manufacturing companies configure, manage, and implement quality specifications that reduce the cost and increase the efficiency and accuracy of capturing and monitoring critical quality information on the plant floor. Information is captured and monitored either directly from process equipment or by operators using sample plan procedures.

Wonderware MES/Quality software incorporates electronic records of operational performance and quality sampling procedures for compliance with internal, governmental, or safety regulations that require tighter quality control and improved quality procedure management in many manufacturing industries.

Wonderware QI Analyst

Wonderware QI Analyst SPC software is an important part of any quality management program. Performing both online and historical SPC, QI Analyst supports real-time process monitoring and alarms, as well as historical reports to view process “health” over any period of time. Real-time SPC, analysis, and reporting are equally easy. By storing process data in the QI Analyst database and linking to external data sources, users can leverage enterprise-wide SPC to reduce variation, reduce costs of manufacturing, and increase productivity.

Wonderware InBatch Software

Wonderware InBatch flexible batch management software optimizes the management of any batch process. InBatch software automates recipe management using a graphical procedure environment featuring Sequential Function Charts (SFC). Consistent with the ISA S88 flexible batching standard, InBatch software offers comprehensive batch execution and equipment history, material genealogy, stringent security, web-based reporting, and the ability to facilitate the design and implementation of systems that are compliant with FDA 21 CFR Part 11 regulations.

Wonderware SCADAAlarm

SCADAAlarm alarm and event-notification software provides a telecommunications link to industrial automation software systems. It seamlessly integrates with the comprehensive Wonderware product family and has built-in browsers to enable fast configuration of information from ArchestrA System Platform and InTouch HMI software.

Wonderware Toolkits

Wonderware Toolkits provide powerful extensibility to InTouch HMI and System Platform applications by enabling developers to extend the capabilities of Wonderware products to meet specific system integration needs. The Toolkits promote adherence to industry standards, provide additional customization and intellectual property protection, and enhance the ability to interface Wonderware products with other software and hardware.

Wonderware offers the following Toolkits:

Toolkit	Enables developers to:
ArchestrA Object Toolkit	Extend the ArchestrA architecture with objects that provide specific application or device integration functionality.
GRAccess Toolkit	Create programmatic access to and interaction with System Platform Galaxy configuration data.
MXAccess Toolkit	Create programmatic access to runtime data in a System Platform Galaxy.
DAServer Toolkit	Build custom device integration servers more easily.
Historian Toolkit	Create high-value industrial applications that integrate with data sources from the System Platform and other data sources.
Alarm Toolkit	Produce custom-distributed alarm providers and consumers.
Wizard Toolkit	Produce their own Wizards for inclusion in InTouch HMI.
Script Toolkit	Develop custom InTouch scripts.

Wonderware Device Integration Servers

Connectivity to plant/facility devices is key to real-time information management. Wonderware maximizes choices with the broadest possible communication options for industrial automation and information devices. In collaboration with more than 100 third-party interface developers, Wonderware provides the largest selection of connectivity options to hundreds of control systems, such as PLCs, RTUs, DCSs, flow controllers, loop controllers, scales, gauges, bar code readers, and other hardware devices. Wonderware has also fully embraced the openness of OPC technology, exposing data via OPC from Wonderware products as an OPC Client, as well as providing the means to connect to any third-party OPC Server.

Device integration can be maintained more easily using Device Integration Objects (DI Objects) within the ArchestrA System Platform for seamless connectivity to any data source. Wonderware also offers the DAServer Toolkit, which empowers companies to create their own connectivity server.

Wonderware Enterprise Integration Application

The Wonderware Enterprise Integrator enables fast and reliable information exchange between Wonderware MES Software, ArchestrA System Platform, and enterprise systems.

Typical integration scenarios include connecting business systems with Wonderware MES, InBatch, or Intelligence and other shop floor applications.

The Wonderware Enterprise Integrator modular design allows any application to be integrated, so that one consistent approach to enterprise integration is adopted, eliminating isolated silos of information and high-maintenance point-to-point integration scenarios.

Wonderware IntelaTrac

Wonderware IntelaTrac is the industry-leading mobile workforce and decision support system.

IntelaTrac includes configurable software and ruggedized mobile hardware solutions that enable workflow, data collection, and general task management for plant operations, maintenance management, production tracking, and compliance applications. IntelaTrac is capable of exploiting integrated barcode and RFID reader technology found in many of today's leading mobile devices. This enables precise location verification of critical field tasks supporting regulatory compliance and continuous improvement initiatives.

IntelaTrac is also a key component of a complete plant-intelligence solution that connects all of your wired and stranded assets. This enables an even broader visibility into the performance of your assets than ever before. The IntelaTrac asset-centric approach makes tracking asset performance straightforward, which supports Reliability Centered Maintenance initiatives.

Section 2 – InTouch HMI Overview

This section introduces the InTouch Human Machine Interface (HMI) and provides an overview of the benefits of using InTouch, its key capabilities, and features.

Introduction

InTouch software provides graphic visualization of operations management and control. No other HMI can match InTouch software for industry leading innovation, architectural integrity, and unequaled device integration and connectivity, uninterrupted software version migration path, and truly legendary ease of use.

Overview

Wonderware InTouch provides a quick and easy way to create HMI applications for the Microsoft Windows 7, Windows XP, Windows Server 2008, Windows Server 2003, and Windows Vista operating systems. InTouch applications are used around the world in the manufacture of all kinds of goods including food processing, semiconductors, oil and gas, automotive, chemical, pharmaceutical, pulp and paper, transportation, utilities, and more.

With InTouch, create applications that use key features of Microsoft Windows, including ActiveX controls, OLE, graphics, networking, etc.

Key Benefits

- Ease-of-use enabling developers and operators to quickly and easily be more productive
- Unequaled device integration and connectivity to virtually every device and system
- Stunning graphic visual representation and interaction with an operation brings the right information to the right people at the right time
- History of uninterrupted software version migration path that means HMI applications investment is protected

Key Capabilities

- Resolution independent graphics and intelligent symbols that visually bring a facility to life right on a computer screen
- Sophisticated scripting to extend and customize applications for specific needs
- Real-time distributed Alarming with historical views for analysis
- Built-in, real-time and historical trending
- Microsoft ActiveX controls and .NET controls integration
- Extensible library of over 500 pre-designed 'intelligent' and customizable graphic and object symbols

Options to Manage InTouch Applications

Manage InTouch applications with either the InTouch Application Manager or with the Arcestra IDE.

There are three types of InTouch applications, categorized by how they are managed or where they were published from:



Stand-Alone InTouch applications

(Created from the InTouch Application Manager)



Managed InTouch applications

(Created from the Arcestra IDE)



Published InTouch applications

(Runtime Application Published from a Managed Application)

This course will focus on Managed InTouch applications and their application in distributed network environments.

Managed InTouch Applications

Manage InTouch applications using the Arcestra Integrated Development Environment (IDE). These applications are called “Managed” InTouch applications. They appear in InTouch Application Manager as Managed. Develop an InTouch application on one node of the Galaxy with WindowMaker, then deploy it to one or more target nodes running WindowViewer.

When using the system platform functionality of the Arcestra IDE to manage InTouch applications, it is possible to:

- See which InTouch applications are running on what node
- Use a central repository for InTouch applications
- Deploy changes to WindowViewer running on the remote nodes
- Create and instantiate Arcestra Graphics

Compare Stand-Alone, Managed, and Published InTouch Applications

Stand-alone, managed, and published InTouch applications have some differences and some similarities, as described in the following table:

	Stand-Alone InTouch Applications	Managed InTouch Applications	Published InTouch Applications
Create Application	Application Manager	ArchestrA IDE • New application • Importing stand-alone application • Importing SmartSymbols	Not possible
Edit Application	WindowMaker started from the Application Manager	WindowMaker started from within the IDE	WindowMaker started from the Application Manager
Delete Application	Delete folder and remove from Application Manager	Delete InTouchViewApp template	Delete folder and remove from Application Manager
Support of ArchestrA Symbols	No	Yes for all operations	Yes, but only for viewing, not for creating and editing
Support of DB Dump and DB Load	Yes, function within Application Manager	Yes, function within the IDE	Yes, function within Application Manager
Editing application in original resolution requires conversion	Yes	No	Yes
Management of Distributed Applications	Network Application Development (NAD)	ArchestrA IDE	Network Application Development (NAD)
Configuring how new InTouch application versions are accepted	Configured in Application Manager (Network Application Development)	Configured in WindowMaker	Configured in Application Manager (Network Application Development)
Use Fast-Switch to test application	Yes	Yes	Yes
Use tag value and tag parameter retention	Yes	Yes, also requires configuration of the local working directory	Yes

Section 3 – System Requirements, Licensing, and Support

This section provides a detailed explanation of the system requirements necessary for System Platform, discusses Licensing details, and covers Support services.

System Requirements for InTouch 2012 R2

This section describes the hardware and software requirements to install InTouch 2012 R2. For updates to this information, see the Wonderware Technical Support website at www.wonderware.com/support.

IMPORTANT

- Installing the InTouch HMI on a computer used as a domain controller is not supported.
- UAC - Disable User Account Control (UAC) before installing Wonderware Application Server 3.5 and InTouch 2012 R2. This is applicable for Windows Vista and newer operating systems.
- If Wonderware Application Server and the InTouch HMI are installed on the same node, they both must be upgraded to Application Server 3.5 and InTouch 2012 R2, respectively.
 - If your version of Wonderware Application Server is 3.1 SP3 or earlier, then upgrade it to Application Server 3.5 before upgrading InTouch 10.1 SP3 to 2012 R2.
 - If you are using managed InTouch applications with ArchedrA symbols from Application Server 3.1 SP3 or earlier, run the ArchedrA Symbol Analysis and Repair Utility to ensure all issues reported by the utility are addressed before installing Wonderware Application Server 3.5.
- InTouch Memory Usage - The default value for InTouch internal shared memory management (WWHeap) has been changed to use more memory.
 - In a 32-bit operating system, the default maximum memory of WWHeap is increased to 1.5 GB from 256 MB. Special configuration can increase the maximum limit to nearly 2 GB. In a 32-bit environment, the InTouch memory management can be extended to 3 GB by adding the /3GB switch in the boot.ini file. An entry in the boot.ini file with the /3GB switch looks like the following:

[operating systems]

```
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"  
/fastdetect /3GB
```

Note: This example is specific to a Windows XP computer. For other operating systems, the Windows name is different.

- If the /3GB switch is enabled in Windows, the paging file size should be set to a system managed size, and then InTouch can access up to 3 GB of memory.
- In a 64-bit operating system, it is not necessary to edit the boot.ini file and InTouch can use up to 4 GB of memory.

For additional information, refer to Microsoft Knowledge Base articles, such as KB 328882.

- InTouch Window Caching - View.exe supports hybrid functionality to keep some selected windows in memory, and load the rest from the disk. To enable this functionality, you must configure the memory health check interval and set the memory properties.
- Alarm DB Logger - In earlier releases that supported Windows Vista, Windows 7, or Windows Server 2008, the Alarm DB Logger could not be enabled to run as a service. The Alarm DB Logger now runs as a Service in Windows Vista and later operating systems primarily to support Galaxy Alarms and InTouch Alarms from Terminal sessions. However, there is a limitation in that the combination of Alarm DB Logger configured as a service and InTouch running locally as a console application is still not supported. If InTouch needs to run in the Console session, the Alarm DB Logger must be configured in the Alarm DB Logger Manager as a "Normal Application" instead of a "Windows Service".

Refer to the updated Tech Note 725, "Running InTouch and AlarmDBLogger Services on Vista and Later Operating Systems," for full details of the supported scenarios and applicable alarm query syntax.

- Galaxies with Arcestra symbols or InTouchViewApp objects containing Arcestra symbols that leverage the Runtime Language Switching functionality must always be modified on the same language operating system and InTouch language version on which it was initially developed. Attempts to modify these components in a different language operating system, or a different InTouch language version, may cause unexpected functional or operational issues that are not currently supported. Published applications running in an operating system language that is different from the originally developed language are supported. Refer to Known Issues CRs L00088974, L00088993, and L00088998.
- Review the Antivirus Software Exclusions list.
- Windows Server 2008 R2 no longer supports the /console switch as a means of setting the Remote Desktop Client (RDC), also known as a console session.
- If you are running WindowViewer within a Terminal Server session and want to access alarms from WindowViewer in a client session, you must use the syntax `\\terminalservernode:<IP Address>\InTouch!$System` to access the alarms, with a colon (:) after the node name. The IP Address is that of the client computer connected to the session. Click this Terminal Services Behavior link for more information about Terminal Services behavioral changes.
- Starting the Arcestra Licensing Utility from an existing License CD may not work on a 64-bit operating system. Start the utility from the installed shortcut instead.
- All the restrictions applicable to Windows Vista and Windows Server 2008 are also applicable to Windows 7 and Windows Server 2008 R2, respectively.
- If you copy a command from other applications, issues may occur with command line printing. To avoid any issue, delete any quotes (") that you may have copied and type them again manually.

Hardware Recommendations

The following shows the recommended specifications to install and run Wonderware InTouch HMI Version 2012.

- The recommended operating system for InTouch development is Windows 2008 R2 SP1 or Windows 7 SP1.
- The recommended operating system for run-time nodes is Windows 7 SP1.
- SQL Server 2008 SP1 Express, Standard, or Enterprise (32-bit) is the only supported version at this time.

Hardware Requirements	Small 1 – 5,000 I/O	Medium 5,000 – 20,000 I/O	Large > 20,000 I/O
CPU (GHz)	1.2	2.5	2.5
Minimum Cores	1	2	4
RAM (Min GB)	1	2	4
Hard Disk Size (GB)	50	100	250
Monitor	1024 X 768	1280 X 1024	1280 X 1024
Network Card (Mbps)	100	100	1000

Note: The Microsoft Windows Vista operating system imposes hardware requirements that exceed the minimum requirements for InTouch HMI version 2012 R2. If you intend to run InTouch 2012 R2 with Windows Vista, see the following Microsoft web site for current hardware requirements: www.microsoft.com/windows/products/windowsvista/editions/systemrequirements.aspx

Software Requirements for InTouch 2012 R2

The following table lists the supported operating systems that can be installed on computers that run InTouch server and client components.

Operating Systems	Product name Components				
	Window Maker	Window Viewer	ArchestA IDE	Archestra Run Time	Galaxy Repository
Windows XP SP3 Pro 32-bit	Y	Y	Y	Y	NO
Windows Vista SP2 Business/Ultimate 32/64-bit	Y	Y	Y	Y	Y
Windows 7 Pro/Ultimate 32/64-bit	Y	Y	Y	Y	Y
Windows 7 SP1 Pro/Ultimate 32/64-bit	Y	Y	Y	Y	Y
Windows Server 2003 SP2 Standard/Enterprise 32-bit	Y	Y	Y	Y	Y
Windows Server 2008 SP2 Standard/Enterprise 32/64-bit	Y	Y	Y	Y	Y
Windows Server 2008 R2 Standard/Enterprise 64-bit	Y	Y	Y	Y	Y
Windows Server 2008 R2 SP1 Standard/Enterprise 64-bit	Y	Y	Y	Y	Y
Windows XP Tablet	Y	Y	NO	Y	NO

Note: Microsoft Windows 7 (32-bit or 64-bit) is the recommended operating system to run InTouch client components and Microsoft Windows Server 2008 R2 is the recommended operating system to run InTouch server components.

Wonderware Product License

Wonderware enforces the use of its products with a software license. After you install a Wonderware product, you must install a single license file, ARCHESTRA.LIC, which contains your customer information, product data, and software features. Until you apply your Wonderware license, you can run the InTouch HMI only in demonstration mode.

You use the ArchestrA License Manager to install the contents of the ARCHESTRA.LIC file to the computer where you have installed the InTouch HMI. The ArchestrA License Manager is included as a common ArchestrA component when you install the InTouch HMI.

Product Support

Wonderware provides responsive, award-winning teams of specialists committed to delivering world-class customer support, training, and consulting services. For information on the various customer support programs, contact a local distributor or access the Wonderware Technical Support site online at <http://www.wonderware.com/support/mmi/>

There is a variety of information on the Technical Support site including the Wonderware Developer Network (WDN) Library, Documentation, FAQs, Tech Alerts, Tech Articles, Tech Notes, and Tech Support Forums. When first entering the site, there is limited access. Register to gain access to the different areas and services.

Also, on the Technical Support site is the *Technical Support Policies, Terms & Conditions* guide, which provides information on how to contact Technical Support, information on the support policies and procedures, and much more.



Module 2 – Application Setup

Section 1 – Managing InTouch Applications Using ArchastrA IDE	2-3
Lab 1 – Galaxy and Application Setup	2-7
Section 2 – Tag Functionality	2-15
Lab 2 – Tag Viewer	2-21

Module Objectives

- Import a backup of an InTouch Application into ArchestrA IDE
- Connect an InTouch Application to a data source
- Work with the Tag Viewer to view InTouch tag data

Section 1 – Managing InTouch Applications Using ArchestrA IDE

This section describes how InTouch is integrated and used within the ArchestrA IDE. It also discusses creating a new Galaxy and importing existing applications.

Introduction

The ArchestrA IDE provides automation objects to build complex system environments and a suite of graphic tools. Using automation objects, integrate InTouch applications with other Wonderware components. Embed pre-built ArchestrA symbols into applications or use a wide assortment of tools from the ArchestrA Graphic Toolbox to create custom symbols.

You can use the ArchestrA IDE to:

- Create a new managed InTouch application or import an existing one
- Start WindowMaker
- Submit the changes made in WindowMaker to a managed InTouch application
- Export and import a managed InTouch application with its InTouchViewApp object
- Publish or delete a managed InTouch application
- Import and export tag data used in a managed InTouch application
- Export tag data to and import tag data from a .csv file

ArchestrA IDE and InTouch

ArchestrA manages the InTouch applications with a specific type of ArchestrA object called the InTouchViewApp object. This type of object is derived from the InTouchViewApp template and then associated with an InTouch application.

This is done by:

- Creating a new InTouch application
- Importing a stand-alone InTouch application

Note: The InTouchViewApp template does not contain the InTouch application data itself, such as the tag configuration and values, but simply references the application.

Once the InTouch application is created or imported, open it for editing using WindowMaker. To test the application while it is open in WindowMaker, simply click the runtime button in the top-right corner of the screen to fast-switch to WindowViewer. When WindowMaker is closed after making changes to the InTouch application, the InTouchViewApp object is automatically checked in.

Managing Memory for WindowViewer

You can configure how WindowViewer uses memory for windows and pop-up symbols to improve performance at run time. Windows and pop-up symbols displayed using ShowSymbol animation and the ShowGraphic script function can be kept in memory at run time in certain conditions to allow for fast retrieval.

Note: For information on memory management for symbols displayed by the ShowGraphic function, see the *Creating and Managing ArchestrA Graphics User's Guide*.

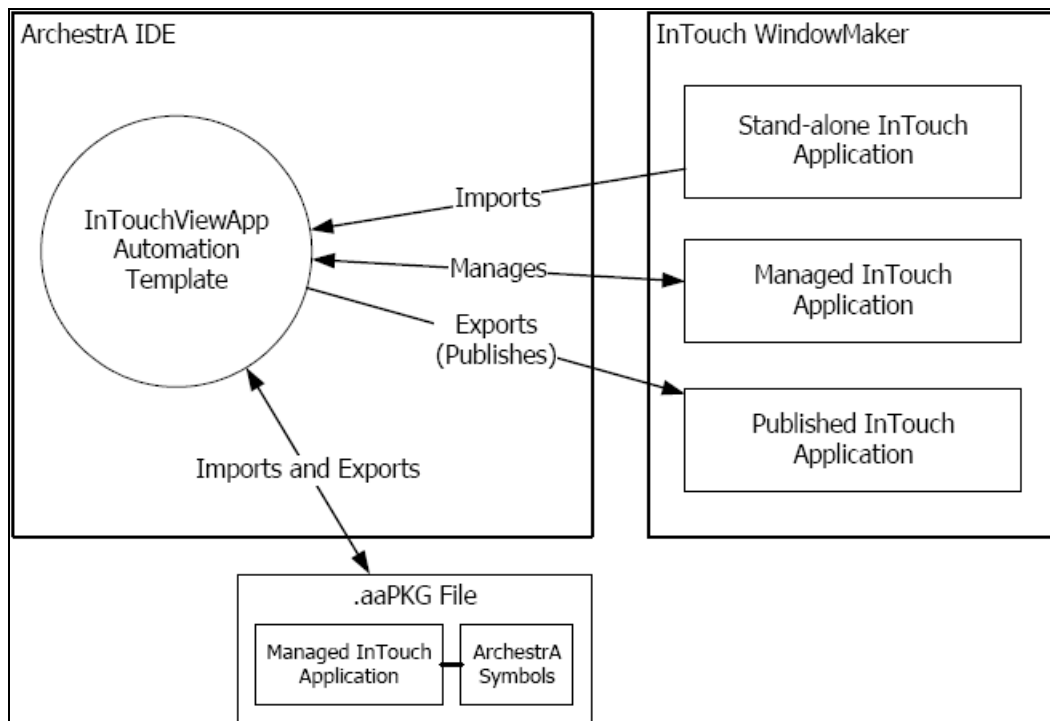
Export, Import, and Publish Features

Export the InTouchViewApp object, for example, to use the managed InTouch application together with its hosting InTouchViewApp object in other Galaxies. When the object is exported, a package file (.aaPKG) is created containing information about the object, the associated managed InTouch application, and any ArchestrA Symbols the application uses.

When an InTouchViewApp object is imported, the ArchestrA IDE also imports the managed InTouch application. If the InTouch application contains references to ArchestrA data, such as galaxy:UDA, then a WinPlatform object needs to be deployed to the node that the InTouch application is being edited on; otherwise, the data shows blank values.

If a managed InTouch application is published, the published InTouch application is no longer associated with the InTouchViewApp template. This means the published application cannot be edited within the IDE or imported into another InTouchViewApp template. The published InTouch application can still communicate with the Galaxy through any embedded ArchestrA Symbol. For example, write data back to the Galaxy or visualize Galaxy data. Edit the ArchestrA Symbol using basic InTouch operations such as copying, cutting, pasting, duplicating, moving, resizing, flipping, rotating, and configuring with InTouch animation links. However, the ArchestrA Symbols cannot be modified, nor can new ArchestrA Symbols be embedded into the InTouch application. These operations are only allowed with managed InTouch applications. Typically, publishing is only utilized in environments that do not support the processing requirements of ArchestrA. For example: in remote plant operations or in small networks.

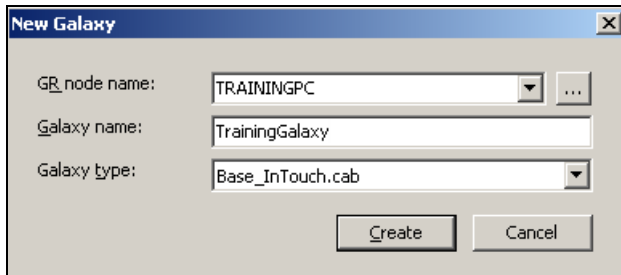
The following graphic shows how applications are imported, exported, managed, and published.



The ArchestrA Galaxy

A Galaxy is a collection of platforms, engines, templates, instances, and attributes that you define as part of your application. This collection is stored within a Microsoft SQL Server Database known as the Galaxy repository. The Galaxy represents the system's configuration and it incorporates a variety of components distributed across multiple computer nodes. The Microsoft SQL Server Database doesn't need to be on every computer, but it must be on at least one computer on the network that everyone has access too.

When creating a new Galaxy, provide the **GR node name**, **Galaxy name**, and **Galaxy type**. The GR node name refers to the location of the Galaxy Repository (i.e., where the Microsoft SQL Server Database is located). Choose a unique Galaxy, but keep in mind it is not easy to change later and cannot contain any spaces.



Galaxy type will be based on specific needs, and can be any of the following:

- Default Galaxy: Full application server Galaxy with support for all Galaxy features.
- Base_Application_Server: Same as the default Galaxy, except that it backs itself up after the Galaxy is built.
- Base_InTouch: Provides only what is necessary to create InTouch tag-based applications.
- Reactor_Demo_Application_Server: Demo based on full application server functionality.
- Reactor_Demo_InTouch: Demo based on a Tag-based InTouch application.

Additional Information

For additional information about any of the topics covered in this training, please refer to one of the following options:

- The InTouch product library consists of a set of Portable Document Files (PDF) available in the following location in the ArchestrA System Platform DVD:

\\Archestra System Platform\\CD-Intouch\\UserDocs\\English

- Context-sensitive documentation is also available with InTouch online help. Select **Help** for information about the current InTouch window open on the computer.

Note: This course provides only basic concepts and procedures of the Wonderware System Platform's ArchestrA IDE interface that are related to building an InTouch tag-based application. To obtain additional information and learn advanced procedures on how to use InTouch and System Platform, attend System Platform – Part 1, and InTouch for System Platform.

Lab 1 – Galaxy and Application Setup

Introduction

In this lab, you will build a Galaxy and import a pre-configured application into the ArcestrA Integrated Development Environment (IDE). The purpose of this lab is to introduce you to the object that you will configure and use in a distributed network environment in later labs.

Objectives

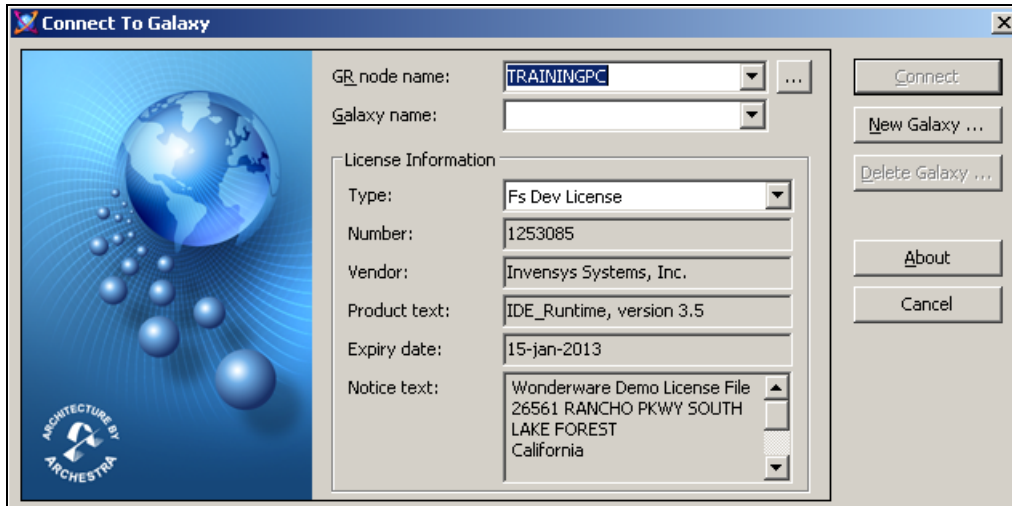
Upon completion of this lab, you will be able to:

- Create and connect to a new Galaxy in the ArcestrA IDE
- Import an existing InTouch application from a back up
- Start the InTouch application from the ArcestrA IDE using WindowMaker

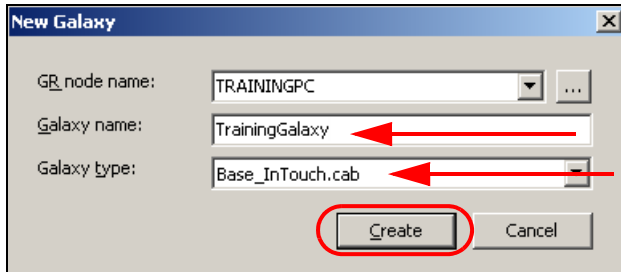
Launch the ArchestrA IDE and Create a New Galaxy

In this lab, you will create a new Galaxy, import an existing InTouch application, and then launch the application using WindowMaker.

1. Start ArchestrA IDE (**Start | All Programs | Wonderware | ArchestrA IDE**).
2. Click **New Galaxy**.

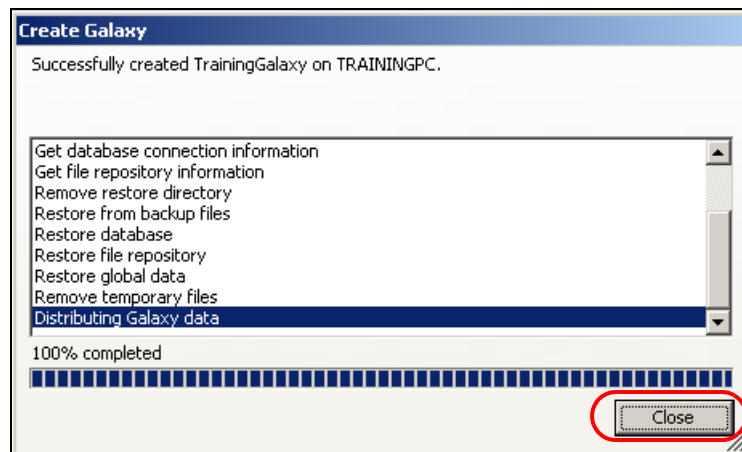


3. Name the new Galaxy **TrainingGalaxy**, select **Base_InTouch.cab** from the drop-down, and then click **Create**.

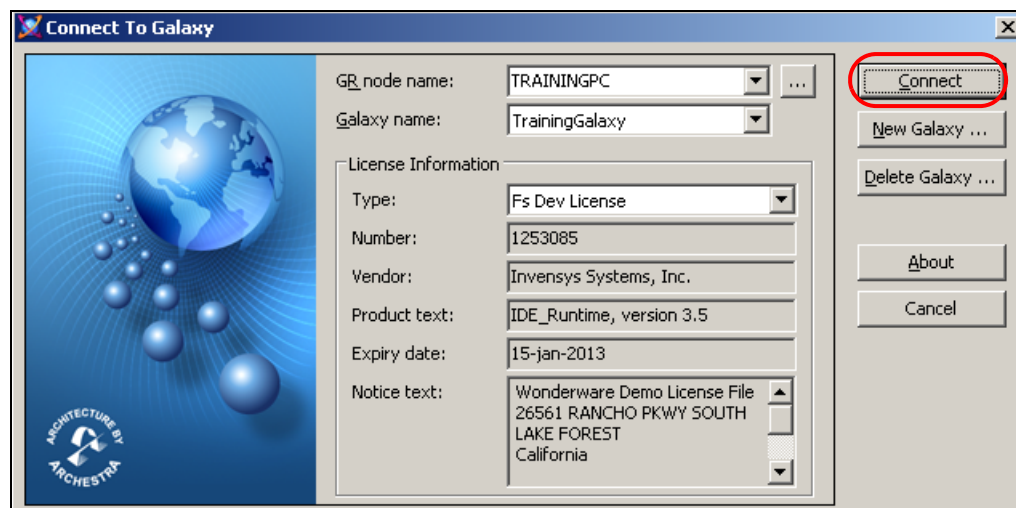


Note: The **GR node name** field self-populates. **Galaxy name** cannot contain any spaces.

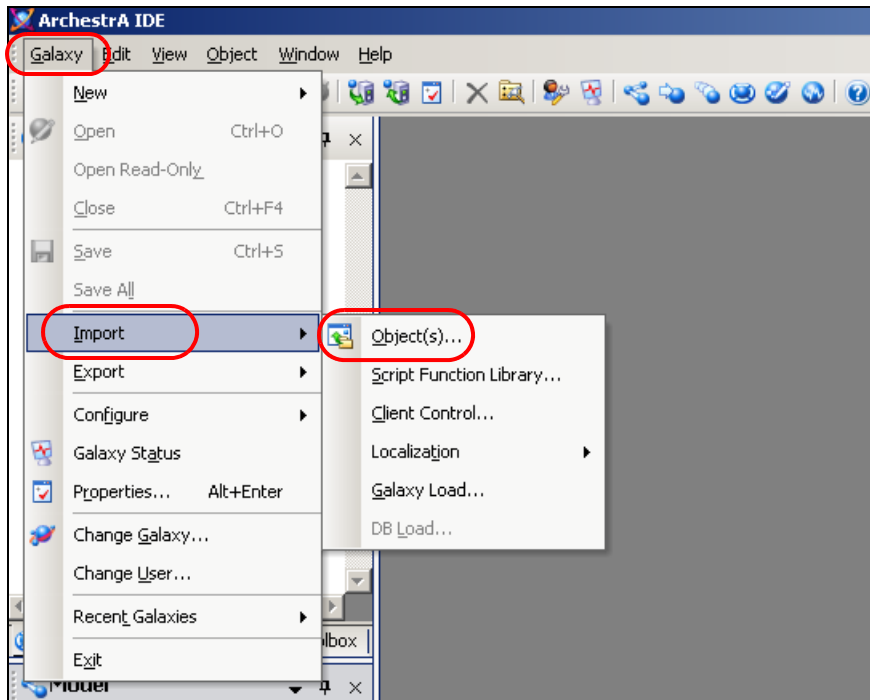
4. After the Galaxy has been created, click **Close**.



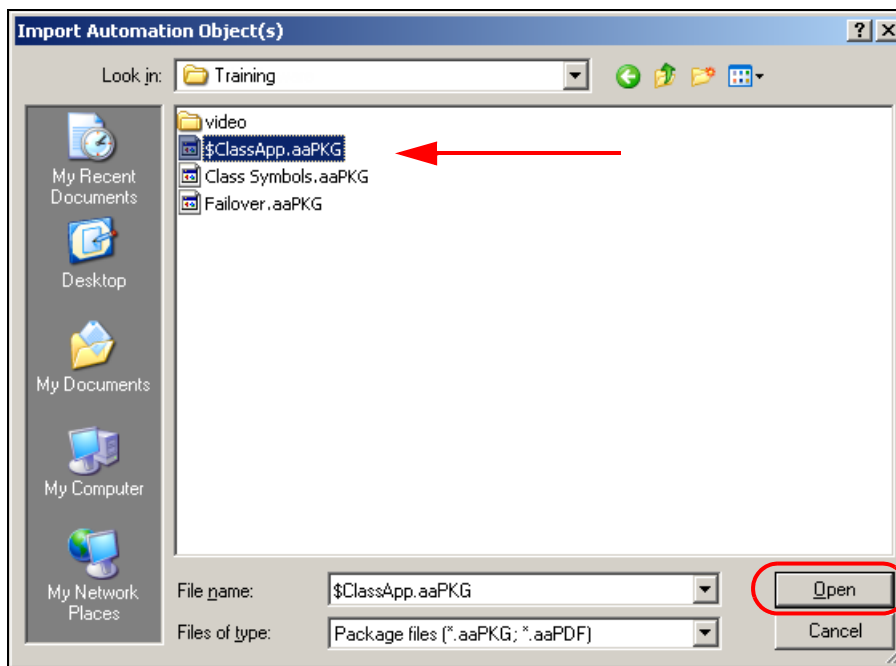
5. Click **Connect** to connect to your Galaxy.



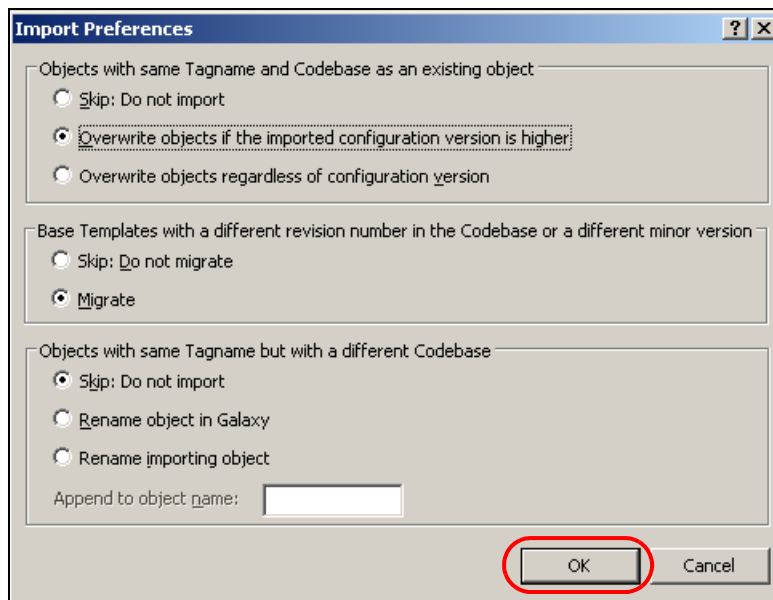
6. On the **Galaxy** menu, click **Import | Object(s)**.



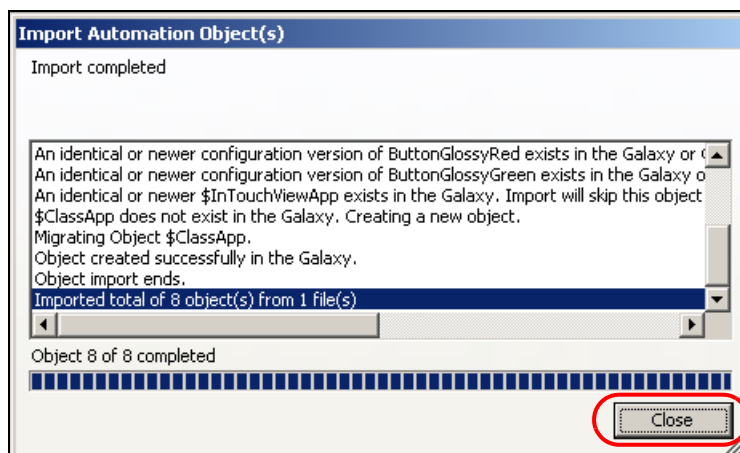
7. Navigate to **C:\Training\ClassApp.aaPKG**, and then click **Open**.



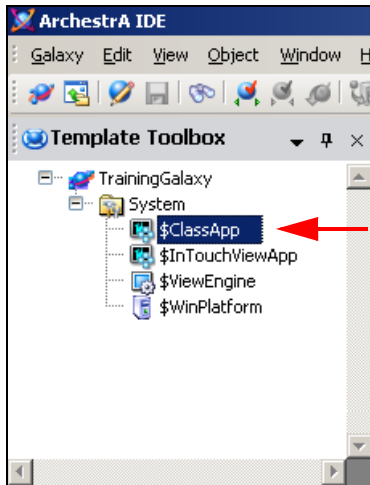
8. Accept the default settings and click **OK**.



9. In the **Import Automation Object(s)** dialog box, after the process is **100% completed** click **Close**.

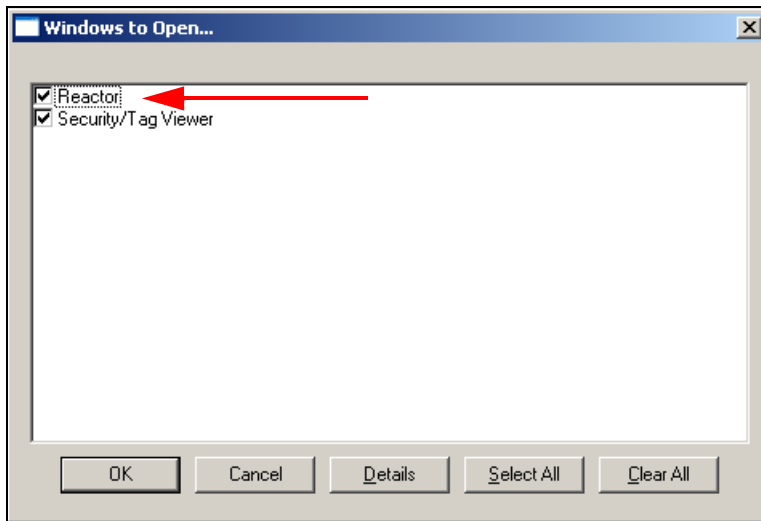


10. In **Template Toolbox**, click **TrainingGalaxy | System toolset**, and then double-click **\$ClassApp** to open the InTouch application in WindowMaker.

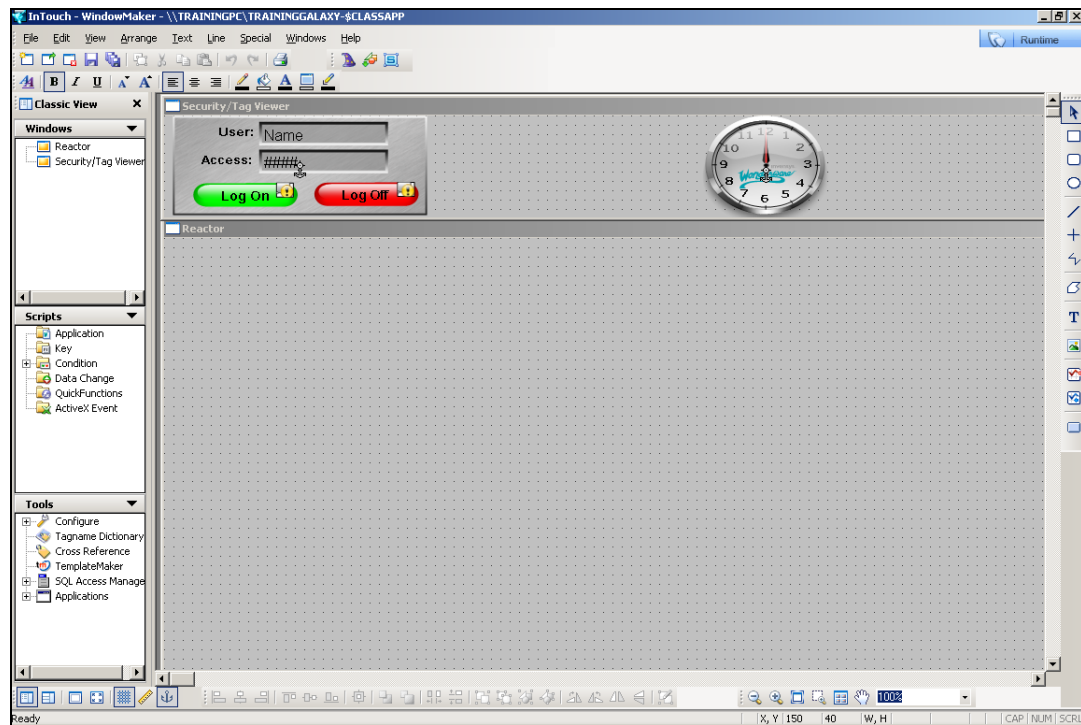


Note: If you are prompted to change the resolution, click **Yes**.

11. Ensure **Reactor** and **Security/Tag Viewer** are checked, and then click **OK**.



After a few moments, the InTouch application opens in WindowMaker.



Section 2 – Tag Functionality

This section covers the concepts of application and configuration of InTouch tags; including SuperTags.

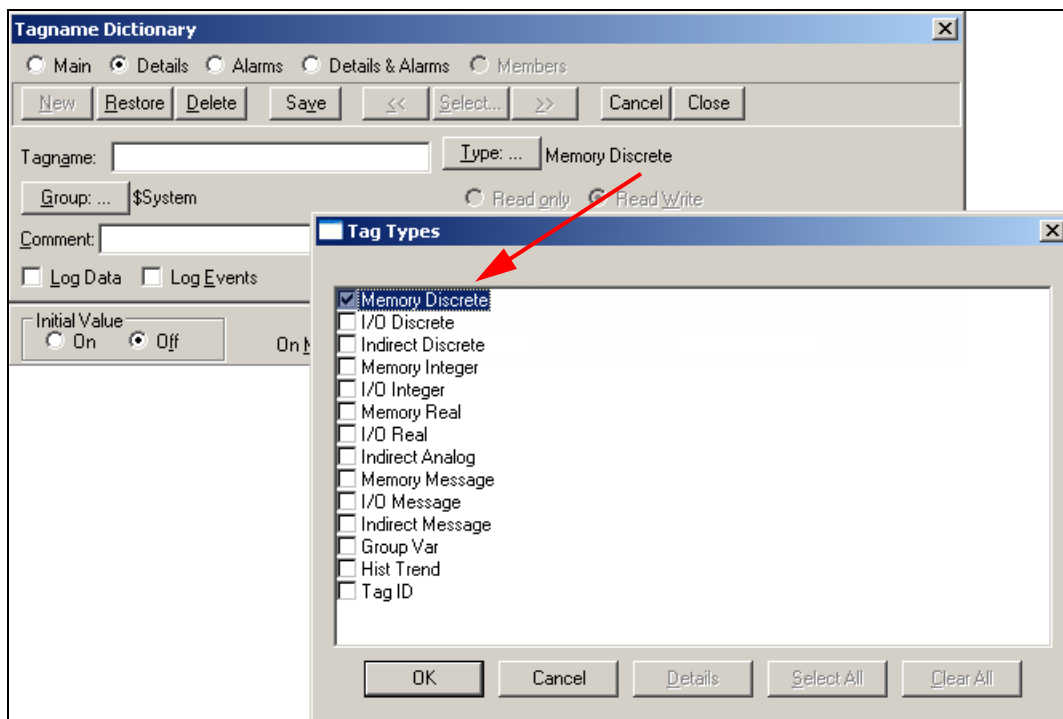
Tags

A Tag is simply defined as a value. The value represents a tank level, temperature, communication status, switch position, valve position, and other real-time values in processes. Each tag has a variety of attributes such as tagname, min/max values, and access name. InTouch requires different information for different tag types. For instance, for I/O-type tags, InTouch requires more information in order to be able to acquire the value from an external data source and convert it for internal use.

The information about the tag is contained within the **Tagname Dictionary**. The Tagname Dictionary (Runtime database) is the heart of InTouch. It contains the current value of all of the tags in the database. In order to create the Runtime database, InTouch requires information about all of the variables being created. Each tag must be assigned a tagname and type. During Runtime, an InTouch application accesses the Tagname Dictionary and provides specific tag information via animation links, scripting, and so on. When tags change (e.g. created, value change, configuration change), the changes are stored in the Tagname Dictionary.

Create and Browse Tags

Manual creation and configuration of tags is performed using the Tagname Dictionary's **Tagname Editor**. Simply name the tag, select the type of data the tag will represent, and fill in the values for the tag. Once tag is saved, it is available for browsing and assigning.



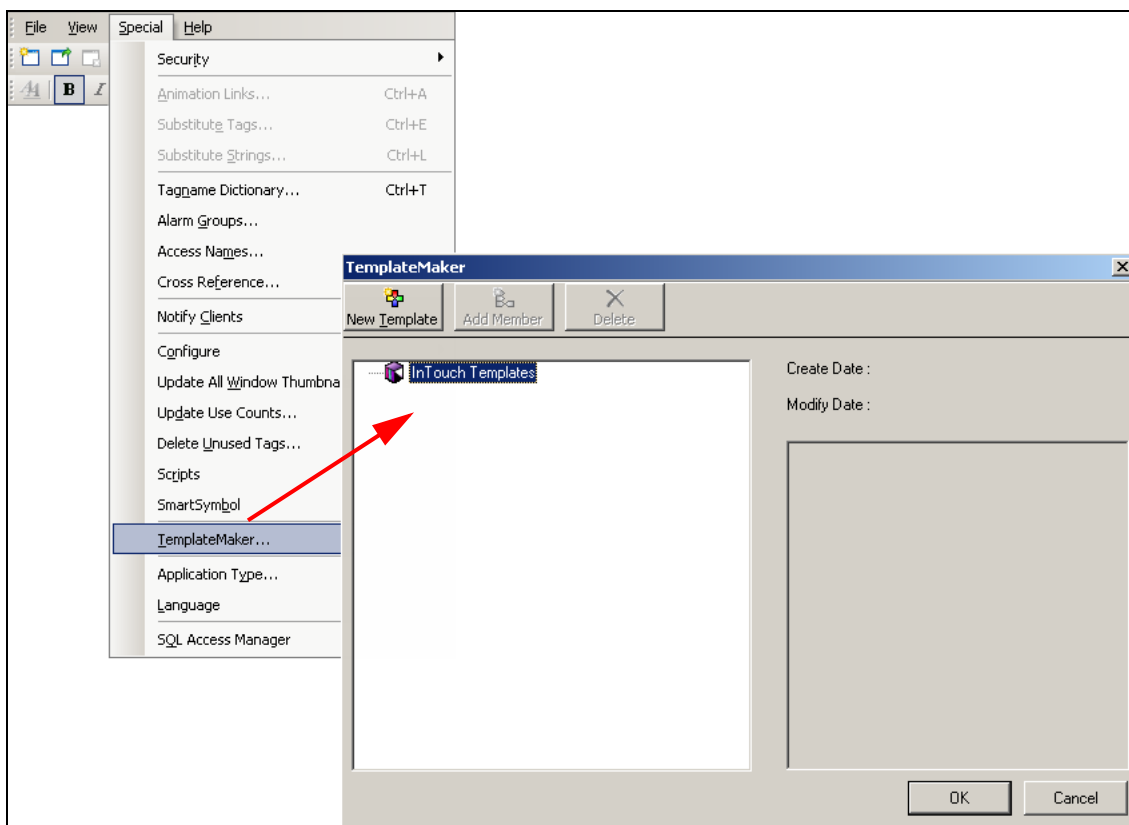
SuperTags

Outside of the standard tag types shown in the Tag Name Dictionary there is another type called a SuperTag, which will be used throughout this training. A SuperTag is a structure of tags that allow users to map InTouch tags to tag structures in a control system. This saves considerable development time when creating tags for similar devices. SuperTags are ideal for “cookie cutter” segments of plants with identical or similar equipment in multiple quantities.

SuperTags can contain up to 64 member tagnames and 2 nesting levels. A SuperTag parent can contain up to 64 embedded child members and each child member can contain up to 64 sub-member tagnames for a total of 4095 member tagnames. When one SuperTag template parent is embedded into another SuperTag template, it becomes a “child member.”

All SuperTag template sub-member tagnames behave exactly like normal tagnames. They support trending, alarming and all tagname Dot Fields. SuperTag members and sub-members can be any supported data types (Discrete, Integer, Real, Memory). In addition, Indirect SuperTags are supported.

For convenience, InTouch provides a “TemplateMaker” that can be used to create templates for creating SuperTags. The TemplateMaker allows the creation, editing, and deletion of SuperTag templates and members. InTouch saves all SuperTag templates in the file SUPERTAG.DAT in the InTouch installation directory, not the application directory. This allows the templates to be used in any application or copied to another InTouch installation.



InTouch also provides the ability to create SuperTags in several alternative ways:

- Directly from the Tagname Dictionary
- Animation links
- Expression input boxes
- QuickScripts
- External files that can be uploaded using the InTouch DBLoad utility

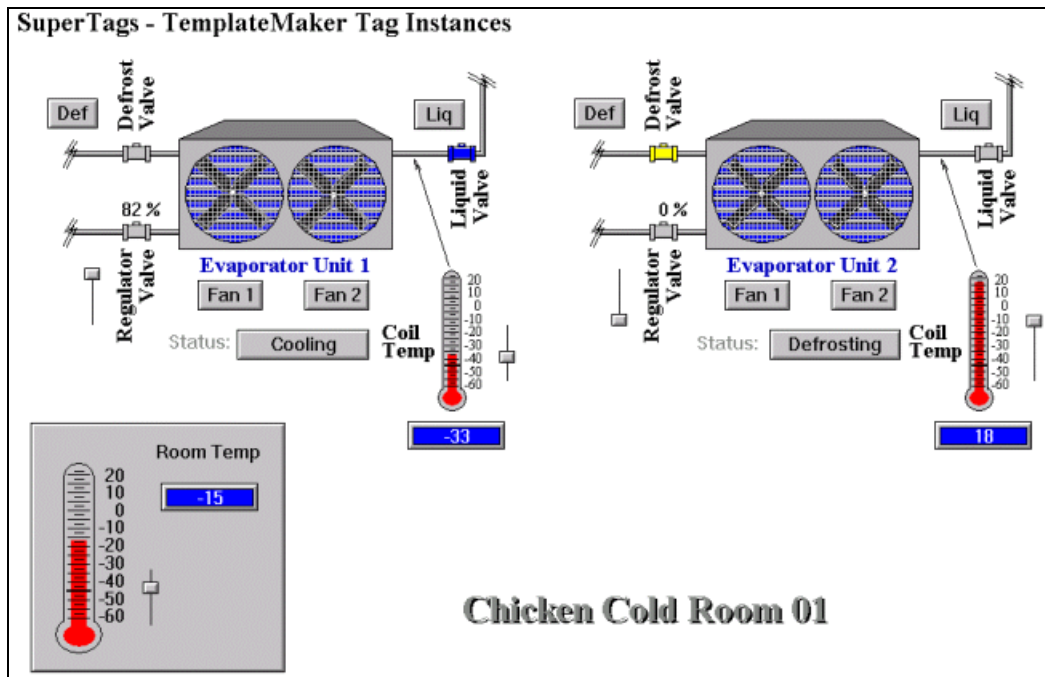
When creating a SuperTag parent template, its name is automatically added to the Tagname Tag Types dialog box in the Tagname Dictionary and is immediately available for selection when a new tagname is created. WindowMaker does not need to be restarted to define tagnames using a newly created SuperTag type.

Creating a SuperTag Template Structure

To realistically illustrate the SuperTag concept in a factory environment, let's assume that you have four identical refrigerated storage rooms in which you store beef, pork, chicken and turkey. Each of these cold rooms has a room temperature and two evaporator units. Each evaporator unit has seven data values that you need to monitor or control in runtime.

For example:

Without the use of SuperTag templates, you would need to manually define each individual tagname for each data value in each cold room multiplied by the total number of cold rooms. In other words, you would have to organize and define dozens of tagnames in the Tagname Dictionary!



By using SuperTags, you can save hours of development time and minimize the chance of making errors. Using the Cold Room scenario described above, you will create one SuperTag parent template called "EvapUnit." This EvapUnit will later become a child member of the ColdRoom parent template. This is a "detail-up" design concept.

Editing SuperTag Templates and Member Tagnames

SuperTag templates or member tagnames can be modified at any time. When an existing SuperTag template or its members are modified, all existing instances of that template are not affected. (Instances are tagnames defined in the Tagname Dictionary that use a SuperTag for their type.) In other words, modifications made to a SuperTag are not retroactive. However, all new instances defined using the modified SuperTag will use the new composition. Similarly, if a member tagname is added to a SuperTag instance through an alternative method, its template is not updated.

Advantages and Disadvantages of Using SuperTags

Advantages

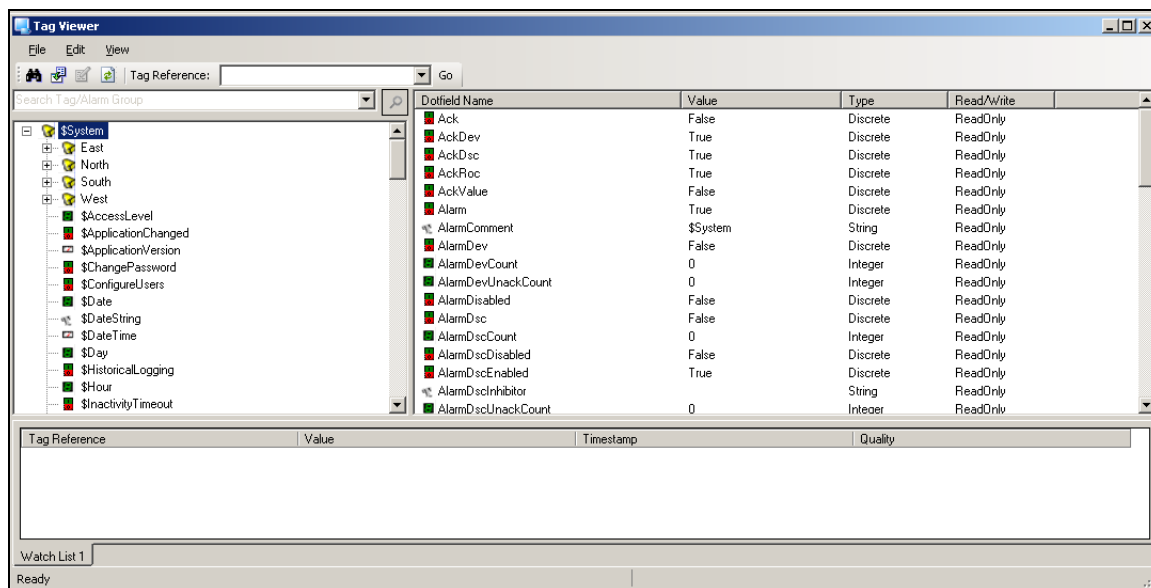
- All SuperTag sub-member tagnames behave exactly like normal tagnames
- Supports alarms and history trends
- Supports Dot Fields
- Development tool
- Template file can be copied

Disadvantages

- One SUPERTAG.DAT file
- No individual SuperTag member deletion
- Slight increase in tag count

Tag Viewer

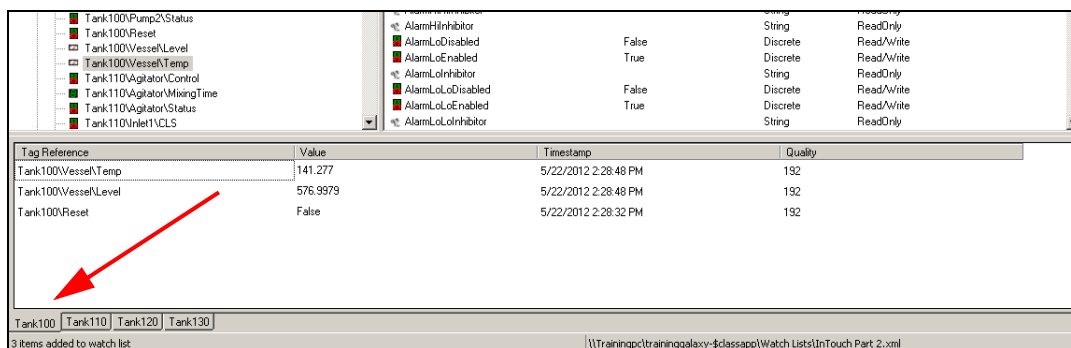
The Tag Viewer displays live tag data values without being in an InTouch window.



Because some tags may be both read and write and Tag Viewer allows direct access to them, it is not enabled by default and must be configured to display during Runtime. It is also recommended that security be set up to restrict access only to those users with the proper access levels.

Watch Lists

Within the Tag Viewer, create customized lists of tags called Watch Lists. Multiple Watch Lists can be displayed as tabs called Watch Windows located across the bottom of the Tag Viewer. These Watch Windows can be named, saved, and retrieved as needed.



Lab 2 – Tag Viewer

Introduction

In this lab, you will configure the InTouch application to access Tag Viewer. The purpose of this lab is to setup and test the Tag Viewer retrieval functionality using both a drop-down menu and a script.

Objectives

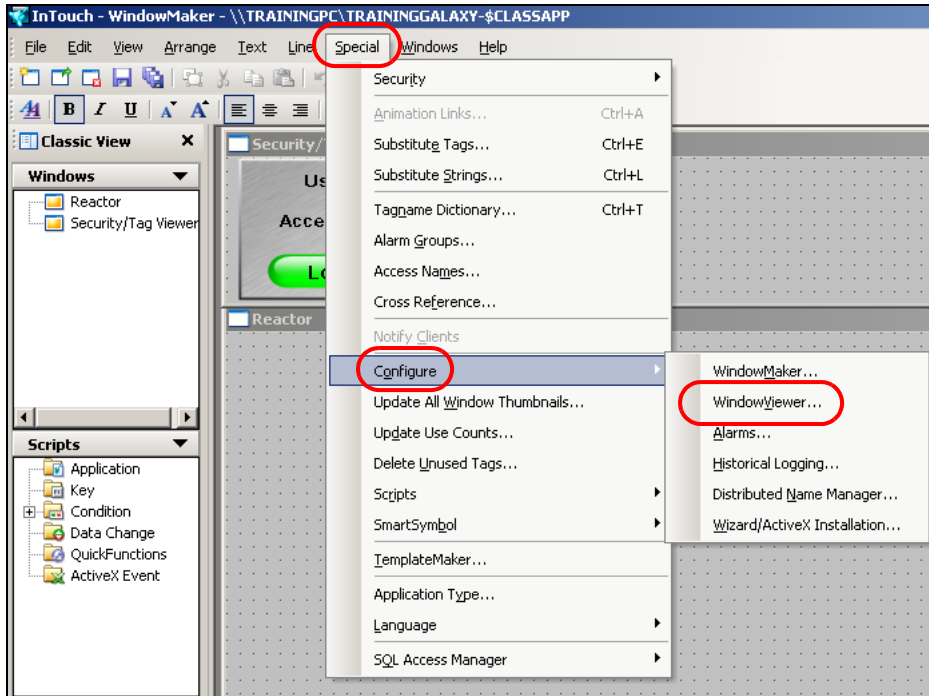
Upon completion of this lab, you will be able to:

- Configure an application for Tag Viewer
- Create users and set security access in the application
- Access Tag Viewer from a drop-down menu and via script
- Create and configure watch lists in Tag Viewer

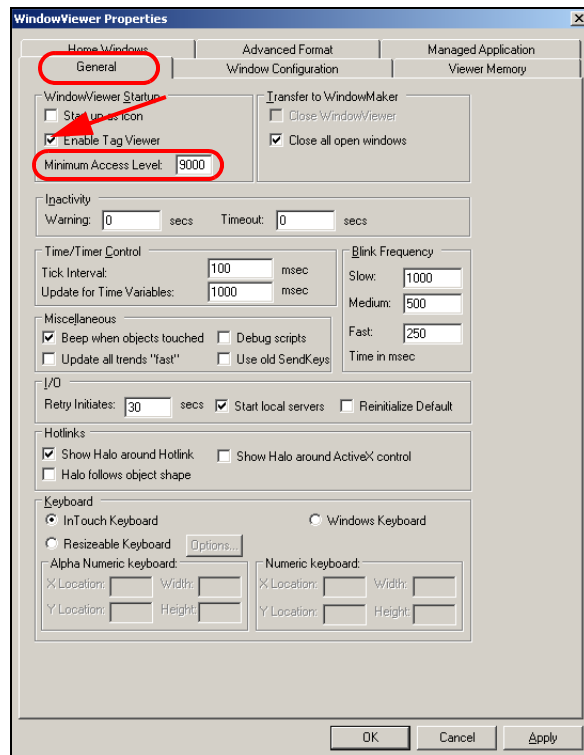
Configure an Application for Tag Viewer

In order to access Tag Viewer, you must first setup the application to enable the Tag Viewer function, using a drop-down menu or via script.

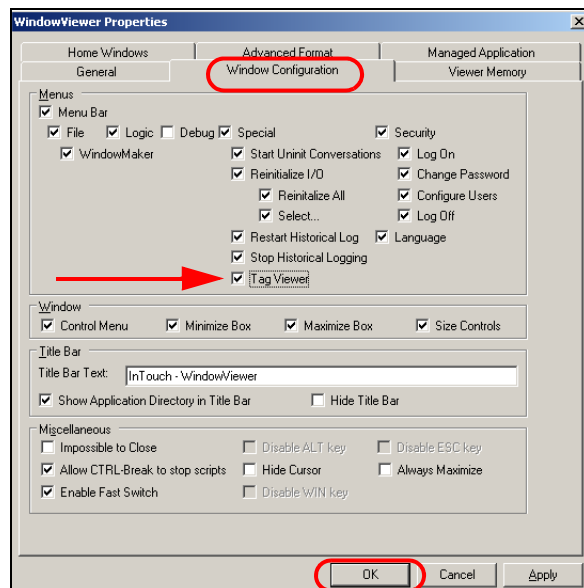
1. In InTouch WindowMaker, on the **Special** menu, click **Configure | Window Viewer**.



2. In the **General** tab, in the **WindowViewer Startup** area, check **Enable Tag Viewer**, and then lower the **Minimum Access Level** to **9000**.



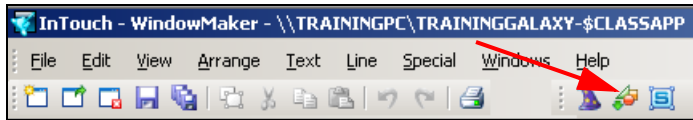
3. Select the **Window Configuration** tab, and then check **Tag Viewer**.
This allows the **Tag Viewer** to be called from a drop-down menu.



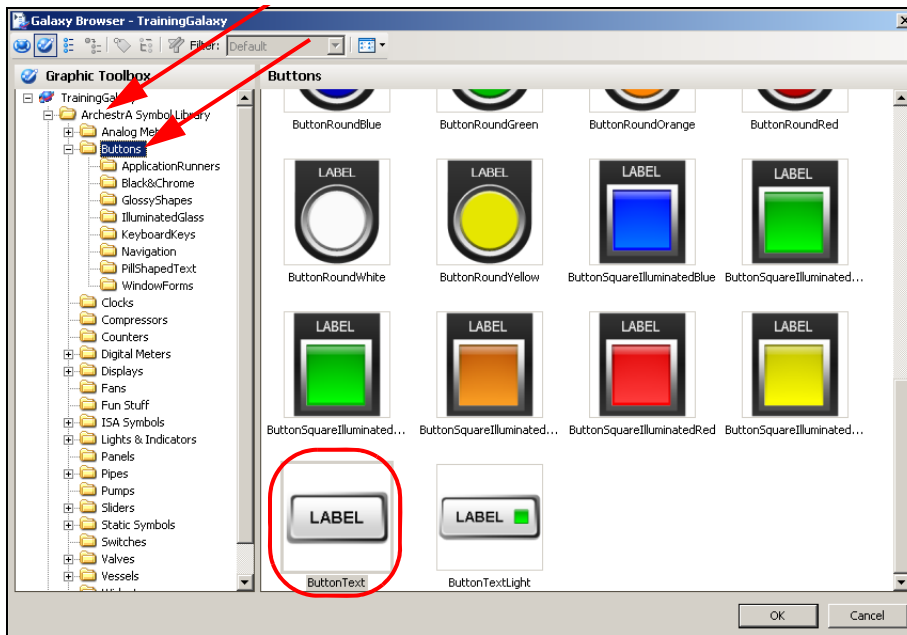
4. Click **OK**.

Next you will add a button to call Tag Viewer via script.

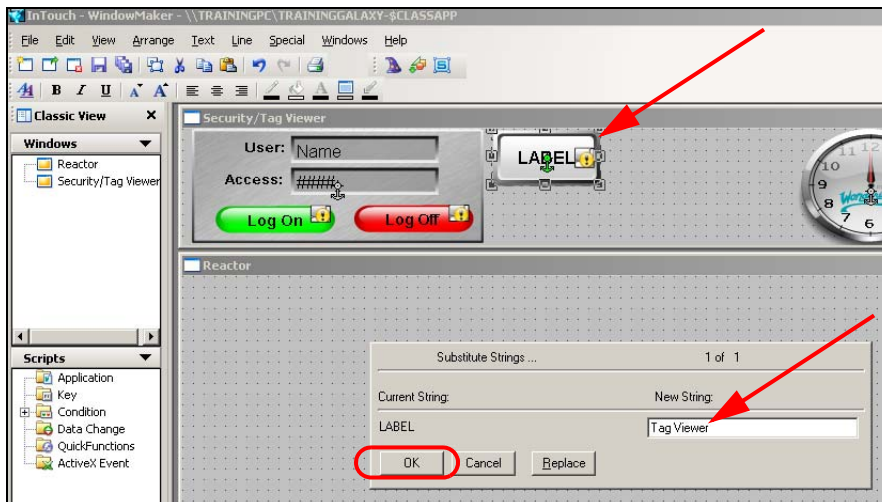
5. Click **Embed Archestra Graphic**.



6. Expand **Archestra Symbol Library**, click **Buttons**, scroll to the bottom and double-click **ButtonText**.

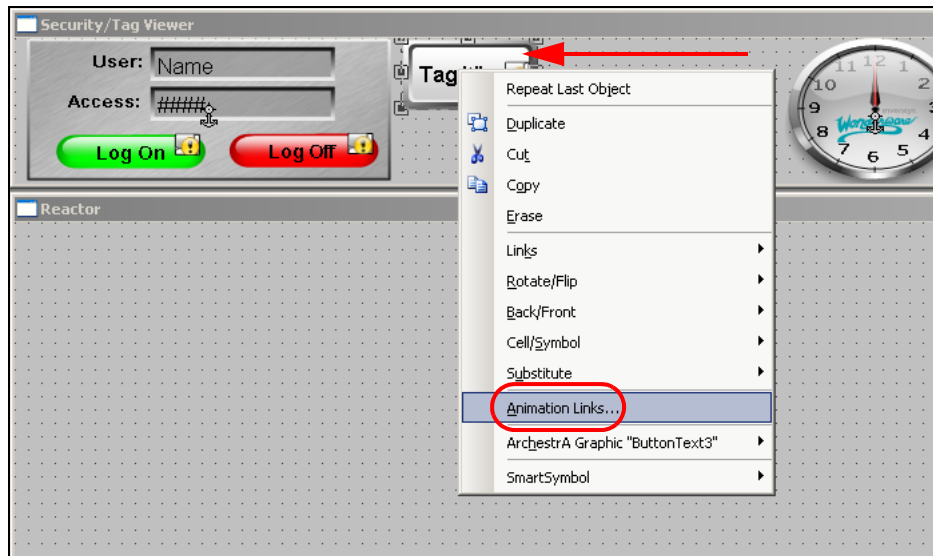


7. Place the object to the right of the **Log On** object, and then use the **Substitute String** command (shortcut key **Ctrl+L**) to change the object text from **LABEL** to **Tag Viewer**.

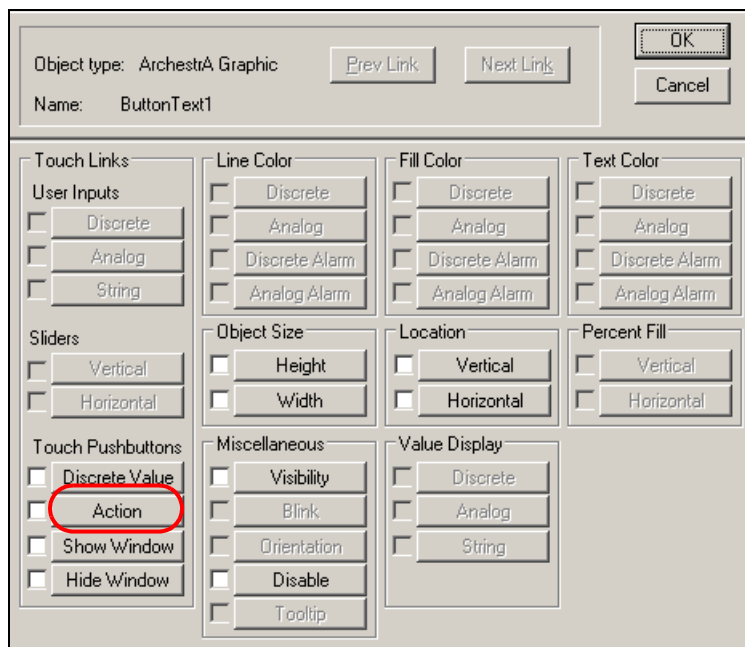


8. Click **OK**.

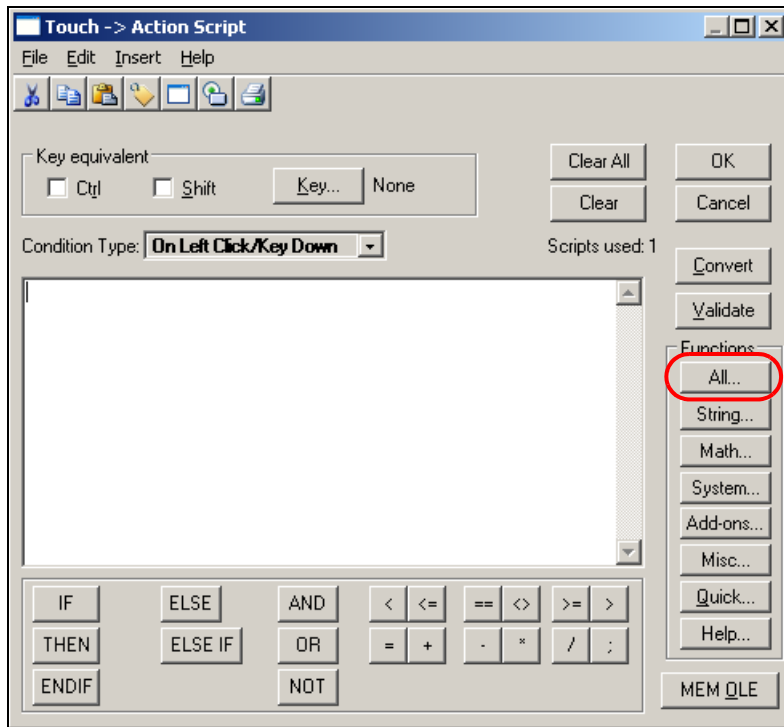
9. Right-click the object and select **Animation Links**.



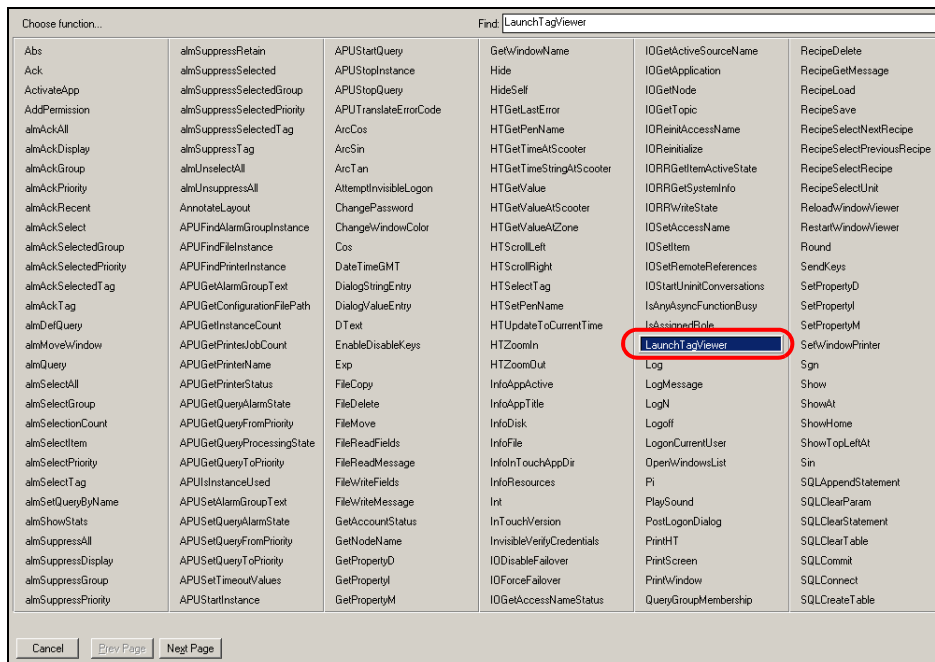
10. In the **Touch Pushbuttons** area, click **Action**.



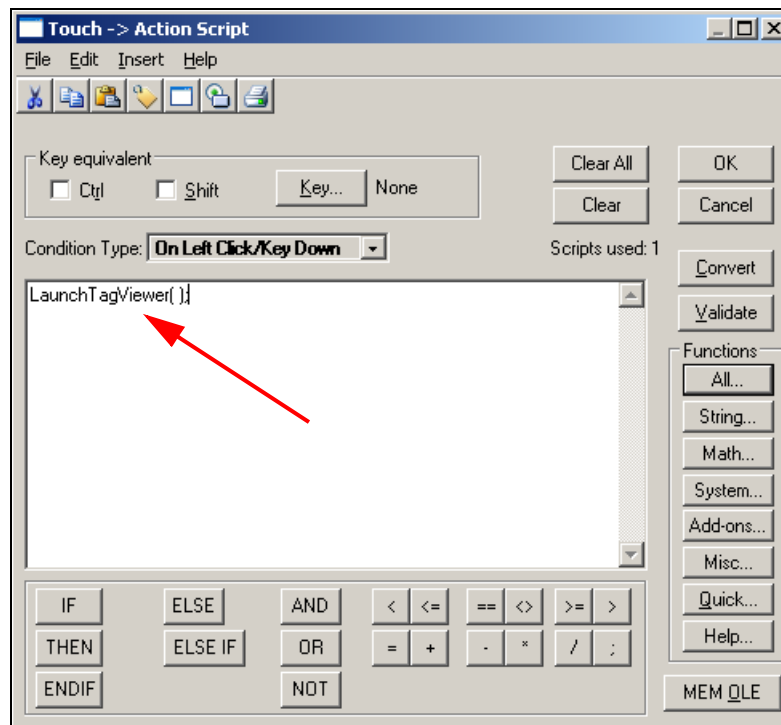
11. Clear any existing script, and then under **Functions** click **All...**



12. Select **LaunchTagViewer**.

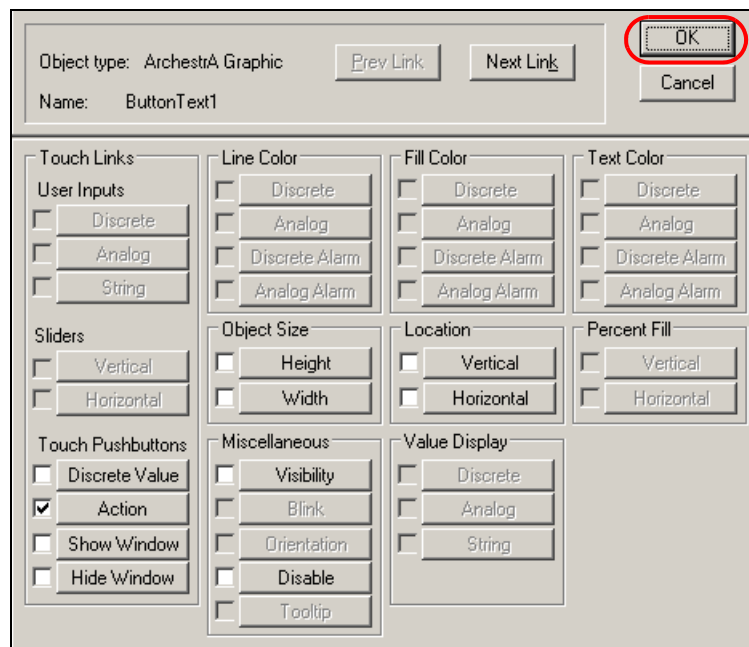


This will place the function into the script pane.



13. Click **OK**.

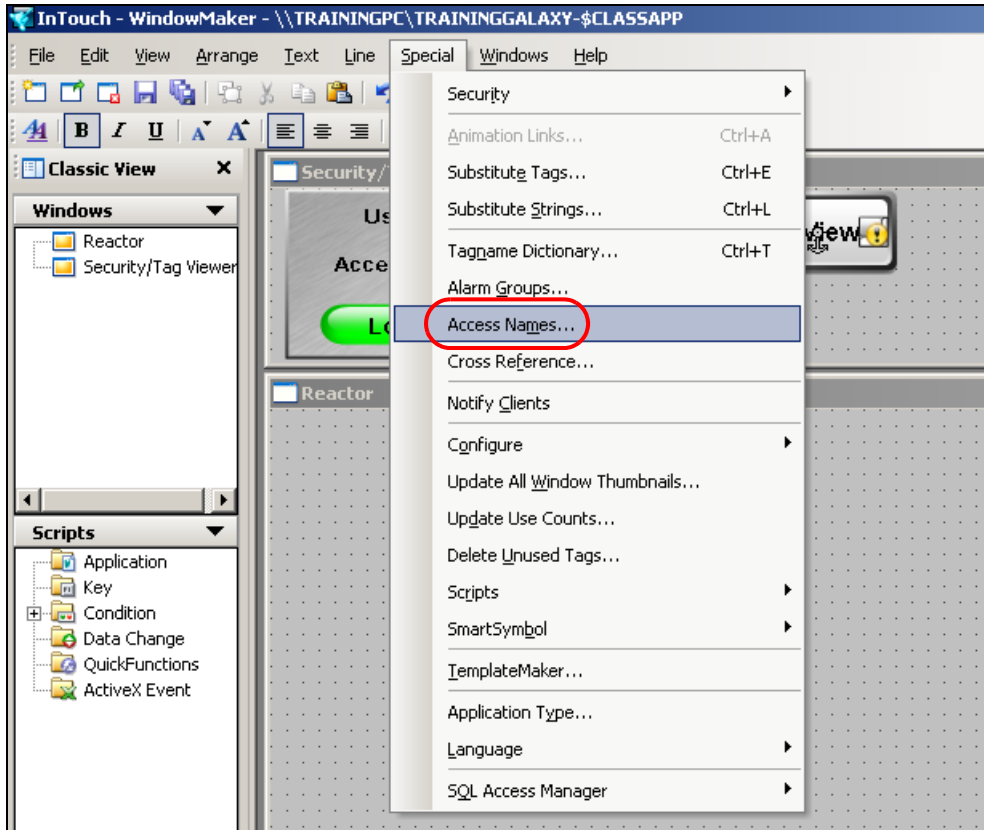
14. Click **OK** again.



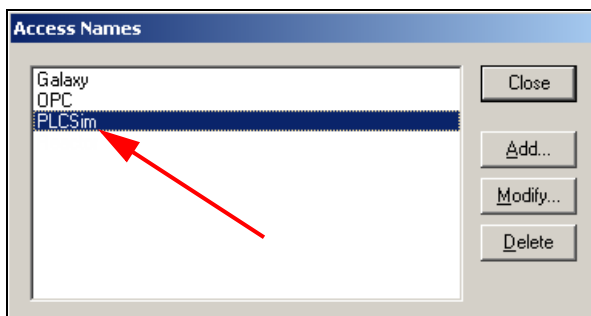
Configure the Access Name

To allow the I/O data necessary for training purposes, you may need to configure the access name to work with the data source.

15. On the **Special** menu, click **Access Names**.



16. In the **Access Names** dialog box, select **PLCSim**, and then click **Modify**.



17. Enter the **Node Name** provided by your instructor, and then click **OK**.

Note: For this example TrainingPC has been used, but node name may vary by location.

Modify Access Name

Access: **OK**

Node Name: **Cancel**

Application Name: **Failover**

Topic Name:

Which protocol to use:

☐ DDE ☒ SuiteLink ☐ Message Exchange

When to advise server:

☐ Advise all items ☒ Advise only active items

☐ Enable Secondary Source

18. Click **Close**.

Access Names

Galaxy
OPC
PLCSim

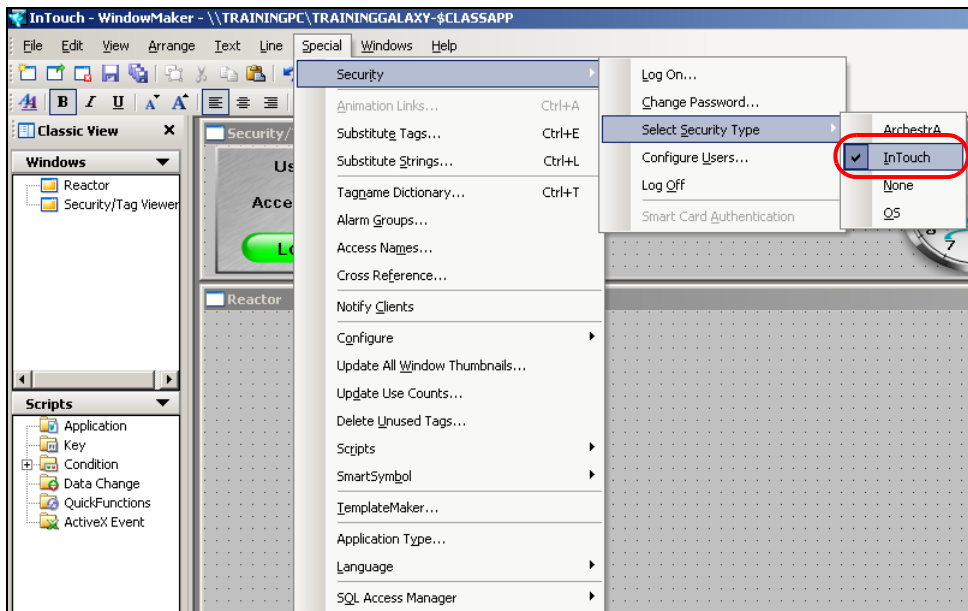
Close

Add...
Modify...
Delete

Create Users and Set Security Access

To test the Tag Viewer access, you will need to create two users - one with the security level necessary to access Tag Viewer, and one without.

19. On the **Special** menu, click **Security | Select Security Type**, and then make sure that **InTouch** is selected.



20. On the **Special** menu, click **Security | Log On**, and then log in with the default administrator account.

Name: administrator

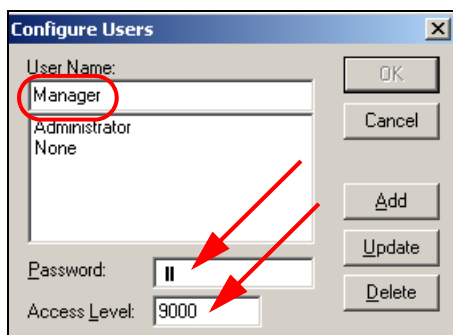
Password: Wonderware

21. Click **OK**.
22. On the **Special** menu, click **Security | Configure Users** to add additional users.
23. To create the first user who will have the ability to access the **Tag Viewer**, enter the following:

User Name: Manager

Password: ww

Access level: 9000

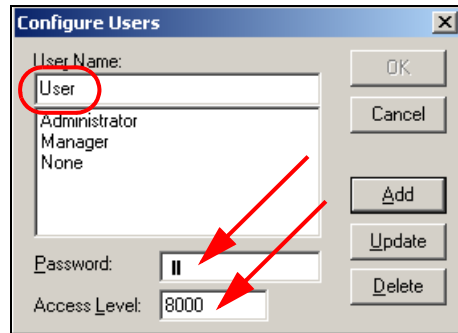


24. Click **Add**.
25. To create the second user who will NOT have the ability to access the **Tag Viewer**, enter the following:

User Name: User

Password: ww

Access level: 8000

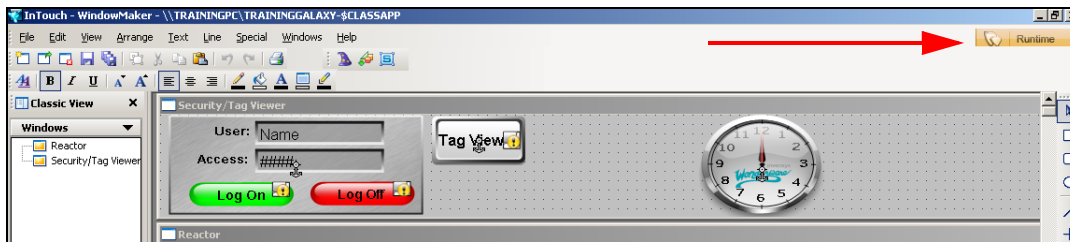


26. Click **Add**.
27. Click **OK**.

Test What You Have Done

Use the Manager and User accounts to check if the Tag Viewer settings are working.

28. Click **Runtime**.



As shown in the next step, notice the **User** is **None** and the access level is set to **0**.

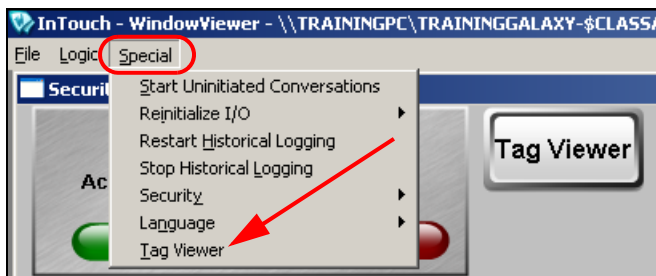
Now verify there is currently no access to the Tag Viewer.

29. Click **Tag Viewer**.



Nothing happens.

30. On the **Special** menu, click **Tag Viewer** to verify you cannot access the **Tag Viewer** from the menu.



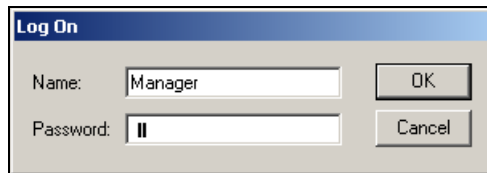
Nothing happens.

Now you will test the settings for the two users that you created.

31. Click **Log On**, and then enter the following:

Name: Manager

Password: ww

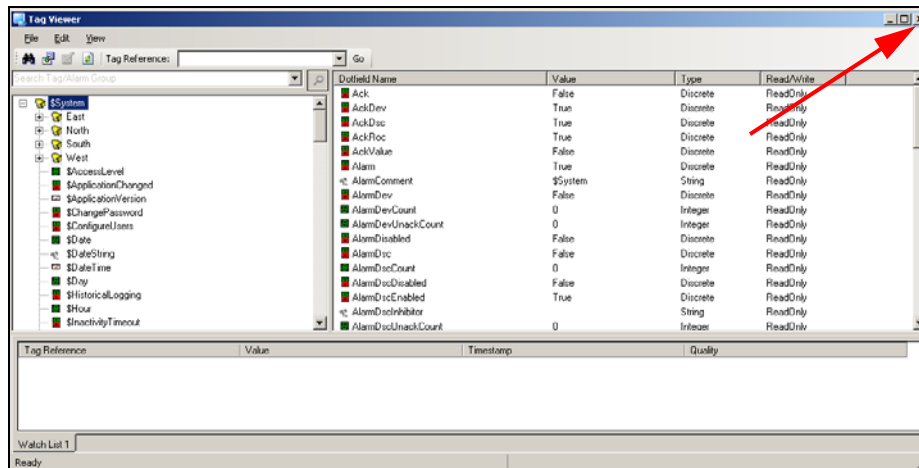


32. Click **OK**.

First, verify that the Manager account has access to the Tag Viewer using both access retrieval methods.

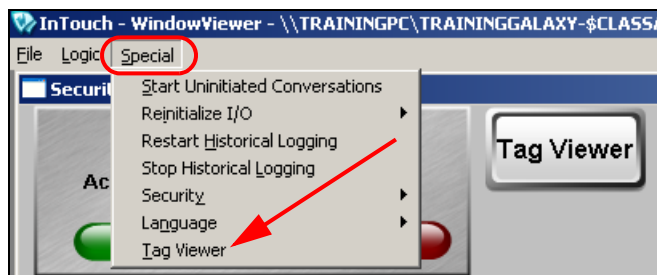
33. Click **Tag Viewer**.

The **Tag Viewer** should display:

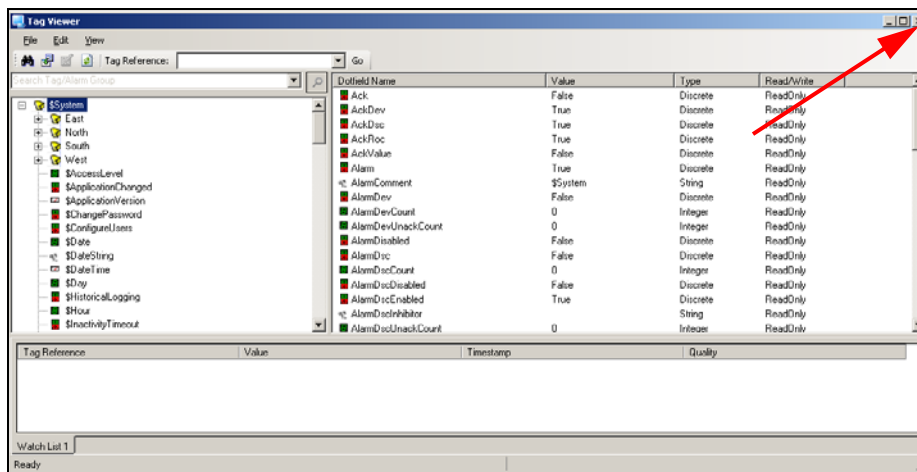


34. Click **X** to close **Tag Viewer**.

35. On the **Special** menu, click **Tag Viewer**.



The **Tag Viewer** should display:



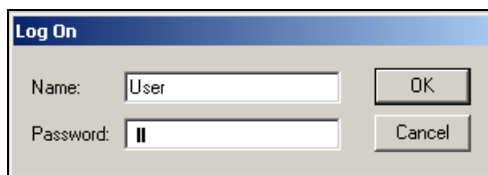
36. Click **X** to close **Tag Viewer**.

Next, verify that the User account cannot access the Tag Viewer using either access retrieval methods.

37. Click **Log Off**, to log off as **Manager**.

38. Click **Log On**, and then enter the following:

Name: User
Password: ww



39. Click **OK**.

40. Click **Tag Viewer**, and then verify that **Tag Viewer** does not display.

41. On the **Special** menu, click **Tag Viewer**, and then verify that **Tag Viewer** does not display.

Create a Watch List

Watch Lists allow you to view the value of a tag or set of tags using the Tag Viewer. You will need to log in as Manager to access the Tag Viewer.

42. Click **Log Off**, and then click **Log On** and enter the following:

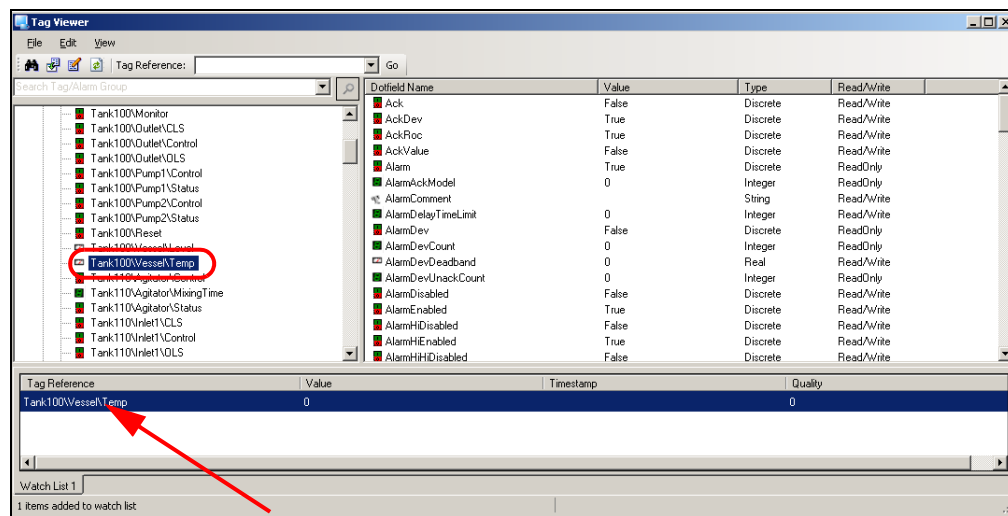
Name: Manager

Password: ww

43. Click **OK**.

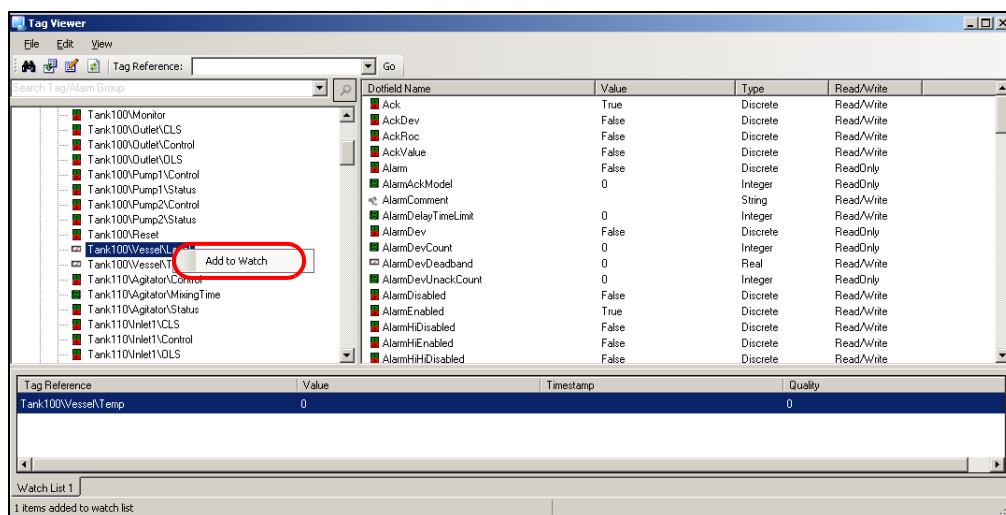
44. Click **Tag Viewer**.

45. Expand **North**, scroll down and double-click **Tank100\VesselTemp** to place it in **Watch List**.



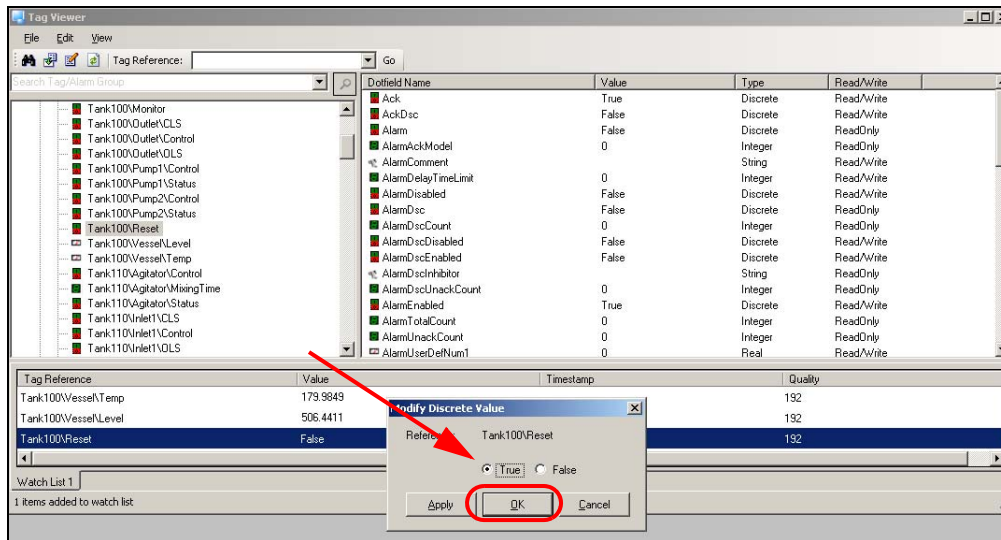
Alternatively, you can select and manually drag it to the lower **Watch List** section, or right-click on the tag and select **Add to Watch**, as shown in the next step.

46. Right-click **Tank100\VesselLevel** and select **Add to Watch**.



Depending on the configuration of the tags, these points may also be read-write.

47. Add **Tank100\Reset** to the **Watch List**.
48. In **Watch List**, double-click **Tank100\Reset**, and then set its value to **True**.



49. Click **OK**.

Note: This will reset the value of the **Tank100\Vessel\Level** to **0**.

Multiple Watch Windows

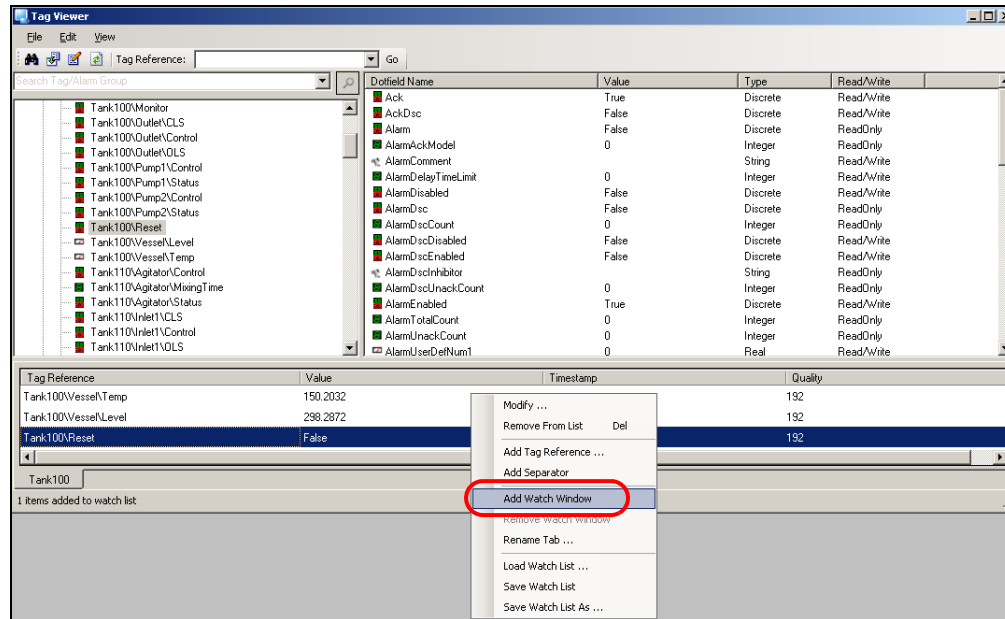
The Tag Viewer can store multiple Watch Windows as tabs across the bottom of the viewer window. To make organizing the individual Watch Windows easier, the Tag Viewer allows you to rename each tab.

50. Right-click the **Watch List 1** tab and select **Rename Tab**.

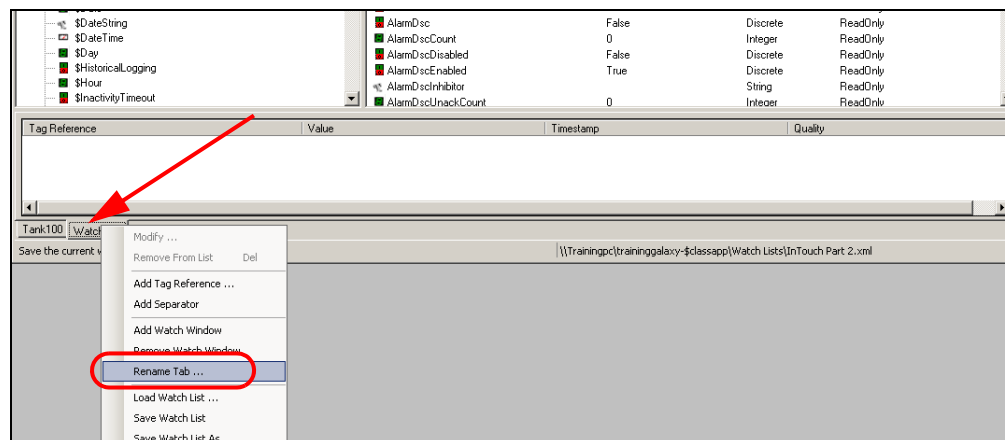
51. Enter **Tank100**, and then, and then click **OK**.

You will now add another Watch Window.

52. In **Watch List**, right-click and select **Add Watch Window**.



53. On the new tab, right-click and select **Rename Tab**.

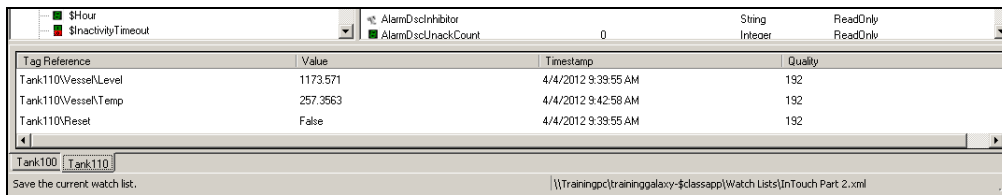


54. Enter **Tank110**, and then click **OK**.



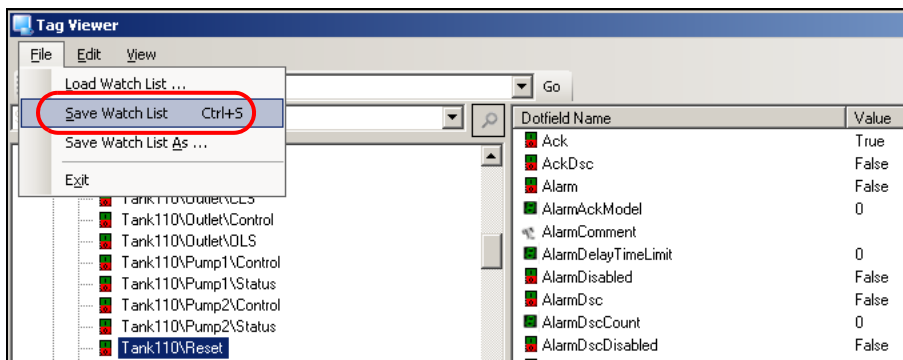
55. Add the following tags to **Tank110 Watch List**, using any of the methods described previously.

- **Tank110\VesselLevel**
- **Tank110\VesselTemp**
- **Tank110\Reset**

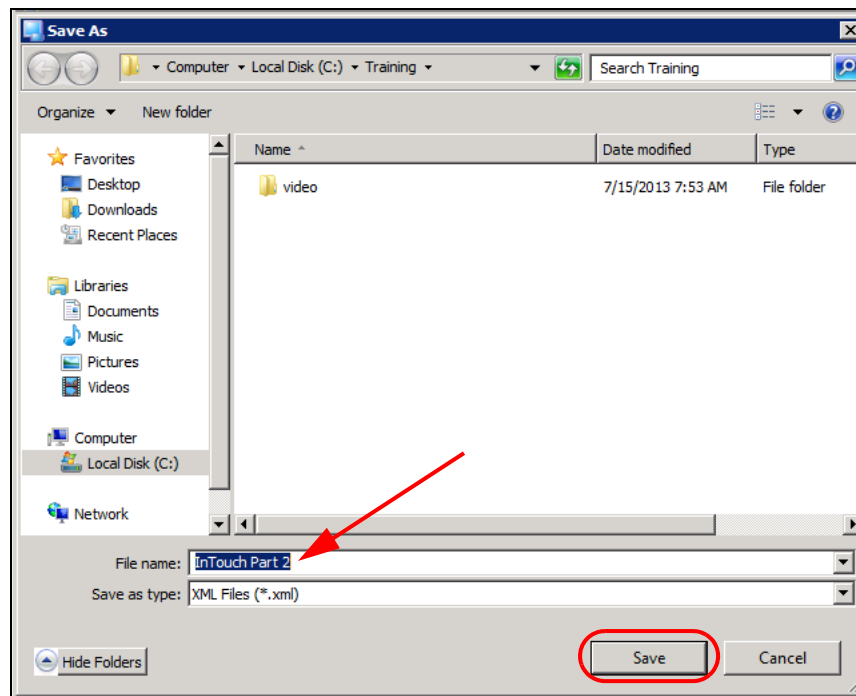


You will need to save these **Watch Windows** if you want to access them later.

56. In **Tag Viewer**, on the **File** menu, click **Save Watch List**.



57. Navigate to **C:\Training** folder and name the file **InTouch Part 2**, and then click **Save**.



Note: If you exit WindowViewer and need to access your previously saved **Watch Lists**, simply click **File | Load Watch List** from **Tag Viewer**, select your previously saved file, and then click **Open**.

58. Close **Tag Viewer**.



Module 3 – Application Navigation Tools

Section 1 – Visualization and Indirect Addressing	3-3
Lab 3 – Indirect Addressing	3-7
Section 2 – The Show Symbol Animation Link	3-39
Lab 4 – Show Symbol Animation Link	3-41
Section 3 – The ShowGraphic Script Function	3-51
Lab 5 – Basics of the ShowGraphic Function	3-53
Section 4 – Panel Display	3-71
Lab 6 – Panel Navigation	3-73

Module Objectives

- Define SuperTags and how they are used
- Build an expandable user interface with indirect tags
- Use Show Symbol animation and the ShowGraphic function to call a symbol not embedded within the InTouch application

Section 1 – Visualization and Indirect Addressing

This section discusses how InTouch is used to visualize data and how that data can be displayed through the use of Indirect Addressing.

Visualization

InTouch allows users to create graphical representations of their plants, equipment, machinery, etc. Once the graphics have been created, users are able to link them to tag data using custom properties to simulate and display what is going on in the real world.

Indirect Addressing

Indirect type tagnames allow the creation of one window and reassign the tagnames in that window to multiple sources. For example, create a Data Change QuickScript that would modify the source for all tagnames in a window, based on a value that has changed.

When equating an indirect tagname to another source tagname, both the indirect tagname and the source tagname become linked to each other in every aspect including Dot Fields, scripts, etc. If the value of the source tagname changes, the indirect tagname reflects the change. If the indirect tagname's value changes, the source tagname changes accordingly. Define indirect tagname values in the database as retentive, which allows them to retain their last Tagname assignment on startup.

.Name Field

Indirect tagnames are assigned by using the .Name field. For example, if creating an indirect analog tagname called "Setpoint" and using the expression below in a QuickScript, "Setpoint1" would become the source for the value of "Setpoint" and vice versa:

```
Setpoint.Name = "Setpoint1";
```

OR

```
Setpoint.Name = Setpoint1.Name;
```

Note: Indirect SuperTags allow the assignment of the .Name field to all members of the SuperTag with one line of scripting, otherwise a script is needed for each included tag.

Concatenation

Concatenate tagnames for use in indirect tags. For example, if a Data Change QuickScript is created that executes each time the value of the tagname "Number" changes, the indirect tagname, "Setpoint," would change accordingly:

```
Number=1;
```

```
Setpoint.Name = "Setpoint" + Text(Number, "#" );
```

When this QuickScript executes, the value of the analog tagname "Number" is converted to text and added to the analog tagname "Setpoint," making "Setpoint.Name" equal to "Setpoint1." Indirect analog type tagnames are used for both integer (whole numbers) and real (floating point) tagnames.

Advantages of Indirect Tag Addressing

- Saves on window development
- Scalable
- Supports alarm and history trends
- Supports Dot Fields (attributes of a tag)
- Supports Remote Tagname Referencing
- Works With Dynamic Reference Addressing

Disadvantages of Indirect Tag Addressing

- Increases tag count
- Requires planning (e.g., tagnames, scripting)
- Special steps must be taken to be sure that tags assigned to the indirect tags are not deleted

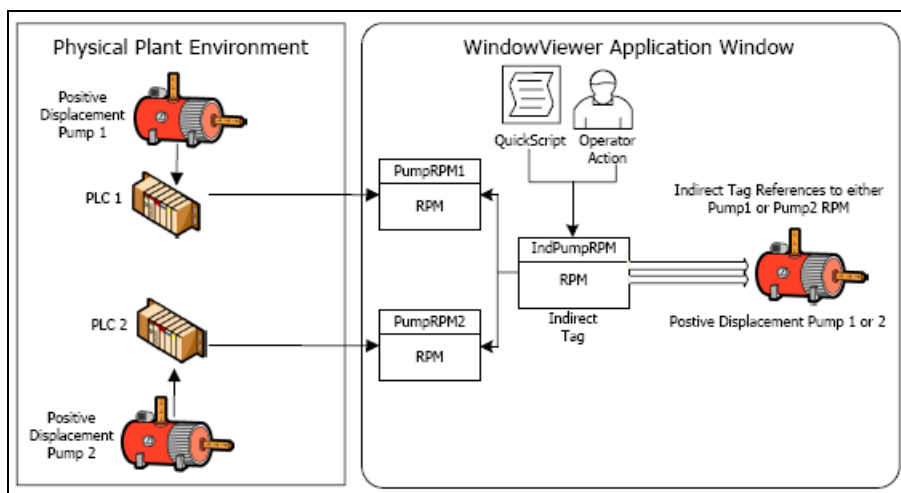
Note: If the Indirect assignment does not exist or the tag has not been created, the last assignment is retained.

Indirect Tag

Indirect tags serve as pointers to other tags. When equating an indirect tag to another source tag, the indirect tag acts as if it is the source tag.

Using indirect tags, create applications with window objects that show values from multiple tags. Indirect tags are tags that can act on behalf of another tag. In the programming world, these are called pointers. Point indirect tags at direct tags and then use the indirect tags as if using direct tags.

Using indirect tags reduces application development time. Instead of creating a separate set of tags and window objects for each process component, create applications with window objects that show values from multiple tag sources.



- Indirect tags can be equated to integer and real tags. Indirect tags can be mapped to any other tag as long as they are the same tag type
- When equating an indirect tag to another source tag, the indirect tag behaves as if it is the source tag. If the value of the source tag changes, the indirect tag reflects the change. If the indirect tag's value changes, the source tag changes accordingly
- Use scripts to assign input source tags to an indirect tag. The script can be triggered by an application event or an operator action like clicking a window button
- Assign retentive attributes to indirect tags. With retention, the indirect tag retains its most recent tag assignment when the InTouch application starts again

Lab 3 – Indirect Addressing

Introduction

In this lab, you will build an expandable user interface using both super tags and indirect addressing. The design of the user interface will allow you to add additional tags as needed without requiring a redesign or a rebuild of the user interface.

Objectives

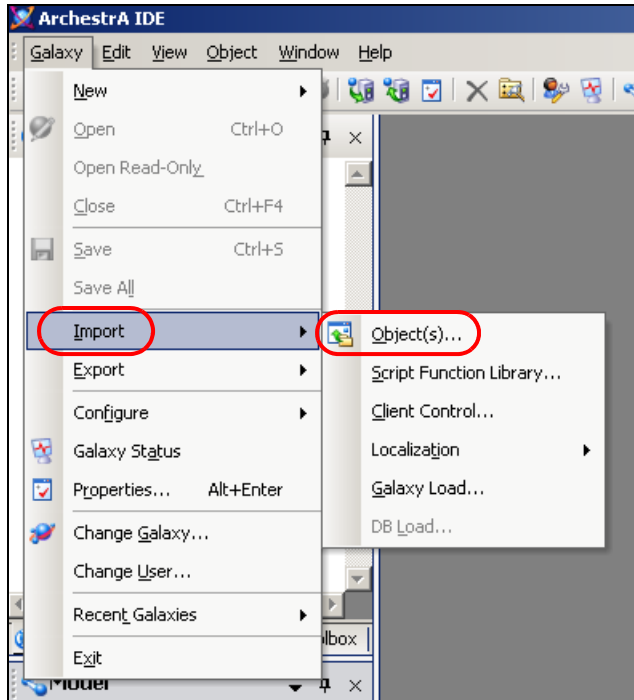
Upon completion of this lab, you will be able to:

- Import Custom Symbols
- Build a user interface
- Implement scripting for indirect addressing
- Modify an application to reference indirect tags

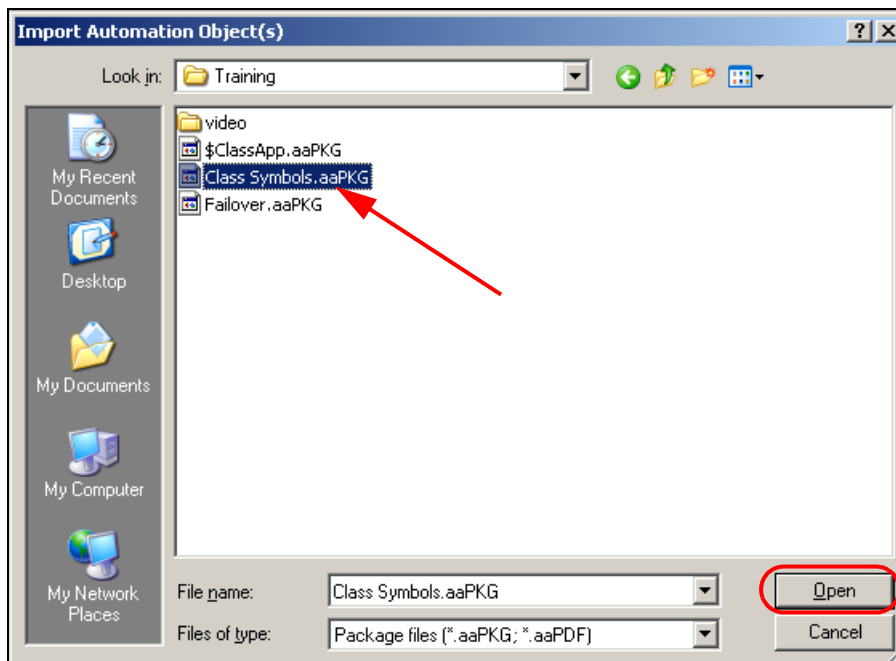
Import Custom Symbols

Archestra IDE allows you to import custom built symbols and place them into your InTouch application.

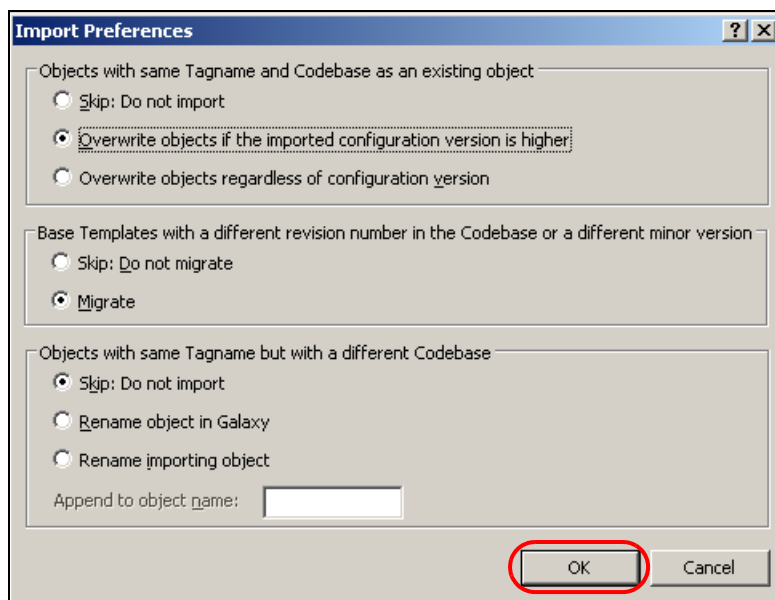
1. In Archestra IDE, on the **Galaxy** menu, click **Import | Object(s)**.



2. Navigate to **C:\Training\Class Symbols.aaPKG**, and then click **Open**.

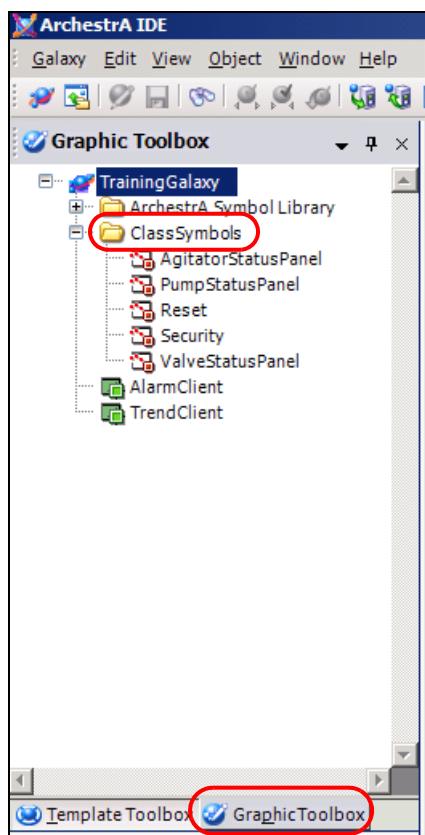


3. Accept the default settings, and then click **OK**.



4. When the import is completed, click **Close**.

The **ClassSymbols** toolset has been added to **Graphic Toolbox**.



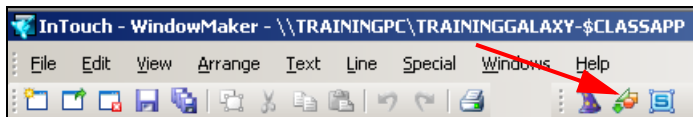
Build the User Interface

The ArchemstrA Symbol Library comes stocked with a large selection of pre-built symbols to assist in building your user interface (UI). In this portion of the lab, you will use a combination of custom and pre-built symbols from ArchemstrA IDE to build your UI.

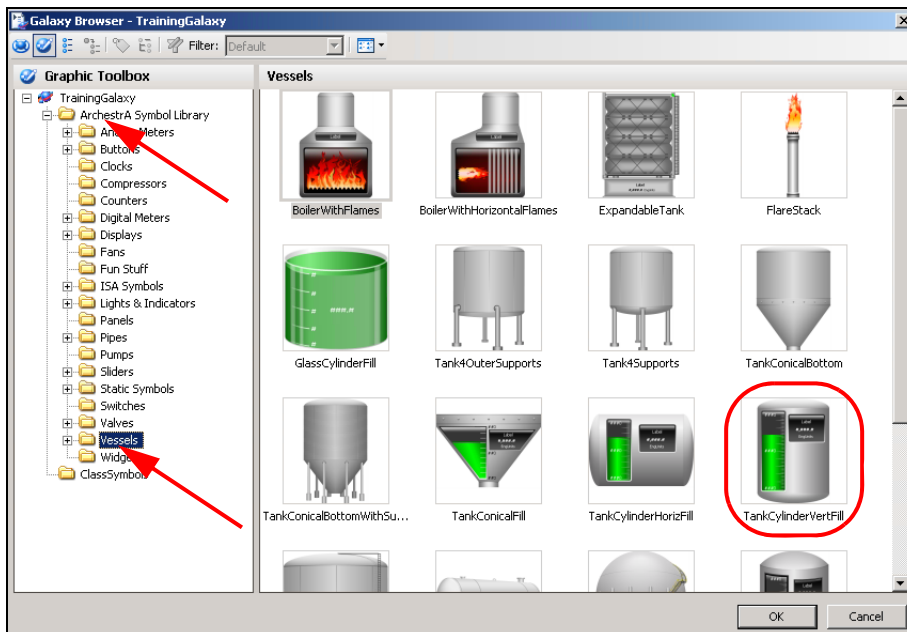
The Tank

ArchemstrA Symbol Library provides a selection of Vessels to represent Tanks in a process.

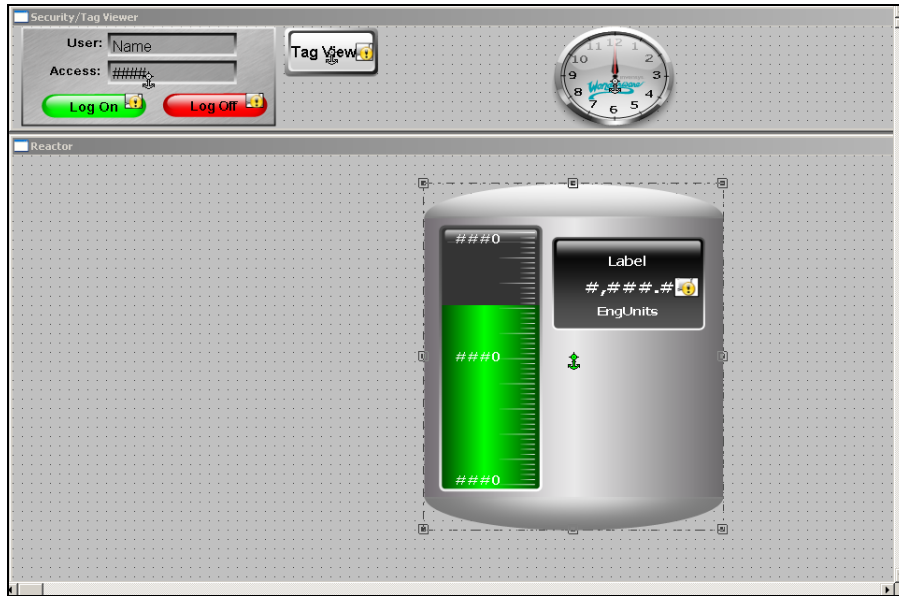
5. In WindowMaker, open the **Security/Tag Viewer** and **Reactor** windows.
6. Click **Embed ArchemstrA Graphic**.



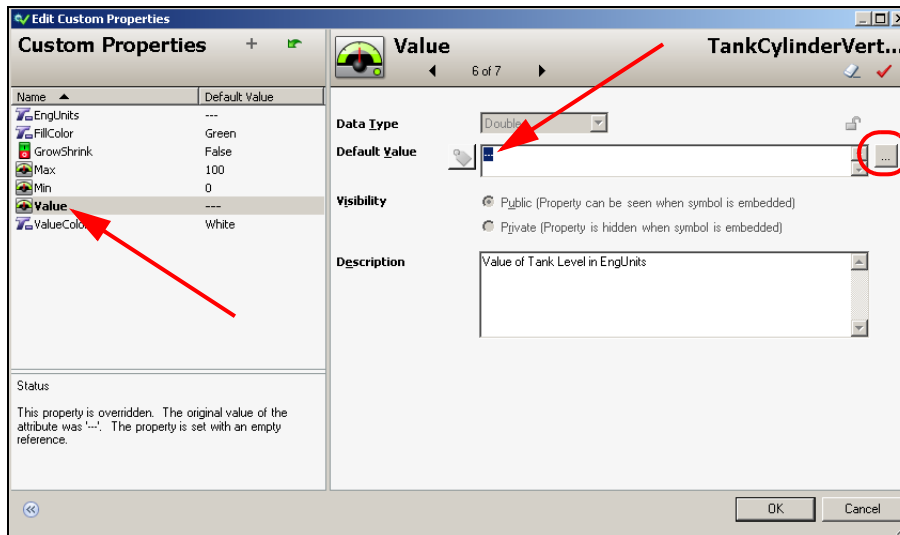
7. In **ArchemstrA Symbol Library**, click **Vessels**, and then double-click **TankCylinderVertFill**.



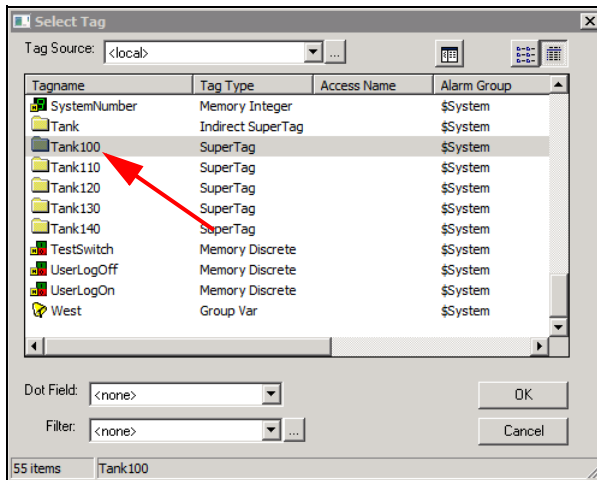
8. In the **Reactor** window, place the symbol and increase its size as needed.



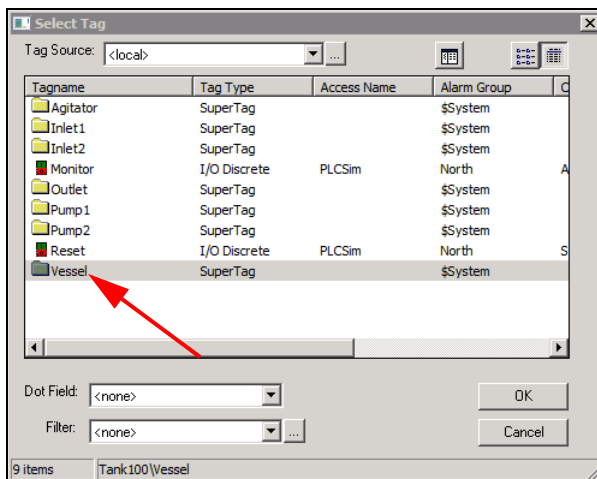
9. Double-click **TankCylinderVertFill** to configure it.
10. Select the **Value** custom property, highlight **Default Value**, and then click **Browse Galaxy**.



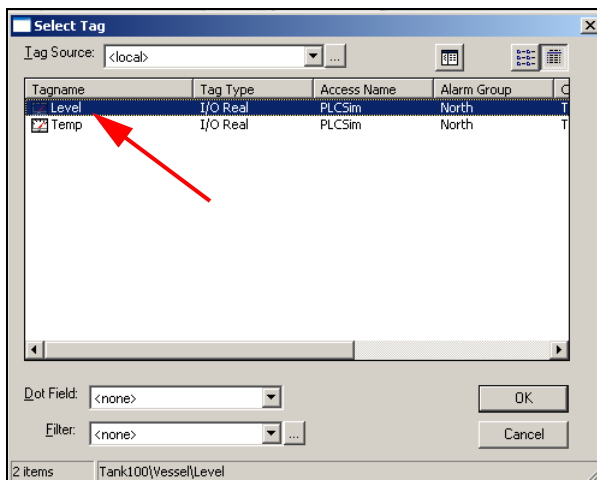
11. Double-click **Tank100**.



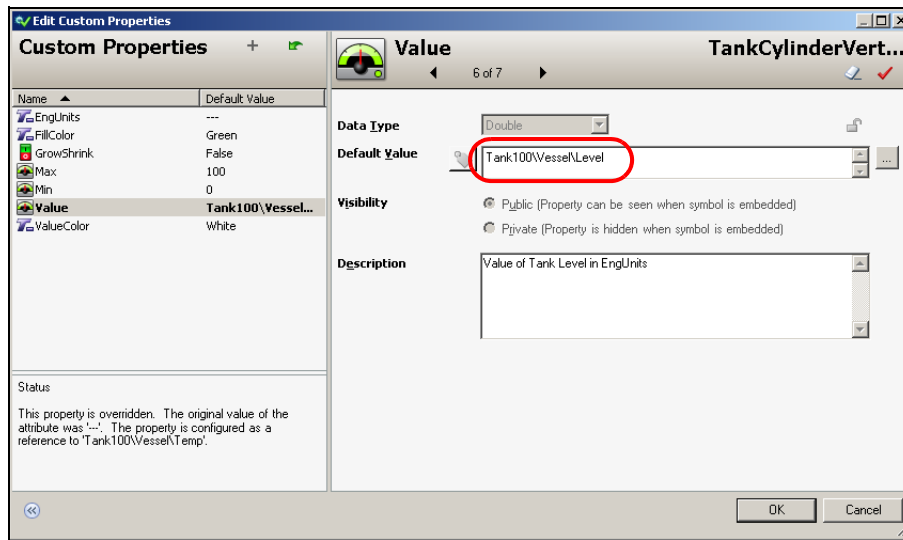
12. Double-click **Vessel**.



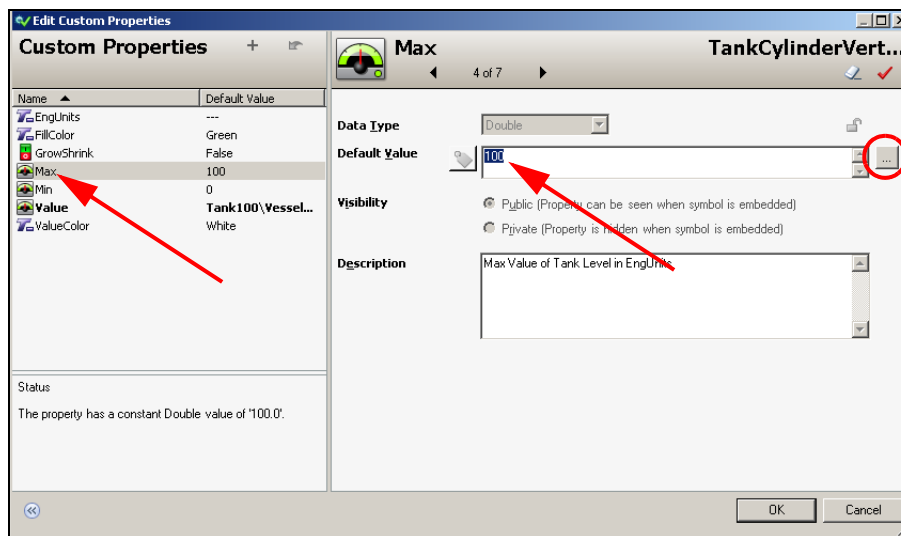
13. Double-click **Level**.



In the **Default Value** field, the highlighted text --- has been replaced by the tagname you selected.

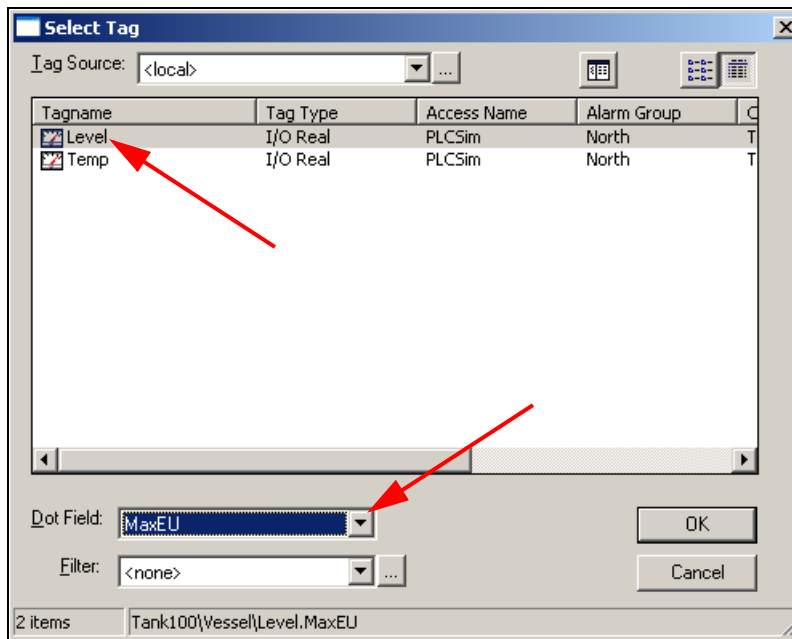


14. Select **Max** custom property, highlight the **Default Value**, and then click **Browse Galaxy**.

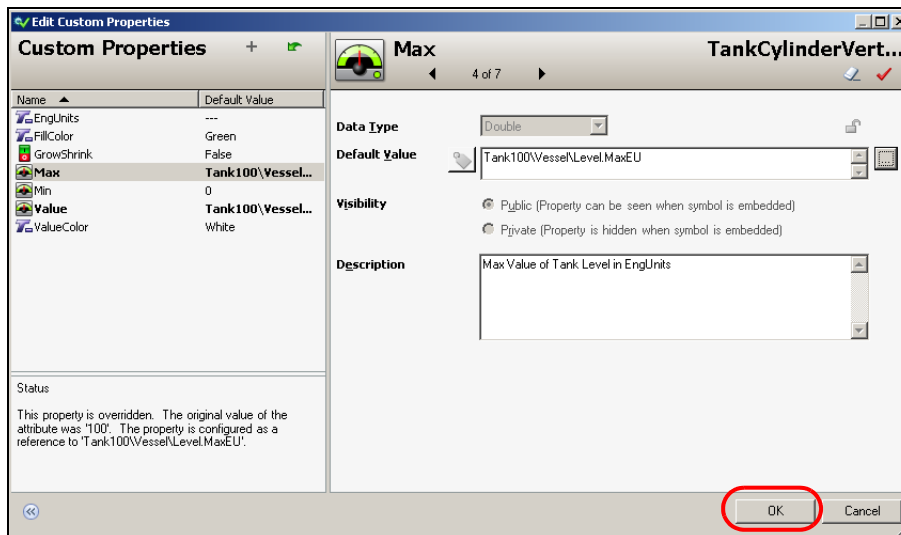


3-14 Module 3 – Application Navigation Tools

15. Double-click **Tank100**, double-click **Vessel**, click **Level**, and then from the **Dot Field** drop-down select **MaxEU**.

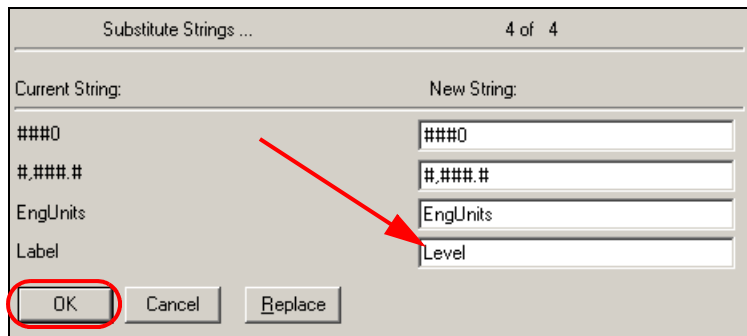


16. Click **OK**.
17. Click **OK** to close the **Edit Custom Properties** window.



Once the custom properties for the tank symbol have been configured, the next step is to change the default label on the tank from **Label** to **Level**.

18. Select the tank symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.
19. Change **Label** to **Level**, and then click **OK**.

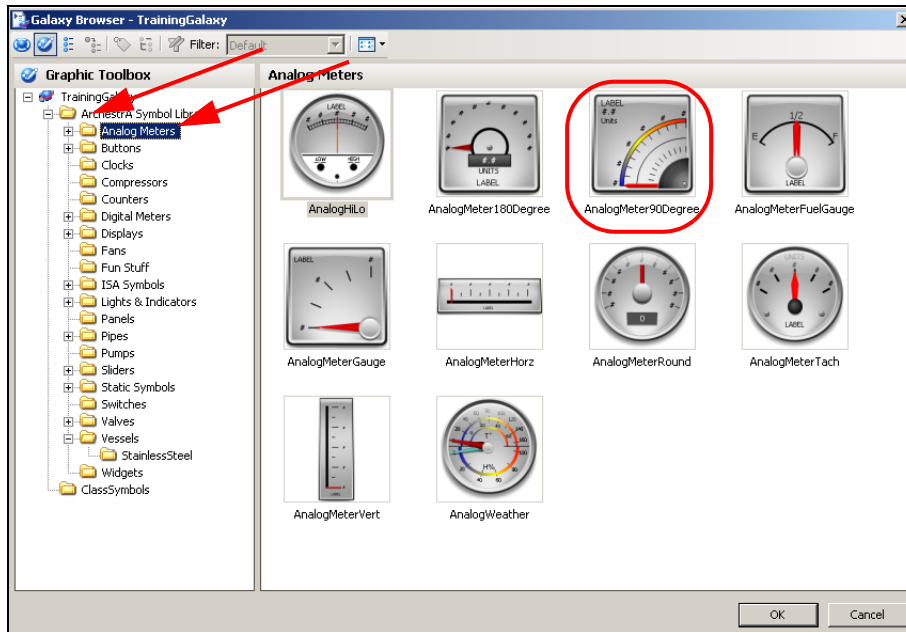


The Analog Meter

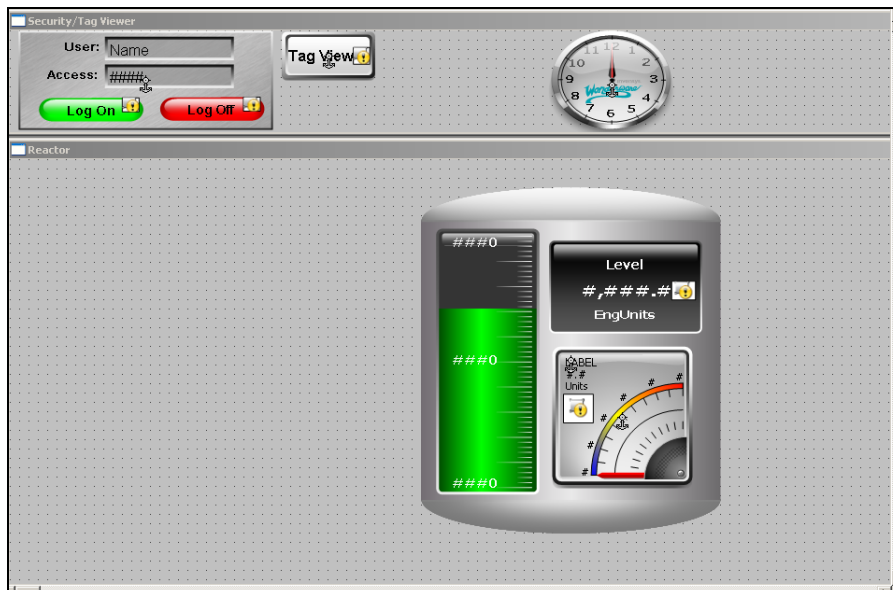
The Analog Meter is another pre-built symbol found in ArchestrA Symbol Library.

20. Click **Embed ArchestrA Graphic**.

21. In **ArchestrA Symbol Library**, click **Analog Meters**, and then double-click **AnalogMeter90Degree**.

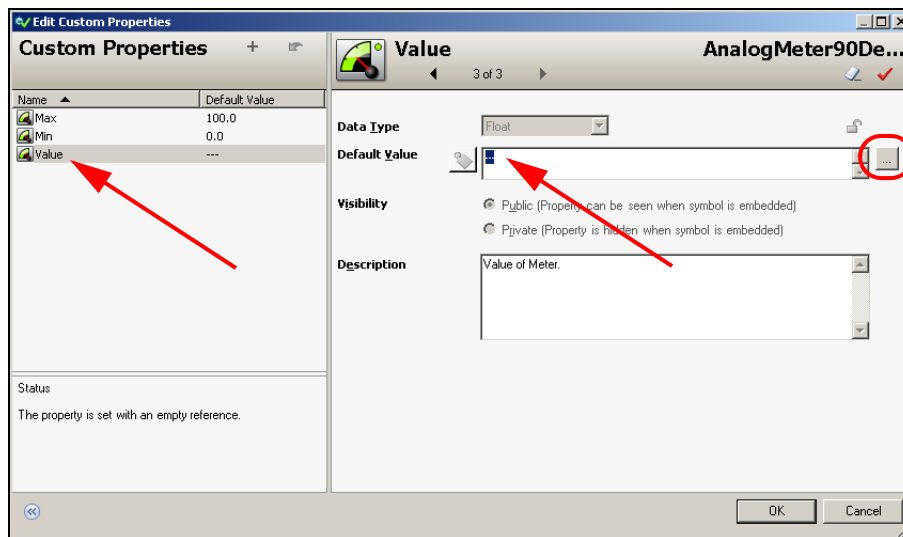


22. Place the **AnalogMeter90Degree** symbol on the tank.

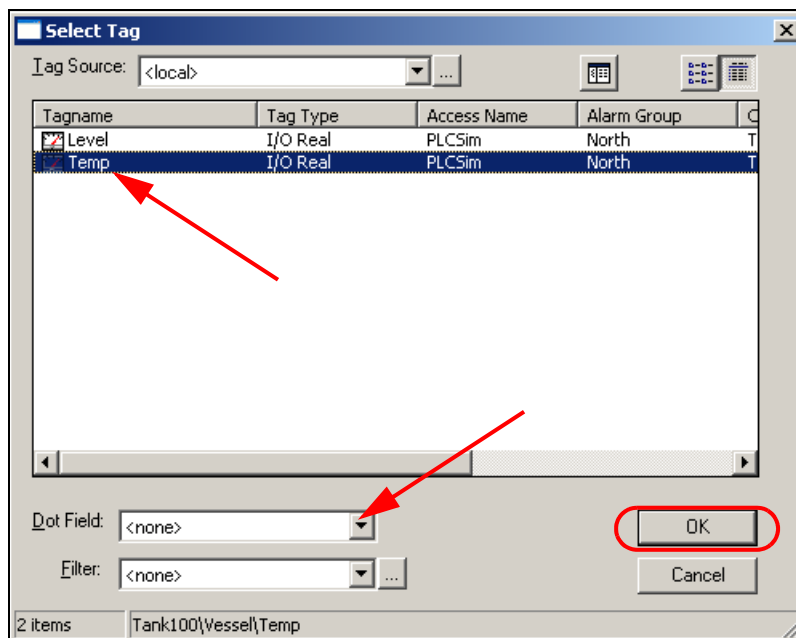


23. Double-click the **AnalogMeter90Degree** symbol to configure it.

24. Select the **Value** custom property, highlight **Default Value**, and then click **Browse Galaxy**.

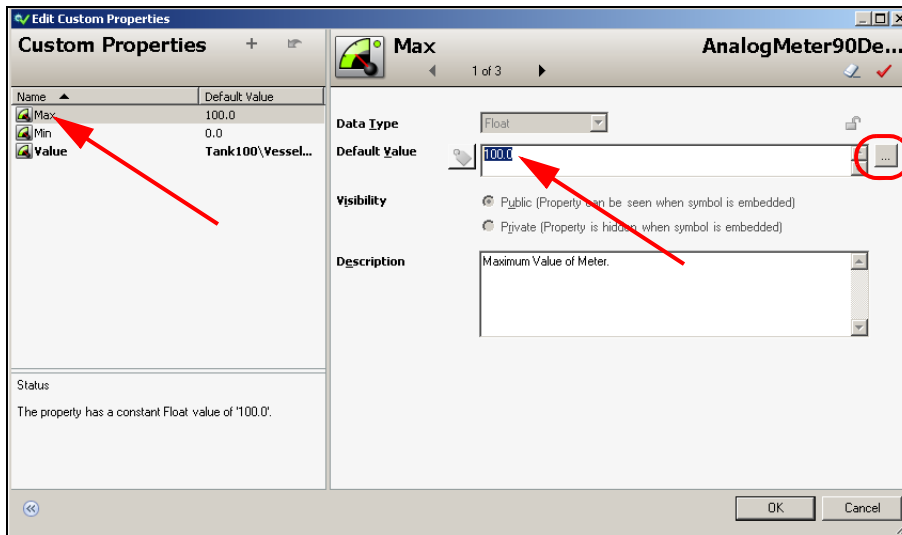


25. Double-click **Tank100\Vessel** and select **Temp**.
26. From the **Dot Field** drop-down, select **<none>**, and then click **OK**.

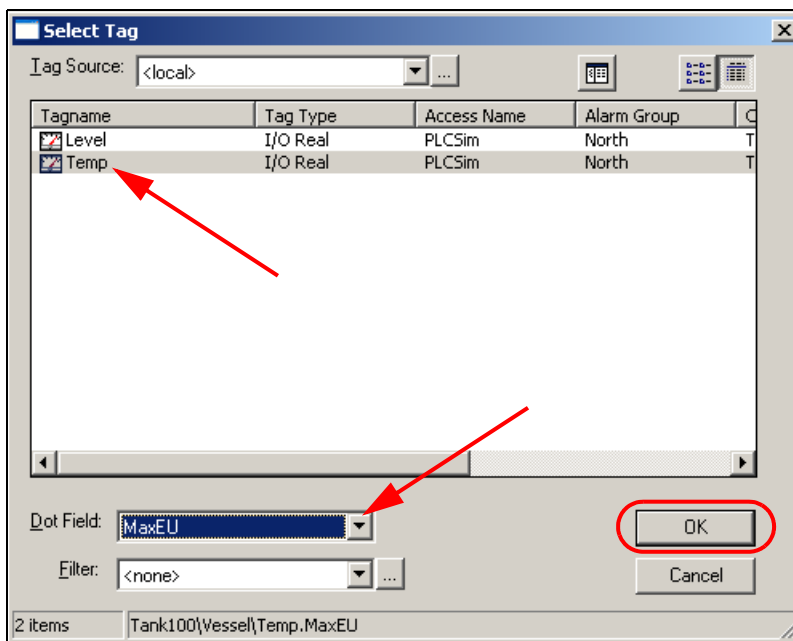


Note: InTouch remembers the settings from the last item you selected, so make sure to review the **Dot Field** selection.

27. Select the **Max** custom property, highlight **Default Value**, and then click **Browse Galaxy**.

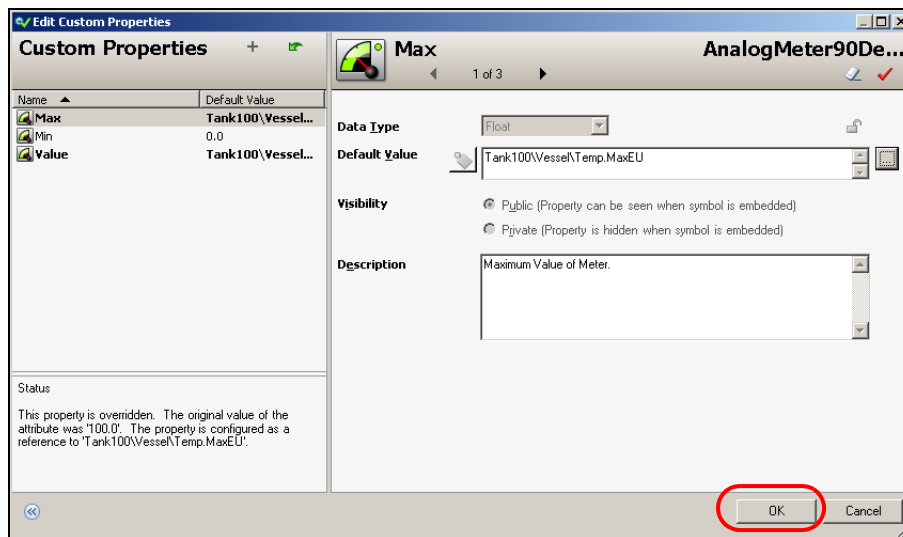


28. Double-click **Tank100\Vessel**, click **Temp**, and then from the **Dot Field** drop-down select **MaxEU**.



29. Click **OK**.

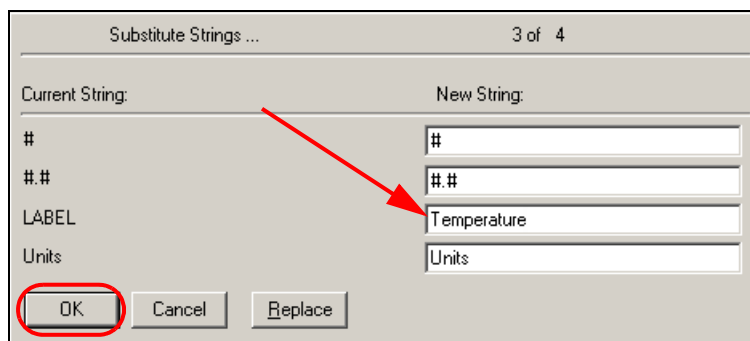
30. Click **OK** to close the **Edit Custom Properties** dialog box.



Once again, now that the custom properties have been configured, you should also change the default label to something more specific.

31. Select the **AnalogMeter90Degree** symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.

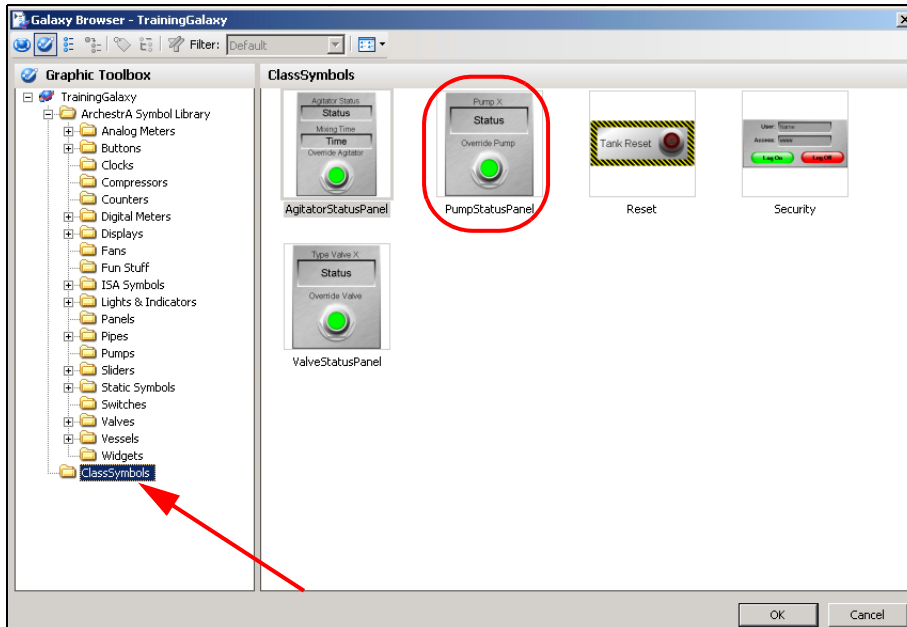
32. Change **LABEL** to **Temperature**, and then click **OK**.



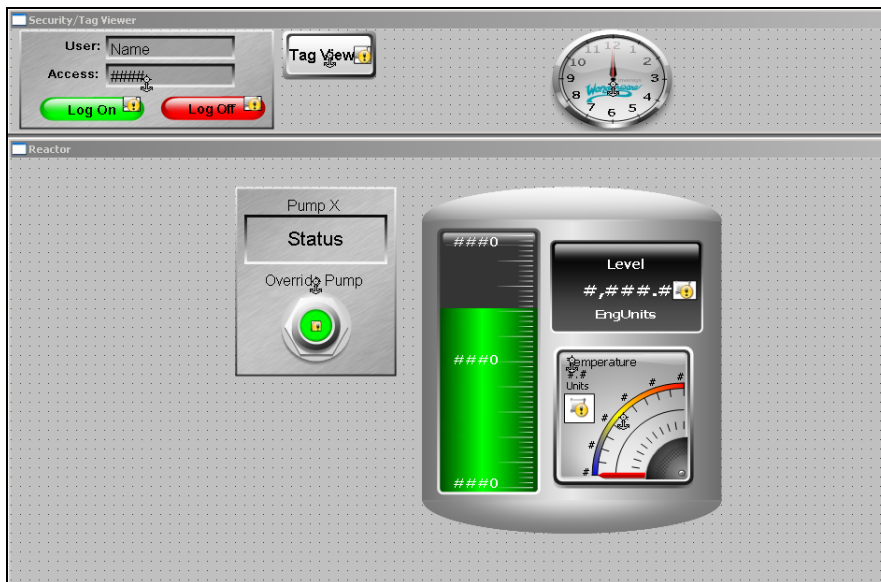
The Pump 1 Panel

Next you will add and configure one of the custom symbols you previously imported.

33. Click **Embed Archestra Graphic**.
34. Click **ClassSymbols**, and then double-click **PumpStatusPanel**.



35. Place the **PumpStatusPanel** symbol next to the tank on your display.



36. Double-click the **PumpStatusPanel** symbol and configure it as follows:

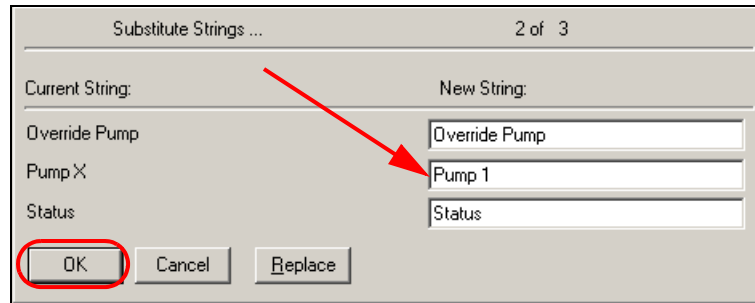
Custom Property	Default Value
OverrideControl.Value	Tank100\Pump1\Control
Value	Tank100\Pump1\Status

37. Click **OK** to close the **Edit Custom Properties** dialog box.

Now that the custom properties have been configured, you should also change the default label to something more specific.

38. Select the **PumpStatusPanel** symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.

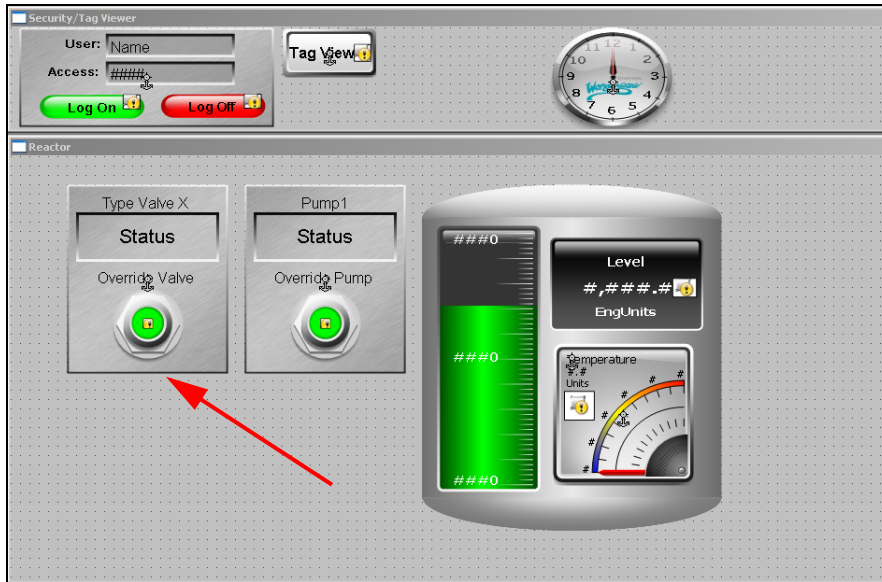
39. Change **Pump X** to **Pump 1**, and then click **OK**.



The Inlet 1 Panel

The Inlet 1 Panel is another of the custom symbols you previously imported.

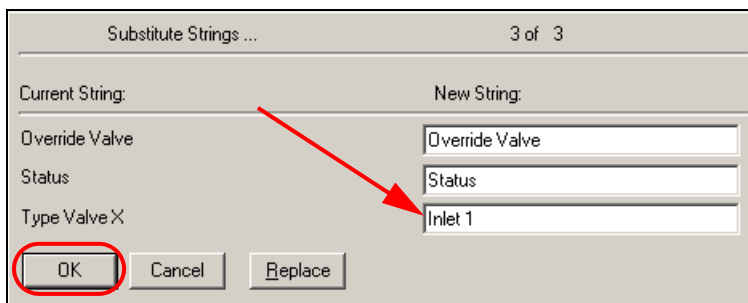
40. Click **Embed ArchestraA Graphic**.
41. Click **ClassSymbols**, and then double-click **ValveStatusPanel**.
42. Place the **ValveStatusPanel** symbol to the left of **PumpStatusPanel**.



43. Double-click the **ValveStatusPanel** symbol and configure it as follows:

Custom Property	Default Value
CLS	Tank100\Inlet1\CLS
OLS	Tank100\Inlet1\OLS
OverrideControl.Value	Tank100\Inlet1\Control

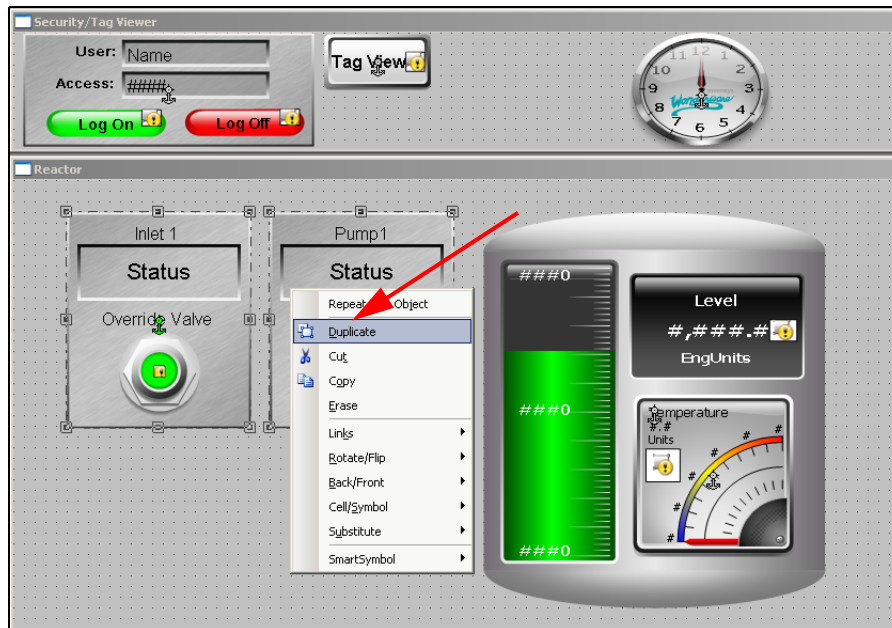
44. Click **OK** to close the **Edit Custom Properties** dialog box.
45. Select the **ValveStatusPanel** symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.
46. In the **Type Valve X** field, enter **Inlet 1**, and then click **OK**.



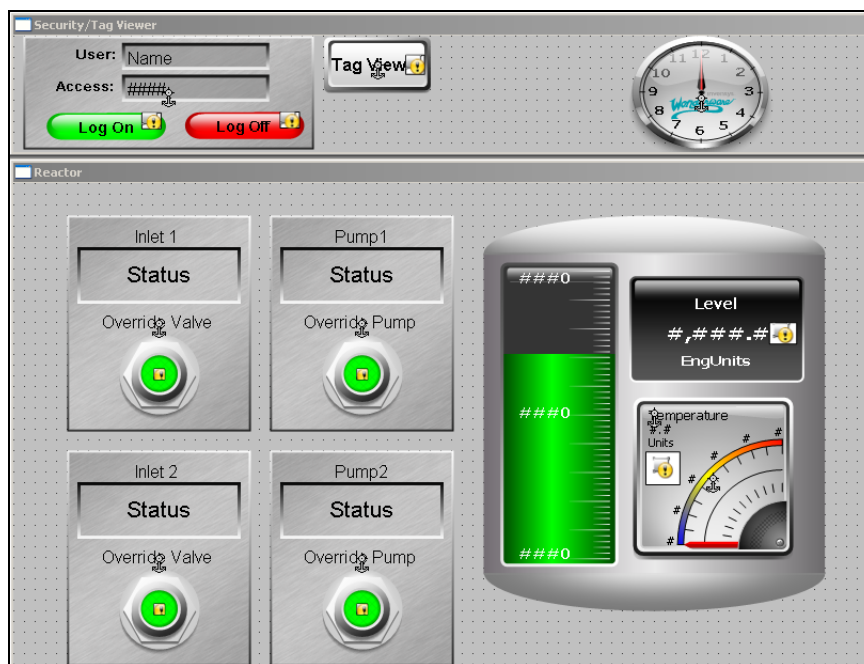
The Pump 2 and Inlet 2 Panels

Next, you will create two more symbols with similar configurations to Pump 1 and Inlet 1. Rather than building them from scratch, the quickest way to accomplish this is to duplicate the two you have already created and make minor changes.

47. Select both **Pump 1** and **Inlet 1**, and then right-click and select **Duplicate**.



48. Place them directly below the existing symbols.
49. Select both of the new objects, and then press **Ctrl+L** to access the **Substitute Strings** function. Change **Inlet 1** to **Inlet 2**, and **Pump 1** to **Pump 2**.



50. Select both **Inlet 2** and **Pump 2**, and then press **Ctrl+E** to access the **Substitute Tags** function.

Change the tags as follows:

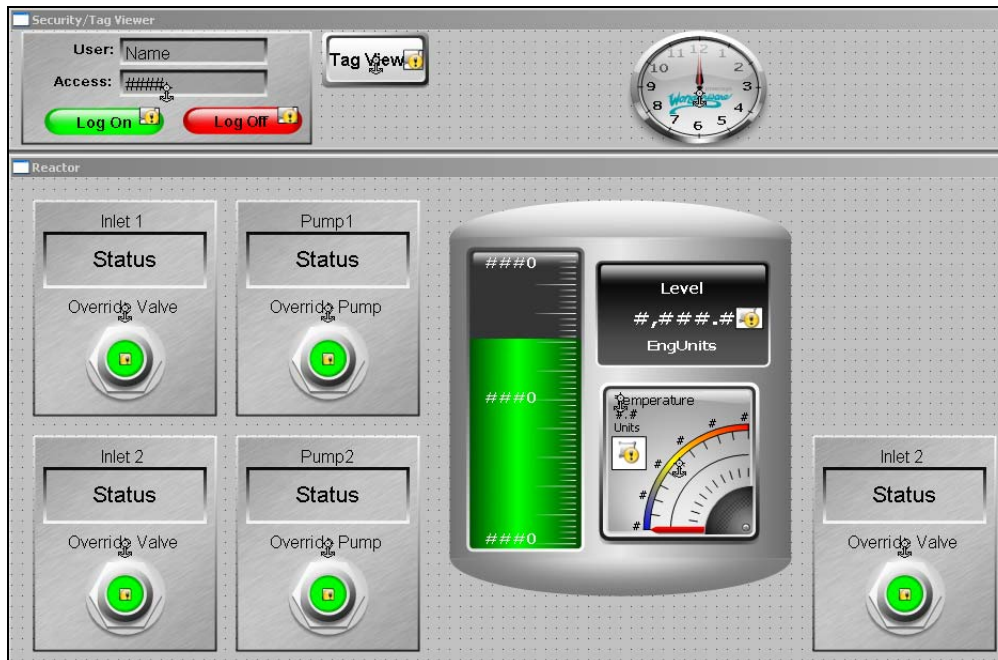
Original Tag	Change To
Tank100\Inlet1\CLS	Tank100\Inlet2\CLS
Tank100\Inlet1\Control	Tank100\Inlet2\Control
Tank100\Inlet1\OLS	Tank100\Inlet2\OLS
Tank100\Pump1\Control	Tank100\Pump2\Control
Tank100\Pump1>Status	Tank100\Pump2\Stauts

51. Click **OK**.

The Outlet Panel

You will now place an additional symbol labeled Outlet. This is easily done by duplicating the Inlet 2 object and making minor changes.

52. Select **Inlet 2**, duplicate it, and place it to the right of your tank.

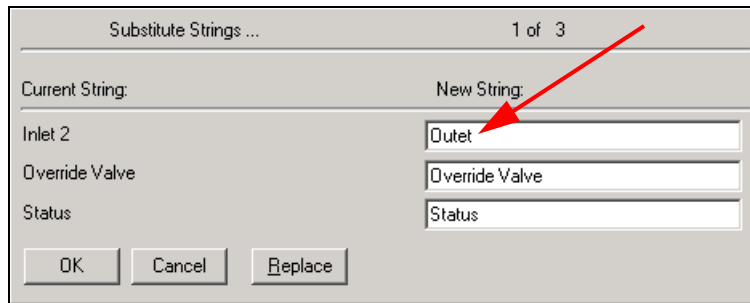


53. Double-click on the object and configure it as follows:

Custom Property	Original	Change To
CLS	Tank100\Inlet1\CLS	Tank100\Outlet\CLS
OLS	Tank100\Inlet1\OLS	Tank100\Outlet\OLS
OverrideControl.Value	Tank100\Inlet1\Control	Tank100\Outlet\Control

54. Click **OK**.

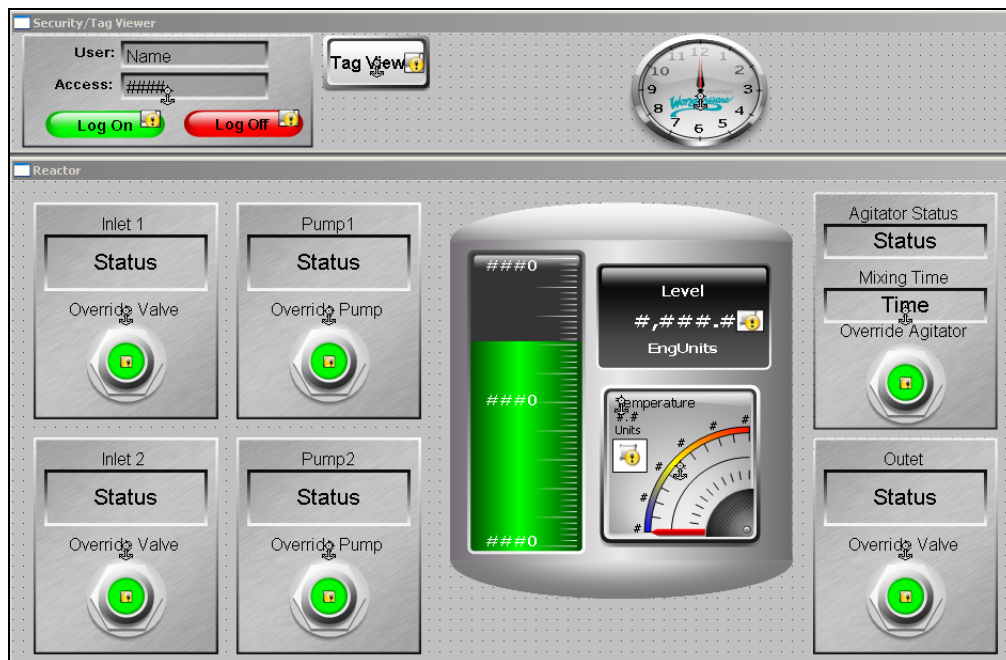
55. Select the object, and then press **Ctrl+L** to access the **Substitute Strings** function.
56. Change **Inlet 2** to **Outlet**, and then click **OK**.



The Agitator Status Panel

Now add and configure **AgitatorStatusPanel**.

57. Click **Embed ArchestrA Graphic**.
58. Click **ClassSymbols**, and then double-click **AgitatorStatusPanel**.
59. Place the new object to the right of your tank above the **Outlet** object.



60. Double-click **AgitatorStatusPanel** and configure it as follows:

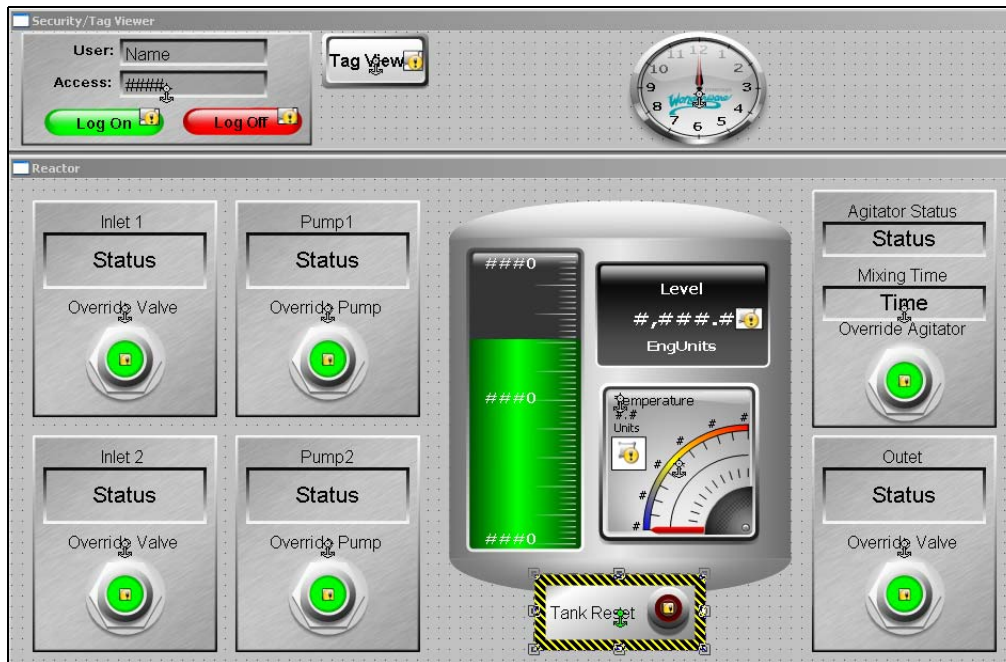
Custom Property	Tag
MixingTime	Tank100\Agitator\MixingTime
OverrideControl.Value	Tank100\Agitator\Control
Value	Tank100\Agitator\Status

61. Click **OK**.

The Tank Reset Button

The last item to add to your display is the Tank Reset custom symbol.

62. Click **Embed ArchestraA Graphic**.
63. Click **ClassSymbols**, and then double-click **Reset**.
64. Place the new object at the bottom of the tank.



65. Double-click **Reset** to configure **ResetButton.Value** as follows:

Tank100\Reset.

66. Click **OK**.

Test What You Have Built

67. Click **Runtime**.
68. Verify that the level and temperature values are changing.
69. Click the override buttons to see if they allow you to start and stop each item.

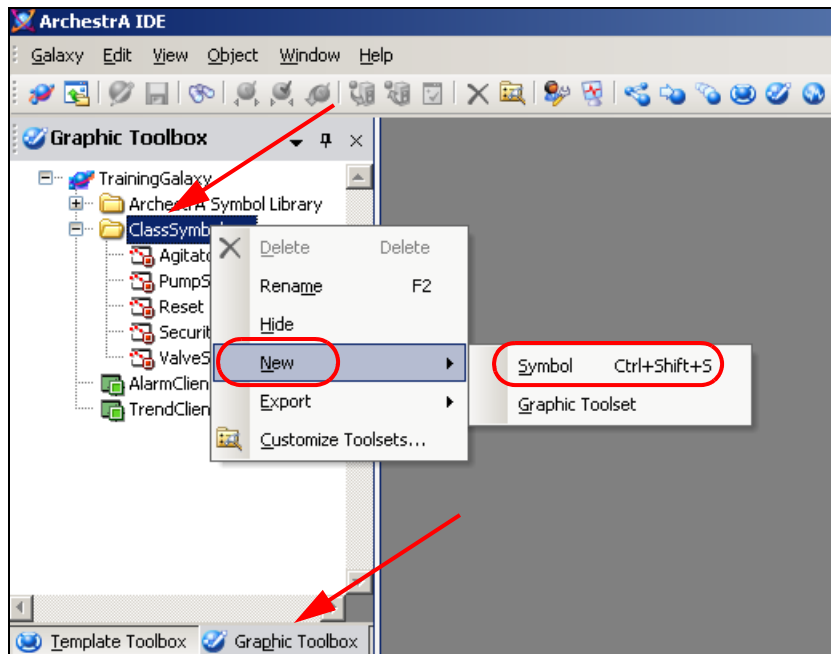
If you are logged in as an **Administrator** or **Manager** you should have full override capabilities. If you log off, the override capabilities should no longer be available.

Use Indirect Tags and a Single Display

You have now built a functioning interface for Tank 100. Rather than repeating all those steps for each additional Tank, you can use indirect tags with a few modifications. This allows the application to use one display window for each of the additional tanks.

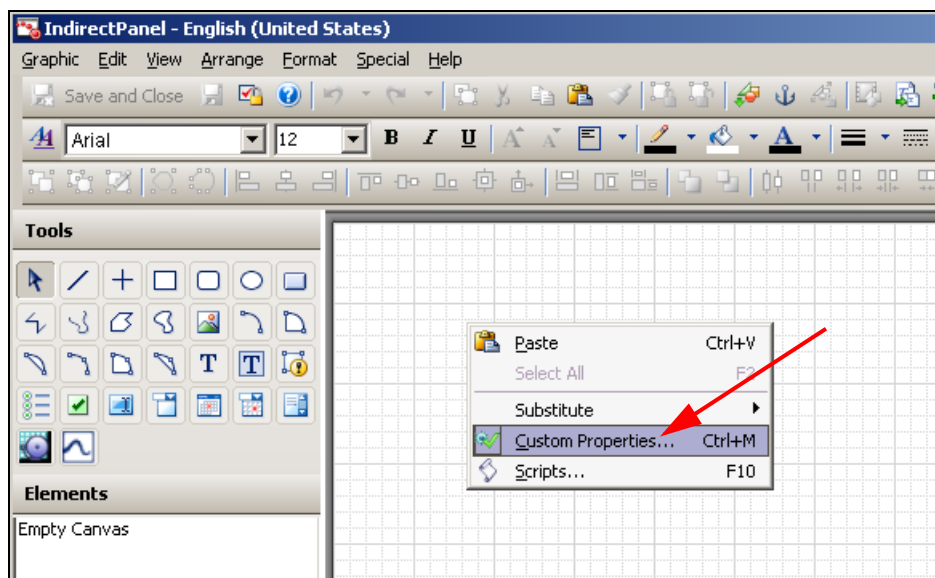
70. Click **Development!** to switch back to WindowMaker.

71. In Archestra IDE, in **Graphic Toolbox**, right-click **ClassSymbols** and select **New | Symbol**.

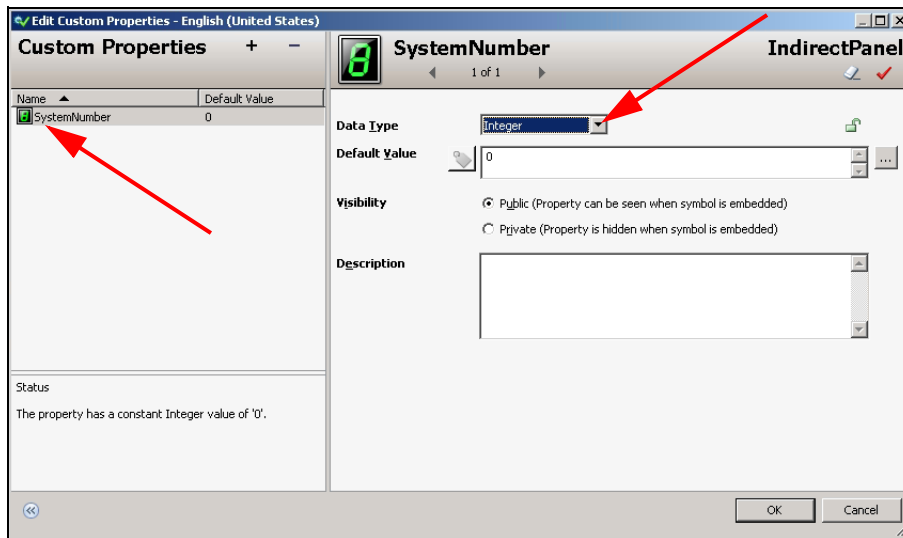


72. Name the new symbol **IndirectPanel**, and then double-click to open it.

73. On the canvas, right-click and select **Custom Properties**.

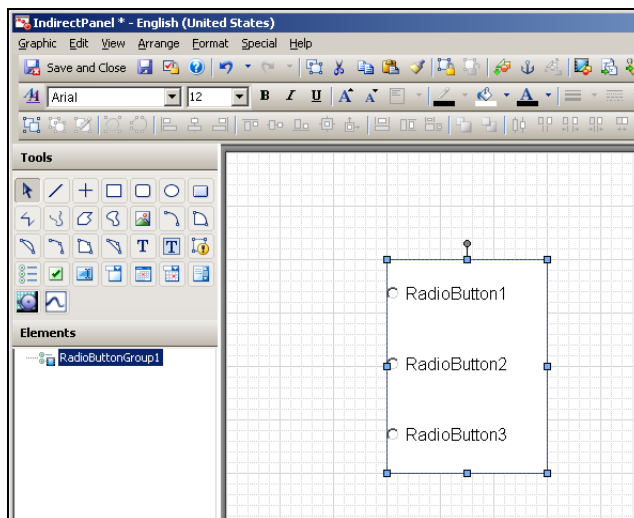


74. Click **+** to add a new custom property, name it **SystemNumber**, in the **Data Type** field select **Integer**, and then click **OK**.



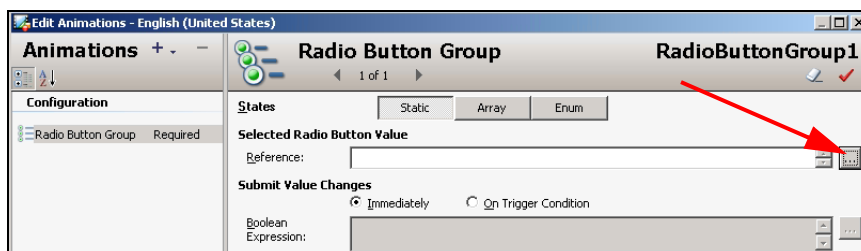
Add a Radio Button Group to the symbol to allow you to switch between systems.

75. Click the **RadioButtonGroup1** element and place it on the canvas.

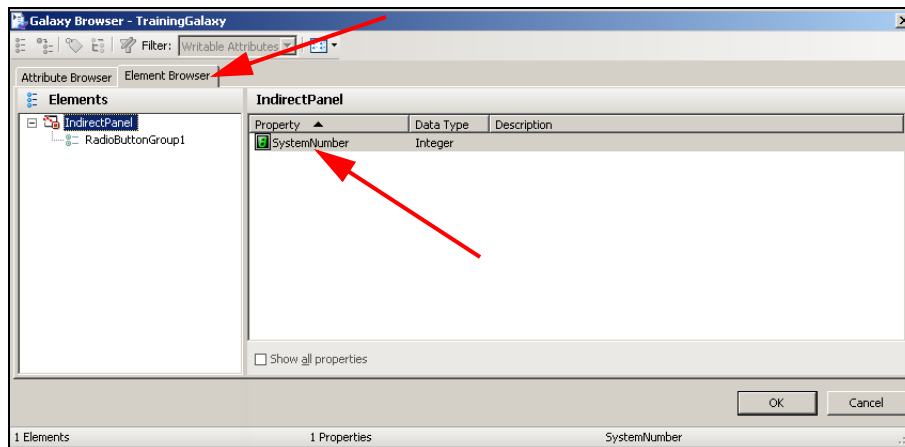


76. Double-click the **RadioButtonGroup1** element to configure it.

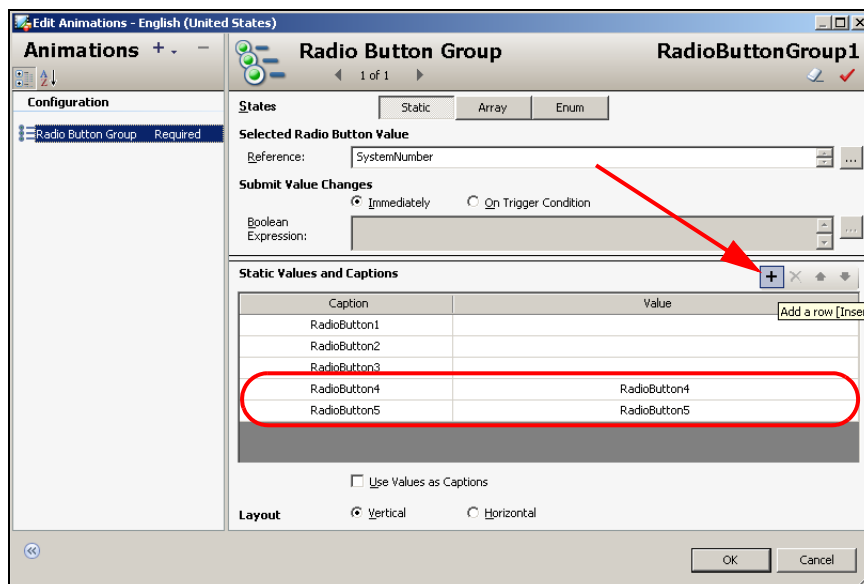
77. Next to the **Reference** field, click the **Browse Galaxy** button.



78. In the **Element Browser** tab, click the **IndirectPanel** element, select **SystemNumber**, and then click **OK**.



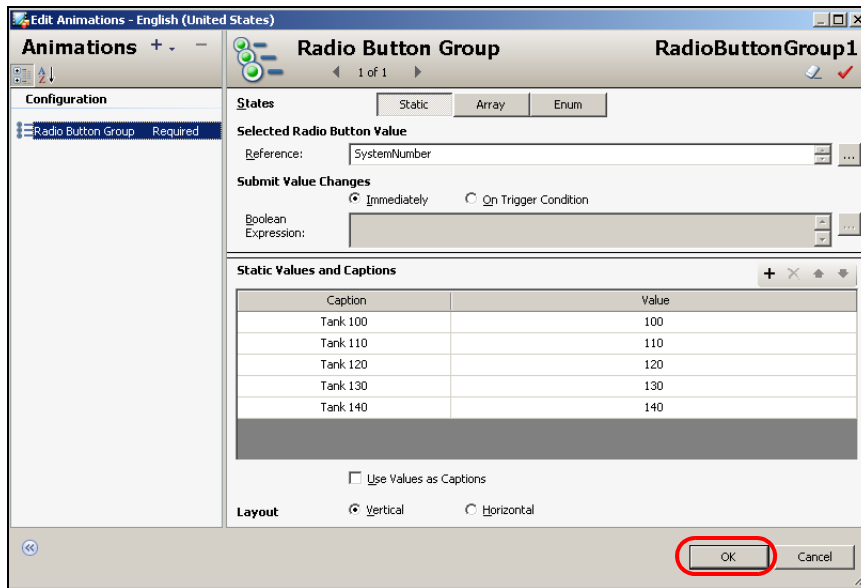
79. Click the **+** button twice under **Static Values and Captions** to add two more rows.



80. Select each radio button and configure as follows:

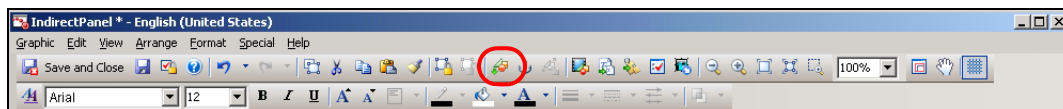
Original	New Caption	Value
RadioButton1	Tank 100	100
RadioButton2	Tank 110	110
RadioButton3	Tank 120	120
RadioButton4	Tank 130	130
RadioButton5	Tank 140	140

81. Click **OK**.

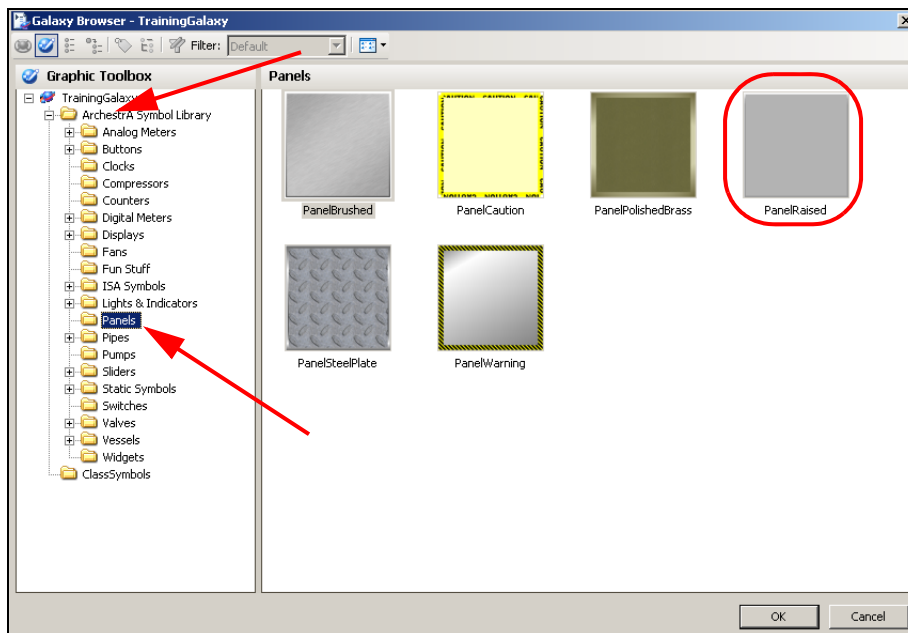


Note: If you need to add additional tanks later, this is where you would add and configure them.

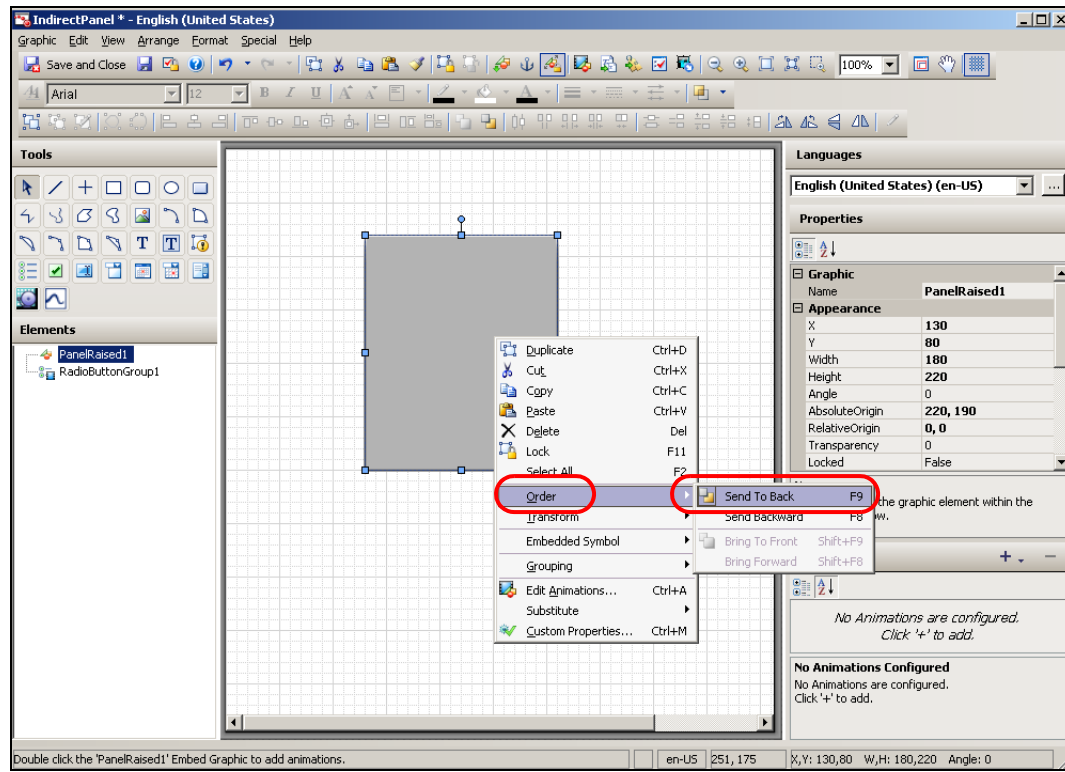
82. Click **Embed Archestra Graphic**.



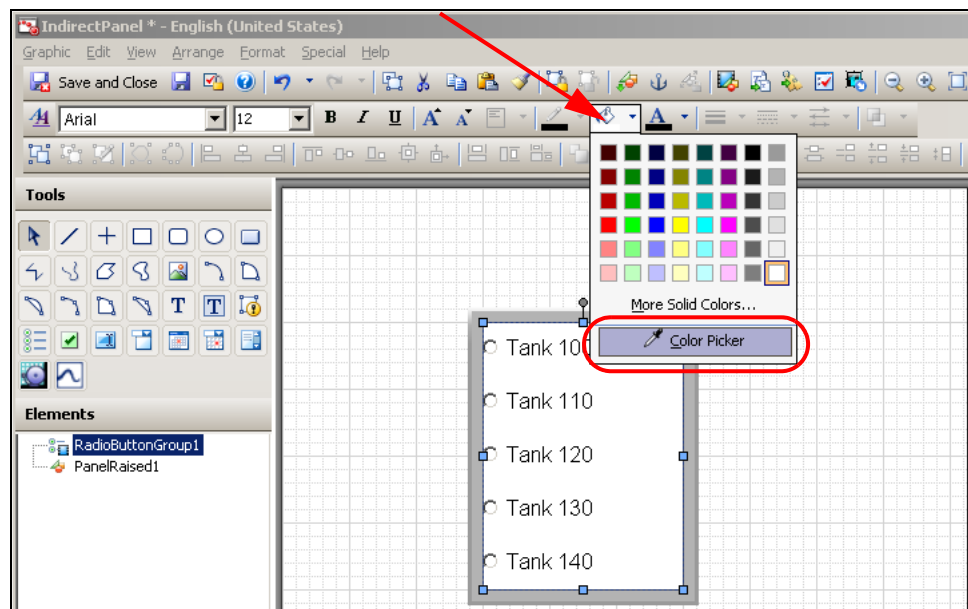
83. Expand **Archestra Symbol Library**, click **Panels**, and then double-click **PanelRaised**.



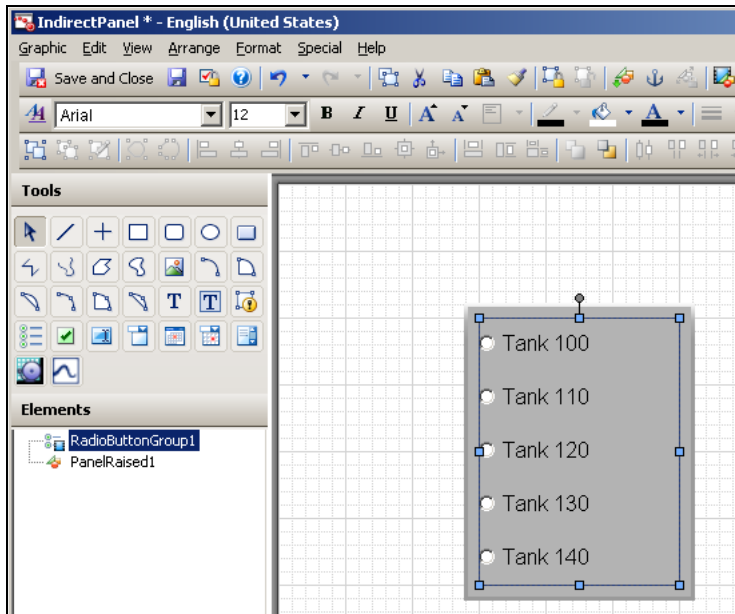
84. Place the symbol over the radio buttons and size it so that it covers the entire group of buttons.
85. Right-click **PanelRaised1** and select **Order | Send to Back**.



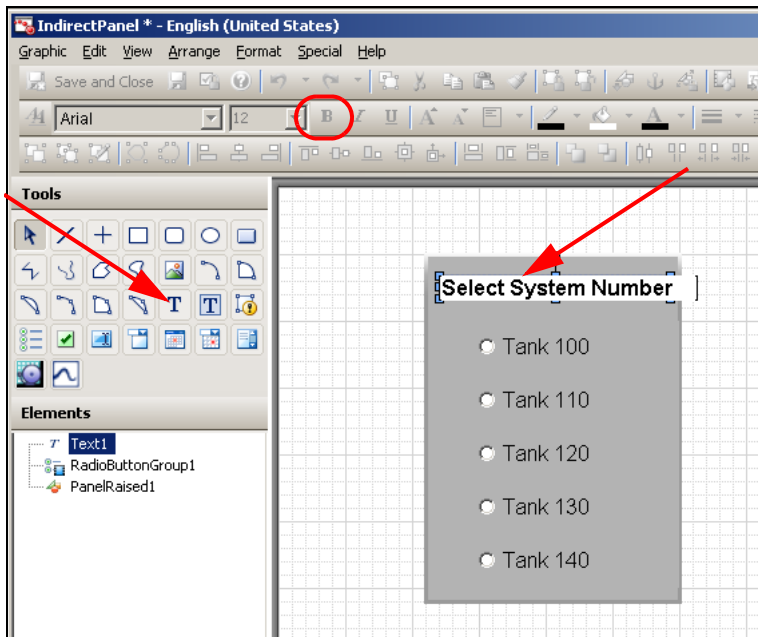
86. Select **RadioButtonGroup1**, and then click **Fill Color | Color Picker**.



87. Using **Color Picker**, click **PanelRaised1** to sample and set the new color for **RadioButtonGroup1** so that it blends in.

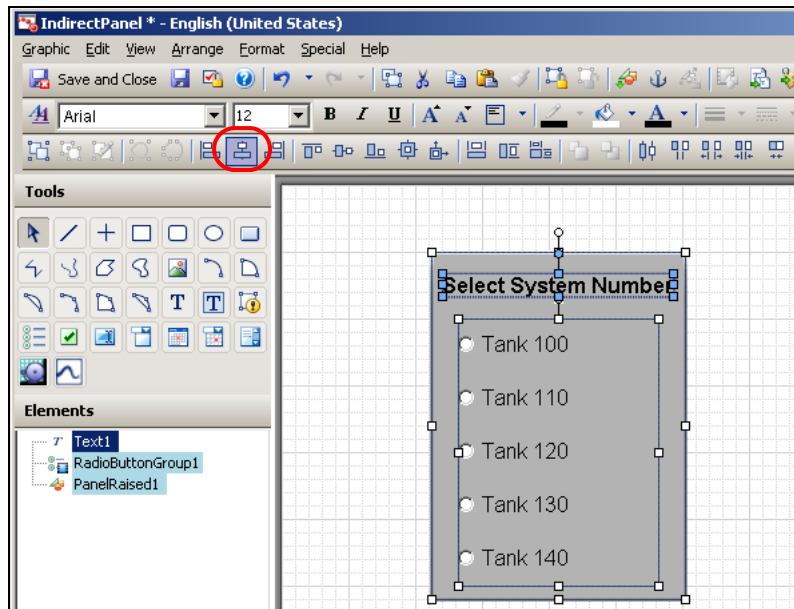


88. Click the **Text** tool, place the text above **RadioButtonGroup1**, enter **Select System Number**, press **Enter**, and then use the toolbar to make it bold.



Note: You may need to increase the size of the **PanelRaised1** object to fit the text element properly.

89. Select all the elements and click the **Align Center** function to organize them.



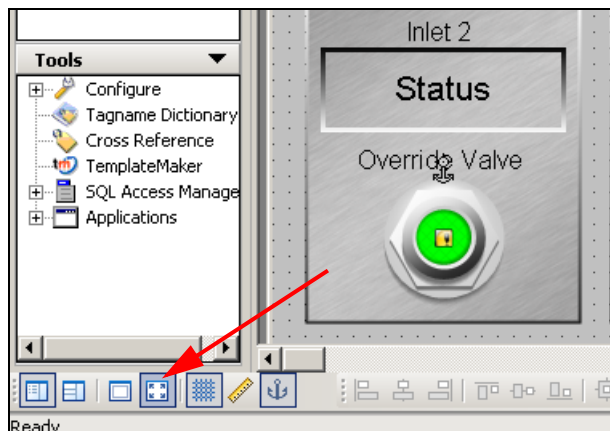
90. Click **Save and Close**, and then check in the symbol when prompted.

Now that you have created a symbol to switch between tanks, you will place it on your display.

91. In WindowMaker, click **Embed ArchestraA Graphic**.

92. Click **ClassSymbols**, and then double-click **IndirectPanel**.

93. Place the symbol anywhere on the display, and then click **Full Screen** located at the bottom-left of the window.

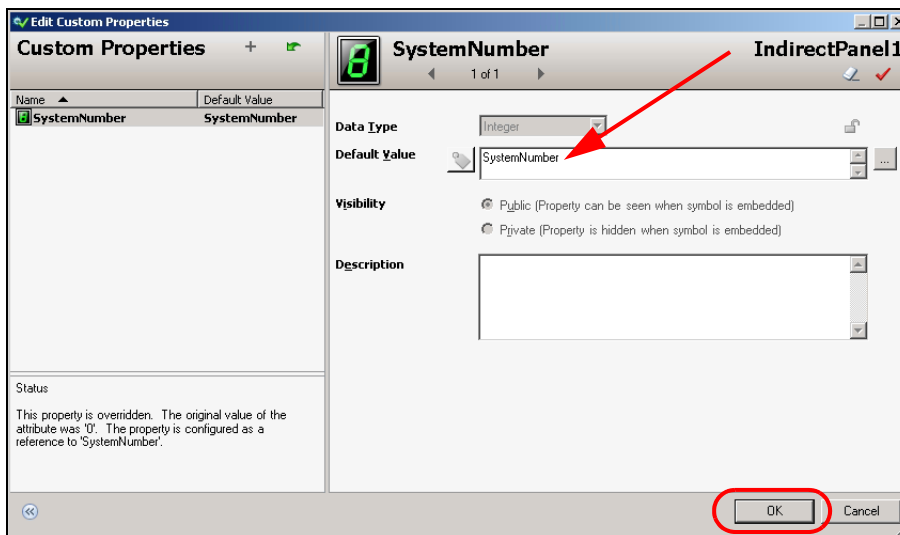


94. Drag the **IndirectPanel** symbol to the right of **Agitator Status**.



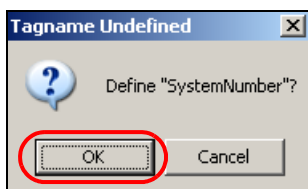
95. Double-click the **IndirectPanel** symbol to configure it.

96. In the **Default Value** field, enter **SystemNumber**, and then click **OK**.



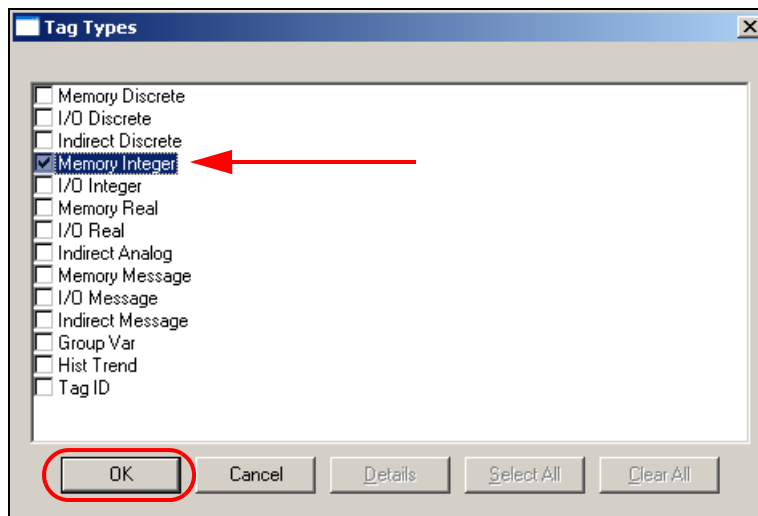
Note: Because this tag does not exist yet, you will be prompted to define it.

97. On the **Tagname Undefined** prompt, click **OK**.

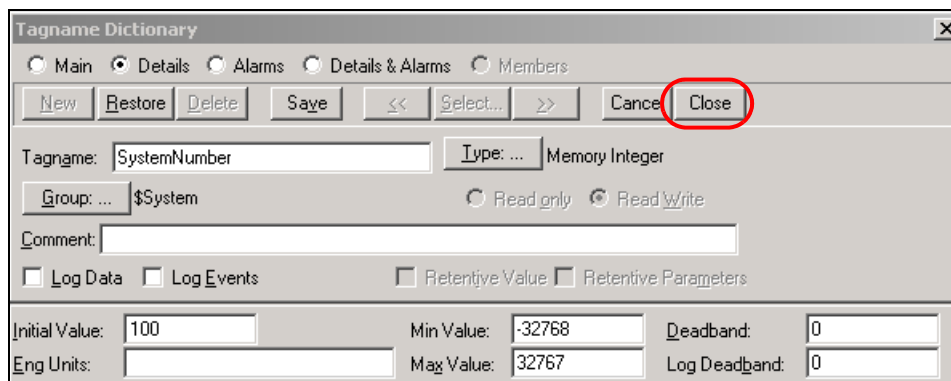


98. In the **Tagname Dictionary** dialog box, click **Type**.

99. In the **Tag Types** dialog box, select **Memory Integer**, and then click **OK**.



100. Set **Initial Value** to **100**, and then click **Close**.



101. Exit out of the full screen mode.



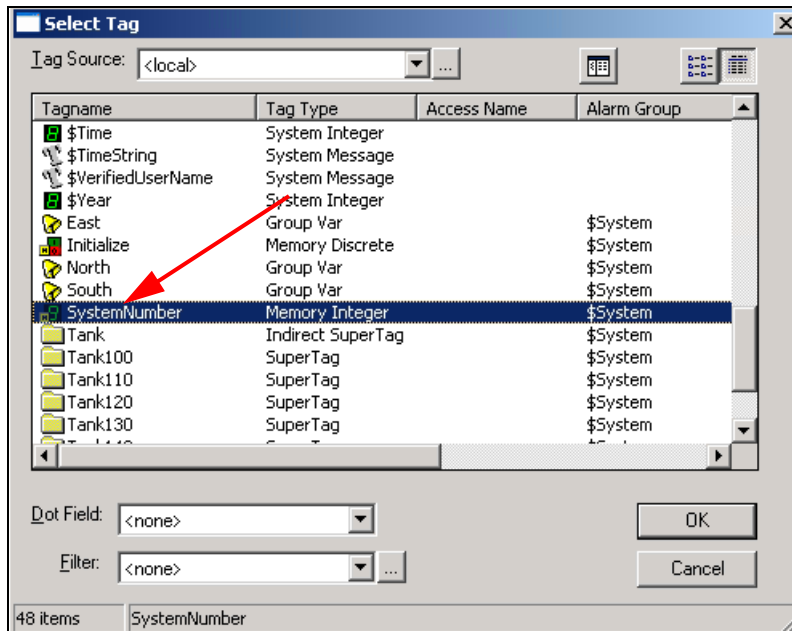
Implement Scripting for Indirect Addressing

In order for the IndirectPanel to switch between tanks, you will need to write a script that allows the indirect assignment of tag values based on the selected system number.

102. On the **Special** menu, click **Scripts | Data Change Scripts**.

103. Double-click in **Tagname[.field]**.

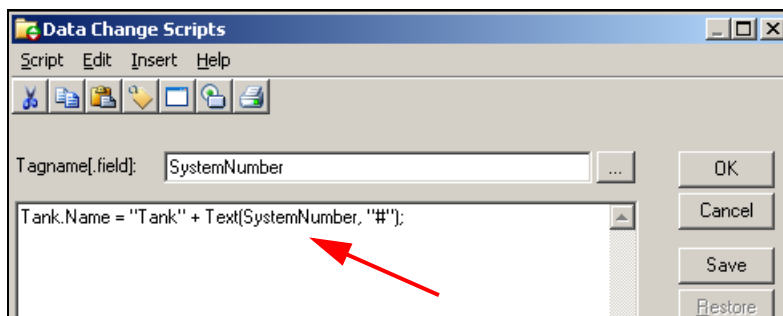
104. Select **SystemNumber**, and then click **OK**.



Because the display was built using SuperTags, you can assign every member of the SuperTag with a single line of script. In this case, the indirect SuperTag is Tank, and the ability to change the indirect assignment is done using the .Name parameter.

105. In the script area, enter:

```
Tank.Name = "Tank" + Text(SystemNumber, "#");
```



Note: Rather than manually inputting the **Text(SystemNumber, "#")** portion of the script, it can also be found by clicking **All** under **Functions** and inserting the **Text** item, and then substitute in the references.

106. Click **OK**.

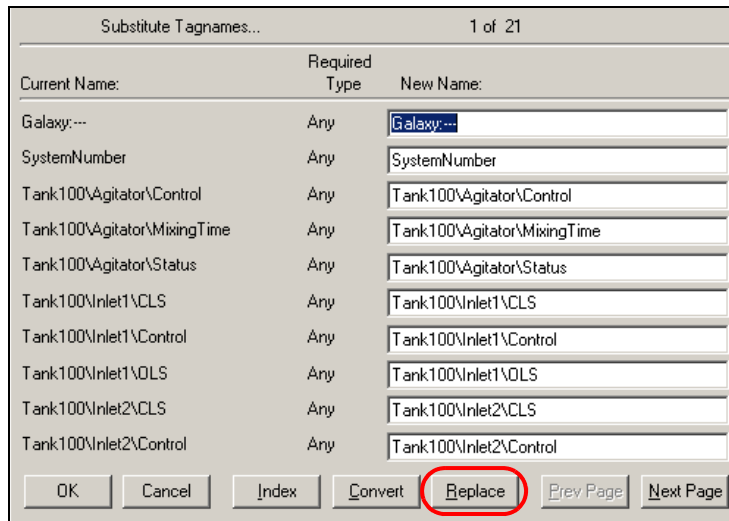
Modify an Application to Reference Indirect Tags

Next, you will need to change the references on the display graphics from Tank100 to the indirect tag Tank. You can quickly do this using the Substitute Tag function.

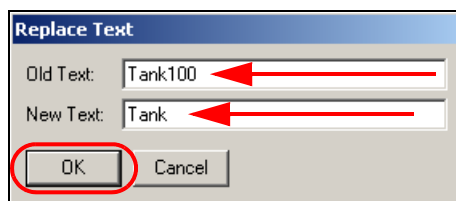
107. Click in the **Reactor** window, and then press **F2** to select all elements.

108. Press **Ctrl+E**.

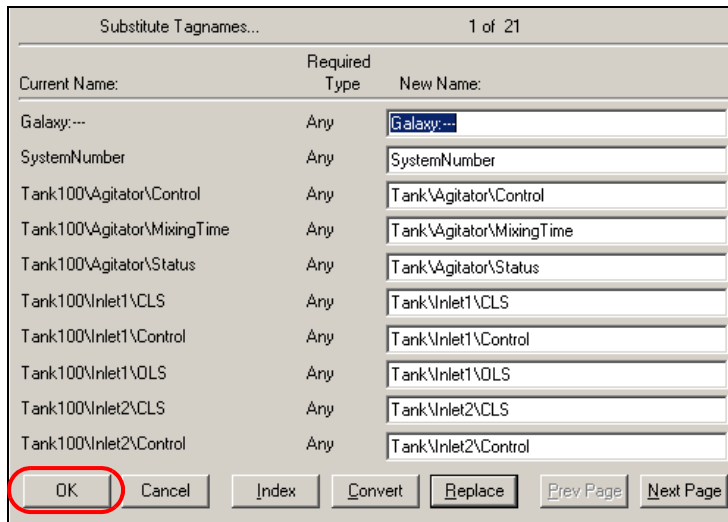
109. Click **Replace**.



110. In the **Old Text** field enter **Tank100**, in the **New Text** field enter **Tank**, and then click **OK** to close the **Replace Text** dialog box.



111. Click **OK** to close the **Substitute Tagnames** dialog box.



Test What You Have Built

112. Click **Runtime**.

113. Click through each of the tanks using the **Select System Number** panel, and verify that the values and level ranges change.

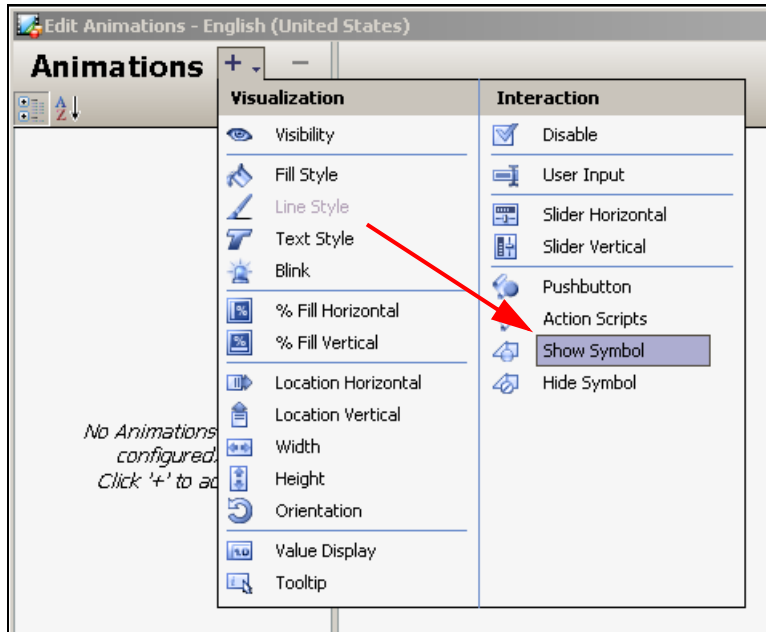
114. Click **Development!**.

Section 2 – The Show Symbol Animation Link

This section explains the use and configuration of the Show Symbol animation, which allows one symbol to reference and control another.

Show Symbol Animation

The Show Symbol Animation allows a symbol to be called from within another symbol rather than directly placing it within the InTouch application. This feature is available from the Edit Animations window within Archestra IDE.

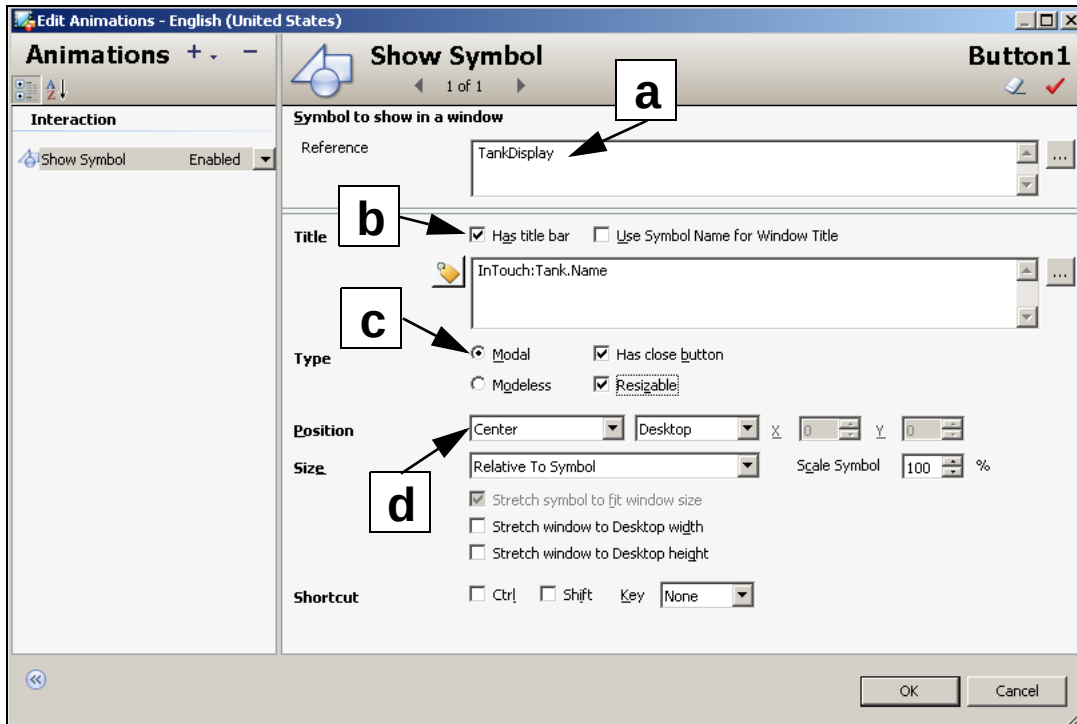


Configuring a Show Symbol Animation

Configure an element with a Show Symbol animation. When the element is clicked, a Show Symbol animation shows a specified symbol in a new window.

Configure:

- Which symbol appears in the new window
- If the window has a title bar, and if so, what the title bar will be
- If the window is modal or modeless, has a close button, and can be resized
- The initial position and size of the window



Note: Modal vs. Modeless refers to whether the symbol once called, will allow you to interact with anything in the background. A Modal setting demands focus on the called symbol until it is closed — Modeless does not.

Lab 4 – Show Symbol Animation Link

Introduction

In this lab, you will use the Show Symbol animation link to display an ArchestrA symbol that is not directly embedded within InTouch. The purpose of this lab is to illustrate the use of the Show Symbol animation link when using indirect tags.

Objectives

Upon completion of this lab, you will be able to:

- Use the Show Symbol animation
- Use indirect tags in an ArchestrA symbol
- Build and show simplified Tanks using the same symbol

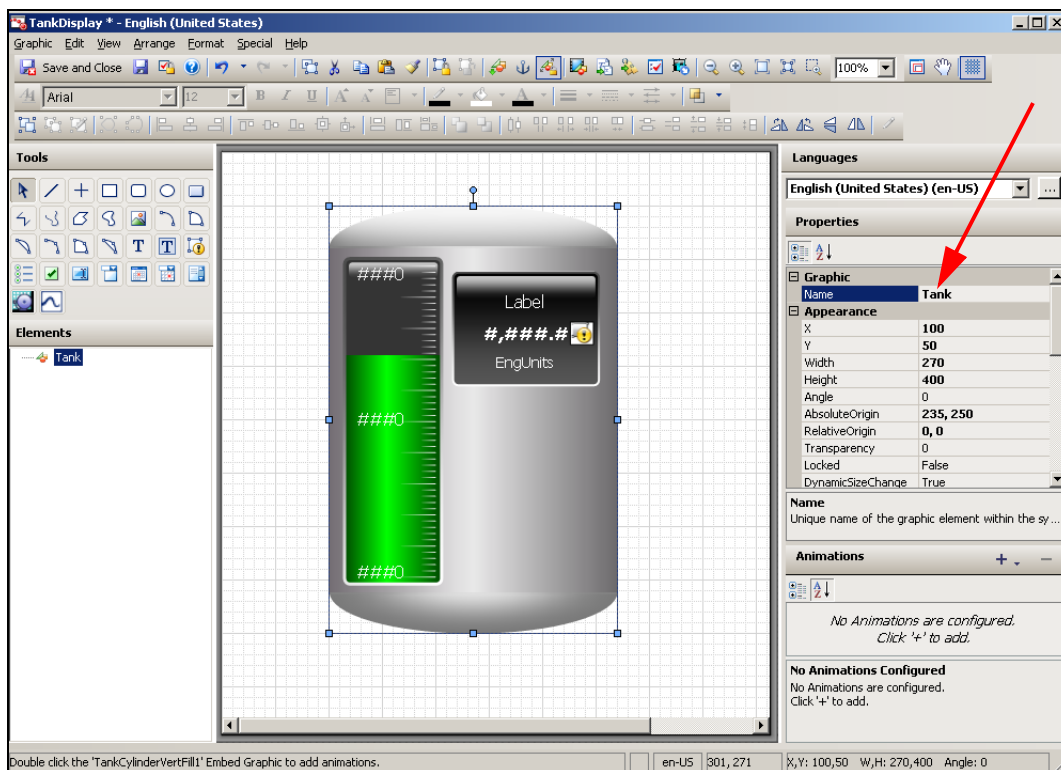
Create an Indirectly Accessed Symbol

For this lab, you will create an indirectly accessed symbol using ArchestrA IDE. This graphic will never be placed into an InTouch window, but will be called into InTouch during Runtime using a secondary symbol and indirect tag addressing.

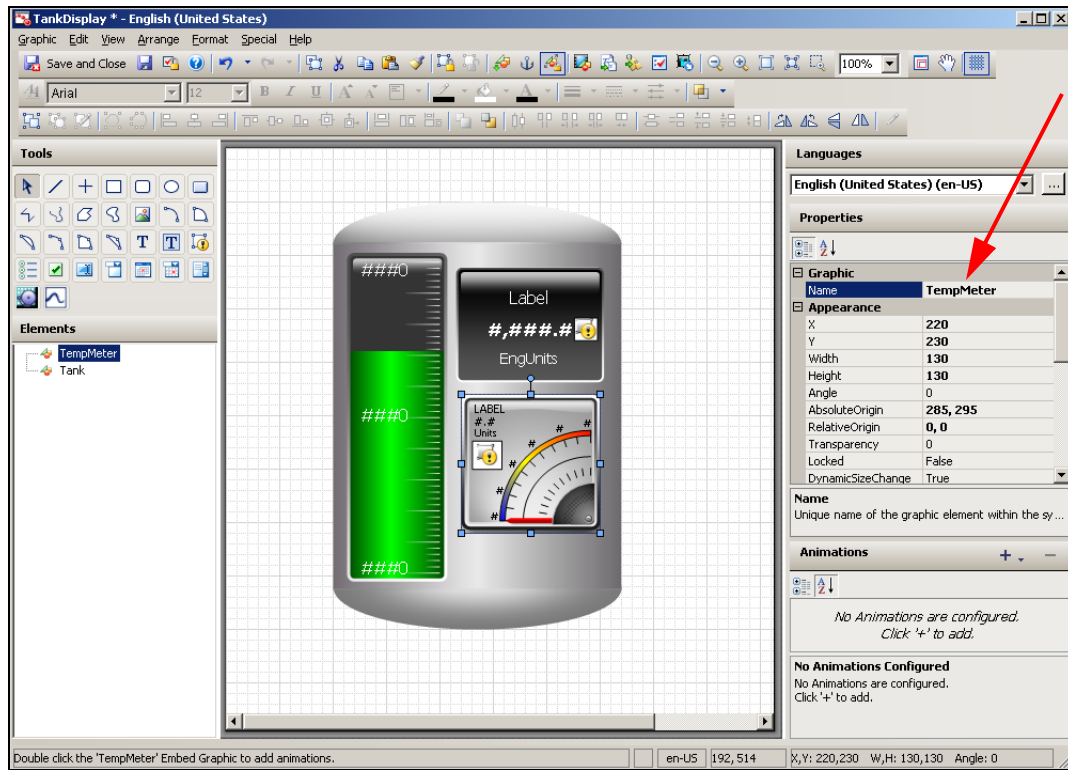
1. In ArchestrA IDE, in **Graphic Toolbox**, right-click **ClassSymbols** and select **New | Symbol**.
2. Name the symbol **TankDisplay**, and then double-click to edit it.

Within this symbol you will embed and configure two additional symbols.

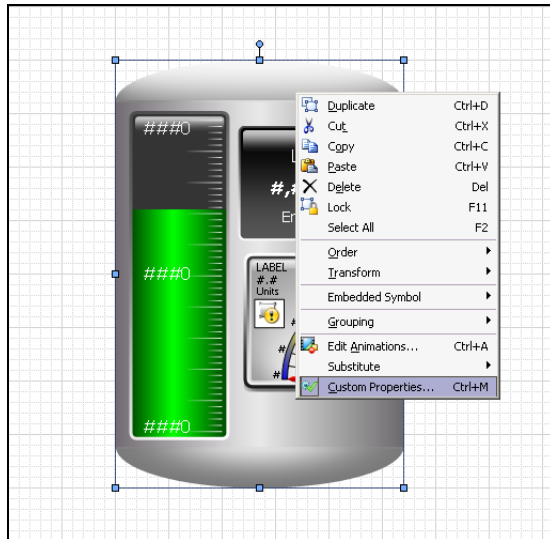
3. Click **Embed ArchestrA Graphic**.
4. Expand **ArchestrA Symbol Library**, click **Vessels**, and then double-click **TankCylinderVertFill**.
5. Place and enlarge the symbol, and then in the right pane rename it **Tank**.



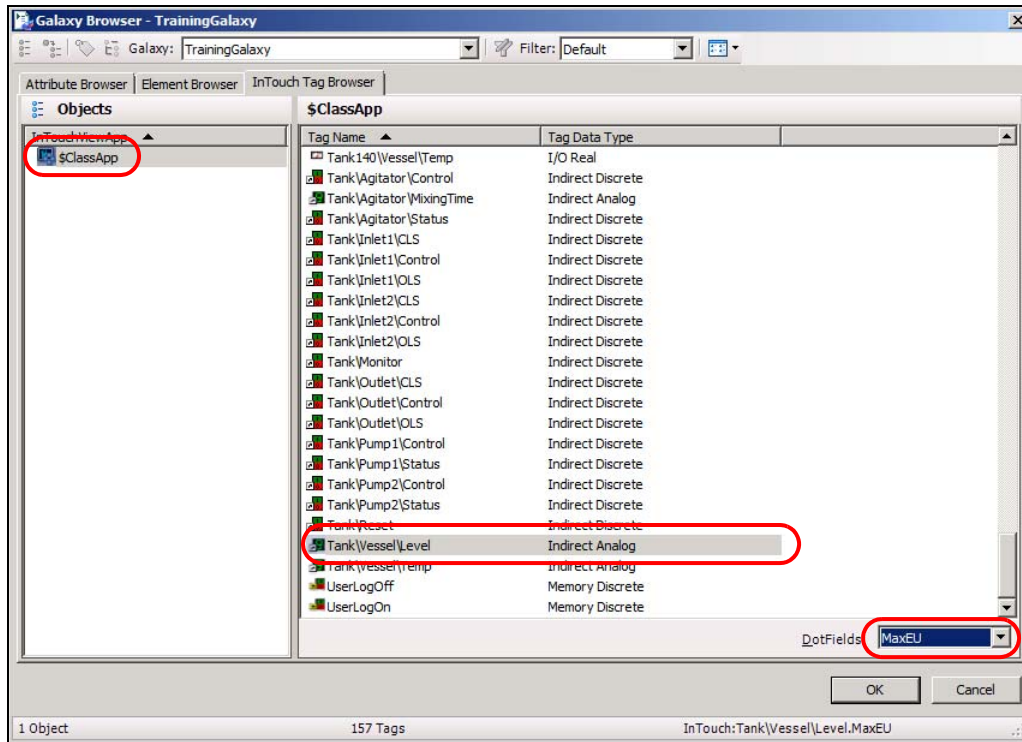
6. Click **Embed Archestra Graphic**.
7. Click **Archestra Symbol Library\Analog Meters**, and then double-click **AnalogMeter90Degree**.
8. Place the symbol on the tank, resize as needed, and then in the right pane rename it **TempMeter**.



9. Right-click **Tank** and select **Custom Properties**.

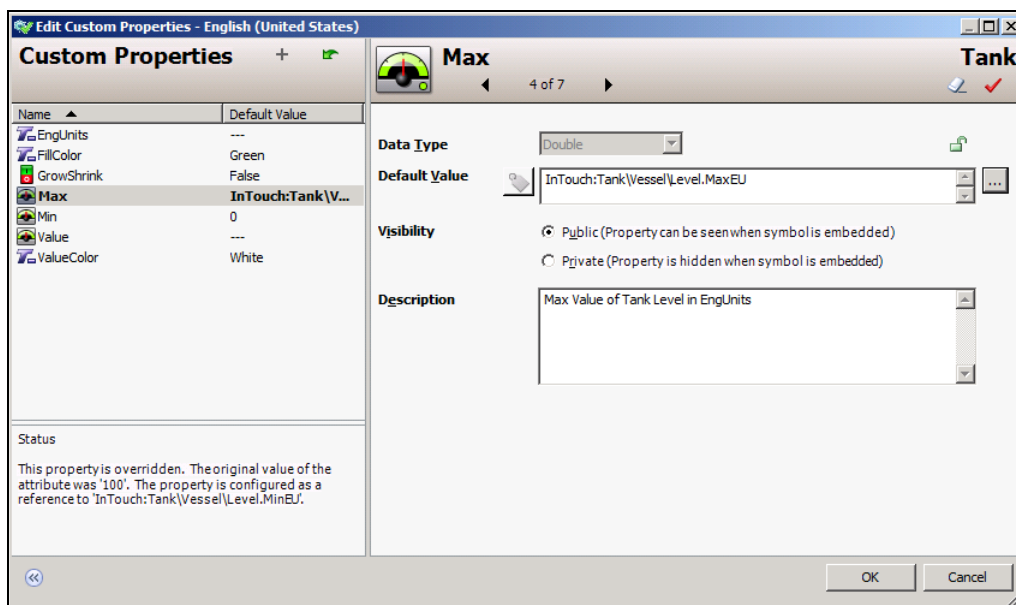


10. In the **Custom Properties** pane, click **Max**.
11. In the **Default Value** field, highlight **100**, and then click **Browse Galaxy**.
12. Click the **InTouch Tag Browser** tab.
13. In the **Objects** pane ensure **\$ClassApp** is selected, in the tag list click **Tank\Vessel\Level**, and from the **Dot Field** select **MaxEU**.

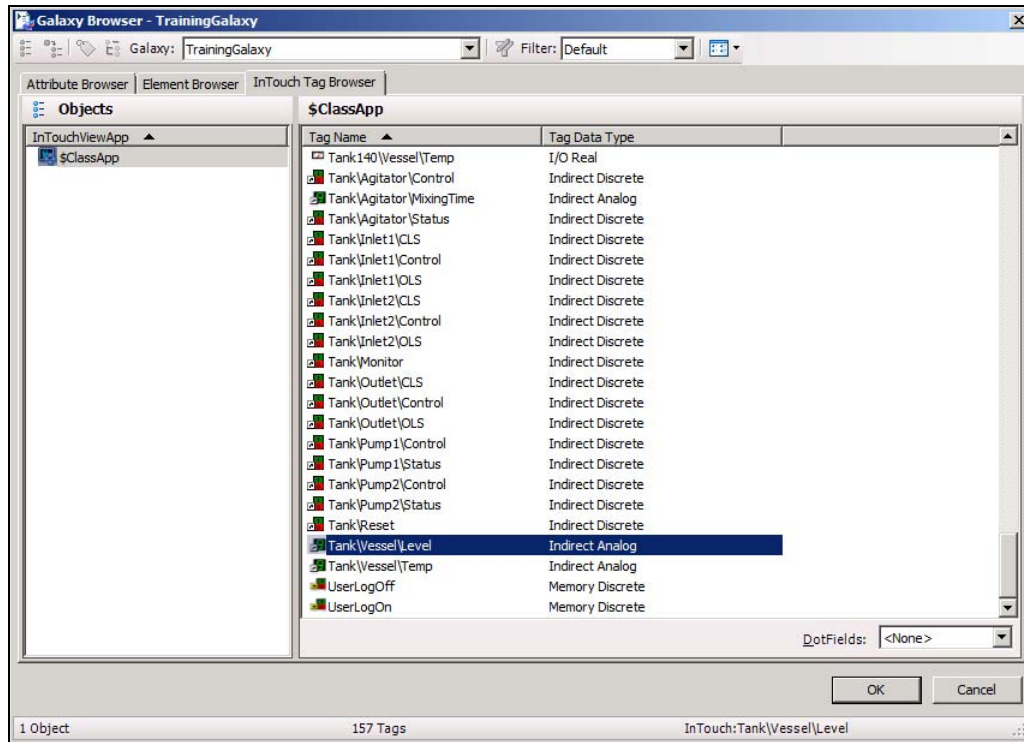


14. Click **OK** to close **Galaxy Browser**.

Notice the **Default Value** field shows **InTouch:Tank\Vessel\Level.MaxEU**.



15. In the **Custom Properties** pane, click **Value**.
16. In the **Default Value** field, highlight ---, and then click **Browse Galaxy**.
17. Click the **InTouch Tag Browser** tab.
18. In the **Objects** pane ensure **\$ClassApp** is selected, in the tag list click **Tank\Vessel\Level**, and leave the **Dot Field** as **<none>**.



19. Click **OK** to close the **Galaxy Browser**.
Notice the **Default Value** field shows **InTouch:Tank\Vessel\Level**.
20. Click **OK** to close the **Edit Custom Properties** dialog box.

As in the previous lab, you should change the text on the symbol to something more specific.

21. Select the **Tank** symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.

22. Change **Label** to **Level**, and then click **OK**.

Repeat the same process to configure the TempMeter.

23. Right-click **TempMeter** and select **Custom Properties**.

24. Configure as follows:

Custom Property	Default Value (enter manually)
Max	InTouch:Tank\Vessel\Temp.MaxEU
Value	InTouch:Tank\Vessel\Temp

25. Click **OK**.

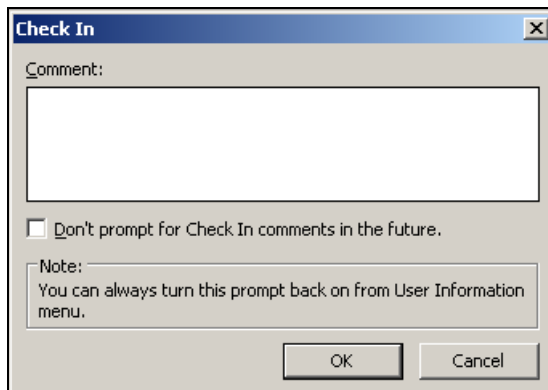
Once again, change the text on the symbol to something more specific.

26. Select the **TempMeter** symbol, and then press **Ctrl+L** to access the **Substitute Strings** dialog box.

27. Change **LABEL** to **Temperature**, and then click **OK**.

28. **Save and Close** the **TankDisplay** symbol.

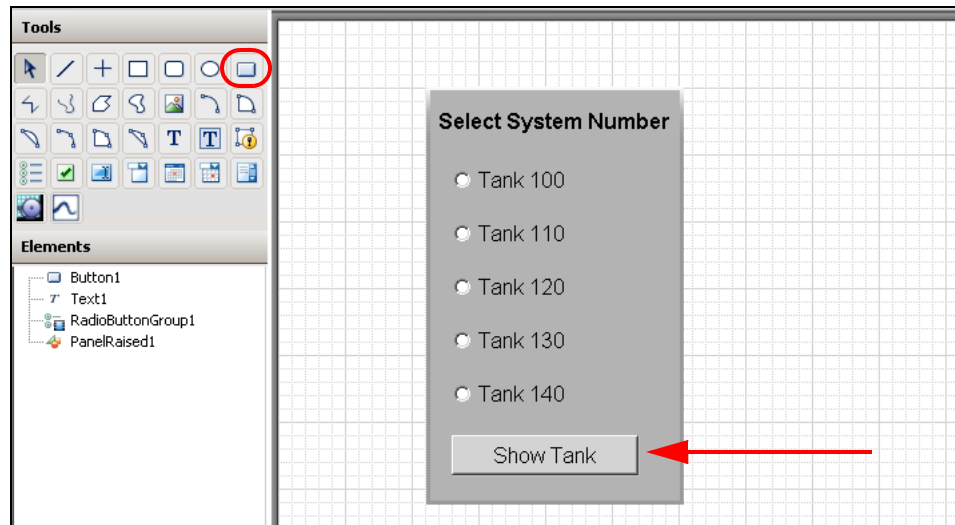
29. Click **OK** to check it in.



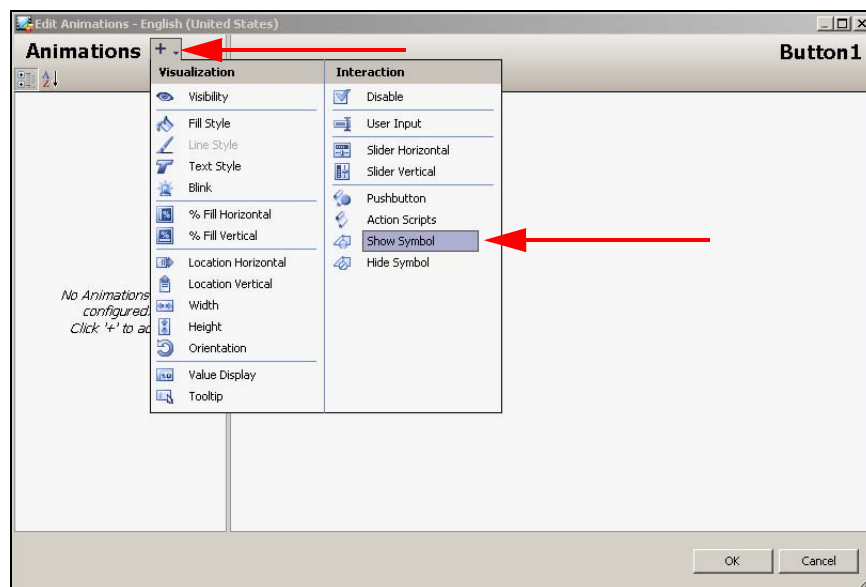
Apply Show Symbol Functionality

Next, you will create a secondary symbol to call the TankDisplay from within the InTouch application. This will be based off the IndirectPanel symbol you created in the previous lab.

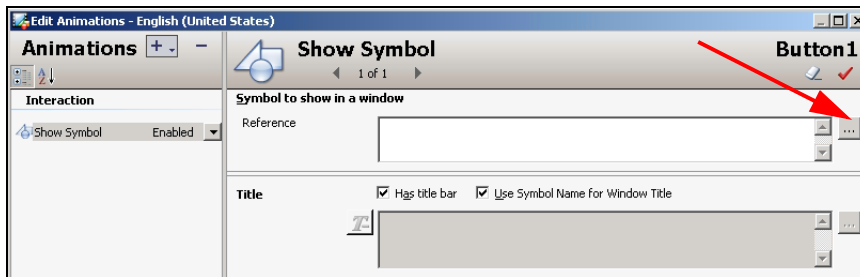
30. In Archestra IDE, in **ClassSymbols**, right-click **IndirectPanel** and select **Duplicate**.
31. Rename the new symbol **TankPanel**, and then double-click to open it for editing.
32. Lengthen the **PanelRaised1** element.
33. Add a button below the **Tank140** radio button, and then change the text to **Show Tank**.



34. Double-click the **Show Tank** button to configure it.
35. Click **+** and select **Show Symbol**.



36. Next to the **Reference** field, click the ellipsis button.



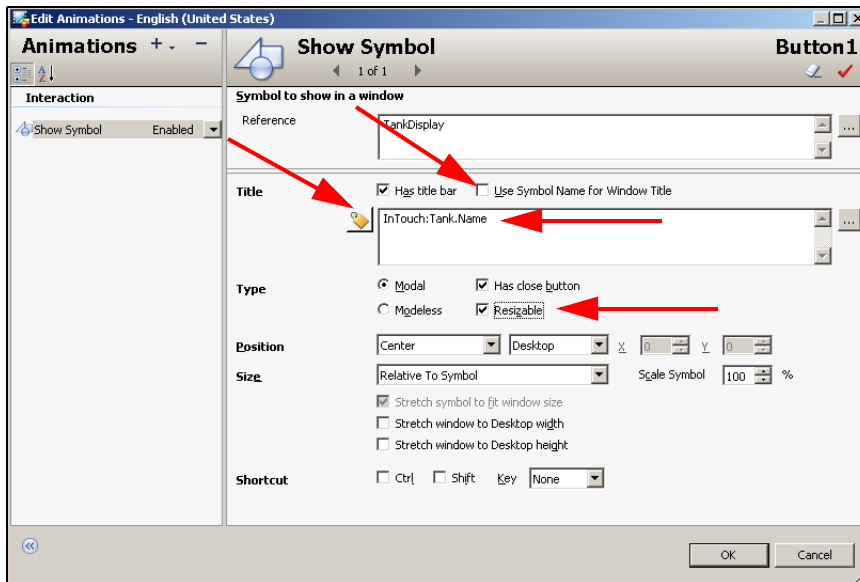
37. Click **ClassSymbols**, and then double-click **TankDisplay**.

38. Ensure **Has title bar** box is checked, and then uncheck **Use Symbol Name for Window Title**.

39. Click the **Text Reference** button  to change from static text to an expression .

The following InTouch tag cannot be selected from the Tag Browser and must be entered manually into the expression field.

40. In the expression field, enter **InTouch:Tank.Name**, check **Resizable**, and then click **OK**.

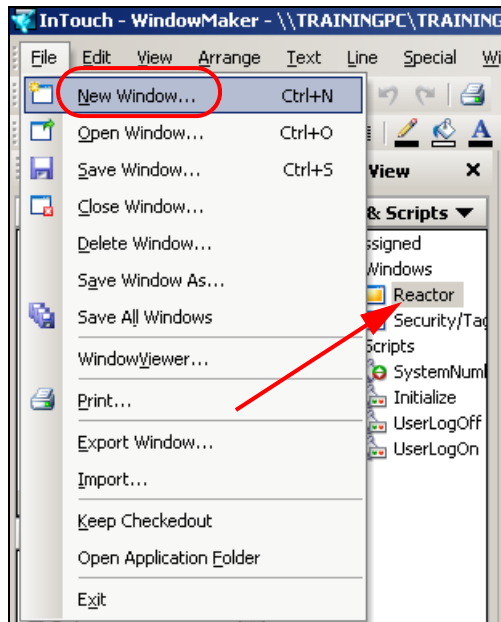


41. **Save and Close** the **TankPanel** symbol.

42. Click **OK** to check it in.

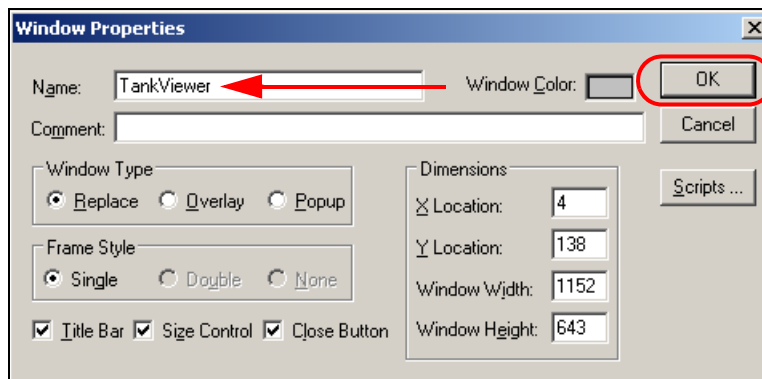
Create a new window to embed the TankPanel symbol.

43. In WindowMaker, click the **Reactor** window, and then on the **File** menu, click **New Window**.



Note: Clicking the **Reactor** window first, ensures that the new window has the same size and location properties of the **Reactor** window.

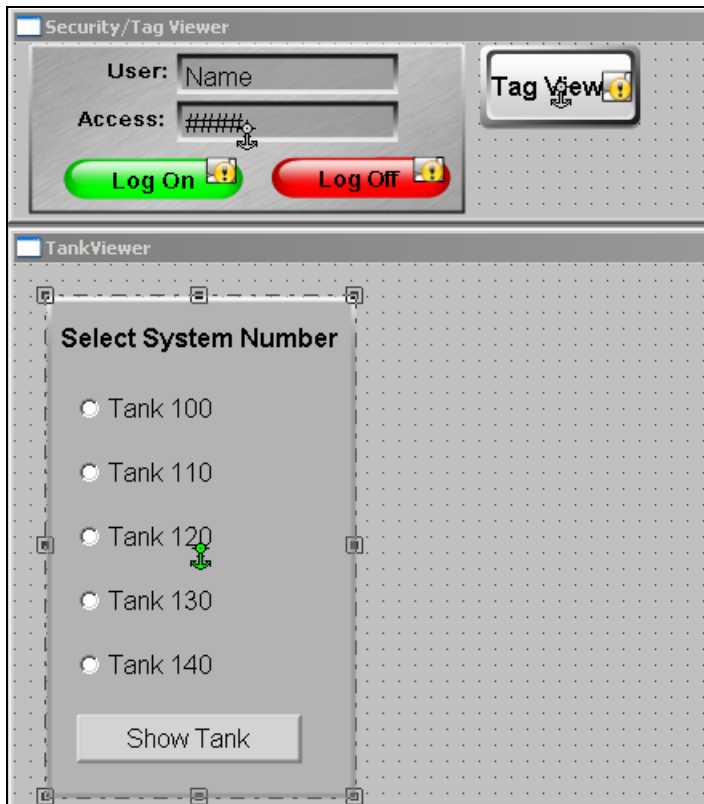
44. Change the name to **TankViewer**, and then click **OK**.



Note: The dimensions and locations may differ based on your current resolution.

45. Close the **Reactor** window.

46. Click **Embed Archestra Graphic**.
47. Click **ClassSymbols**, and then double-click **TankPanel**.
48. In the **TankViewer** window, click to insert **TankPanel**.



Now you will configure TankPanel to link the SystemNumber custom property to the corresponding InTouch tag.

49. Double-click **TankPanel**, and then in the **Default Value** field enter **SystemNumber**.
50. Click **OK**.

Test What You Have Built

51. Click **Runtime**.

Note: If the TankDisplay does not indicate the MaxEU value, try restarting WindowViewer.

52. Select a tank, and then click **Show Tank**.
53. Verify that the title is correct, values are changing, the symbol is resizable, and other functionality is locked out until the symbol is closed.
54. Repeat testing for each of the available tanks.
55. Click **Development!**.

Section 3 – The ShowGraphic Script Function

This section discusses ShowGraphic script function, as another method of calling a symbol into InTouch from within another symbol.

ShowGraphic Script Function

The ShowGraphic script function displays a graphic within a pop-up window in much the same way the Show Symbol animation does.

However, there are some important differences:

- Uses scripting rather than an Animation link
- Does not require the use of indirect tags
- Multiple symbols can be called at the same time
- Multiple instances of the same symbols can be called

Scripting

The scripting for this feature may look extensive at first; however, it can be broken down into four easy to understand parts:

Which symbol to show

```
Dim graphicInfo as aaGraphic.GraphicInfo;  
graphicInfo.Identity = "Display1";  
graphicInfo.GraphicName = "Vessel";
```

How the symbol displays

```
graphicInfo.Resizable = true;  
graphicInfo.HasTitleBar = false;  
graphicInfo.width = 250  
graphicInfo.height = 300  
graphicInfo.X = 100  
graphicInfo.Y = 100
```

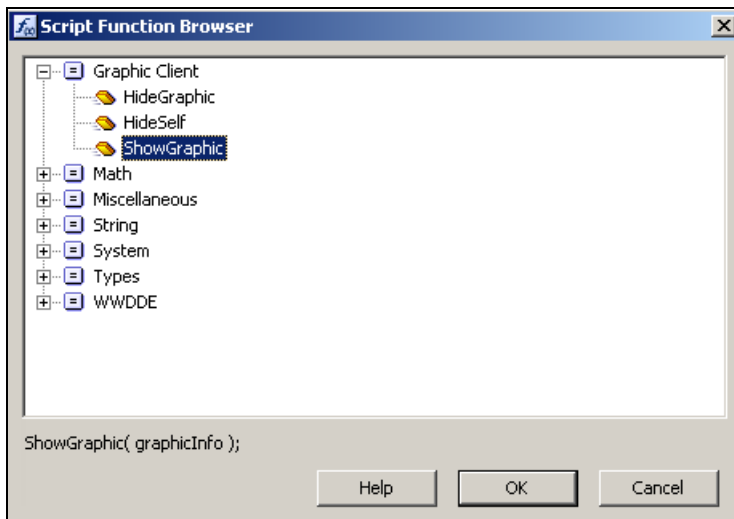
Which values to display

```
Dim cpValues [3] as aaGraphic.CustomPropertyValuePair;  
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);  
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", "InTouch:Tag1",  
False);  
cpValues[3] = new aaGraphic.CustomPropertyValuePair("CP3", "InTouch:Tank" +  
var1, true);  
graphicInfo.CustomProperties = cpValues;
```

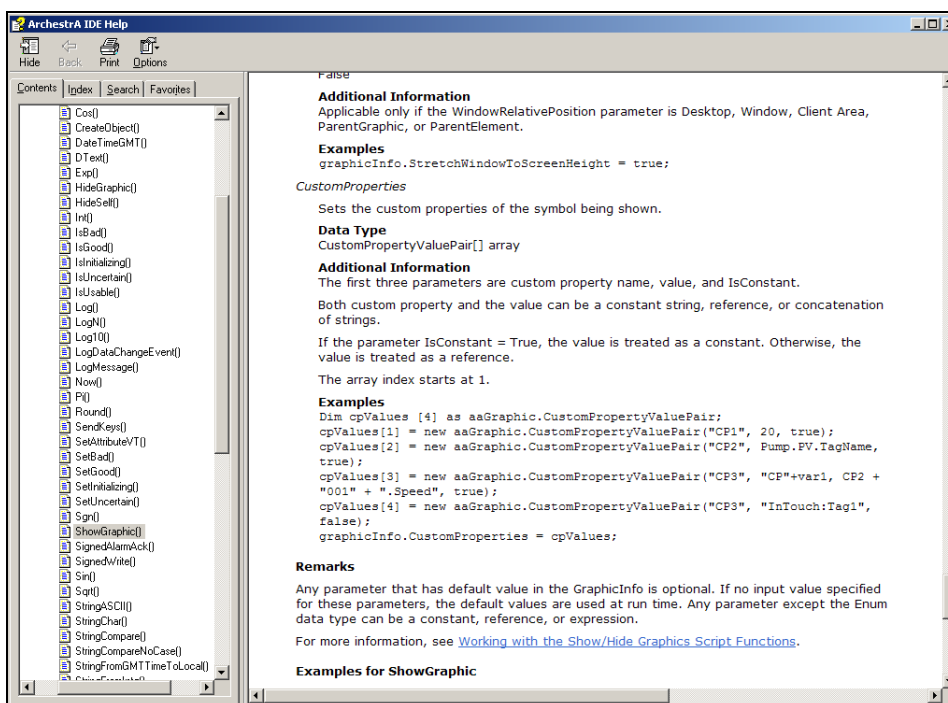
Command to show symbol

```
ShowGraphic( graphicInfo )
```

The Help button located within ArchemstrA IDE's Script Function Browser, provides context sensitive scripting assistance.



Scroll down to the example section, copy and paste it into the script window, and then simply swap out the specific references. This is a useful technique not only for this feature, but whenever scripting assistance is needed.



Lab 5 – Basics of the ShowGraphic Function

Introduction

In this lab, you will use the ShowGraphic script to display an ArchestrA symbol that is not directly housed within InTouch. The purpose of this lab is to illustrate the similarities and differences between the Show Symbol animation link and ShowGraphic script function.

Objectives

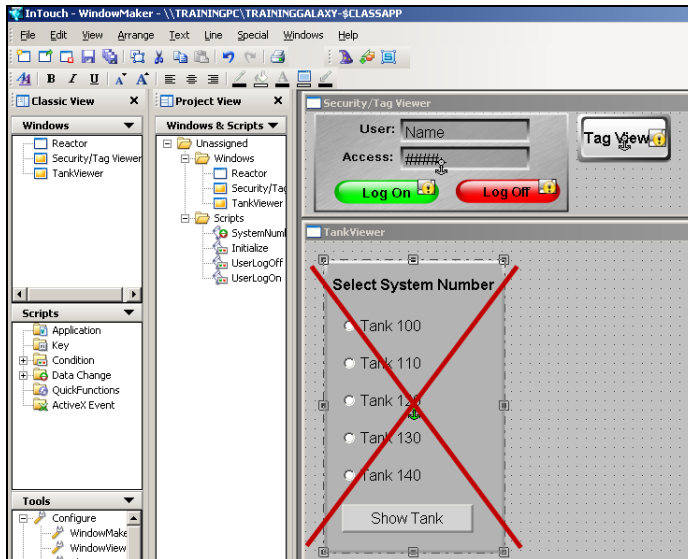
Upon completion of this lab, you will be able to:

- Compare and contrast the ShowGraphic script function with the Show Symbol animation
- Leverage the previous lab's Tank symbol with a custom property assignment

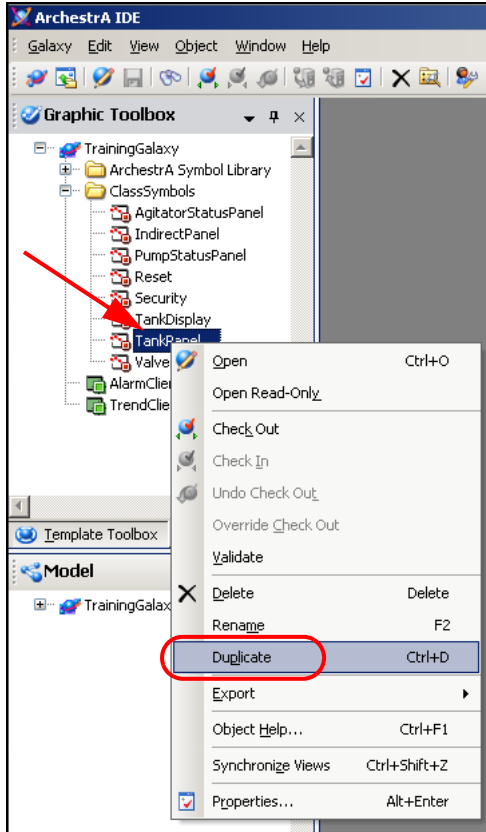
Show a Referenced Symbol with ShowGraphic

As in the previous lab, you will create a referenced symbol using Archestra IDE. However, by using the ShowGraphic script function instead of the Show Symbol animation, you will have the ability to display multiple instances of the same graphic, while bypassing the need for indirect tags.

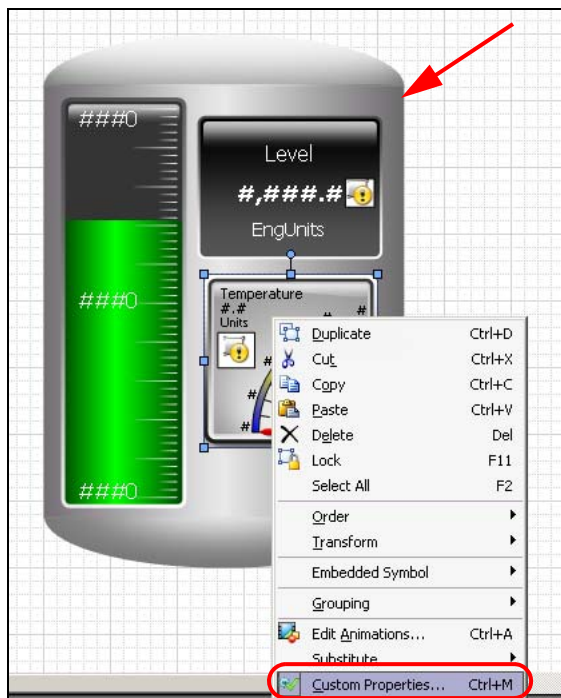
1. In WindowMaker, delete the **Select System Number** panel from the **TankViewer** window.



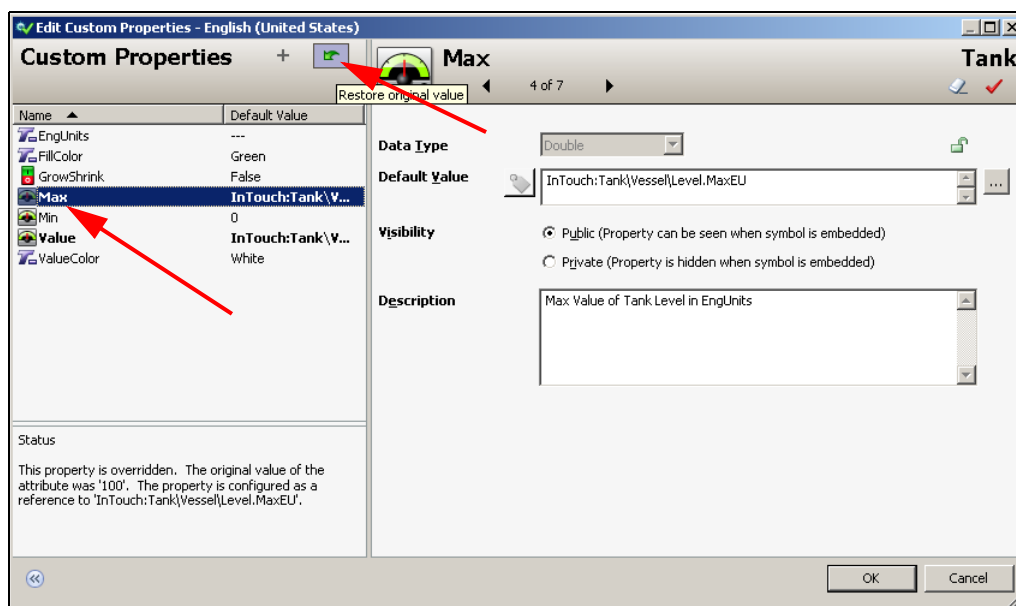
2. In Archestra IDE, expand **ClassSymbols**, right-click **TankPanel** and select **Duplicate**.



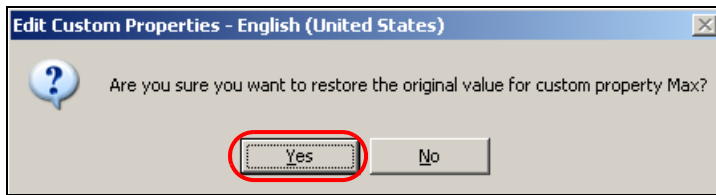
3. Rename the new symbol **TankPanel2**.
4. Right-click **TankDisplay** and select **Duplicate**.
5. Rename the new symbol **TankDisplay2**.
6. Double-click **TankDisplay2** to open it.
7. Right-click the **Tank** symbol and select **Custom Properties**.



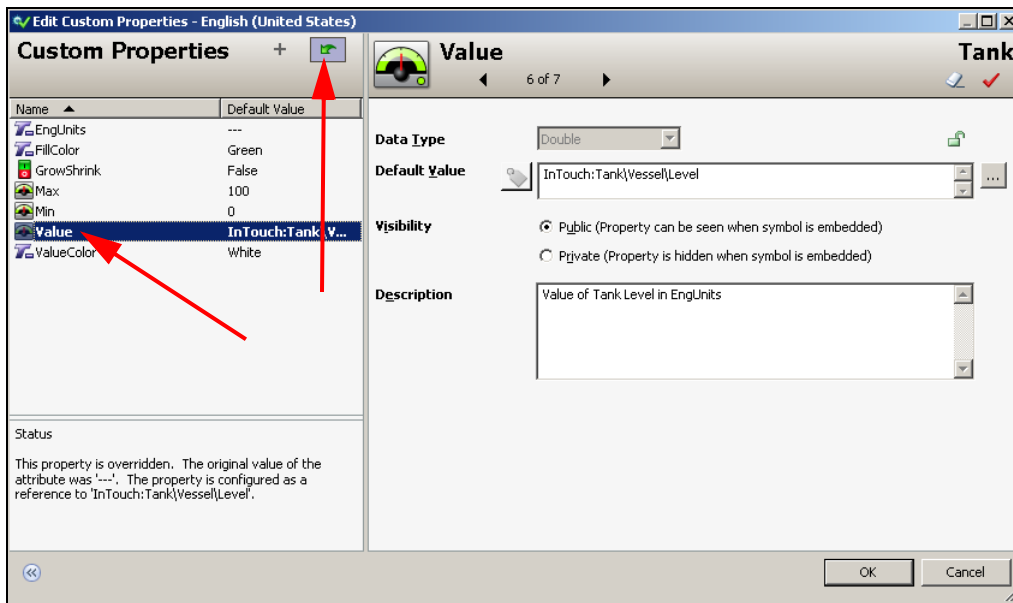
8. Select **Max**, and then click **Restore** to restore it to its original value.



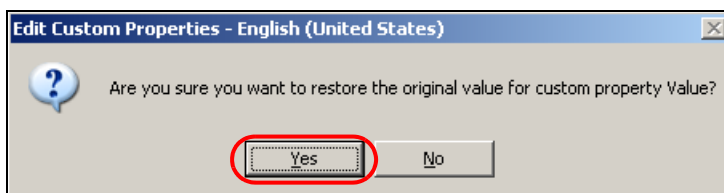
9. Click **Yes**.



10. Select **Value**, and then click **Restore** to restore it to its original value.

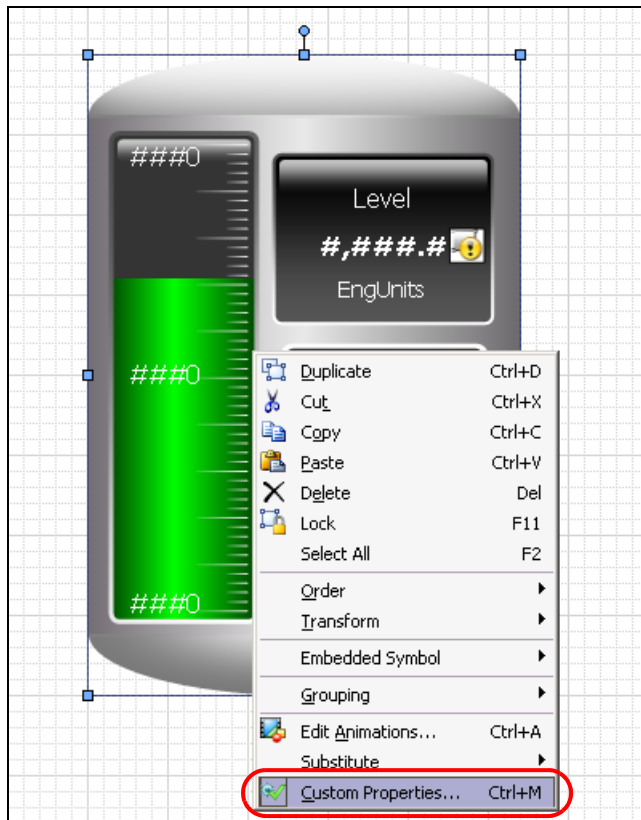


11. Click **Yes**.

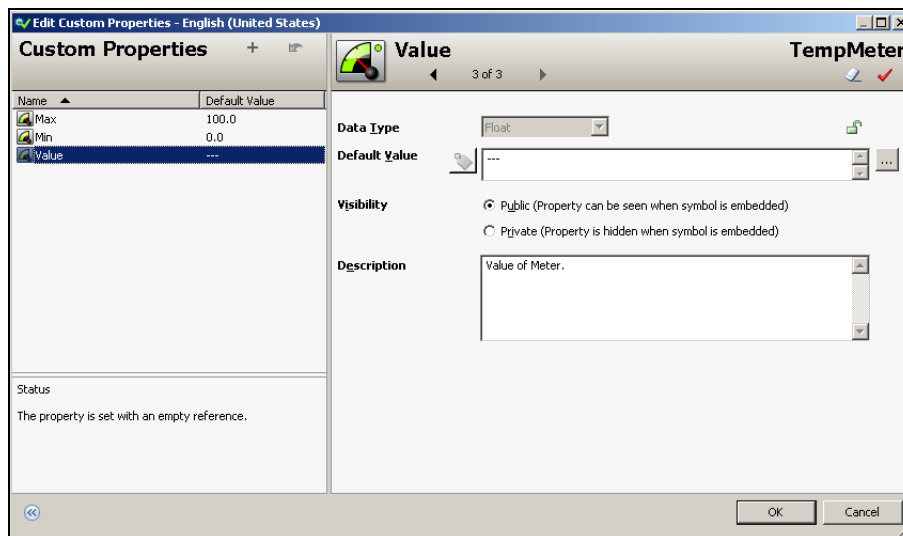


12. Click **OK**.

13. Right-click the **TempMeter** symbol and select **Custom Properties**.



14. Select **Max**, click **Restore** to restore it to its original value, and then click **Yes**.
 15. Select **Value**, click **Restore** to restore it to its original value, and then click **Yes**.
 Your values should look as shown.



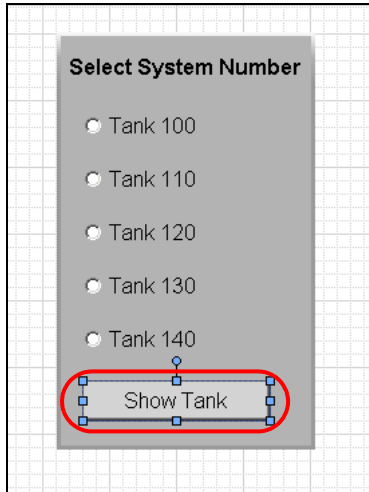
16. Click **OK**.
 17. Click **Save and Close**, and then check in the symbol when prompted.

Add ShowGraphic Scripting

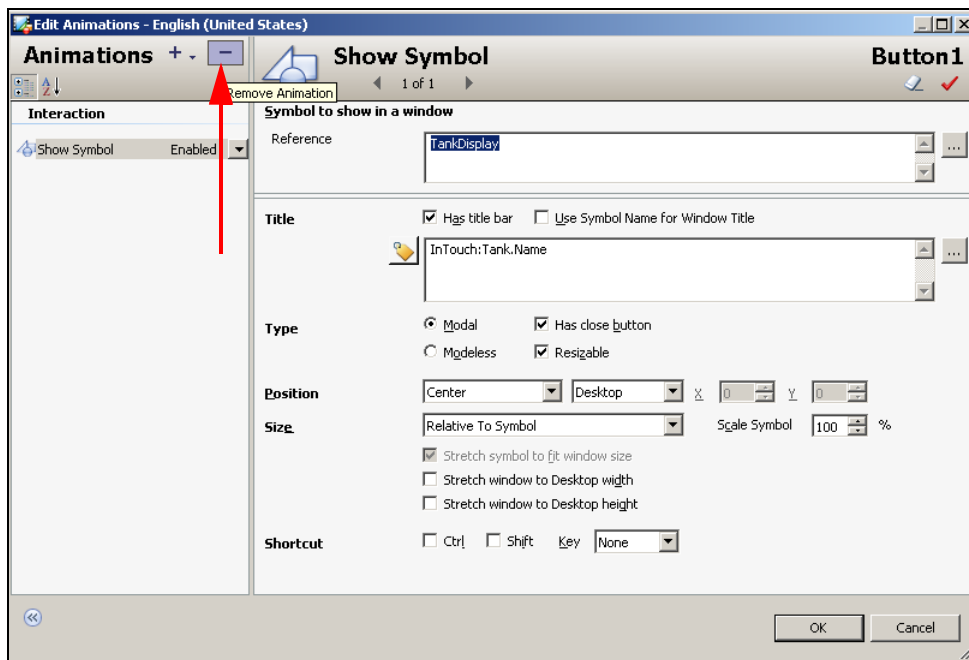
In this section, you will replace the Show Symbol animation link with ShowGraphic action scripting.

Note: All scripts for this lab are available to copy from the file **Lab 5 Script** in your **Training** directory. It is recommended, however, that you use the step-by-step instructions that follow to learn how this script is created.

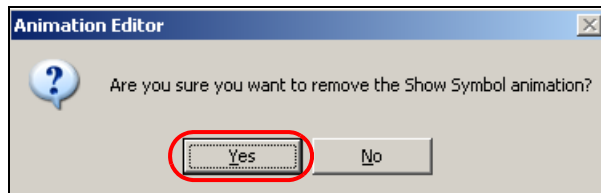
18. In Archestra IDE, expand **ClassSymbols**, and then double-click **TankPanel2** to open it.
19. Double-click **Show Tank** to configure it.



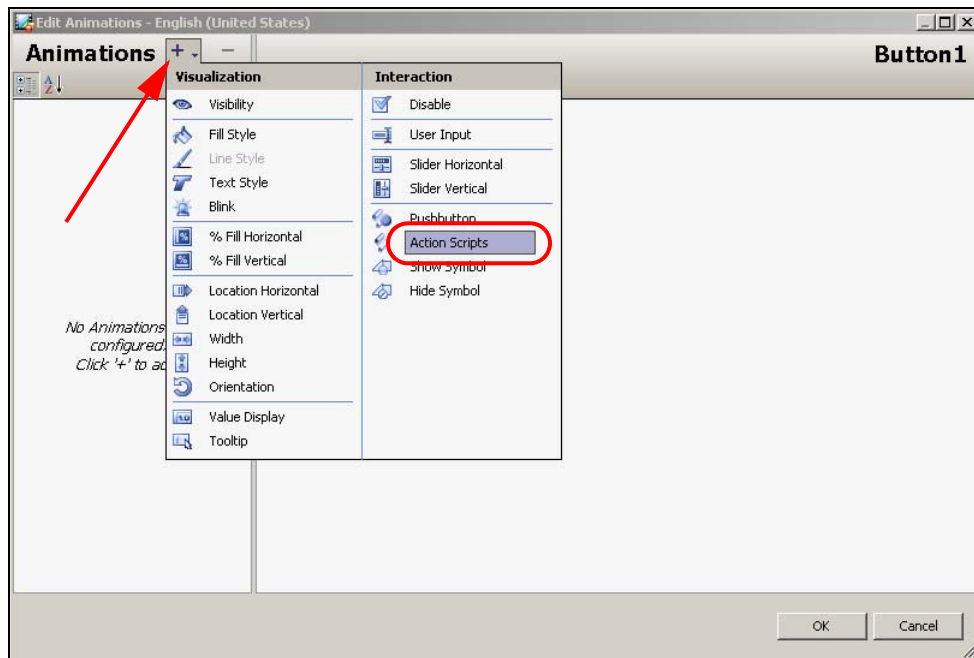
20. Click – to remove the **Show Symbol** animation link.



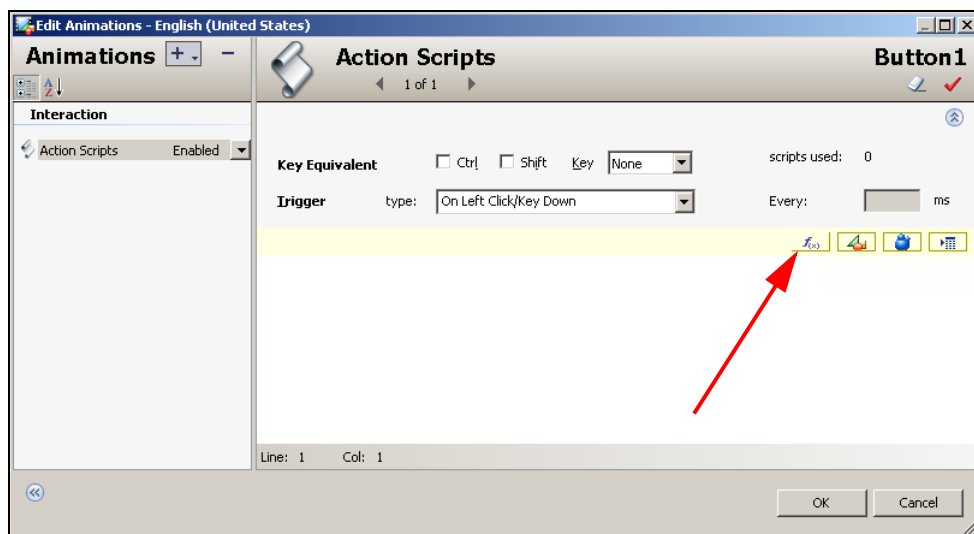
21. Click **Yes**.



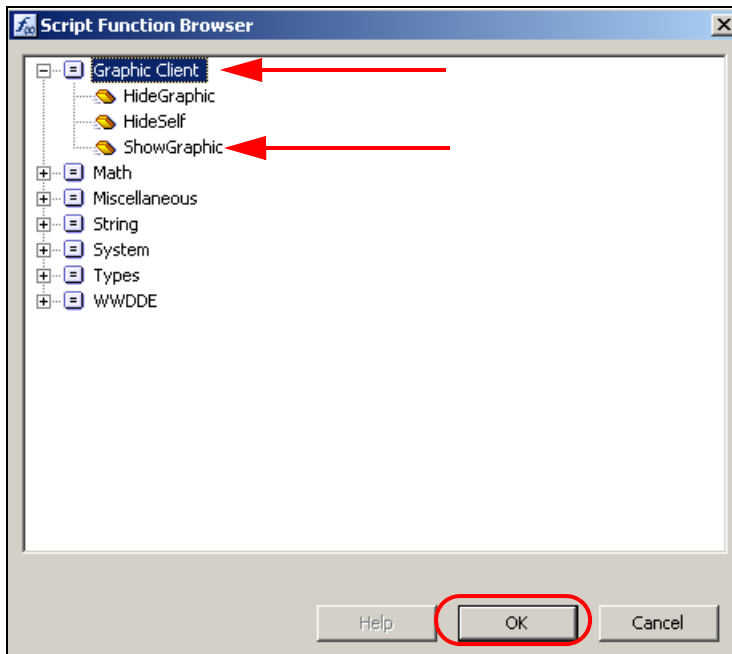
22. Click **+** and select **Action Scripts**.



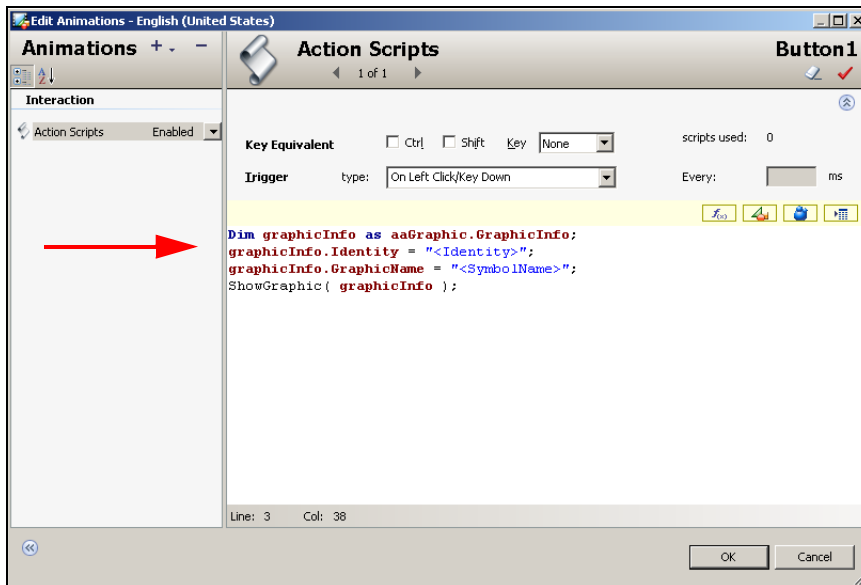
23. Click **Function Browser**.



24. Expand the **Graphic Client** folder, select **ShowGraphic**, and then click **OK**.



This enters four lines of scripting into the script field.



You will leave the first line as is. This line, which is the dimension statement, allows the aliasing of the function.

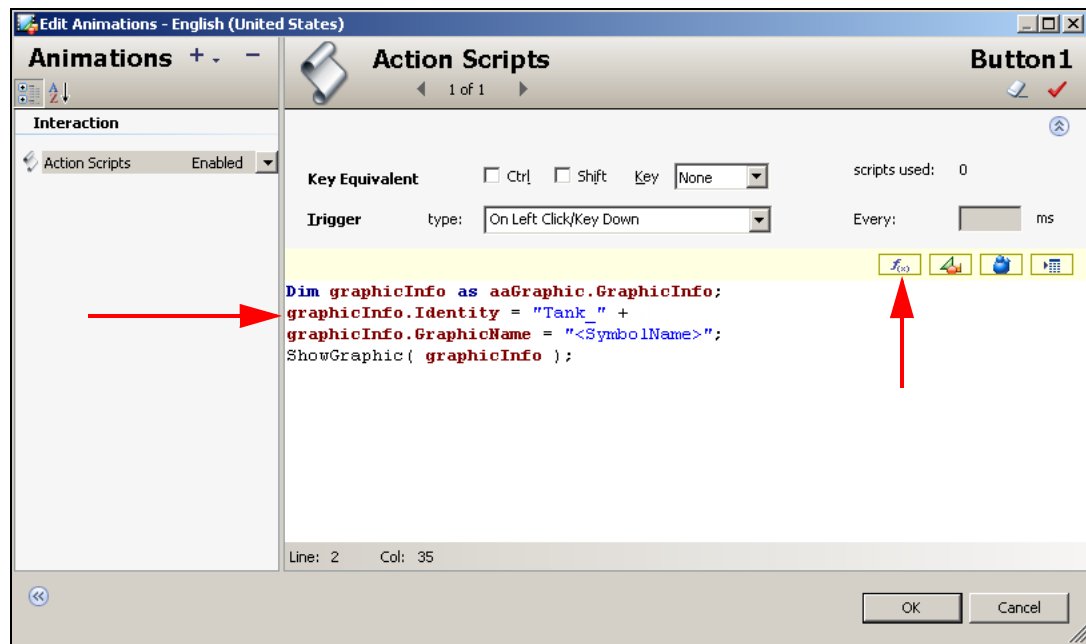
Dim graphicInfo as aaGraphic.GraphicInfo;

The second line refers to the symbol's identity (or name).

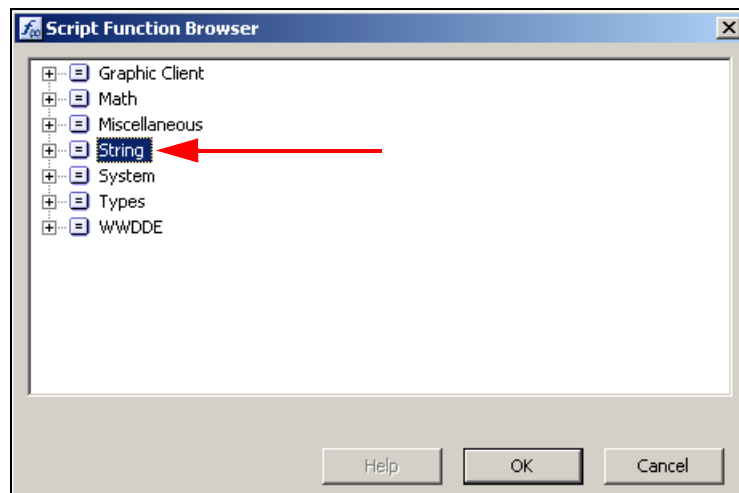
graphicInfo.Identity = "<Identity>";

To ensure each symbol has its own name, each instance of the symbol derives a unique name from the **SystemNumber** parameter. This will allow the application to show multiple instances of the same symbol.

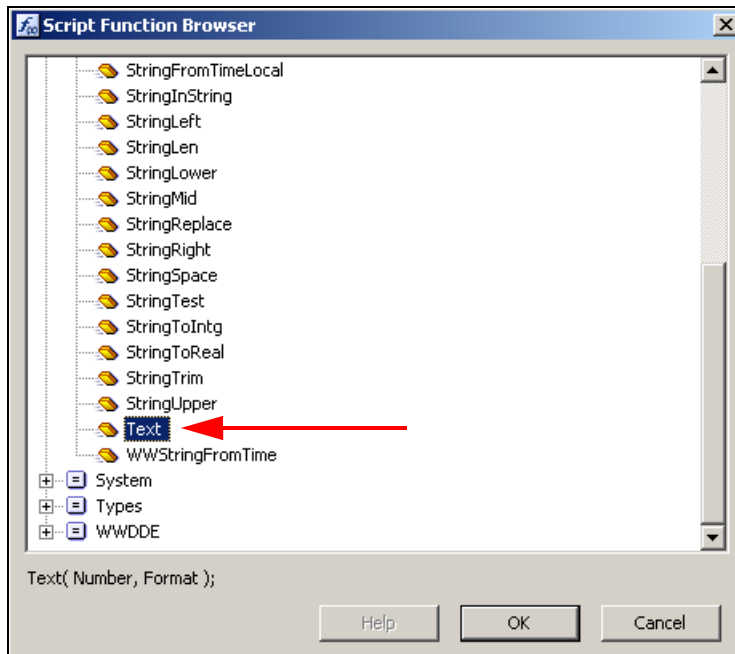
25. Replace “<Identity>”; with “Tank_” +, and then click the **Function Browser**.



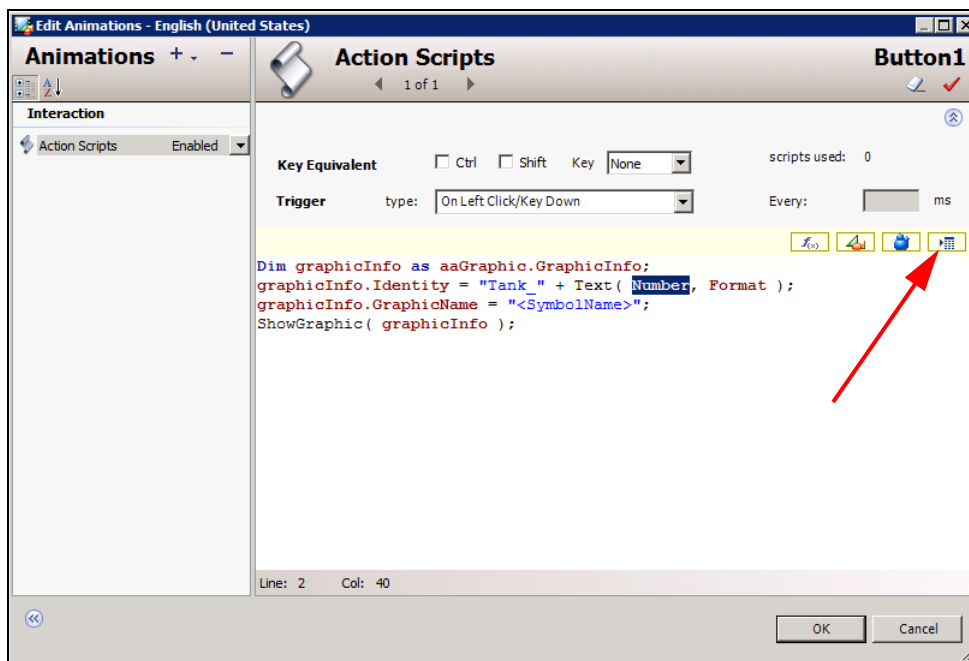
26. Expand the **String** folder.



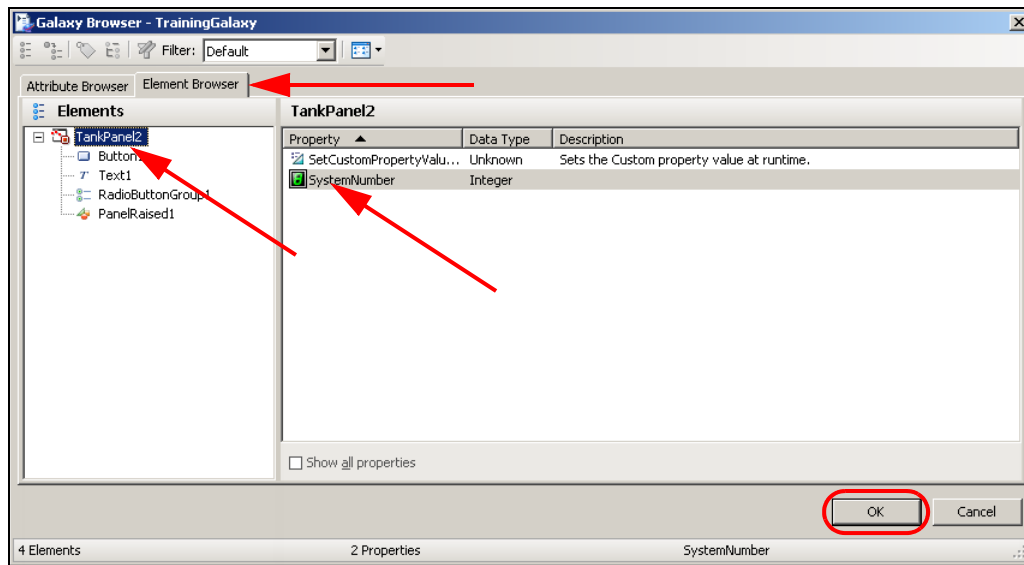
27. Scroll down, select **Text**, and then click **OK**.



28. Highlight **Number**, and then click **Display Attribute Browser**.



29. From the **Element Browser** tab, click **TankPanel2** and select **SystemNumber**, and then click **OK**.



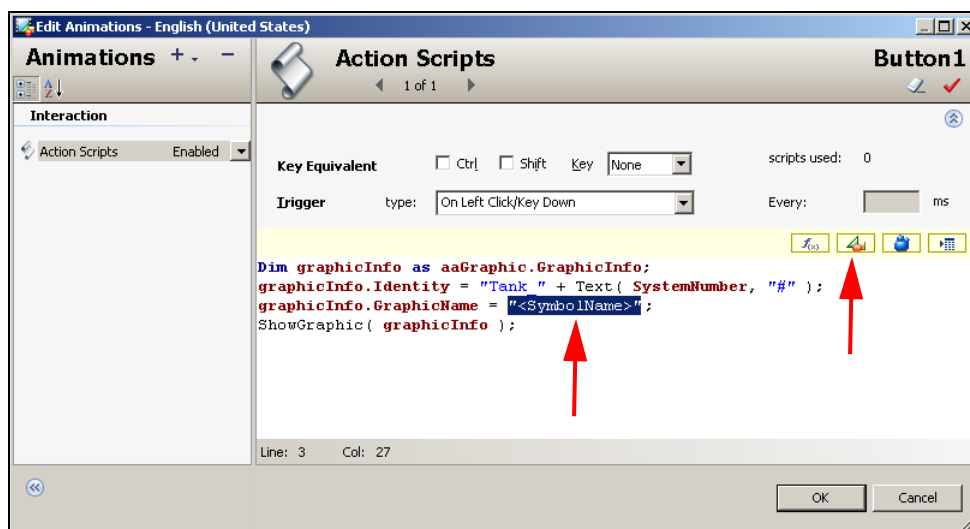
30. Replace the **Format** string with “#”, and then remove the extra semi-colon from the end of the line.

Line 2 should now look like this:

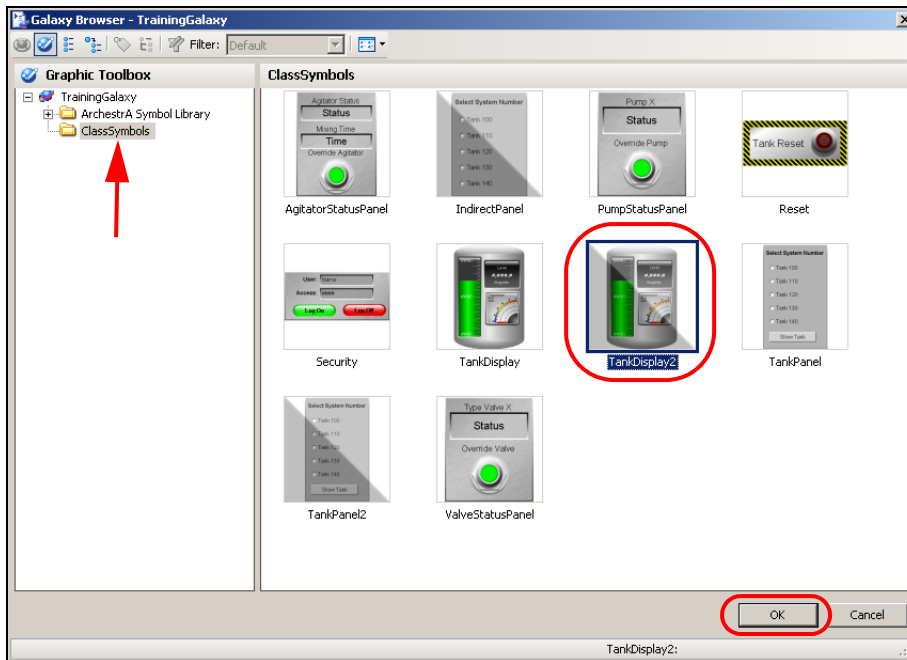
```
graphicInfo.Identity = "Tank_" + Text (SystemNumber, "#");
```

Next, enter the graphical information on Line 3.

31. Highlight “<SymbolName>”, and then click **Display Graphic Browser**.



32. In **ClassSymbols**, select **TankDisplay2**, and then click **OK**.

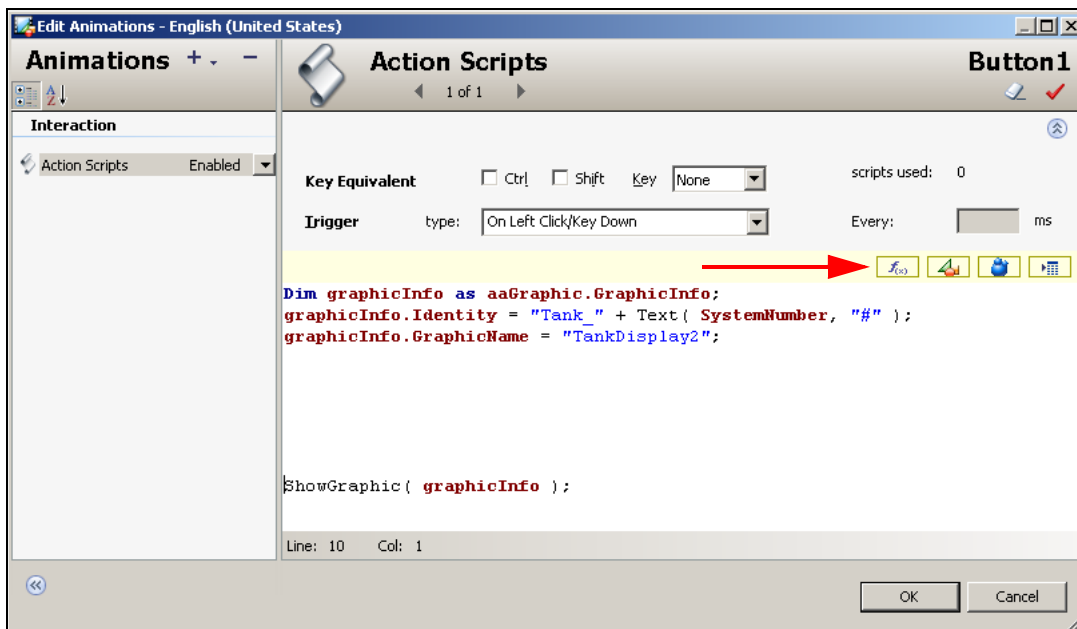


Line 3 should now look like this:

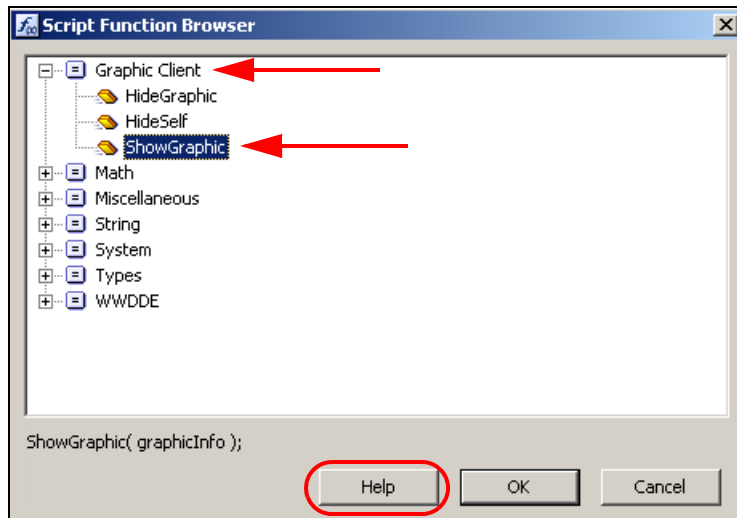
```
graphicInfo.Identity = "TankDisplay2";
```

Next, add the script to allow resizing.

33. Add several spaces between lines 3 and 4, and then click **Script Function Browser**.

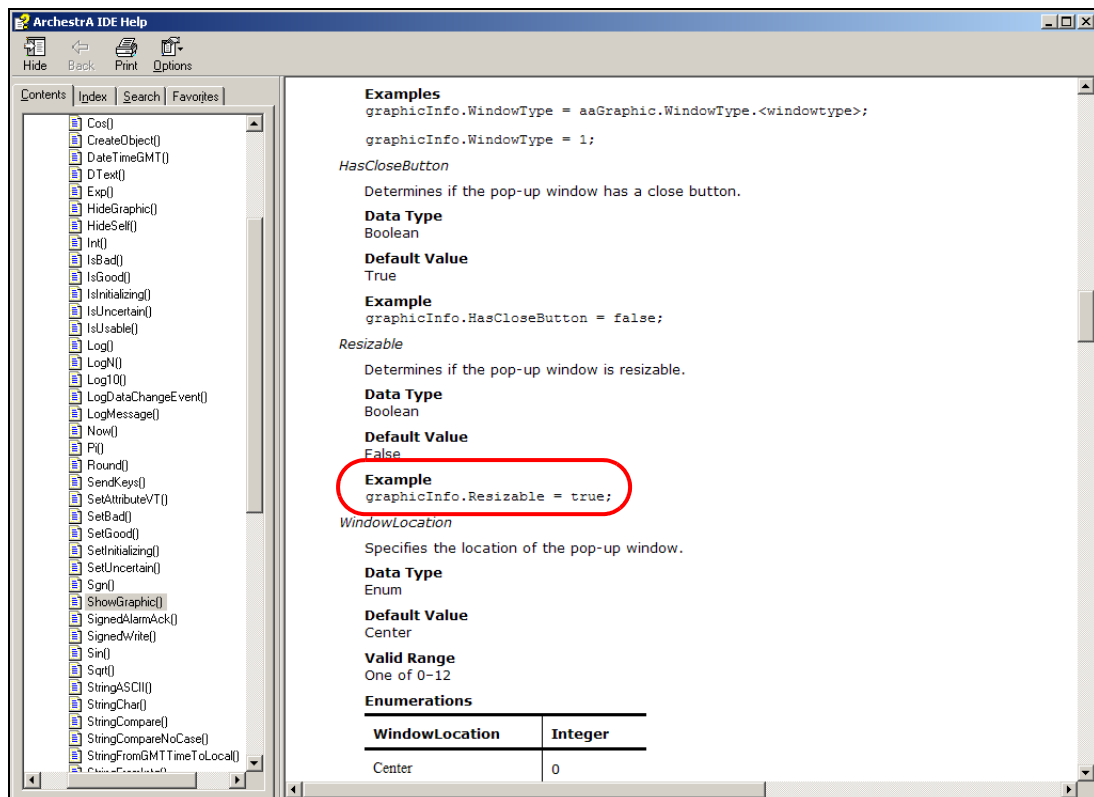


34. Expand **Graphic Client**, select **ShowGraphic**, and then click **Help**.

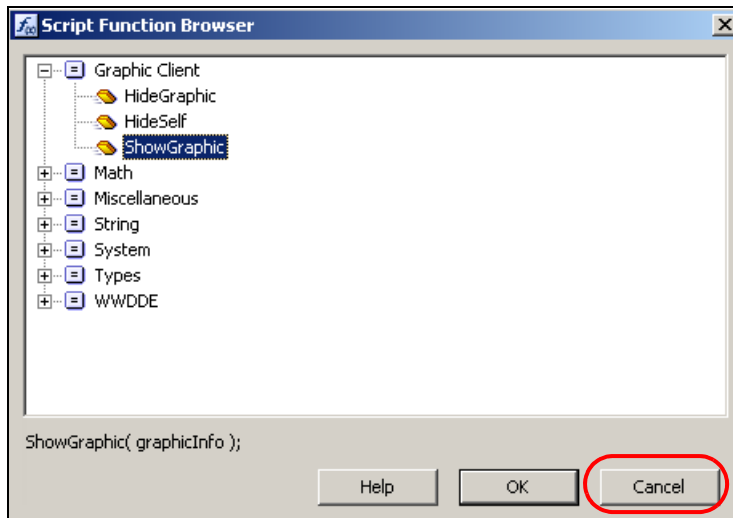


Note: The help tool provides context sensitive scripting assistance.

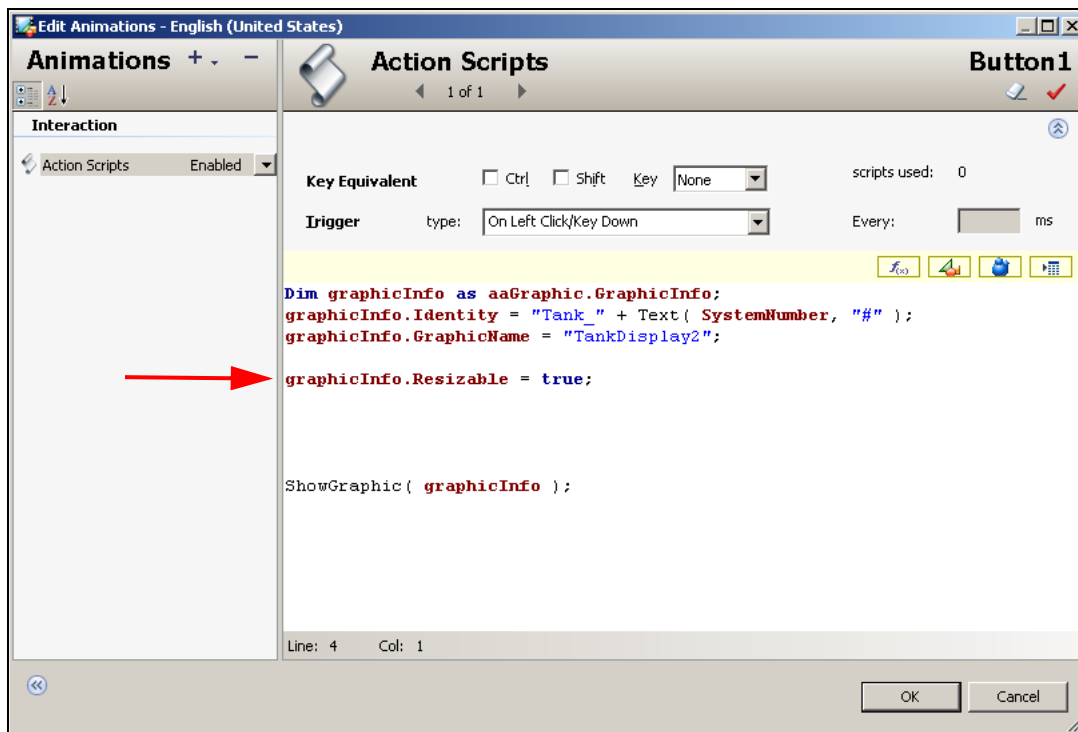
35. Scroll down to the **Resizable** parameter and copy the **Example** script.



36. Minimize the help window, and then in **Script Function Browser** click **Cancel**.

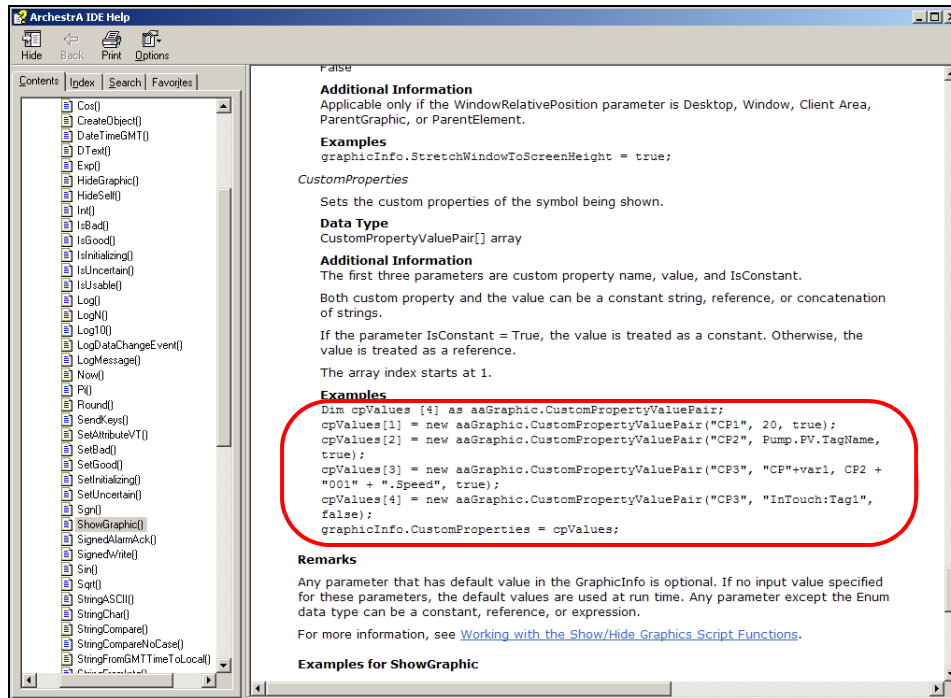


37. Paste the copied script below the third line of the existing script to enable resizability.

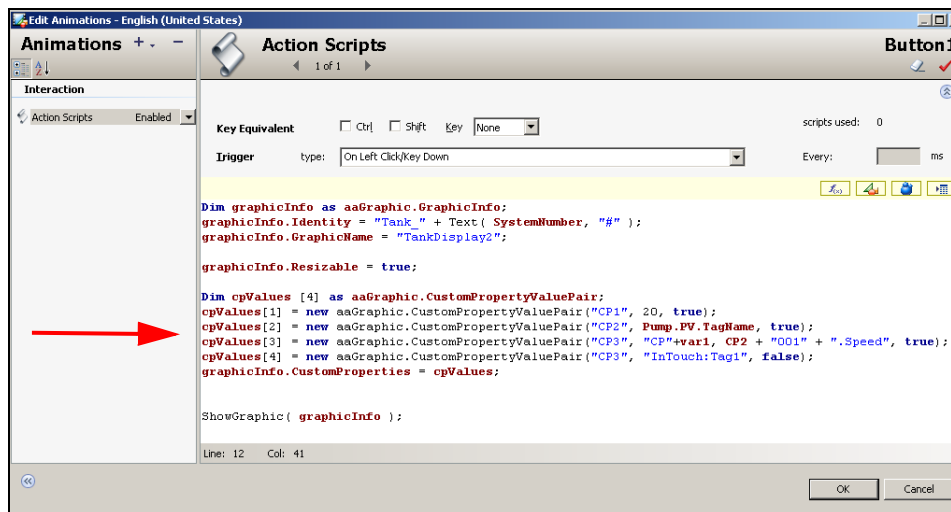


You will need to tell the symbol which tags to use when the symbol is called. To do this, use an array function that allows the application to swap the custom properties with tags from InTouch.

38. Reopen the help file, scroll down to the **Custom Properties** heading, and then copy the **Examples** script.



39. Paste it below the resizing command.



The pasted script has two parts; the **cpValue** information and the default values:

```
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
```

.....

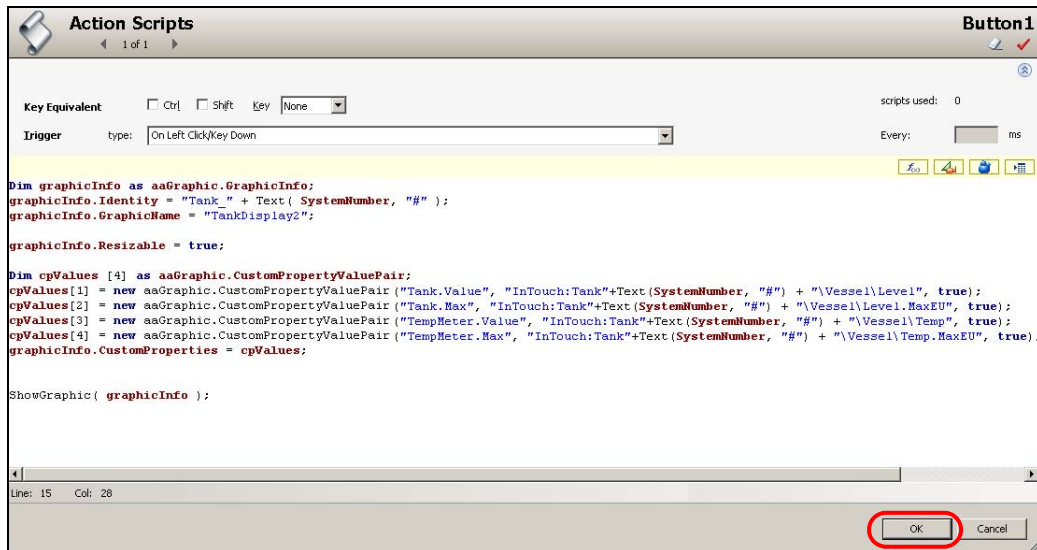
cpValue information

default values

40. Alter the default values portion of the script for each line as follows:

Cp Value	New Values
1	("Tank.Value", "InTouch:Tank"+Text(SystemNumber, "#") + "\Vessel\Level", true);
2	("Tank.Max", "InTouch:Tank"+Text(SystemNumber, "#") + "\Vessel\Level.MaxEU", true);
3	("TempMeter.Value", "InTouch:Tank"+Text(SystemNumber, "#") + "\Vessel\Temp", true);
4	("TempMeter.Max", "InTouch:Tank"+Text(SystemNumber, "#") + "\Vessel\Temp.MaxEU", true);

When you are done, the script should look like this:



41. Click **OK**.

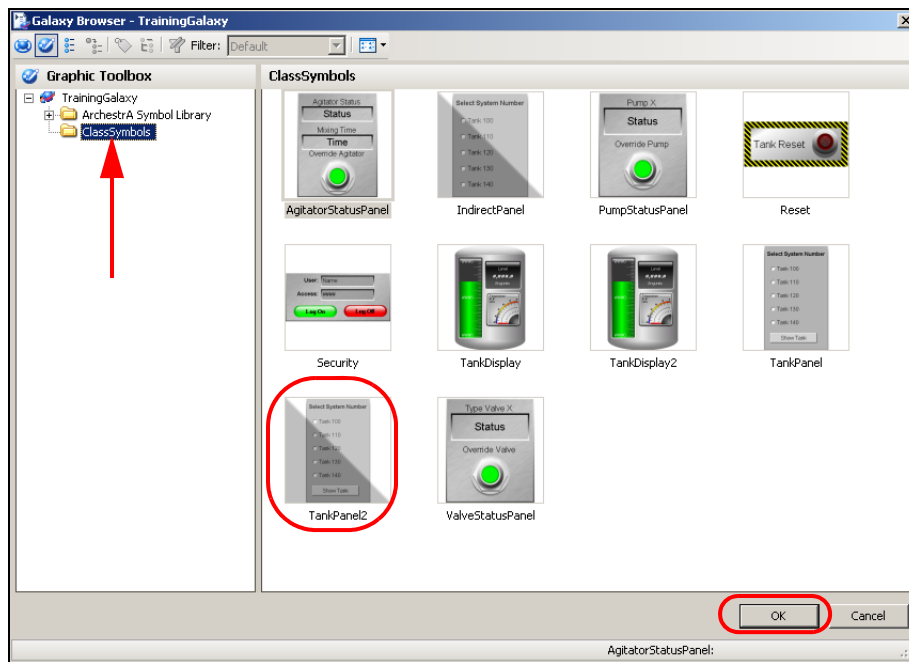
42. Click **Save and Close**, and then check in the symbol when prompted.

Embed the TankPanel2 Symbol

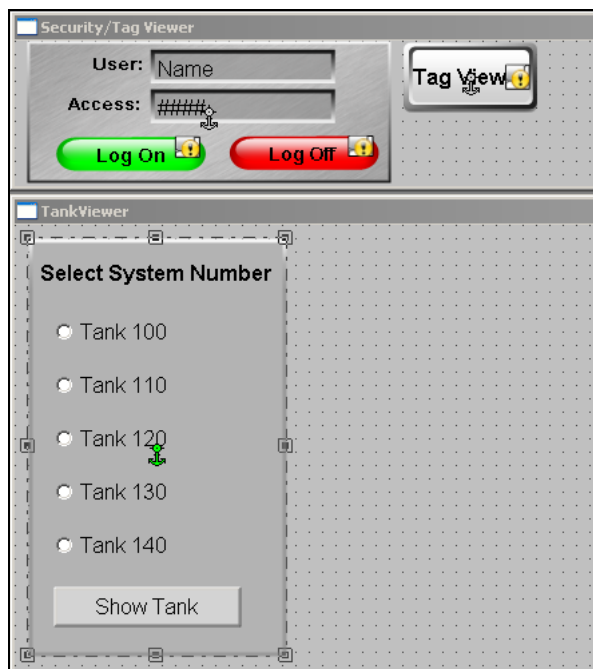
Lastly, place the newly made symbol into your display.

43. In WindowMaker, click **Embed ArchestraA Graphic**.

44. Click **ClassSymbols**, select **TankPanel2**, and then click **OK**.



45. In the **TankViewer** window, place **TankPanel2**.



Test What You Have Built

46. Click **Runtime**.
47. Select a tank, and then click **Show Tank**.
48. Verify that the title is correct, values are changing, and the symbol is resizable.

Note: Unlike in the last lab, this time the pop-up window does not have to be closed before interacting with other items in the display. You can also open multiple tanks at the same time, which will work independently of one another.

49. With a tank window still open, select a new tank and click **Show Tank**.

Observe more than one tank graphic at a time by moving the tank windows around and resizing the windows.

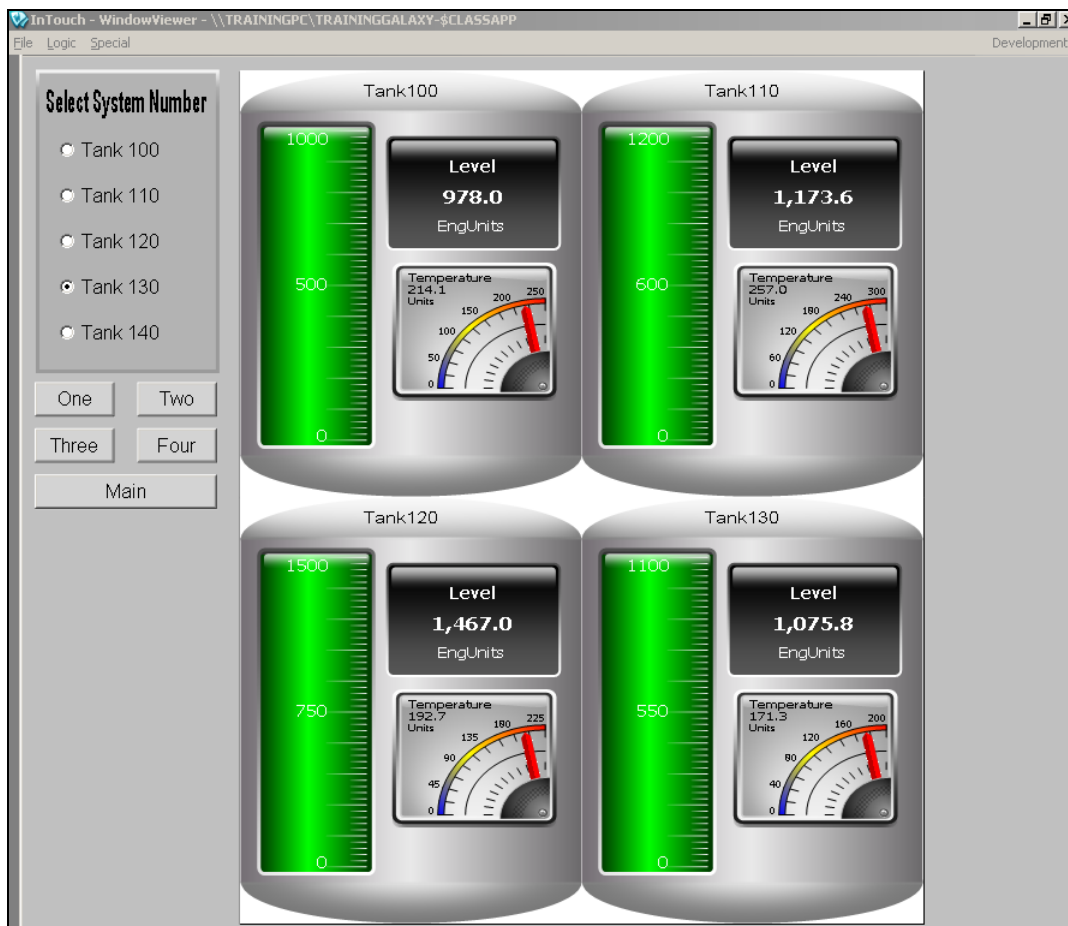
50. Click **Development!**.

Section 4 – Panel Display

This section builds upon the concepts of the ShowGraphic script function from the previous section, but uses scripting to alter the size and location of the symbol to be displayed in a panel display setup.

Control for Size and Location

By slightly altering the script in the second portion of the ShowGraphic script function, the size and location of where the symbol will be displayed can be controlled. Combine this with the ShowGraphic's ability to display multiple instances of the symbol and create a panel display setup that will allow the apparent changing of symbols within a single display.



Note: Each instance of a symbol uses up system resources, so it is a best practice to add HideGraphic scripting to close symbols that are either no longer needed or sitting behind other symbols and not visible on the display.

Lab 6 – Panel Navigation

Introduction

Similar to the previous labs, this lab will call an ArchestraA symbol into InTouch that is not directly housed within the application. However, rather than having free floating symbols, this lab will call the symbols into specific locations using a panel format.

Objectives

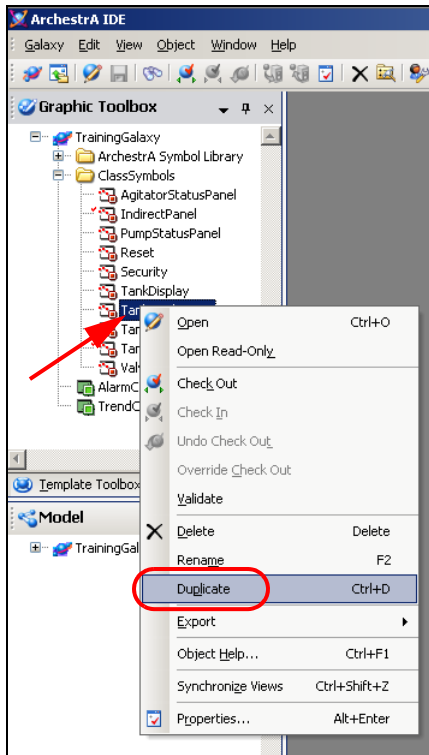
Upon completion of this lab, you will be able to:

- Create a new Panel Navigation symbol reusing elements from the previous labs
- Control the size and location of where the symbol will display

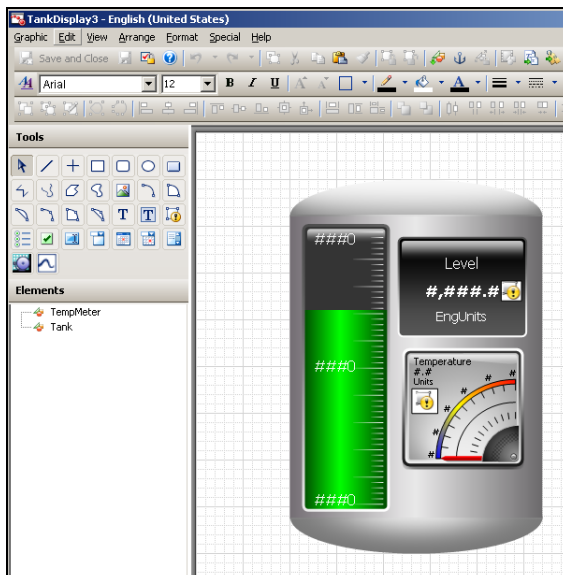
Create the Symbol

Your first step in this lab is to create a new symbol based on the TankPanel2 symbol from the previous lab.

1. In the ArchestrA IDE **Graphic Toolbox**, in **ClassSymbols**, right-click **TankDisplay2** and select **Duplicate**.

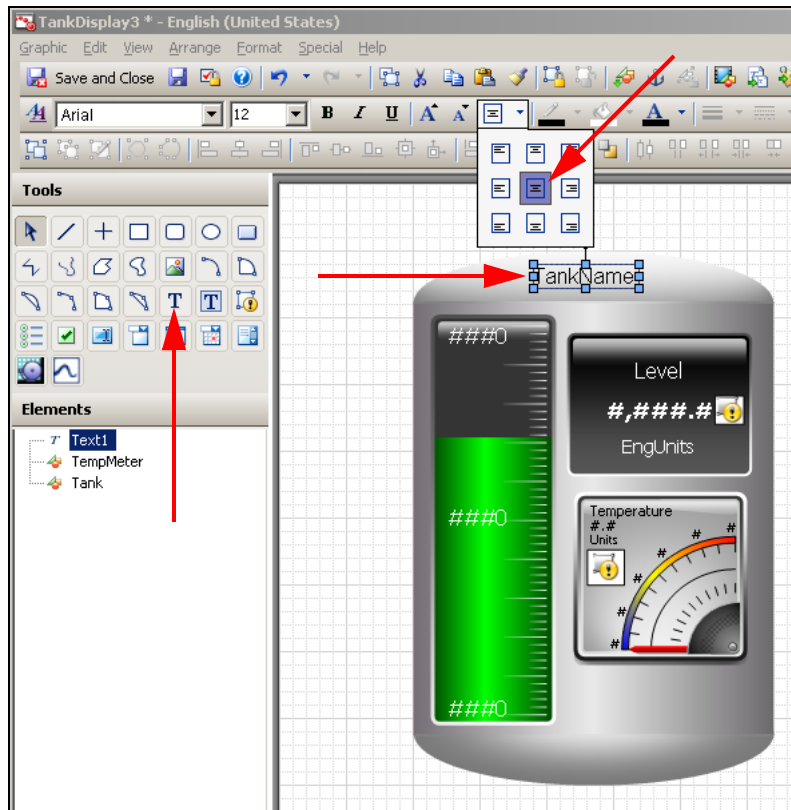


2. Rename the new symbol **TankDisplay3**, and then double-click to open it.

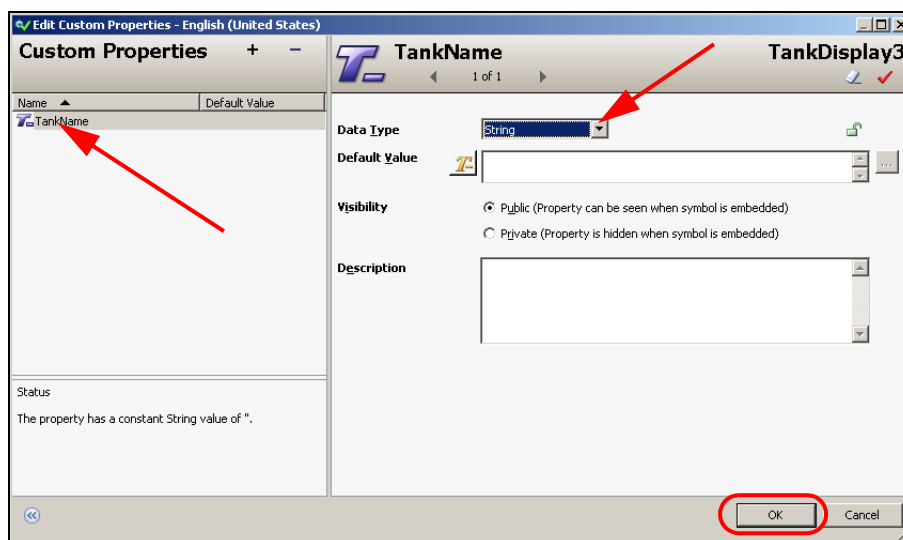


In the previous lab, a title bar was used to display the tank name. In this lab, there is no title bar, so you will add a text element to the tank to show the tank name.

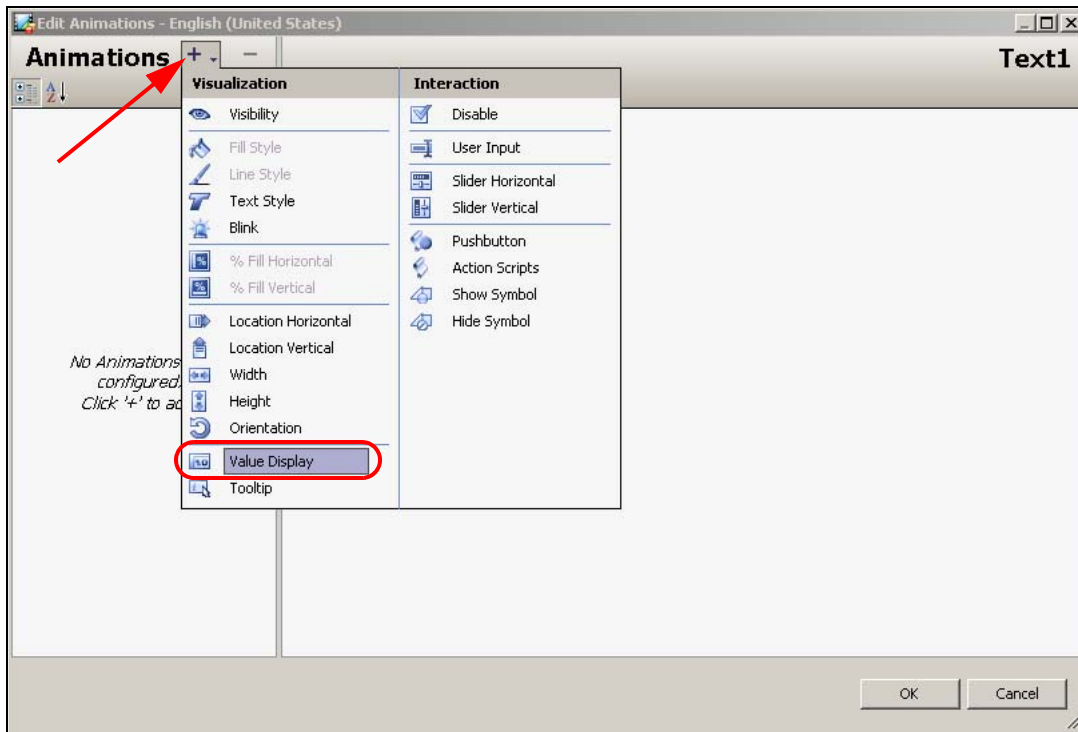
3. Using the **Text** tool, enter **TankName** and place it at the top of the tank center-justified.



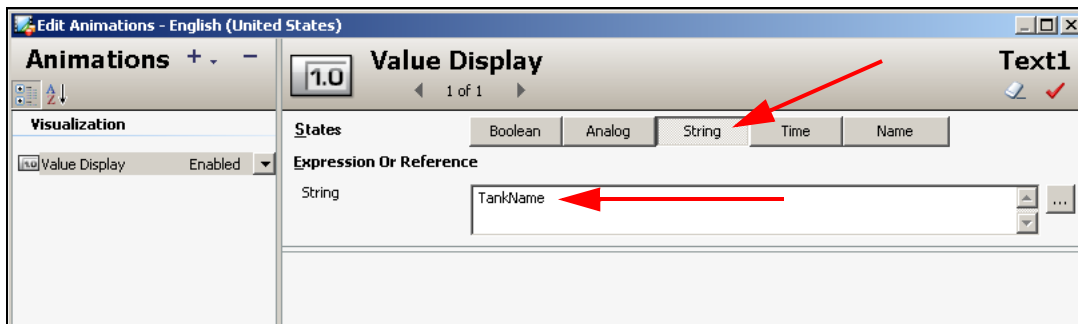
4. On the canvas, right-click and select **Custom Properties**.
5. Click **+** to add a new custom property.
6. Name the new property **TankName**, in the **Data Type** field select **String**, and then click **OK**.



7. Double-click the **TankName** text element.
8. Click **+** to add an animation, and then click **Value Display**.



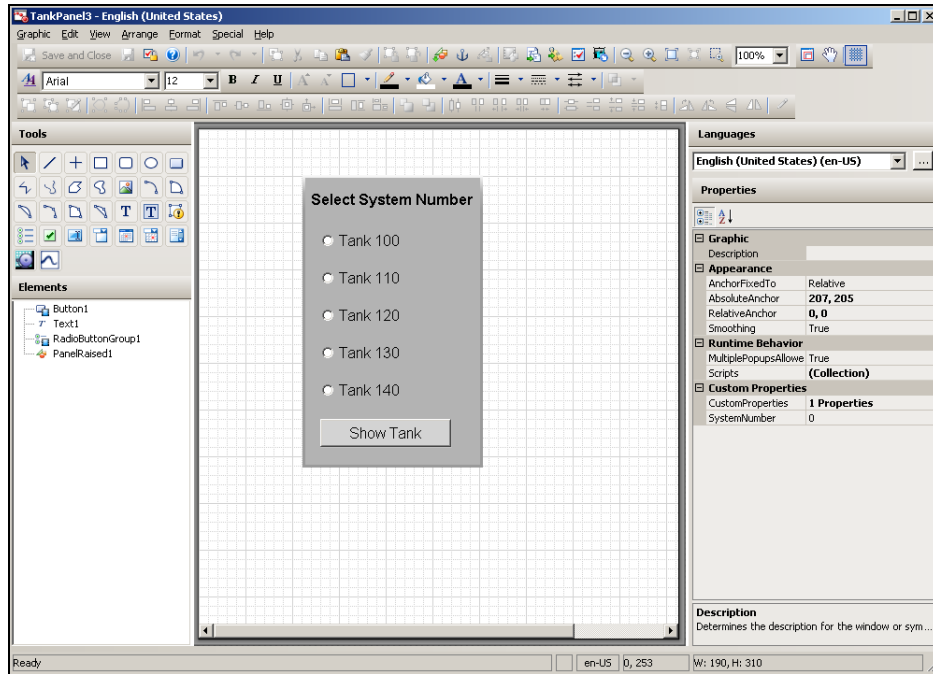
9. Click **String**, in the **String** field enter **TankName** to link the text element to the **TankName** custom property, and then click **OK**.



10. Click **Save and Close**, and then check in the symbol when prompted.

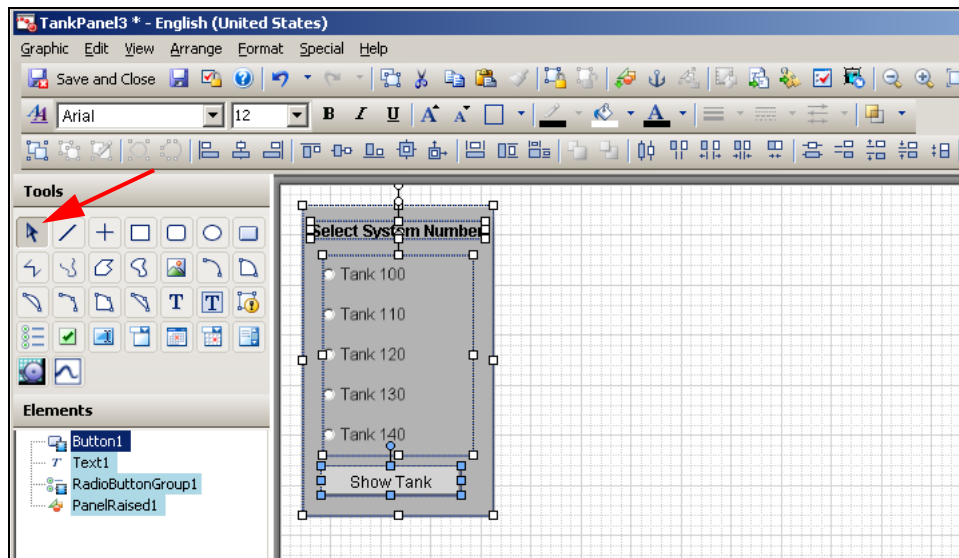
Next, create the panel that will call the symbol.

11. In Archestra IDE, in **ClassSymbols**, right-click **TankPanel2** and select **Duplicate**.
12. Rename the new symbol **TankPanel3**.
13. Double-click **TankPanel3** to open it.

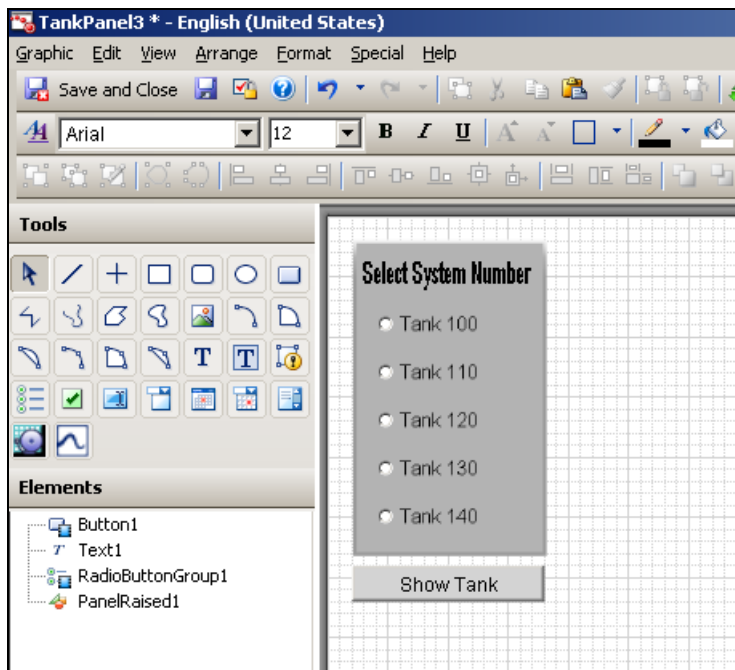


Note: You may wish to zoom out to give yourself more room to design the symbol.

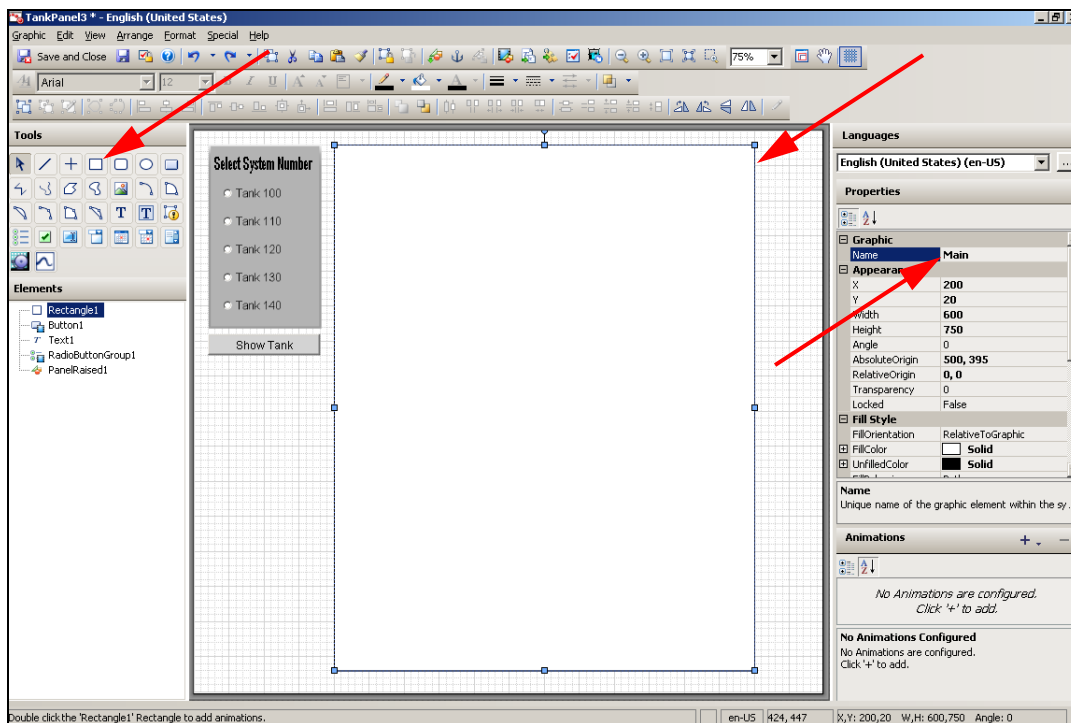
14. Using the **Pointer** tool, select the panel graphic and all of its elements, move it to the top-left corner of the window.



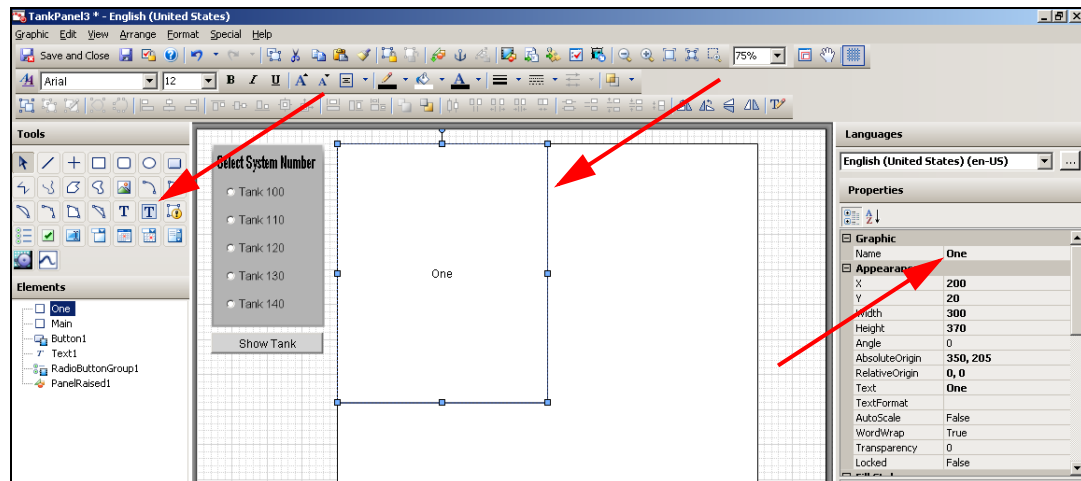
15. Decrease the size of the panel graphic and place the **Show Tank** button below the graphic.



16. Draw a rectangle and name it **Main**.

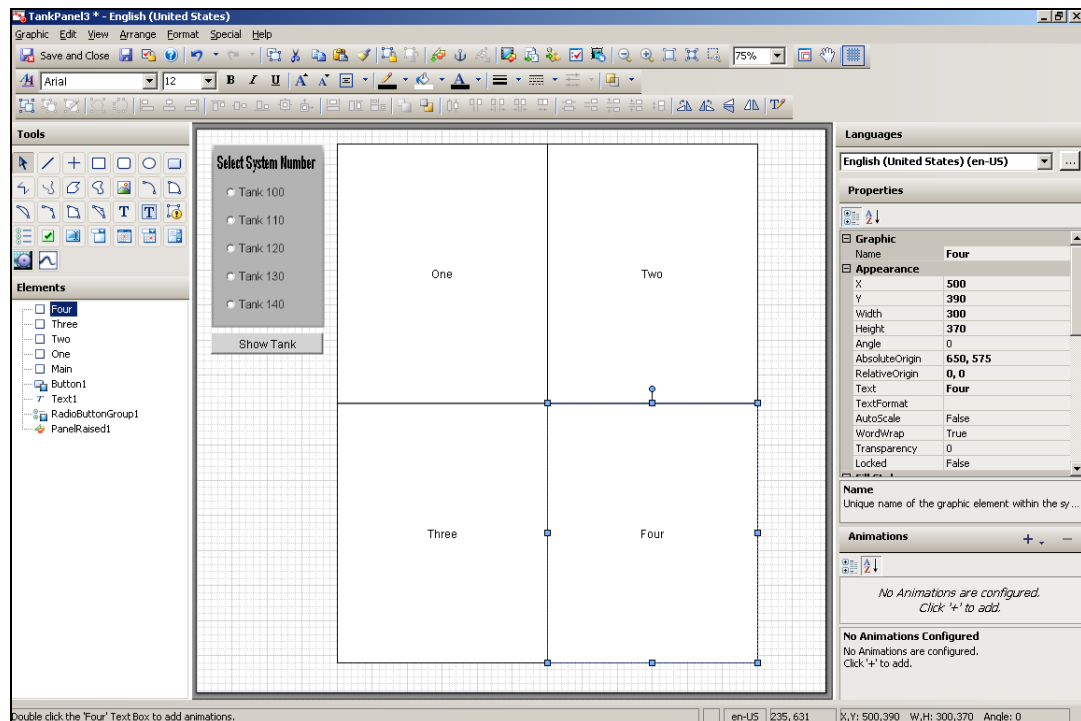


17. Use the **Text Box** tool to draw a text box 1/4th the size of the rectangle you just drew, press **Ctrl+L** to change the text to **One**, and then in the right pane rename the graphic to **One**.



Note: This may require size adjustment of both the smaller and larger rectangle to get it to fit.

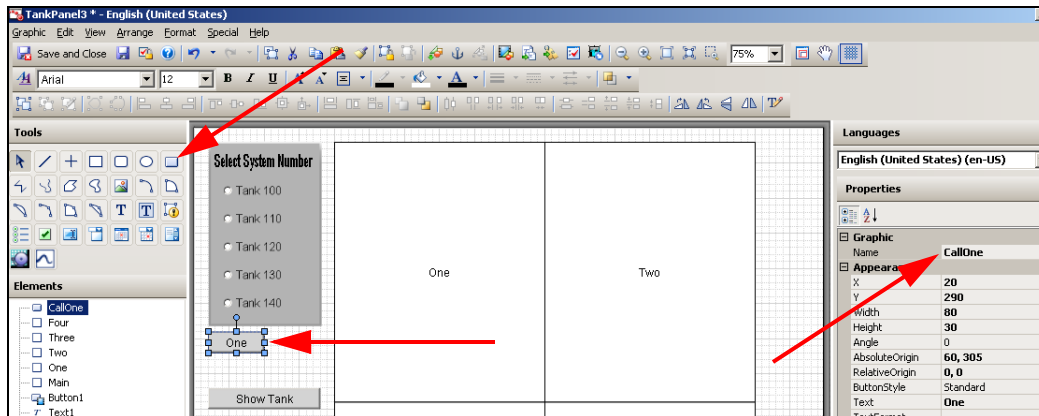
18. Duplicate this rectangle three times, changing the text and name to: **Two**, **Three**, and **Four**. Arrange as shown below.



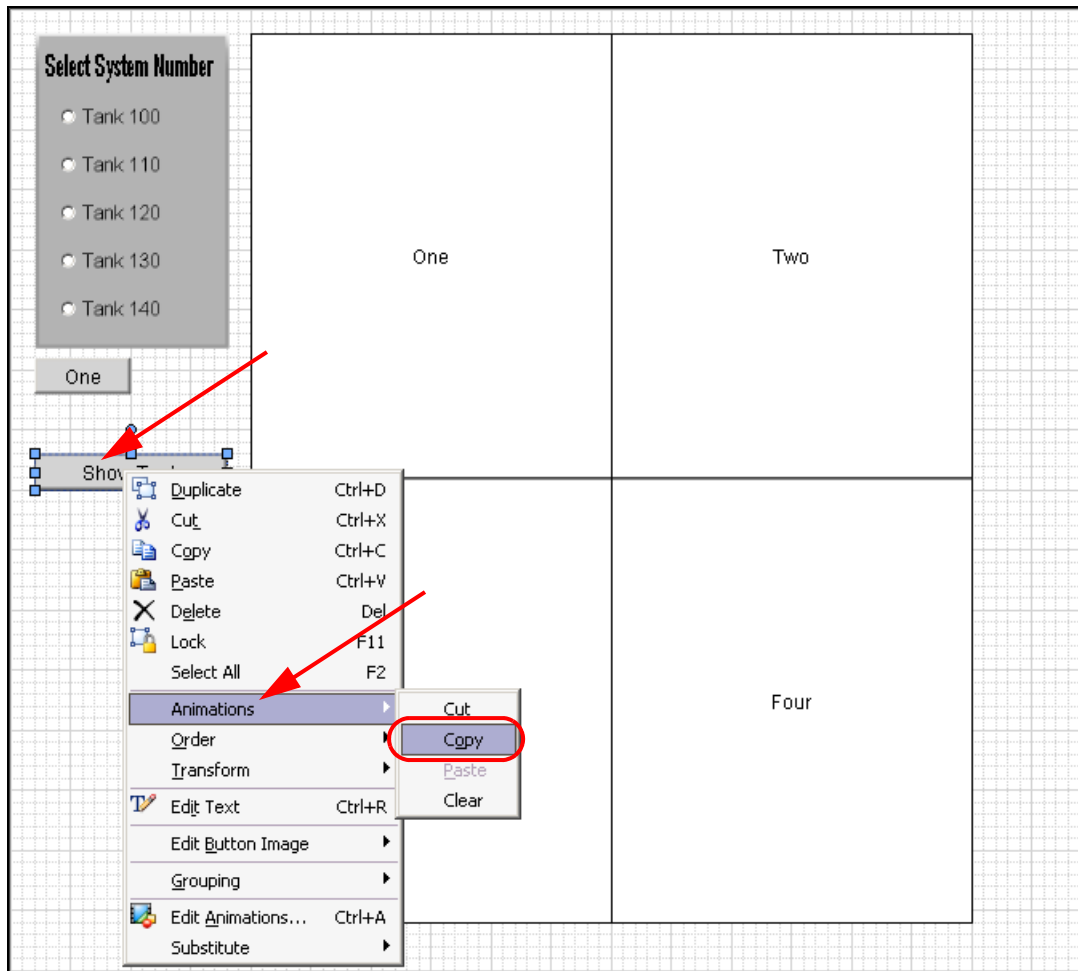
Note: Use **Ctrl+L** to change the text string in each box.

Add Scripts

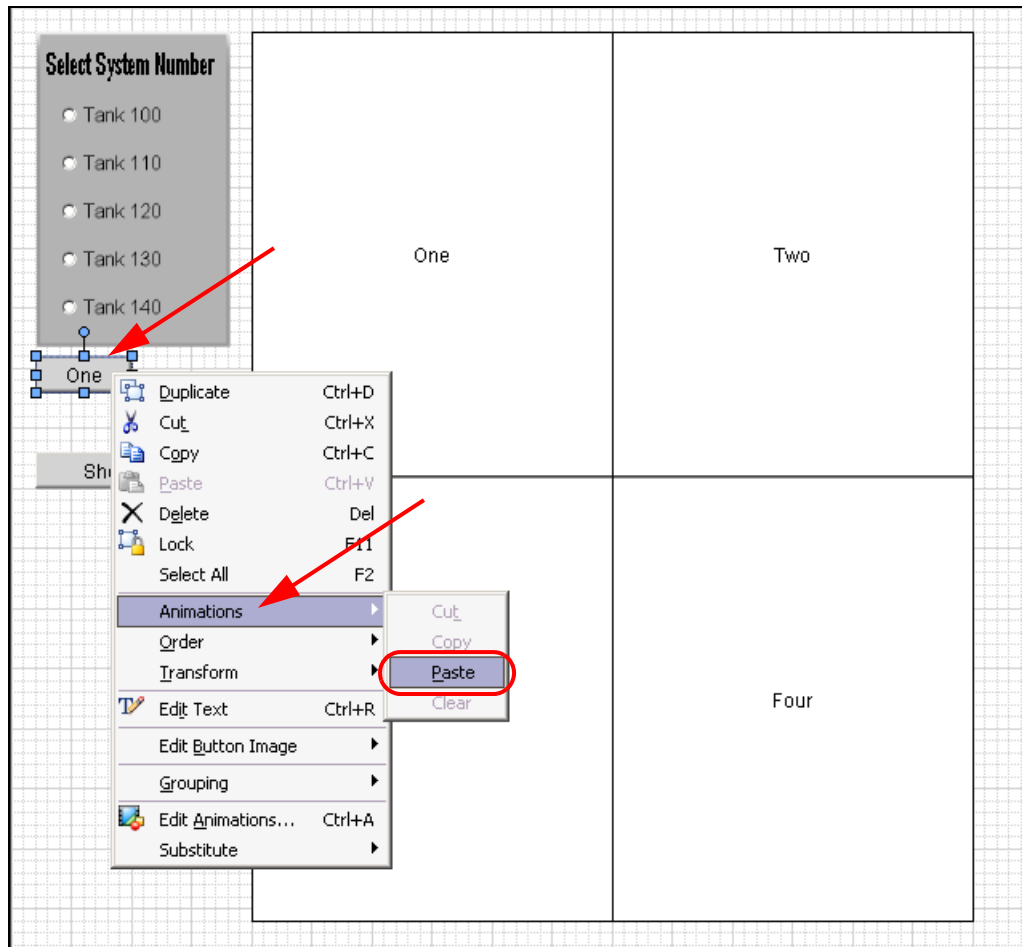
19. Move the **Show Tank** button down to make room for additional buttons, and then draw a button labeled **One**. In the right pane, rename it to **CallOne**.



20. Right-click **Show Tank** and select **Animations | Copy**.



21. Right-click the **One** button and select **Animations | Paste**.



22. Double-click the **One** button to open it for configuration
23. Copy the script from the file or manually alter the script as follows:

Note: All scripts for this lab are available to copy from the file **Lab 6 Script** in your **Training** directory.

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "one";
graphicInfo.GraphicName = "TankDisplay3";
```

```
graphicInfo.HasTitleBar = false;
graphicInfo.WindowLocation = 2;
graphicInfo.WindowRelativePosition = 8;
graphicInfo.RelativeTo = 2;
graphicInfo.X = One.X;
graphicInfo.Y = One.Y;
graphicInfo.Width = One.Width;
graphicInfo.height = One.Height;
```

```
Dim cpValues [5] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new aaGraphic.CustomPropertyValuePair("Tank.Value",
"InTouch:Tank" + Text(SystemNumber, "#")+ "\Vessel\Level", true);
```

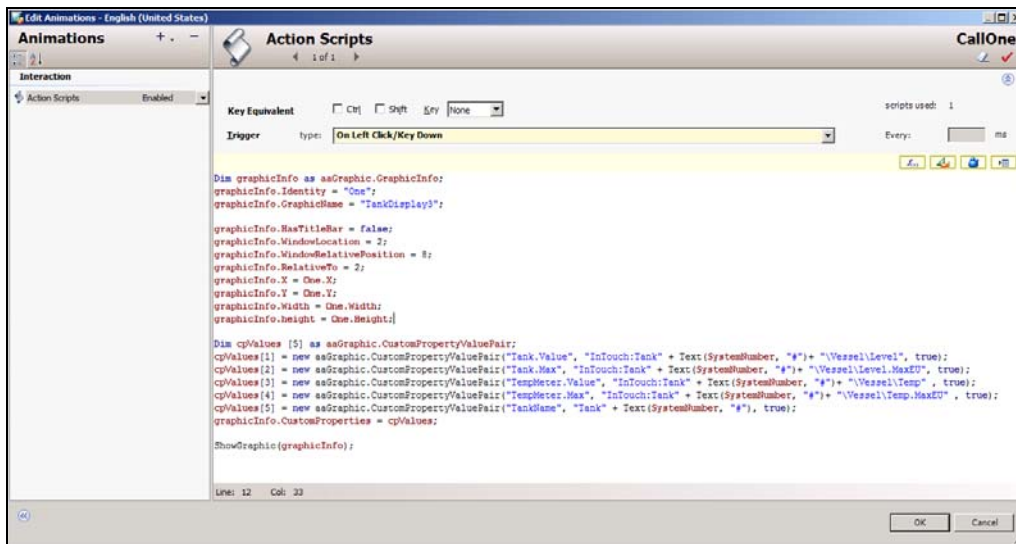
```

cpValues[2] = new aaGraphic.CustomPropertyValuePair("Tank.Max",
"InTouch:Tank" + Text(SystemNumber, "#")+ "\\Vessel\Level.MaxEU", true);
cpValues[3] = new aaGraphic.CustomPropertyValuePair("TempMeter.Value",
"InTouch:Tank" + Text(SystemNumber, "#")+ "\\Vessel\Temp", true);
cpValues[4] = new aaGraphic.CustomPropertyValuePair("TempMeter.Max",
"InTouch:Tank" + Text(SystemNumber, "#")+ "\\Vessel\Temp.MaxEU", true);
cpValues[5] = new aaGraphic.CustomPropertyValuePair("TankName", "Tank" +
Text(SystemNumber, "#"), true);
graphicInfo.CustomProperties = cpValues;

ShowGraphic( graphicInfo );

```

When you are finished, the script should look as follows:



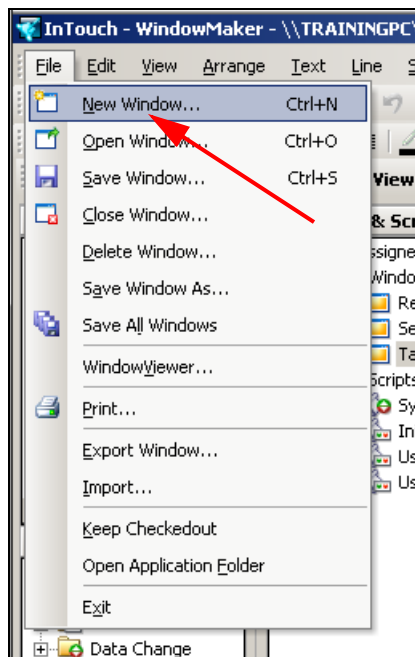
24. Click **OK**, and then click **Save**.

Create a New InTouch Window

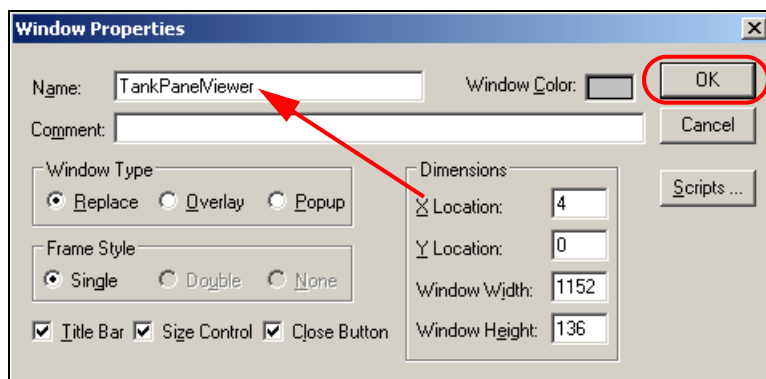
Next, create an InTouch window to test what you have built.

25. In WindowMaker, close all open windows.

26. On the **File** menu, click **New Window**.

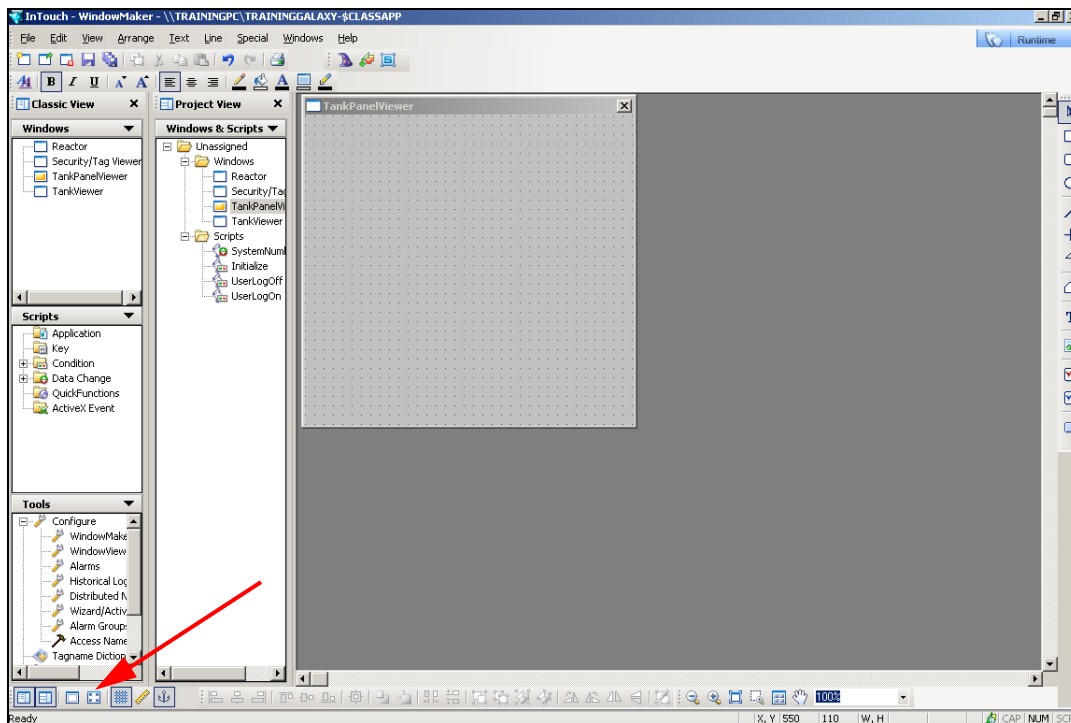


27. Name the new window **TankPanelViewer**, and then click **OK**.

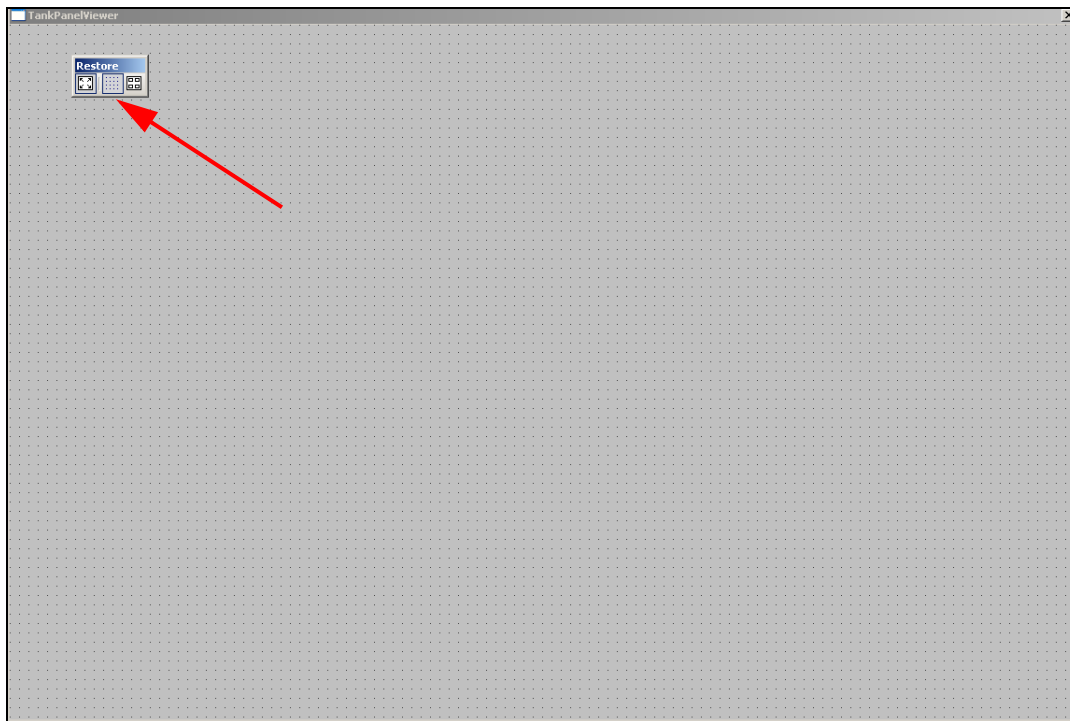


Note: The window size and location parameters shown here may vary from your display. There is no need to change them here as you will be changing them in the next steps.

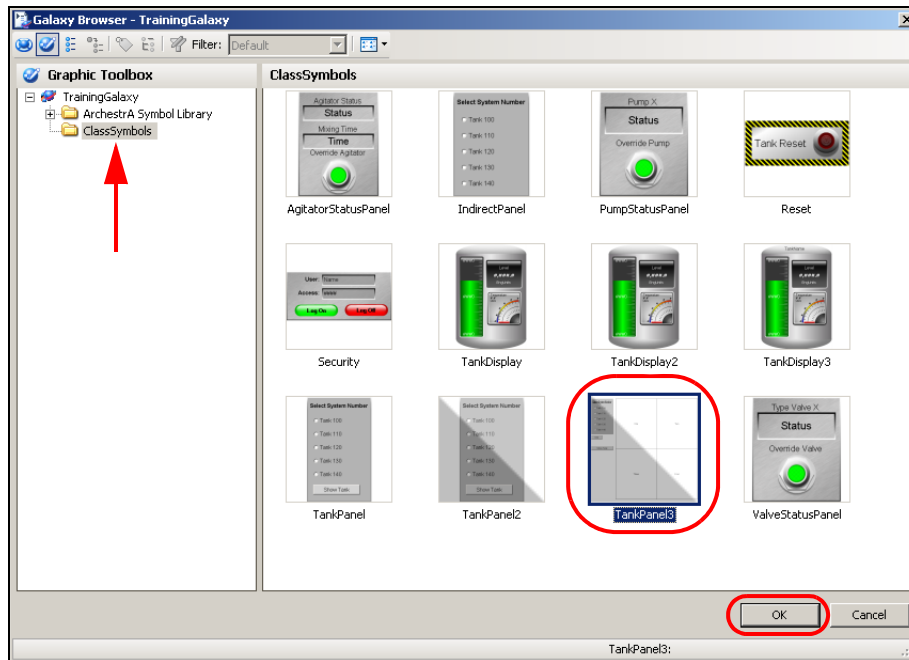
28. At the bottom-left of the screen, click **Fullscreen**.



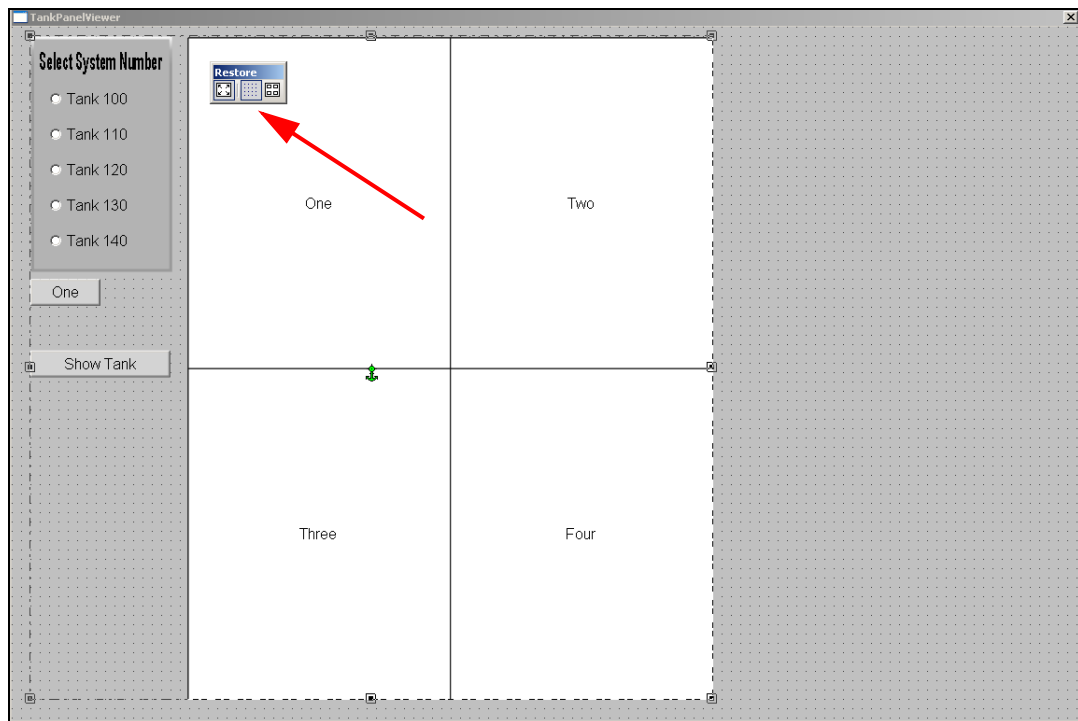
29. Enlarge the window to fill most of the available screen space, and then exit **Fullscreen**.



30. Click **Embed ArchestraA Graphic**.
31. In **ClassSymbols**, select **TankPanel3**, and then click **OK**.



32. At the top-left of the **TankPanelViewer** window, place **TankPanel3**.
33. Click **Fullscreen**.
34. Resize **TankPanel3** so that it fits properly in the window, and then exit **Fullscreen**.



Test What You Have Built

35. Click **Runtime**.
36. From the **System Number** panel, select **Tank100**, and then click **One** and verify that the symbol appears in the text box labeled **One** with the correct text label.
37. Repeat this testing for each of the tanks.

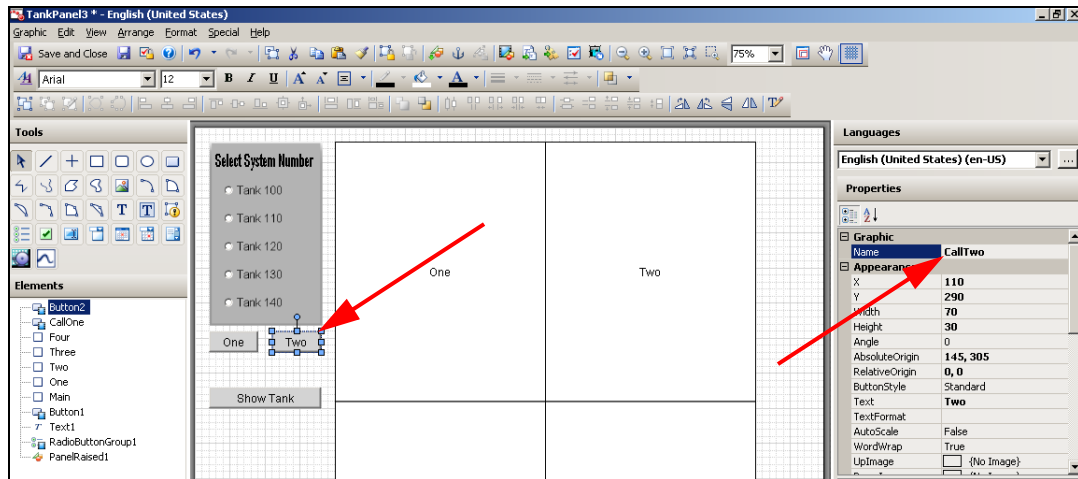
Note: For purposes of this testing only use button One. This is only a preliminary step and the scripting for the Show Tank button is not in place yet.

38. Click **Development!**.

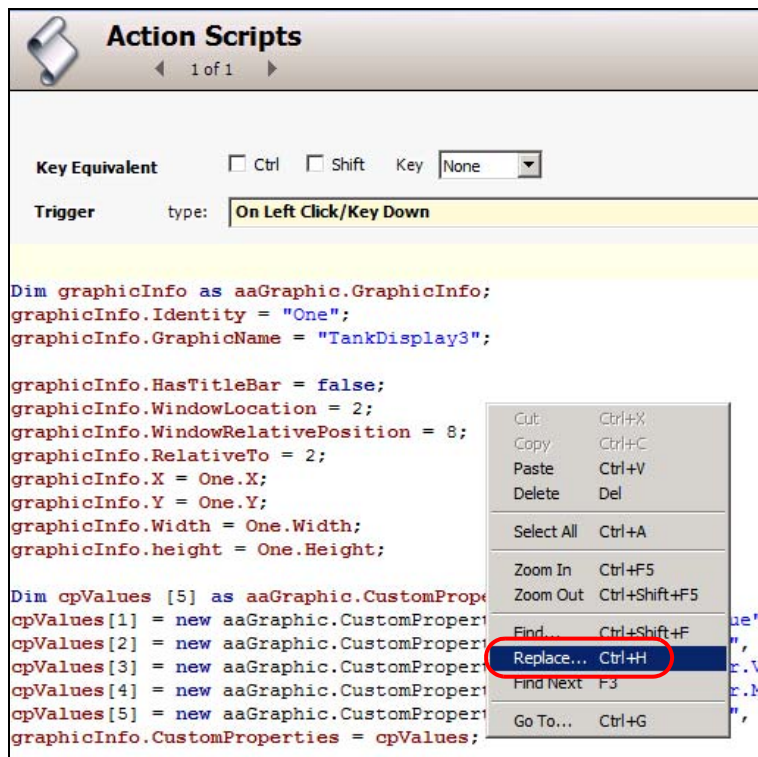
Add Buttons and Scripts for Additional Displays

Now that you have created the script for text box One and tested it, you will need to do the same for the remaining text boxes.

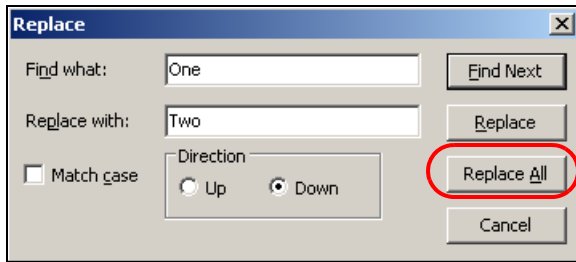
39. Return to **TankPanel3** in **ArchestrA Symbol Editor**.
40. Duplicate button **One**, and then press **Ctrl+L** to change the label to **Two**.
41. In the right pane, rename the button to **CallTwo**.



42. Double-click the **Two** button to edit the script.
43. In the script area, right-click and select **Replace**.



44. In the **Find what** field enter **One**, in the **Replace with** field enter **Two**, and then click **Replace All**.



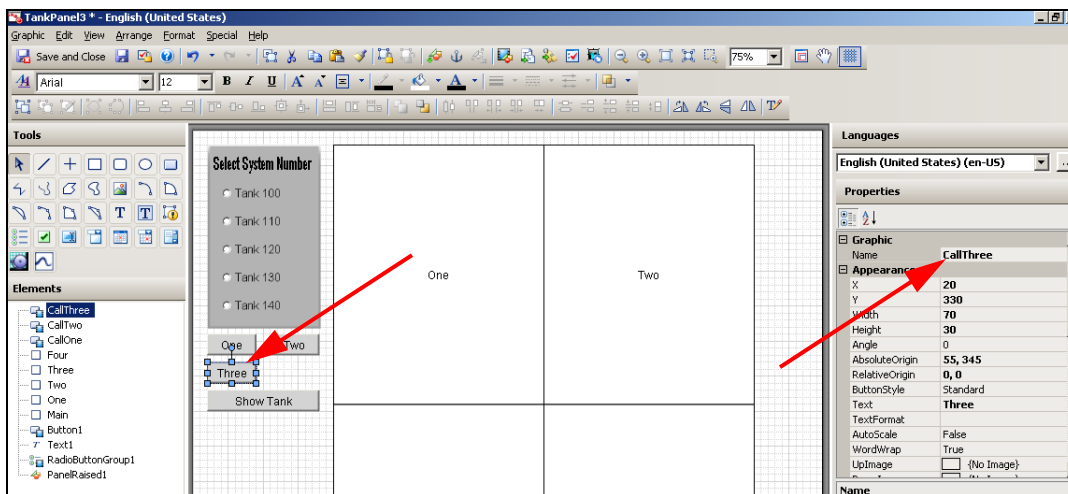
45. Close the **Replace** dialog box.

46. Click **OK**.

Now, you will follow the same process to create button Three.

47. Duplicate the **One** button, and then press **Ctrl+L** to change the label to **Three**.

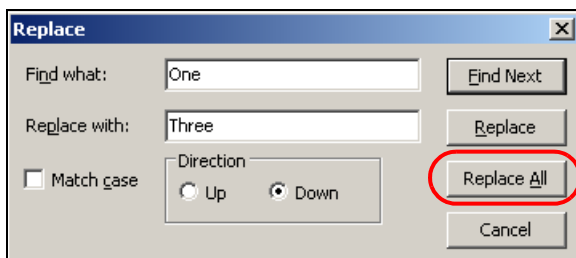
48. In the right pane, rename the button to **CallThree**.



49. Double-click the **Three** button to edit the script.

50. In the script area, right-click and select **Replace**.

51. In the **Find what** field enter **One**, in the **Replace with** field enter **Three**, and then click **Replace All**.



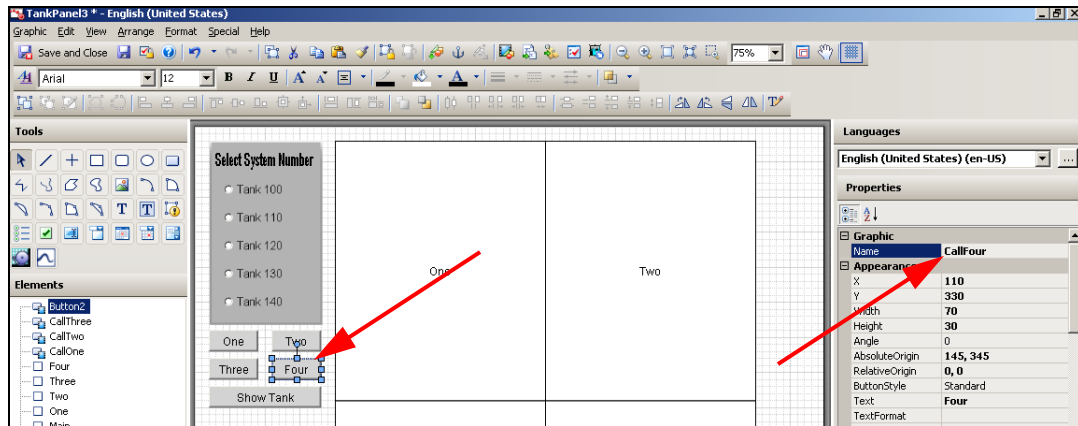
52. Close the **Replace** dialog box.

53. Click **OK**.

Once again follow the same process to create button Four.

54. Duplicate button **One**, and then press **Ctrl+L** to change the label to **Four**.

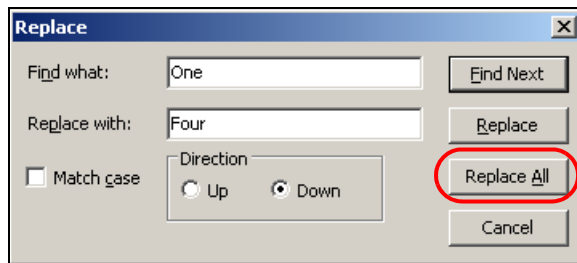
55. In the right pane, rename the button to **CallFour**.



56. Double-click the **Four** button to edit the script.

57. In the script area, right-click and select **Replace**.

58. In the **Find what** field enter **One**, in the **Replace with** field enter **Four**, and then click **Replace All**.

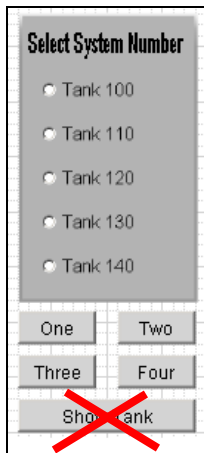


59. Close the **Replace** dialog box.

60. Click **OK**.

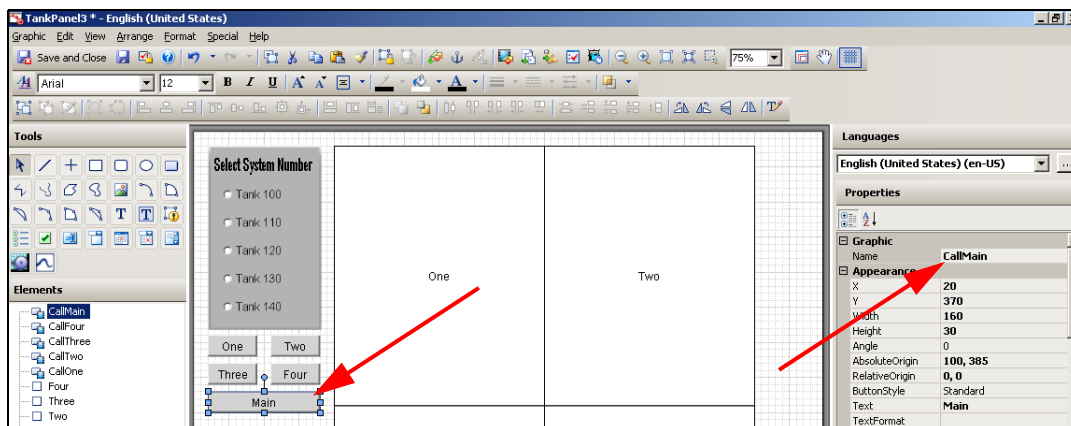
Lastly, you will need to create a button to call the symbol into the main display.

61. Delete the **Show Tank** button.



62. Duplicate button **One**, enlarge it, and then press **Ctrl+L** to change the label to **Main**.

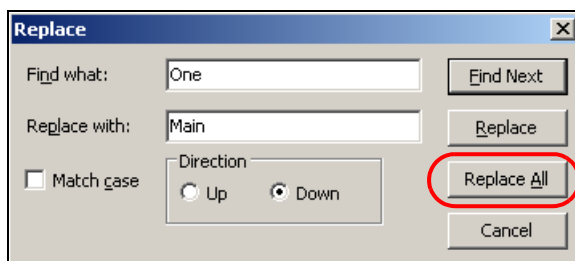
63. In the right pane, rename the button to **CallMain**.



64. Double-click the **Main** button to edit the script.

65. In the script area, right-click and select **Replace**.

66. In the **Find what** field enter **One**, in the **Replace with** field enter **Main**, and then click **Replace All**.



67. Close the **Replace** dialog box.

68. Click **OK**.

69. Click **Save**.

Test What You Have Built

70. In WindowMaker, click **Runtime**.
71. From the **System Number** panel, select **Tank100**, and then click **One** and verify that the symbol appears in the text box labeled **One** with the correct text label.
72. Repeat the same test for the remaining tanks, placing them as follows:

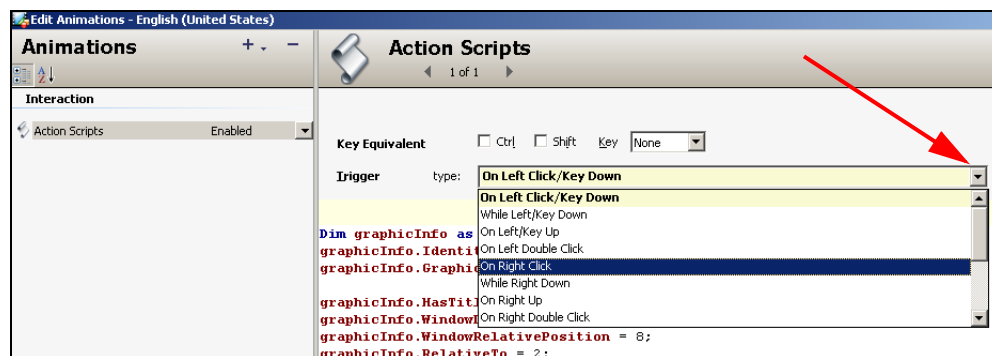
Tank #	Location
Tank110	Two
Tank120	Three
Tank130	Four
Tank140	Main

73. Click **Development!**.

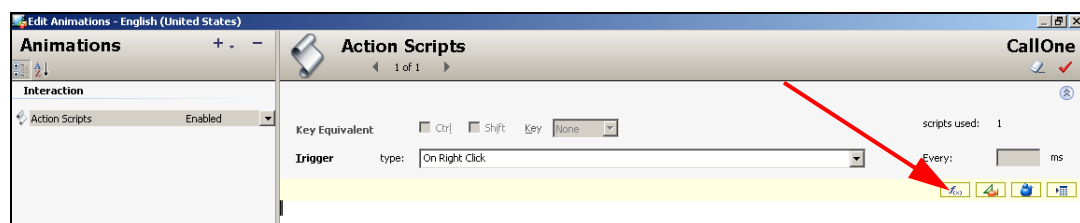
Add Hide Functionality

You may have noticed during your testing that the symbols cannot be closed once opened, to resolve this issue you need to add a hide command to each of the symbols.

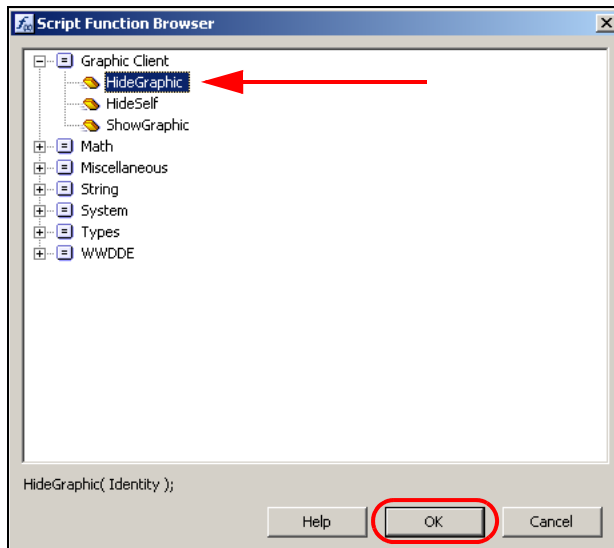
74. Return to **TankPanel3** in **Archestra Symbol Editor**.
75. Double-click the **One** button to open it for editing.
76. From the **Trigger** drop-down, select **On Right Click**.



77. Click **Function Browser**.

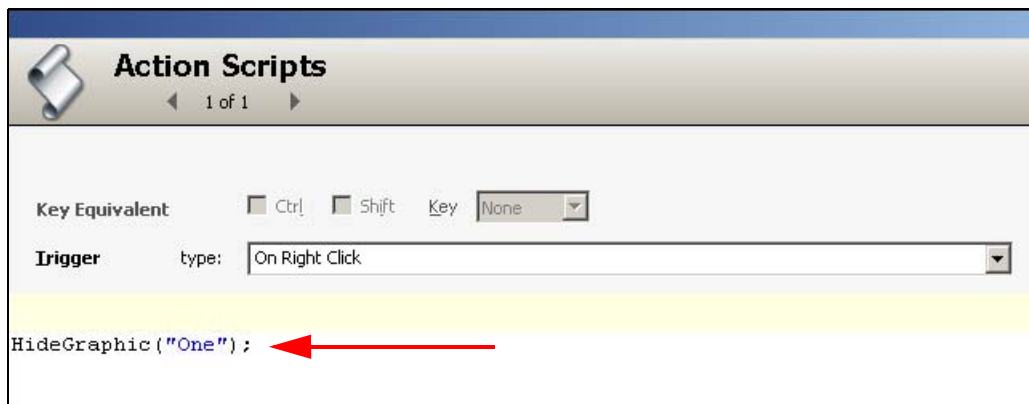


78. Expand **Graphic Client**, select **HideGraphic**, and then click **OK**.



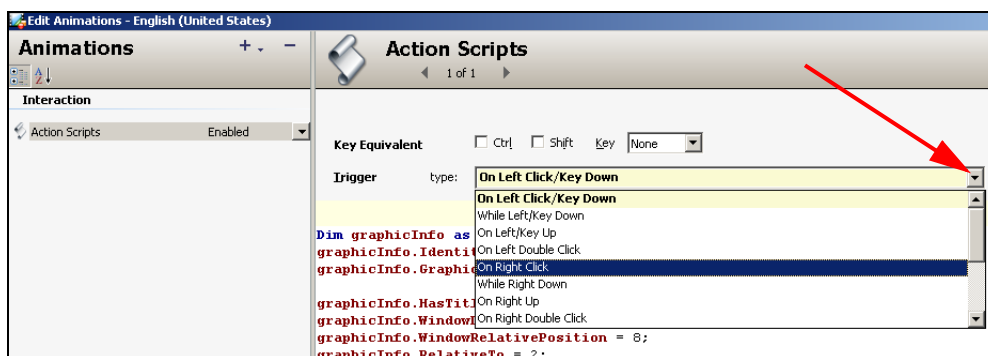
This will place the following script in the script field:
HideGraphic (Identity);

79. Replace **Identity** with “One”, and then click **OK**.

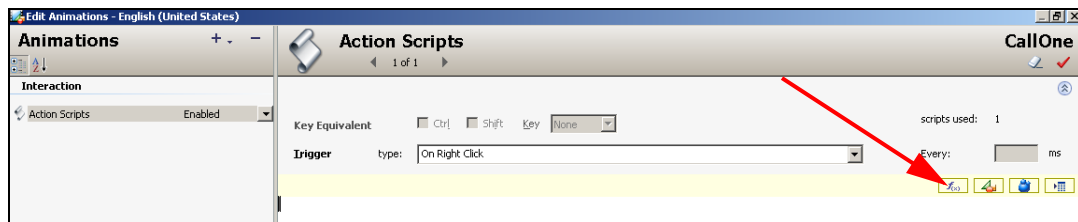


80. Double-click the **Two** button to open it for editing.

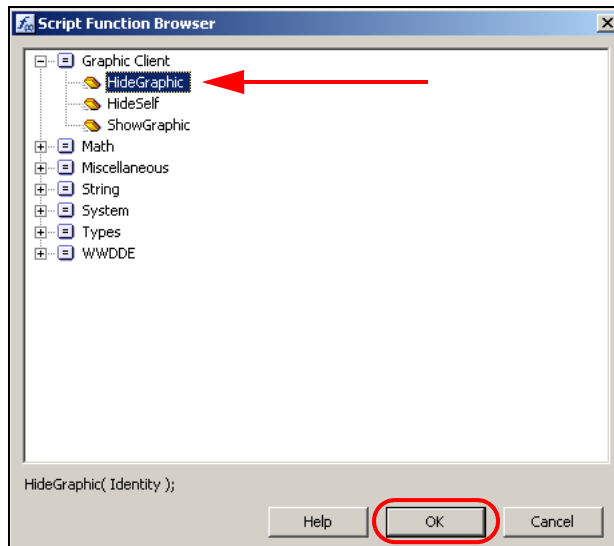
81. From the **Trigger** drop-down, select **On Right Click**.



82. Click **Function Browser**.

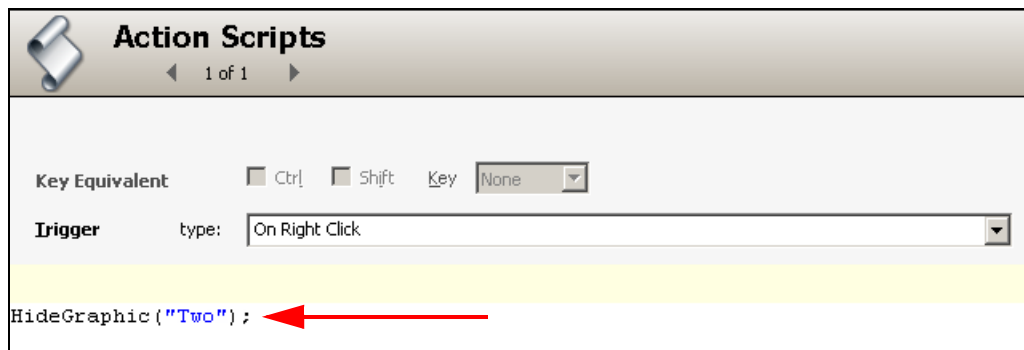


83. Expand **Graphic Client**, select **HideGraphic**, and then click **OK**.



This will place the following script in the script area:
HideGraphic (Identity);

84. Replace **Identity** with “Two”, and then click **OK**.



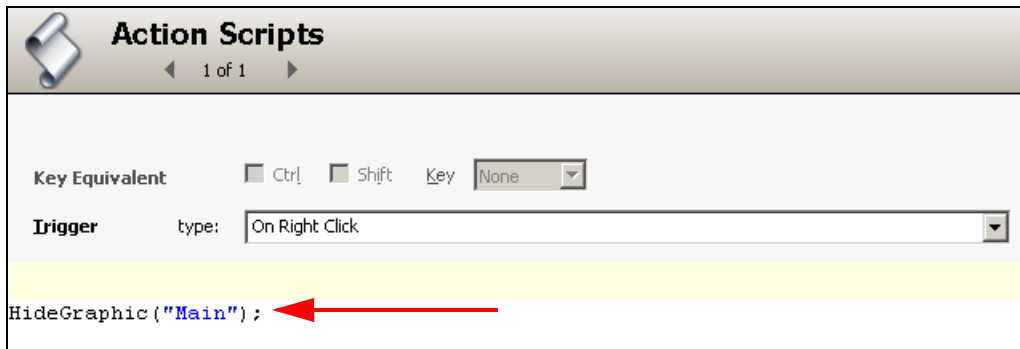
85. Repeat this process for buttons **Three** and **Four**, substituting the word **Identity** with “Three” and “Four” respectively.

86. Double-click the **Main** button to open it for editing.

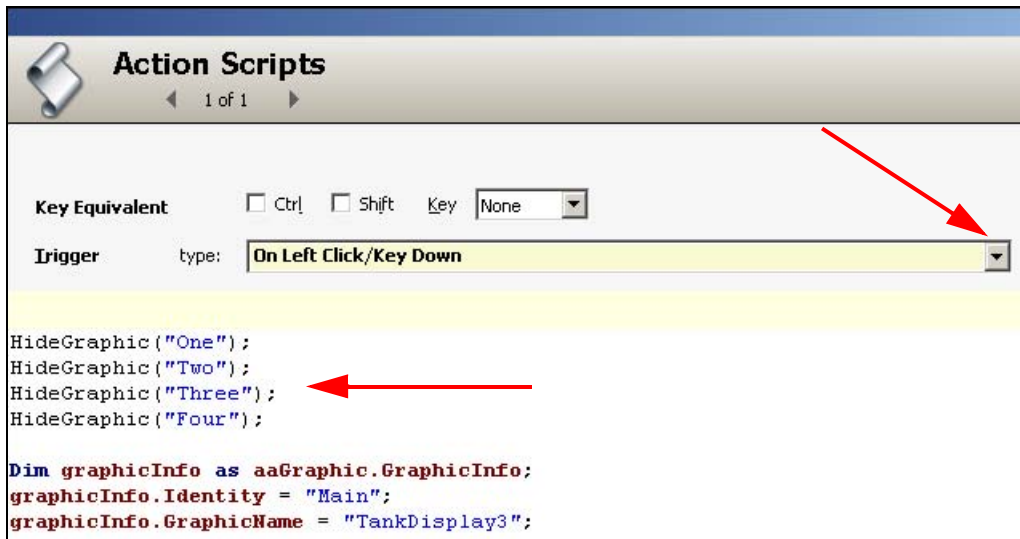
87. From the **Trigger** drop-down, select **On Right Click**, and then click **Function Browser**.

88. Expand **Graphic Client**, select **HideGraphic**, and then click **OK**.

89. In the script area, delete the word **Identity** and replace with “Main”.



90. From the **Trigger** drop-down, select **On Left Click/Key Down**.
91. At the very top of the script, add the following script to hide all of the other symbols:
- ```
HideGraphic("One");
HideGraphic("Two");
HideGraphic("Three");
HideGraphic("Four");
```



---

**Note:** The **Main** button requires this additional script so that you do not have unnecessary symbols open behind it utilizing system resources.

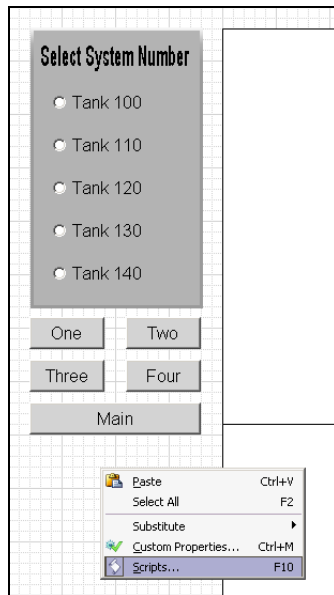
---

92. Click **OK**.



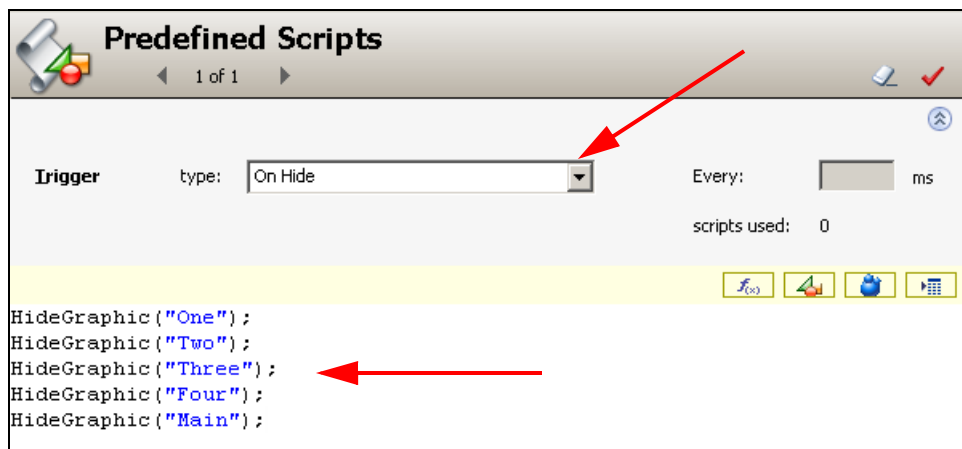
Lastly, make sure all associated symbols close if the window is closed. To do this, you will add a predefined On Hide script.

93. On the canvas, right-click and select **Scripts**.



94. From the **Trigger** drop-down, select **On Hide**, and then enter the following script:

```
HideGraphic("One");
HideGraphic("Two");
HideGraphic("Three");
HideGraphic("Four");
HideGraphic("Main");
```

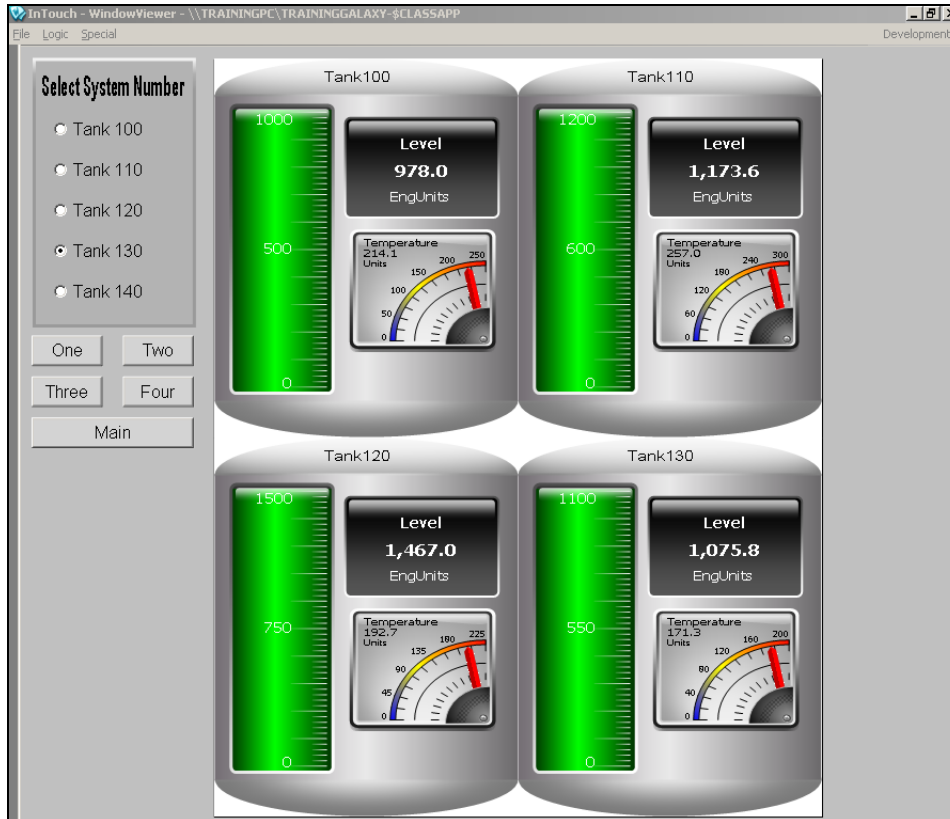


95. Click **OK**.

96. Click **Save and Close**, and then check in the symbol.

## Test What You Have Built

97. In WindowMaker, click **Runtime**.
98. Test each tank and button to ensure they are working as designed, displaying in the correct area, and hiding when right-clicking on each of the buttons.



99. Click **Development!**.



invenys  
Learning Services



## Module 4 – Enhanced Application Development

---

|                                                                  |      |
|------------------------------------------------------------------|------|
| Section 1 – Create Graphic Effects with Animation and Scripting  | 4-3  |
| Lab 7 – Drawing a Pulsating Light                                | 4-7  |
| Section 2 – Grow/Shrink Functionality                            | 4-23 |
| Lab 8 – Creating Re-Sizeable Symbols                             | 4-25 |
| Section 3 – Modify and Enhance Existing Symbols                  | 4-39 |
| Lab 9 – Modifying and Enhancing Existing Symbols                 | 4-41 |
| Section 4 – ActiveX Controls                                     | 4-57 |
| Lab 10 – Using an ActiveX Control as a Universal Document Viewer | 4-61 |
| Section 5 – .NET Controls                                        | 4-71 |
| Lab 11 – Using .Net Controls                                     | 4-75 |

**Module Objectives**

- Enhance existing symbols with scripting, animations, and graphic modifications
- Build an animation using a truth table
- Utilize ActiveX and .NET functionality

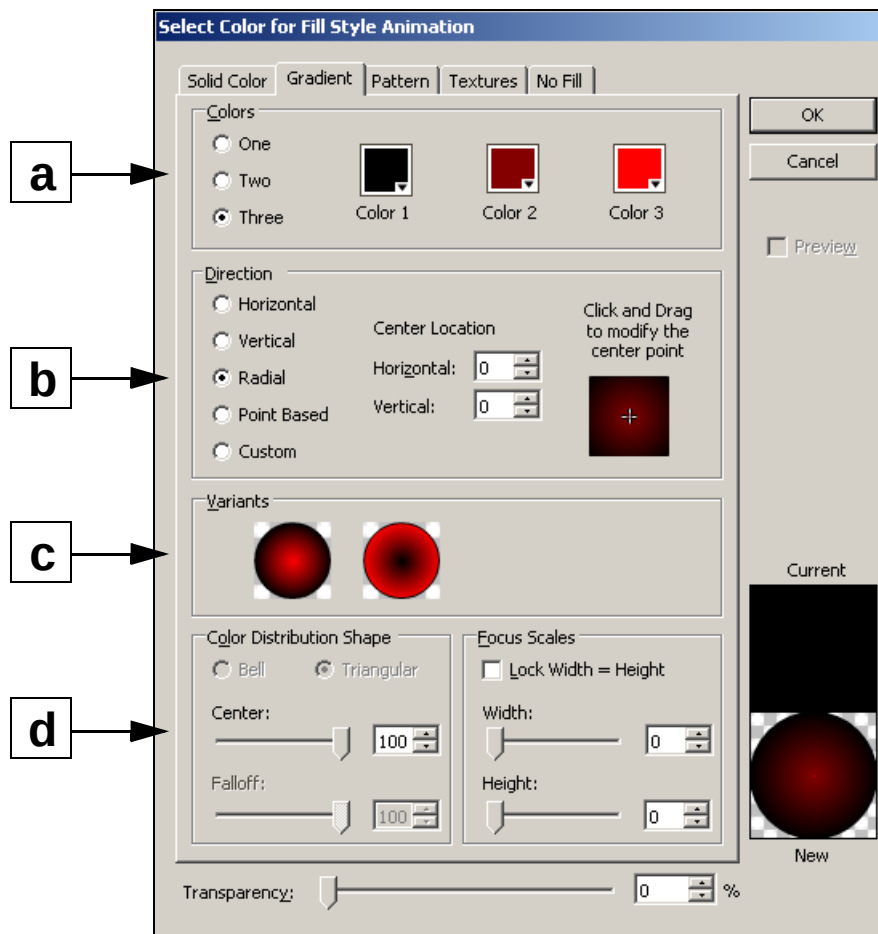
## Section 1 – Create Graphic Effects with Animation and Scripting

This section discusses the drawing tools and scripting necessary to create a pulsation effect in an ArchestrA symbol.

### Color Gradients

Color gradients create an illusion of depth and lighting, making them the foundation of creating a pulsating effect. Gradients are configured on the Gradient tab in the Style Selection dialog box, and have the following parameters:

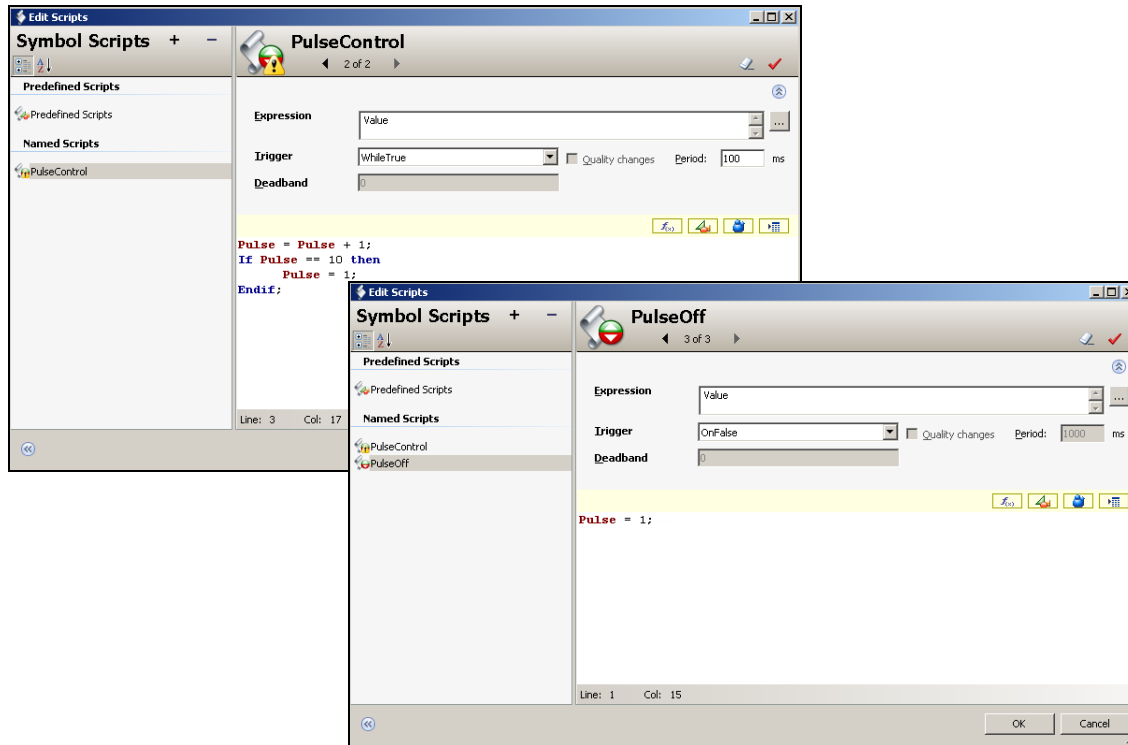
- a. **Colors** - selection number1, 2, or 3.
- b. **Direction** - horizontal, vertical, radial, point based, or customized.
- c. **Variant** - depending on your selection for the number of colors and direction.
- d. **Color Distribution Shape** - bell or triangular with options to configure the center and falloff.



## Symbol Scripts

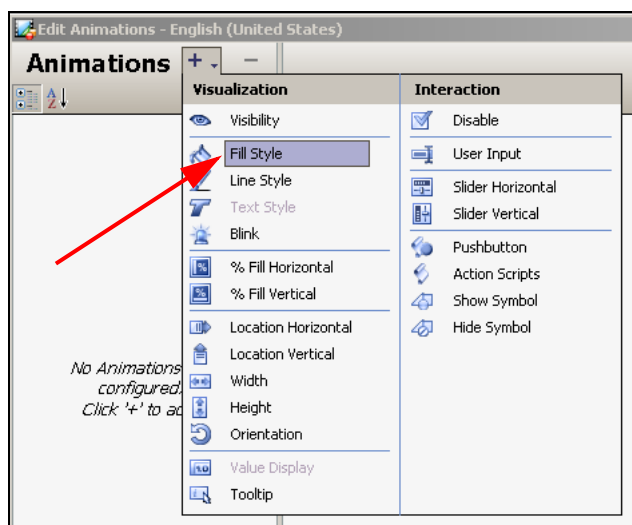
While gradients create the visual representation of pulsation, symbol scripting will be used to generate a series of values that a Fill Style animation will use to drive the pulsation effect.

One script is used to increment the value and another to turn it off and reset it to 1.

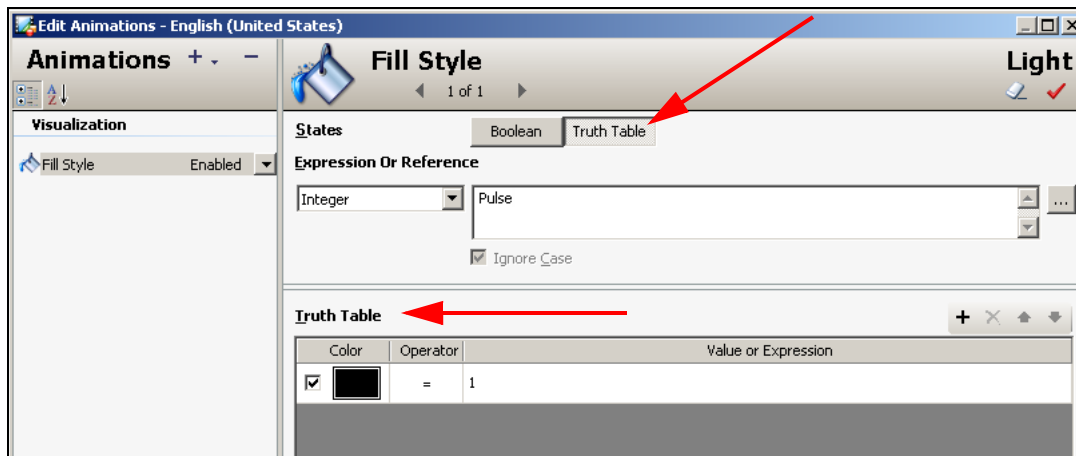


## Fill Style Animation

With the script in place, the gradient effects are combined with a Fill Style animation to create the pulsating light effect.



The animation itself is driven by a truth table of values, which is configured in the Edit Animations window for Fill Style.



A truth table assigns a fill style (in this case a specific gradient) to the value of the assigned property, which is based upon the symbol script created earlier.

| Truth Table                         |          |                     |  |
|-------------------------------------|----------|---------------------|--|
| Color                               | Operator | Value or Expression |  |
| <input checked="" type="checkbox"/> | =        | 1                   |  |
| <input checked="" type="checkbox"/> | =        | 2                   |  |
| <input checked="" type="checkbox"/> | =        | 3                   |  |
| <input checked="" type="checkbox"/> | =        | 4                   |  |
| <input checked="" type="checkbox"/> | =        | 5                   |  |
| <input checked="" type="checkbox"/> | =        | 6                   |  |
| <input checked="" type="checkbox"/> | =        | 7                   |  |
| <input checked="" type="checkbox"/> | =        | 8                   |  |
| <input checked="" type="checkbox"/> | =        | 9                   |  |
| <input checked="" type="checkbox"/> | =        | 10                  |  |





# Lab 7 – Drawing a Pulsating Light

## Introduction

ArchestraA Symbol Library contains several symbols with a light feature, but none of them pulsate. Instead, they are set to an on or off state. In this lab; however, you will use a truth table animation and script to get around this and create your own pulsating light symbol.

## Objectives

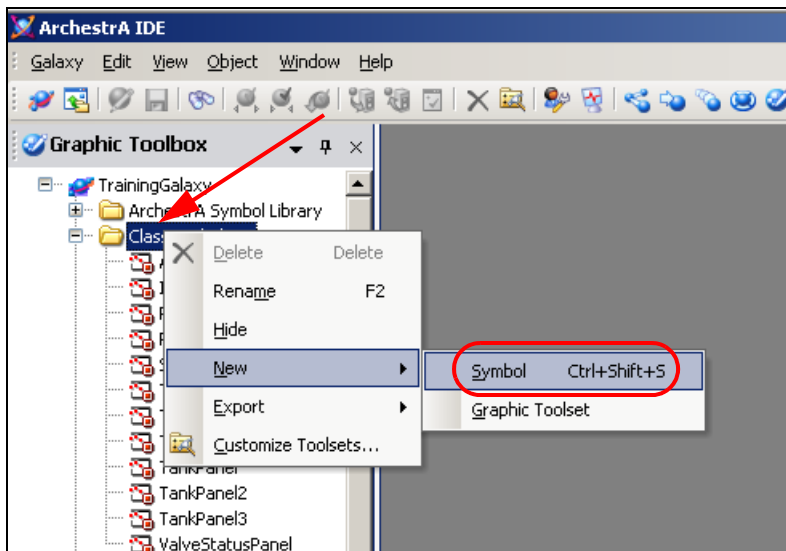
Upon completion of this lab, you will be able to:

- Draw a light symbol
- Add script to control light pulsation
- Add a color truth table to a light element for pulsating effect

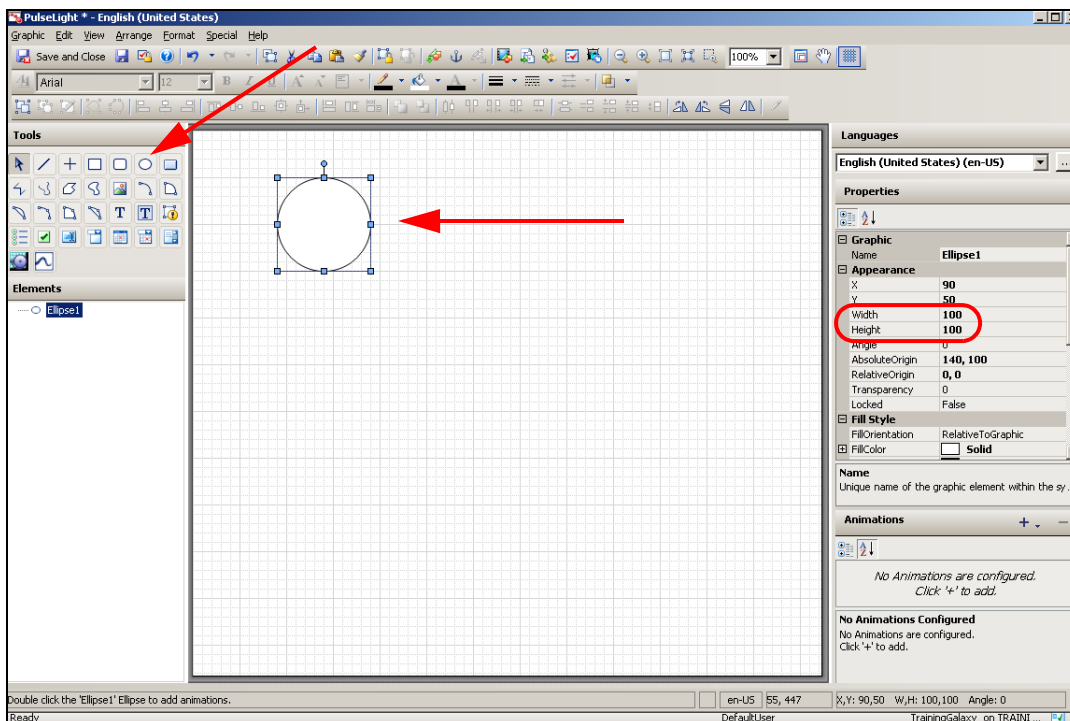
## Create a Pulsating Light

One of the most useful effects for drawing attention to an object is to give it a pulsating light effect.

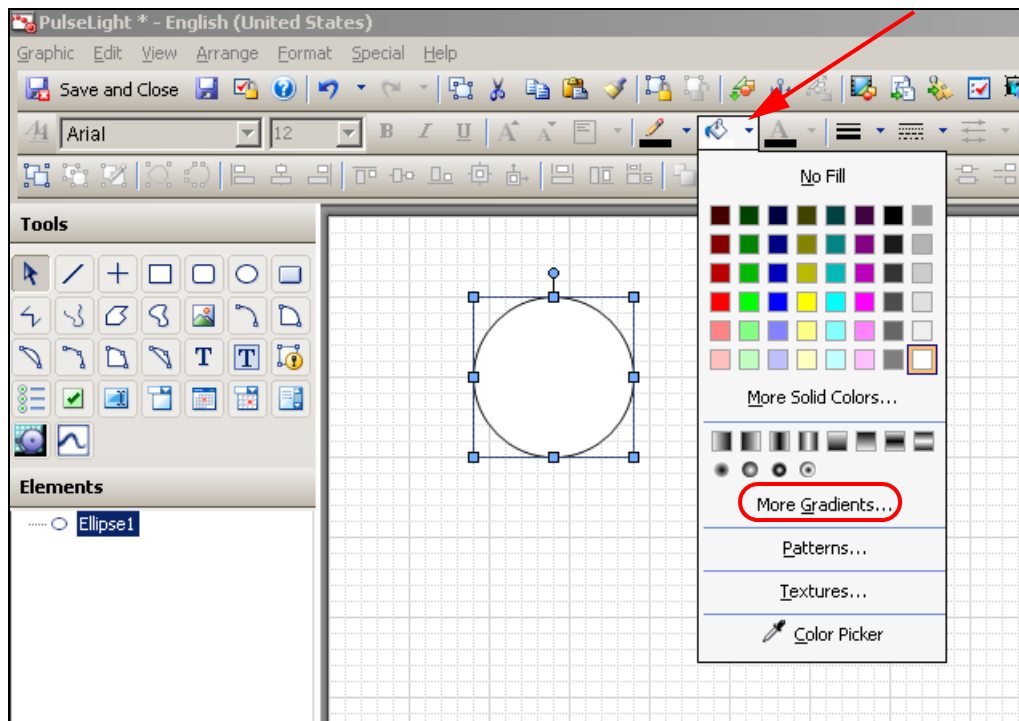
1. In Archestra IDE, right-click **ClassSymbols** and select **New | Symbol**.



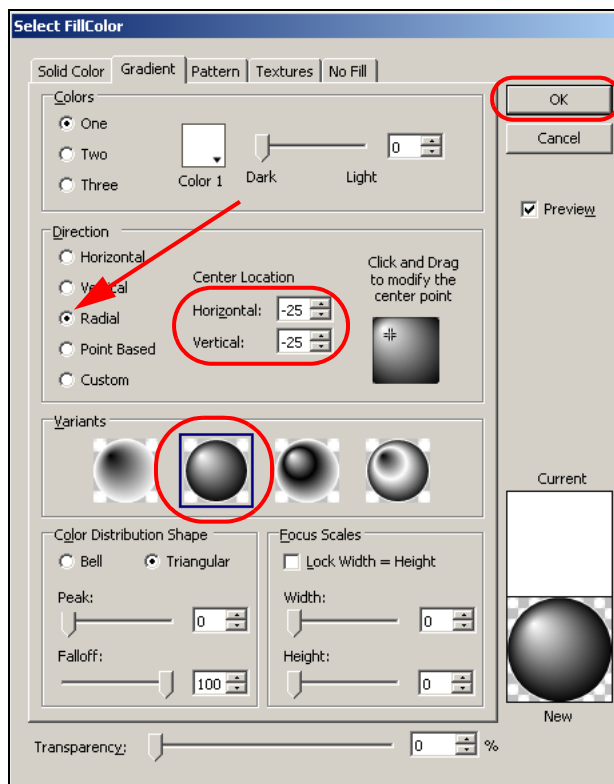
2. Name the new symbol **PulseLight**, and then double-click it for editing.
3. Draw a circle with a **Width** and **Height** of **100**.



- Click **Fill Color**, and then select **More Gradients**.

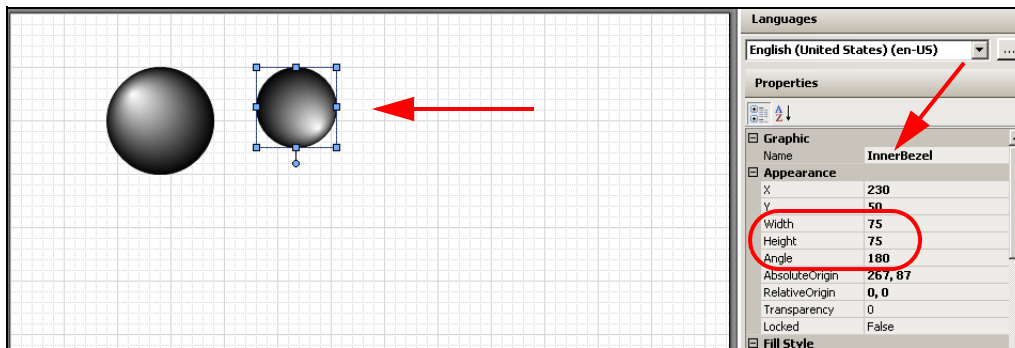


- Select a **Radial** gradient for both **Horizontal** and **Vertical** with a **Center Location** of **-25**, under **Variants** select the second option, and then click **OK**.

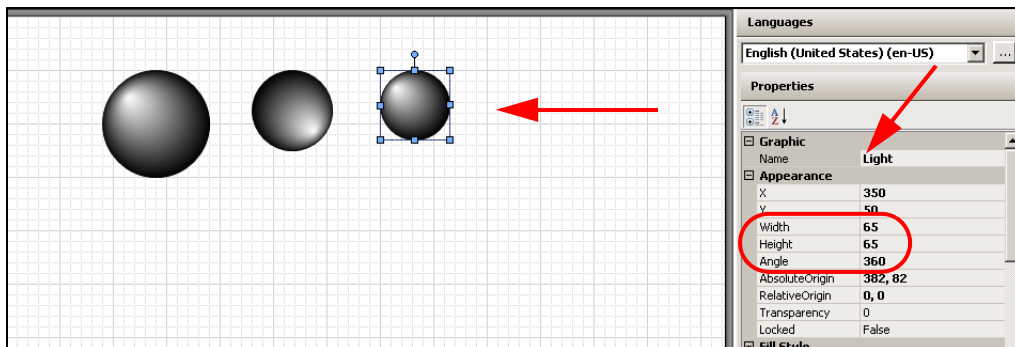


- Name the element **OuterBezel**.

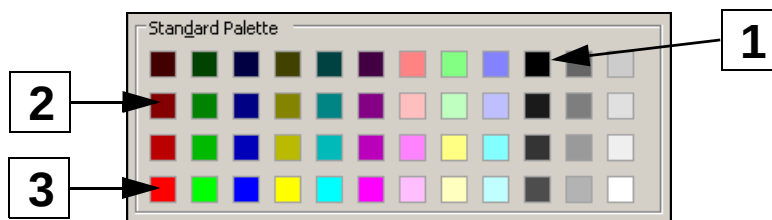
7. Duplicate **OuterBezel** and give it a **Width** and **Height** of **75**, an **Angle** of **180°**, and then name it **InnerBezel**.



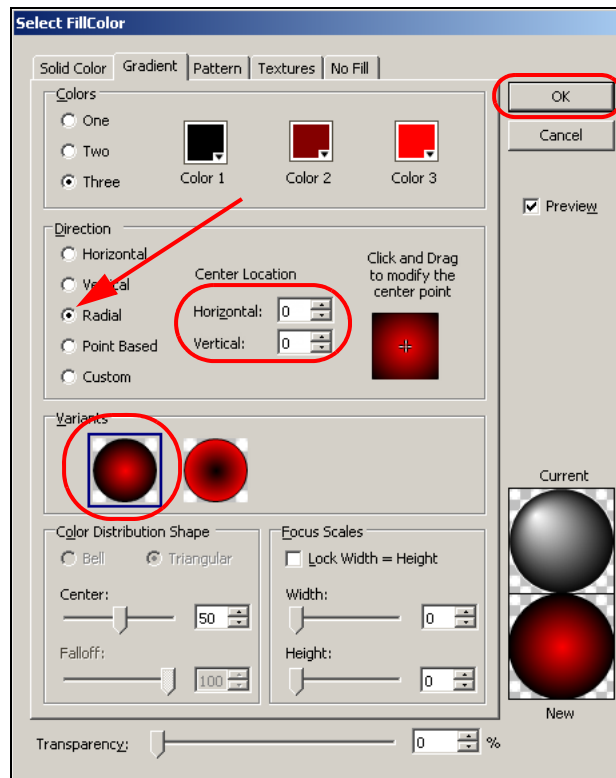
8. Duplicate **InnerBezel** and give it a **Width** and **Height** of **65**, an **Angle** of **360°**, and then name it **Light**.



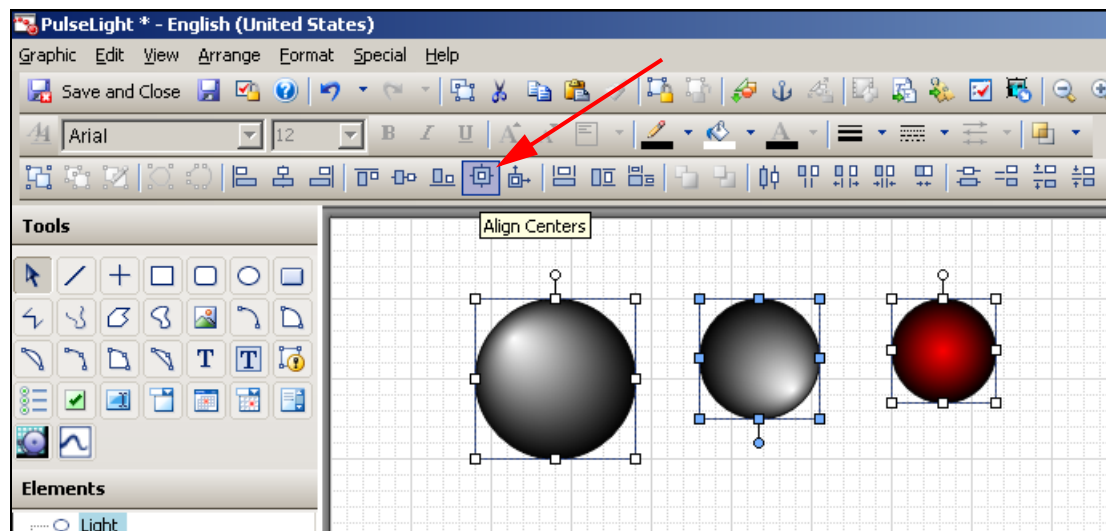
9. Click **Fill**, and then select **More Gradients**.
10. Select a **Three** color fill, with **Color 1** being black, **Color 2** dark red, and **Color 3** bright red.



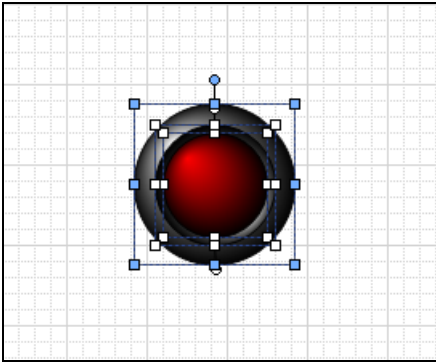
11. Select a **Radial** gradient for both **Horizontal** and **Vertical** with a **Center Location** of 0, under **Variants** select the first option, and then click **OK**.



12. Select all three elements, and then click **Align Centers**.

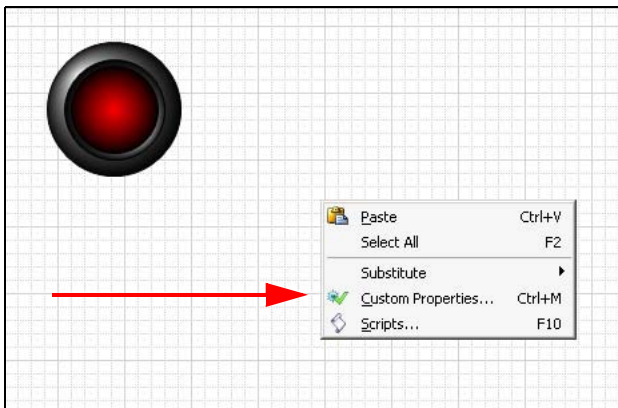


This will layer the objects on top of each other.

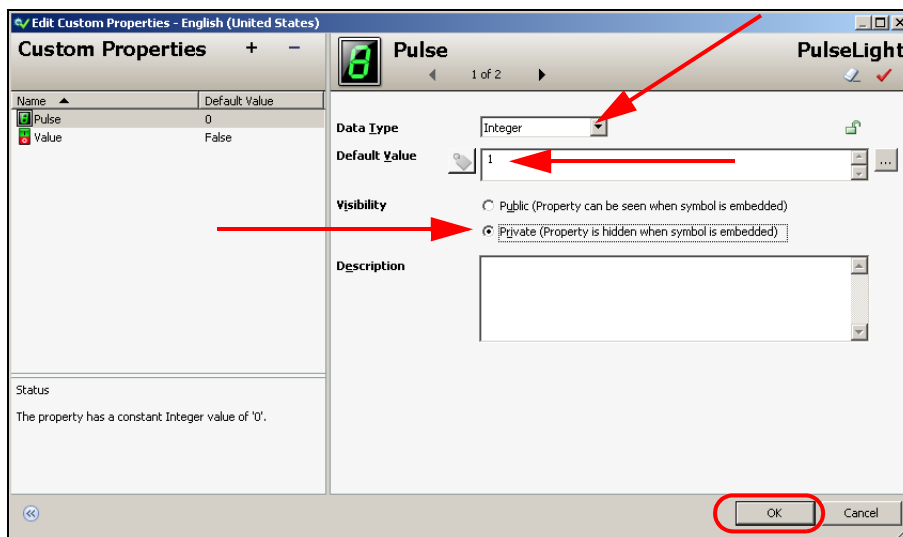


Now add two custom properties.

13. On the canvas, right-click and select **Custom Properties**.

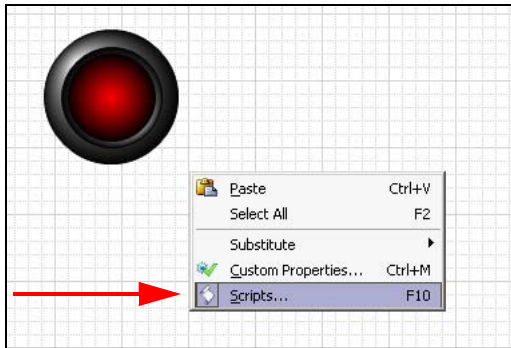


14. Click **+** and name the new custom property **Value**, and ensure the visibility is set to **Public**.
15. Click **+** and name the new custom property **Pulse**.
16. Select **Integer** as the **Data Type**, set **Default Value** to **1**, the **Visibility** to **Private**, and then click **OK**.



Next, add scripting to control the pulsation.

17. On the canvas, right-click and select **Scripts**.

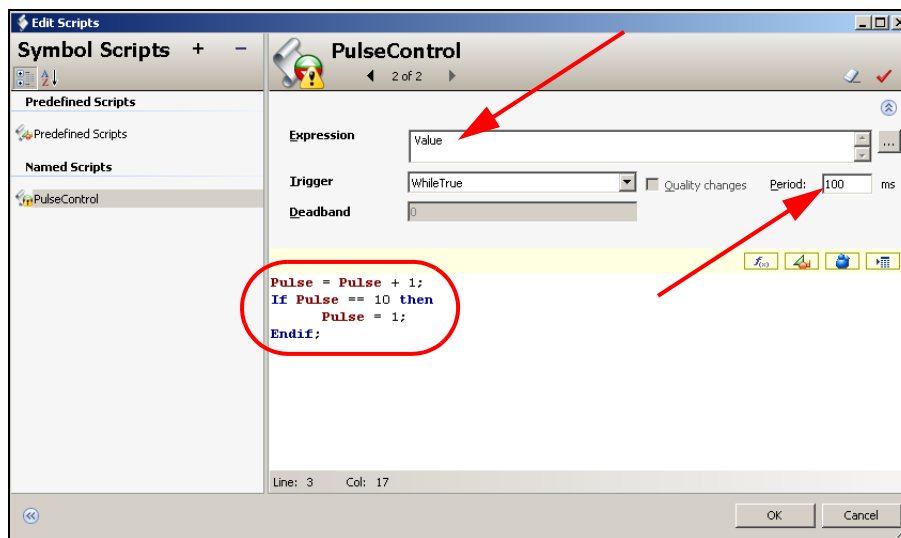


18. Click + and name the new script **PulseControl**.

19. In the **Expression** field, enter **Value**, and then set the **WhileTrue Period** to **100 ms**.

20. Enter the following script:

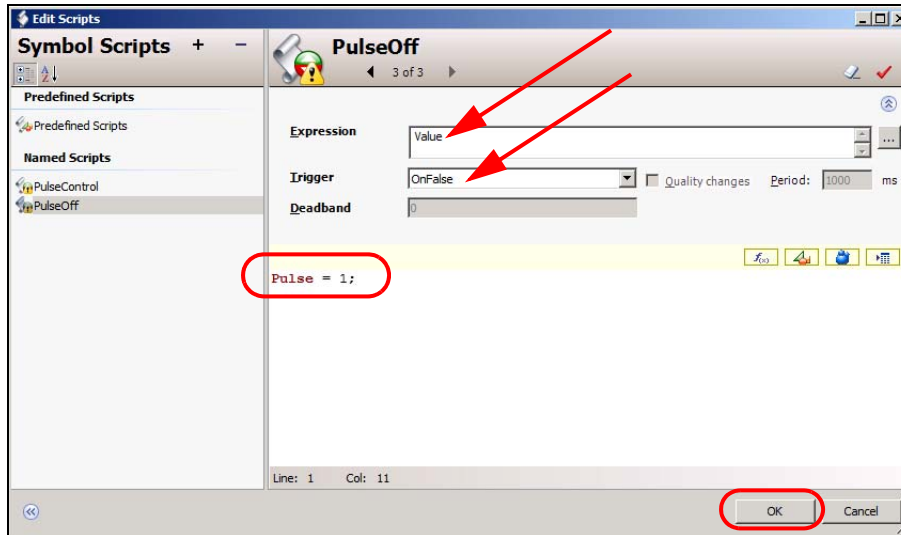
```
Pulse = Pulse + 1;
If Pulse == 10 then
 Pulse = 1;
Endif;
```



## 4-14 Module 4 – Enhanced Application Development

---

21. Click **+** and name the new script **PulseOff**.
22. In the **Expression** field, enter **Value**, and then set the **Trigger** to **OnFalse**.
23. Enter the following script:  
**Pulse = 1;**



Whenever the **Value** custom property goes off, the light control will go back to its starting value.

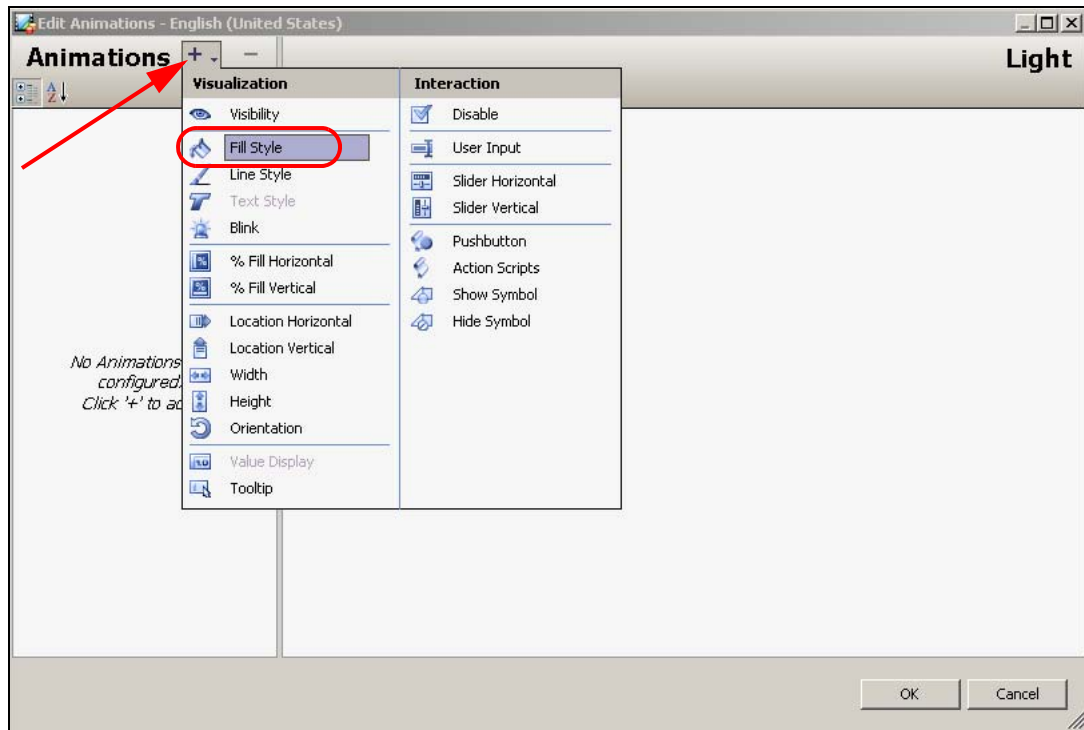
24. Click **OK**.



Next, using the pulse integer you incremented in the script, you will add animation to the light to give it a pulsating effect. This is done by adjusting the color of the light based on the value of the custom property.

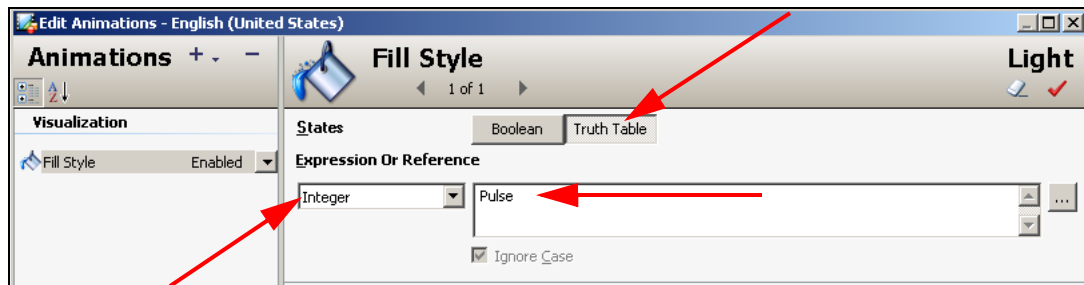
25. Double-click the **Light** element to add animation.

26. Click **+** and select **Fill Style**.

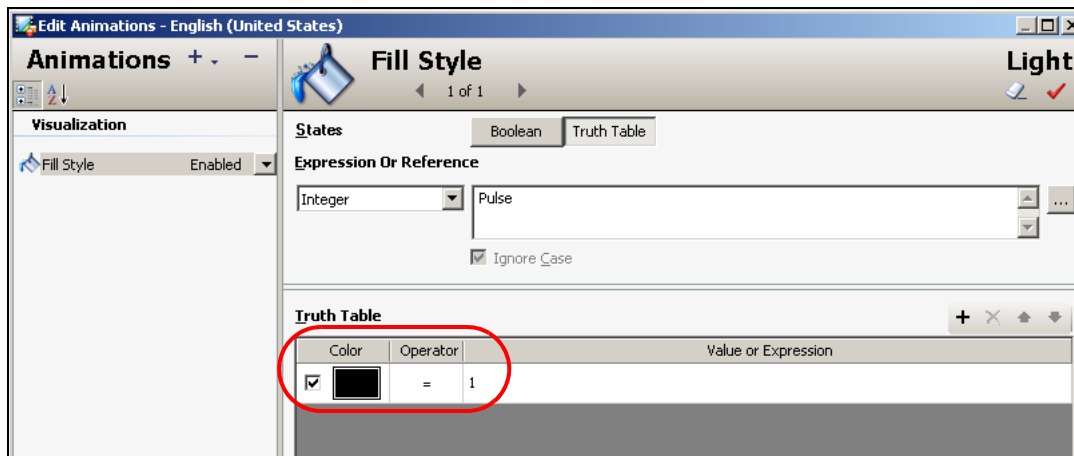


27. Click **Truth Table**.

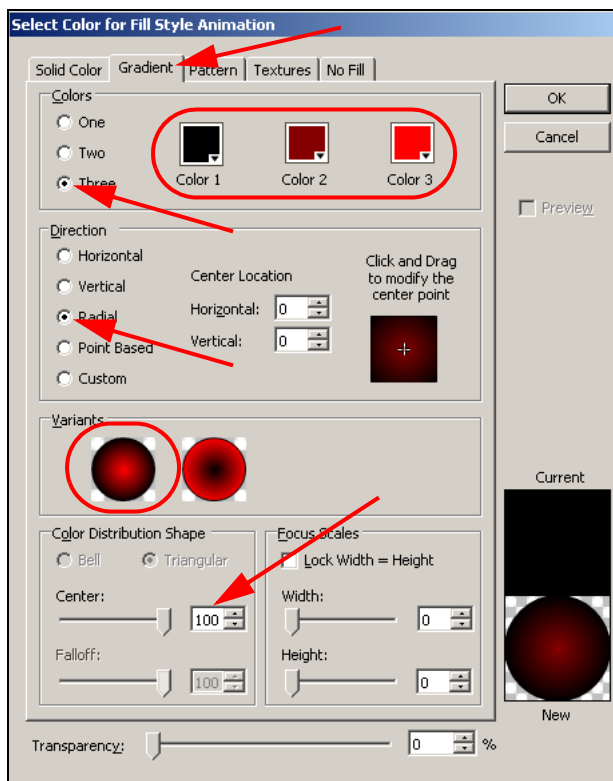
28. In the **Expression Or Reference** area, select **Integer** as the data type, and then enter **Pulse**.



29. In the **Truth Table** area, select the = sign for the **Operator** and **1** as the value, and then click on the black square to set the color.

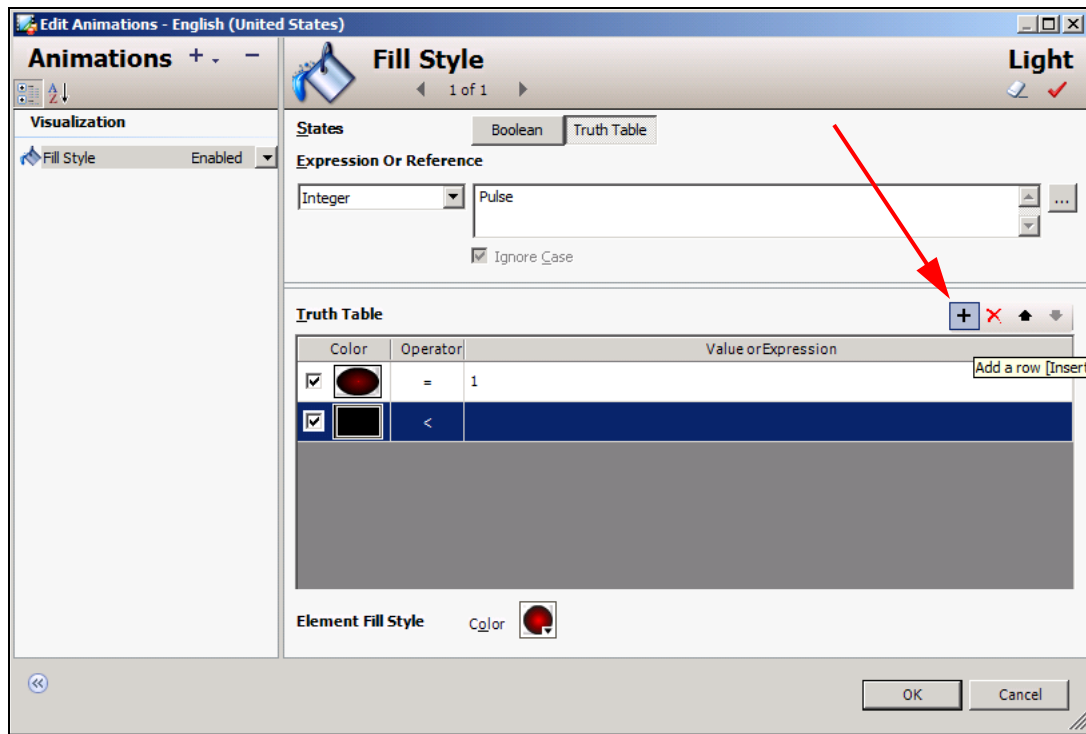


30. Click **Gradient**, select **Three** colors (set them to black, dark red, and bright red in that order), select **Radial** direction, the first **Variant**, and a **Center** distribution of **100**.



31. Click **OK**.

32. In the **Truth Table** area, click + to add additional colors (you will need to add and configure nine more colors).

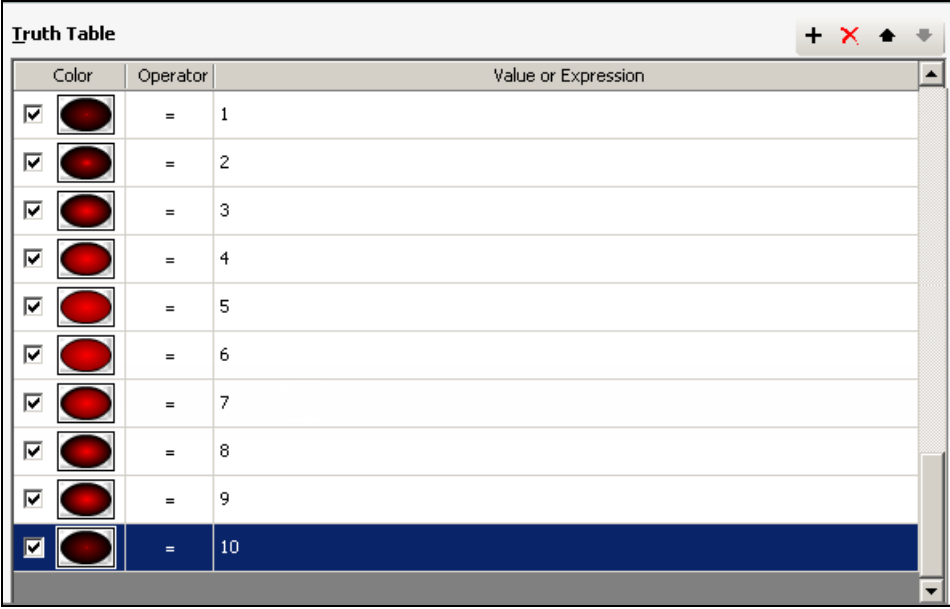


The nine remaining colors will be similar to the first color except they will have different values and center distributions.

33. Configure the nine remaining colors exactly the same as the first one: click **Gradient**, select **Three** colors (set them to black, dark red, and bright red in that order), select **Radial** direction, and choose the first **Variant**.
34. Configure the **Value** and **Center** distribution fields as follows:

| Operator & Value | Center Distribution |
|------------------|---------------------|
| =2               | 75                  |
| =3               | 50                  |
| =4               | 25                  |
| =5               | 0                   |
| =6               | 0                   |
| =7               | 25                  |
| =8               | 50                  |
| =9               | 75                  |
| =10              | 100                 |

When you are finished, your **Truth Table** should look like the one below:



|                                     | Color                                                                             | Operator | Value or Expression |
|-------------------------------------|-----------------------------------------------------------------------------------|----------|---------------------|
| <input checked="" type="checkbox"/> |  | =        | 1                   |
| <input checked="" type="checkbox"/> |  | =        | 2                   |
| <input checked="" type="checkbox"/> |  | =        | 3                   |
| <input checked="" type="checkbox"/> |  | =        | 4                   |
| <input checked="" type="checkbox"/> |  | =        | 5                   |
| <input checked="" type="checkbox"/> |  | =        | 6                   |
| <input checked="" type="checkbox"/> |  | =        | 7                   |
| <input checked="" type="checkbox"/> |  | =        | 8                   |
| <input checked="" type="checkbox"/> |  | =        | 9                   |
| <input checked="" type="checkbox"/> |  | =        | 10                  |

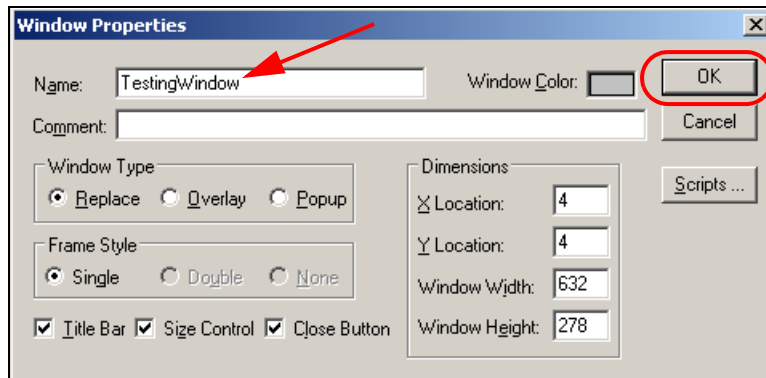
35. Click **OK**.

36. Click **Save and Close**, and then check in the symbol when prompted.

Next you will create a new window in WindowMaker to test the pulsating effect.

37. In WindowMaker, on the **File** menu, click **New Window**.

38. Name the new window **TestingWindow**, and then click **OK**.



---

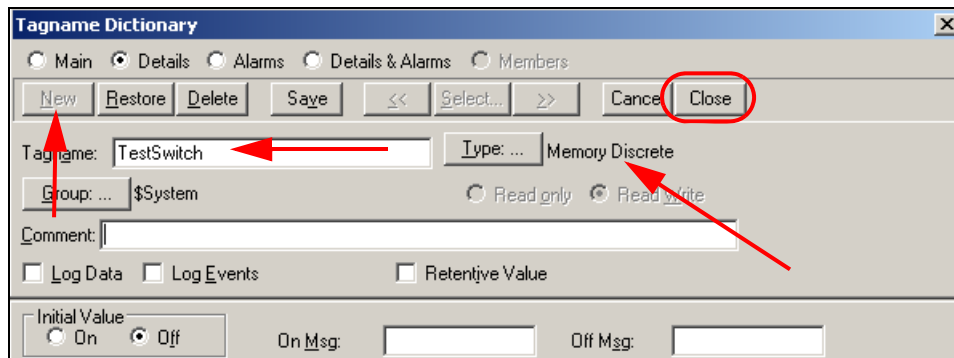
**Note:** The dimensions of the window do not matter, as long as they are sufficient to accommodate your symbols.

---

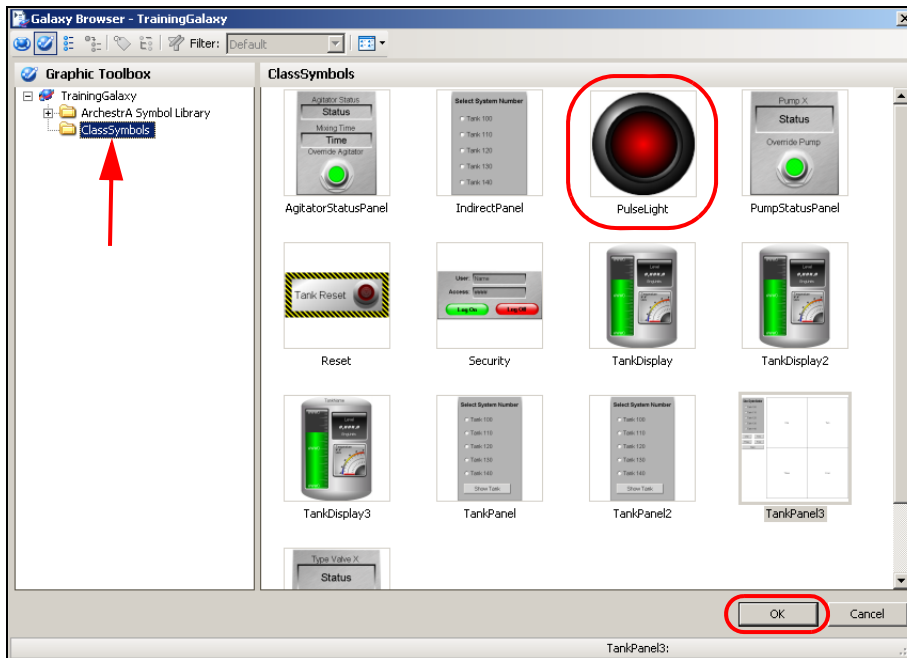
39. Close all other open windows.

40. On the **Special** menu, click **Tagname Dictionary**.

41. Click **New**, in the **Tagname** field enter **TestSwitch**, set it as a **Memory Discrete** tag, erase any existing comments, and then click **Close**.

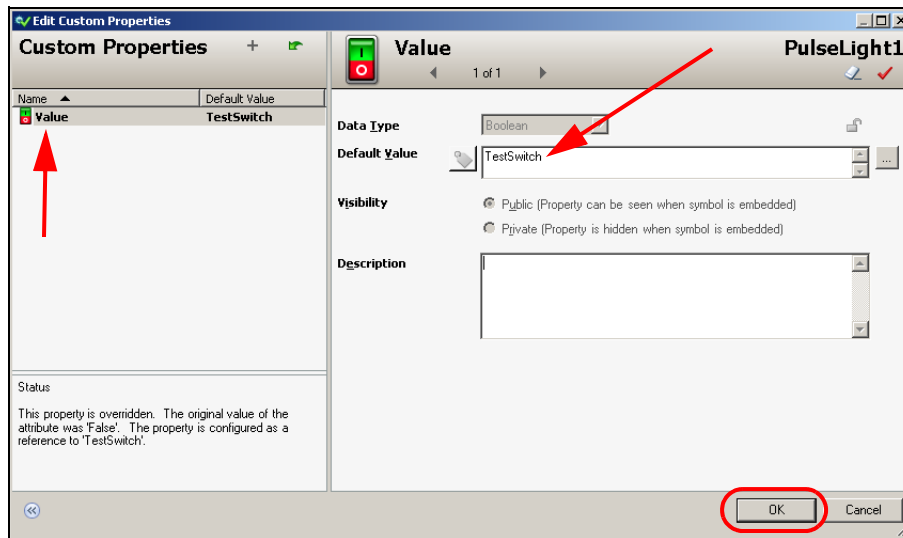


42. Click **Embed ArchestraA Graphic**.
43. In **ClassSymbols**, select **PulseLight**, and then click **OK**.

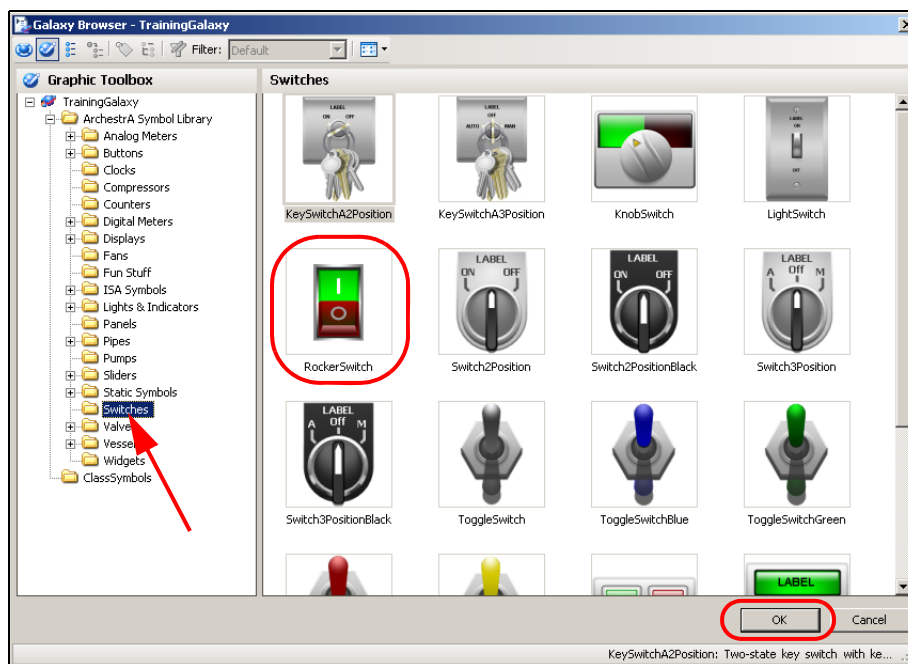


44. In **TestingWindow**, place the symbol, and then double-click to open it for editing.

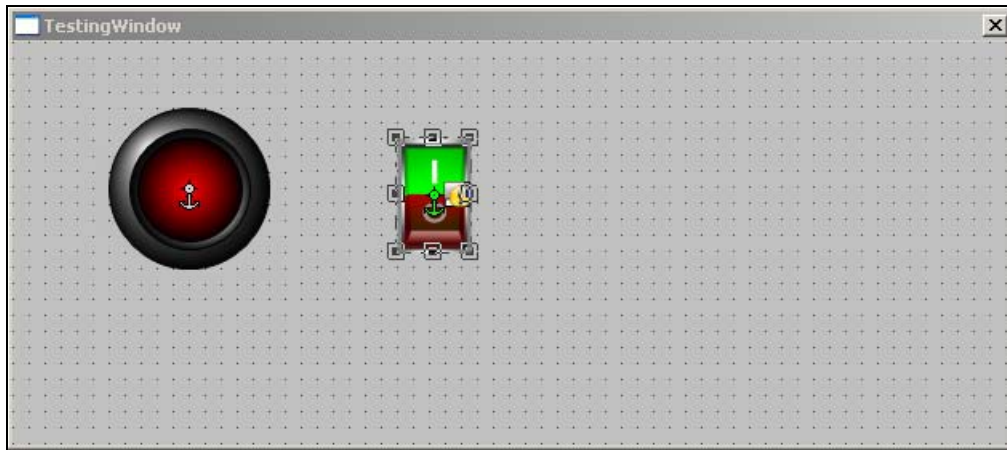
45. Select the **Value** custom property and set the **Default Value** to **TestSwitch** to associate it with the tag you just created, and then click **OK**.



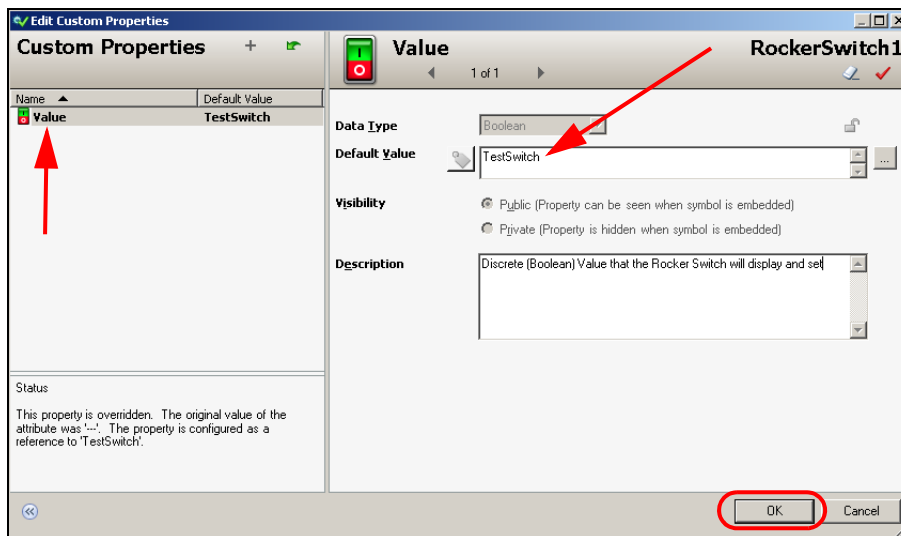
46. Click **Embed ArchestrA Graphic**.
47. In **ArchestrA Symbol Library**, click **Switches** and select **RockerSwitch**, and then click **OK**.



48. In **TestingWindow**, place the **RockerSwitch** symbol, and then double-click to open it for editing.



49. Select the **Value** custom property, and then set the **Default Value** to **TestSwitch**.



50. Click **OK**.

## Test What You Have Built

51. Click **Runtime**.
52. Click the **RockerSwitch** and verify that the light pulsates.
53. Click the **RockerSwitch** again and verify that the pulsation stops and the light defaults to its initial color.
54. Click **Development!**.



## Section 2 – Grow/Shrink Functionality

This section discusses the utilization of growing and shrinking functionality as another method of enhancing symbols, while making the most of limited space in an InTouch display.

### Element Sizing

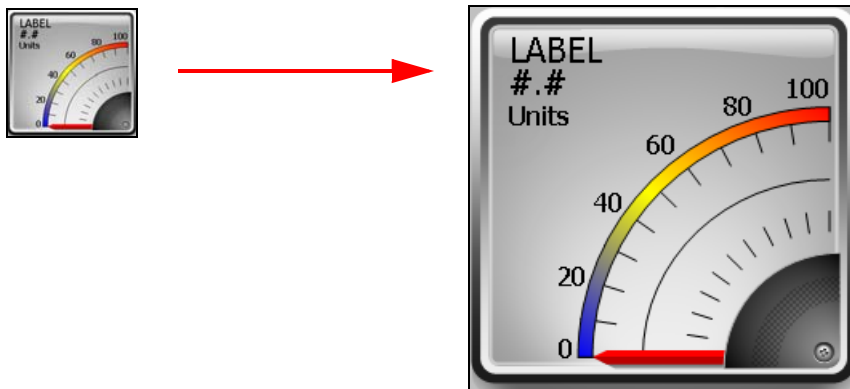
The height and width of an element can be controlled through either animation links or by addressing the height and width properties of the element. The advantage of the height and width properties is that the element can be made smaller or larger, whereas with the animation link, it can only be made smaller.

### Grow/Shrink Functionality

Archestra IDE provides the ability to not only animate resizing that can be triggered through scripts, but to also treat an embedded symbol or grouped element as an icon. This will allow animation of a group of elements as a single object, without disabling any of the animations the symbol might already have (except for interaction animations). Using the resize capabilities along with the TreatAsIcon feature, will give the ability to increase and decrease the size when the symbol is placed in an InTouch window.

| Runtime Behavior |                         |
|------------------|-------------------------|
| Enabled          | True                    |
| OwningObject     |                         |
| SymbolReference  | Symbol:AnalogMeterRound |
| TabOrder         | 0                       |
| TabStop          | True                    |
| TreatAsIcon      | True                    |
| Visible          | True                    |

With some additional scripting, the end result is a smaller graphic, when hovered over, grows into a legible size, then shrinks back down when no longer needed.





# Lab 8 – Creating Re-Sizeable Symbols

## Introduction

The ability to modify the width and height properties of a symbol allows the user to enlarge an otherwise too small to read symbol into something legible, while saving overall display space on the InTouch window. In this lab, you will create a re-sizeable meter that utilizes this capability, then use it as a template for creating other re-sizeable symbols.

## Objectives

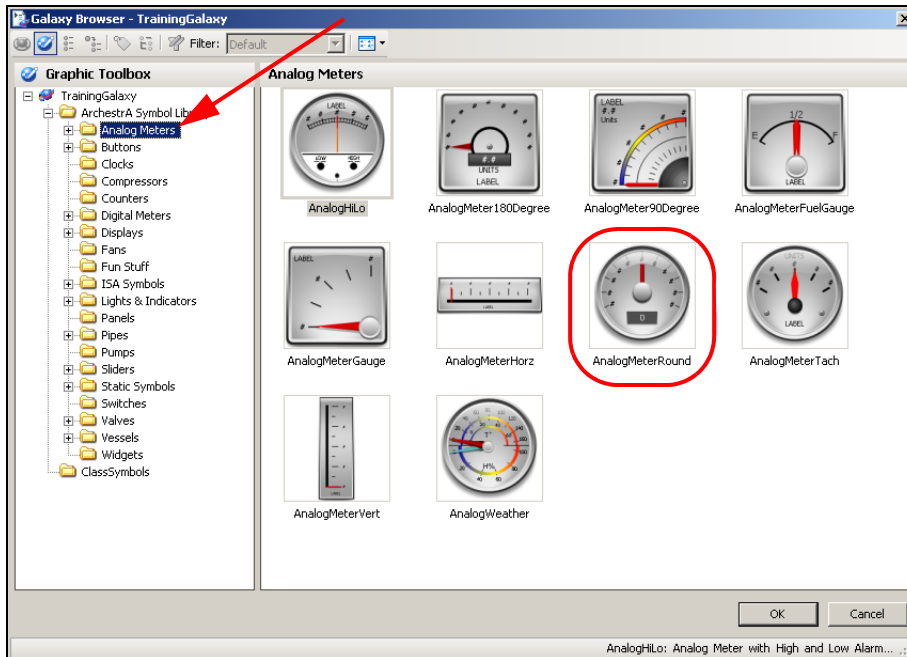
Upon completion of this lab, you will be able to:

- Create a symbol with scripting to grow and shrink an embedded graphic
- Reuse the grow and shrink script with a new symbol without modifying the script

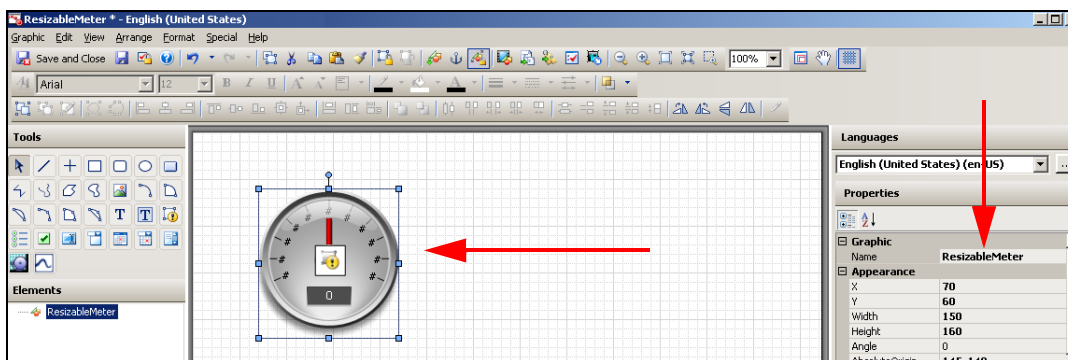
## Create Re-Sizeable Symbols

The first step in this lab is to create the shell symbol that will house all of the scripting and animations, and server as a template for additional symbols.

1. In ArchestrA IDE, right-click **ClassSymbols** and select **New | Symbol**.
2. Name the new symbol **ResizableMeter**, and then double-click it for editing.
3. Click **Embed ArchestrA Graphic**.
4. In **ArchestrA Symbol Library**, click the **Analog Meters** folder, and then double-click **AnalogMeterRound**.



5. On the canvas, place the **AnalogMeterRound** element and name it **ResizableMeter**.

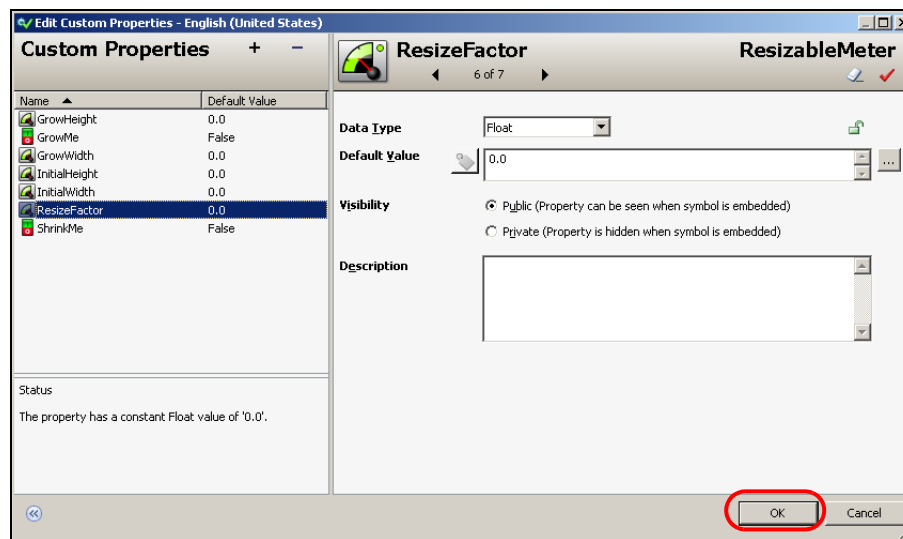


Add the custom properties that will be used in conjunction with scripting to control the size of the symbol.

6. On the canvas, right-click and select **Custom Properties**, and then create the following custom properties:

| Name          | Data Type | Visibility |
|---------------|-----------|------------|
| GrowMe        | Boolean   | Private    |
| ShrinkMe      | Boolean   | Private    |
| GrowHeight    | Float     | Private    |
| GrowWidth     | Float     | Private    |
| InitialHeight | Float     | Private    |
| InitialWidth  | Float     | Private    |
| ResizeFactor  | Float     | Public     |

7. Click **OK**.



## Add Resize Script

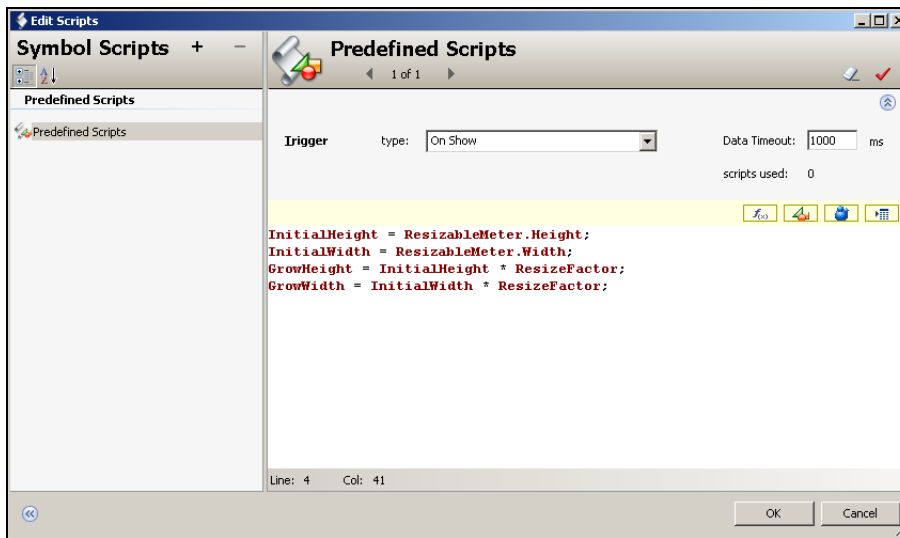
Add script to control the size of the symbol.

---

**Note:** All scripts for this lab are available to copy from the file **Lab 8 Scripts** in your **Training** directory.

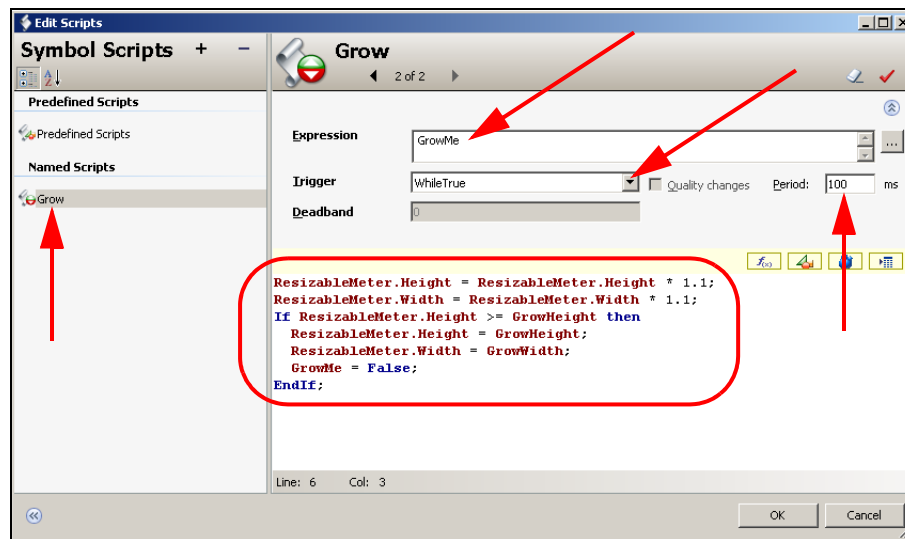
---

8. On the canvas, right-click and select **Scripts**.
9. Select **OnShow** as the trigger, and then enter the following script:  
**InitialHeight = ResizableMeter.Height;**  
**InitialWidth = ResizableMeter.Width;**  
**GrowHeight = InitialHeight \* ResizeFactor;**  
**GrowWidth = InitialWidth \* ResizeFactor;**



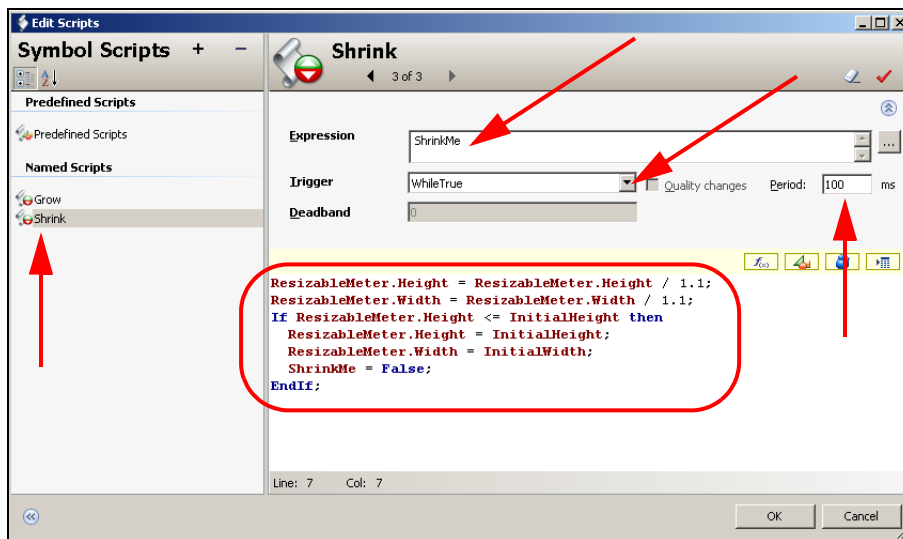
10. Click + and create the following **Named Script**:

| Name | Expression | Trigger   | Period | Script                                                                                                                                                                                                                                                             |
|------|------------|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Grow | GrowMe     | WhileTrue | 100 ms | ResizableMeter.Height = ResizableMeter.Height * 1.1;<br>ResizableMeter.Width = ResizableMeter.Width * 1.1;<br>IF ResizableMeter.Height >= GrowHeight then<br>ResizableMeter.Height = GrowHeight;<br>ResizableMeter.Width = GrowWidth;<br>GrowMe = False;<br>Endif; |



11. Click + and create the following **Named Script**:

| Name   | Expression | Trigger   | Period | Script                                                                                                                                                                                                                                                                        |
|--------|------------|-----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Shrink | ShrinkMe   | WhileTrue | 100 ms | ResizableMeter.Height = ResizableMeter.Height / 1.1;<br>ResizableMeter.Width = ResizableMeter.Width / 1.1;<br>IF ResizableMeter.Height <= InitialHeight then<br>ResizableMeter.Height = InitialHeight;<br>ResizableMeter.Width = InitialWidth;<br>ShrinkMe = False;<br>Endif; |

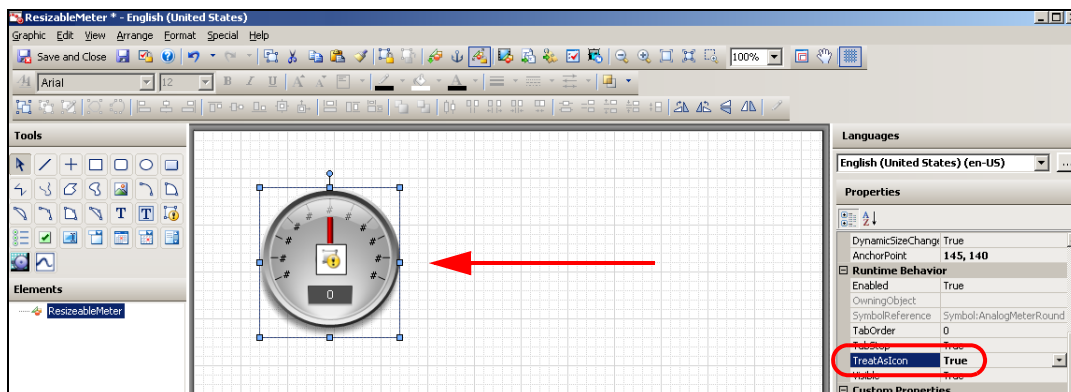


12. Click **OK**.

Next, using the script you just created, add animations to trigger the grow/shrink effect.

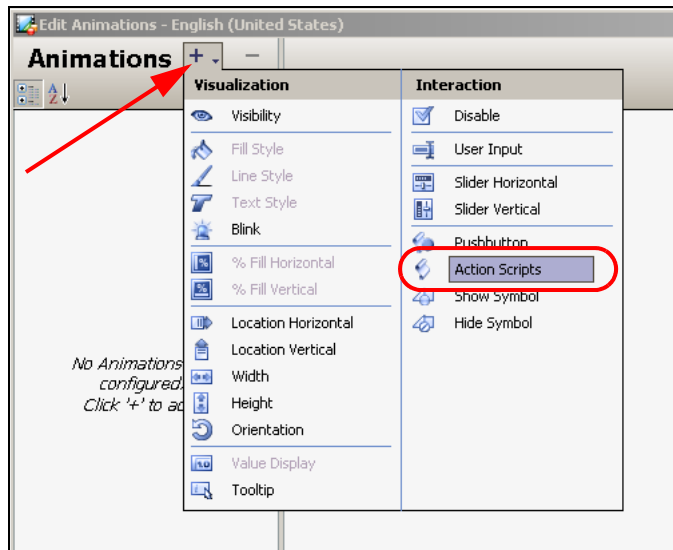
13. Select the **ResizableMeter** symbol.

14. In the **Properties** pane, under **Runtime Behavior**, change **TreatAsIcon** to **True**.

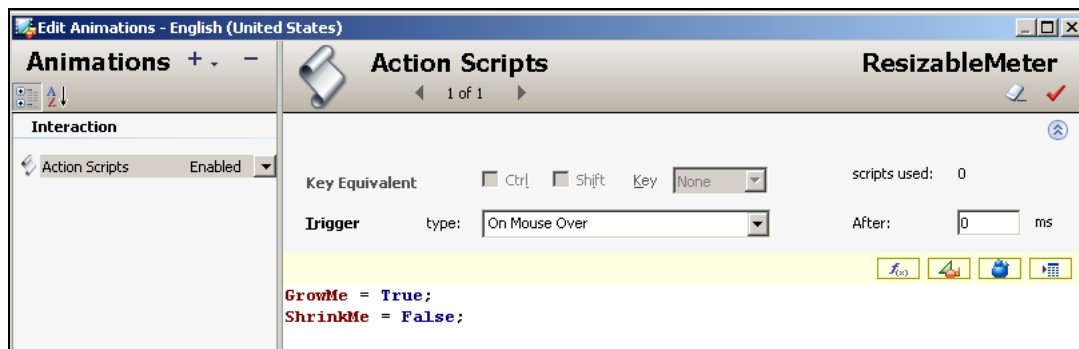




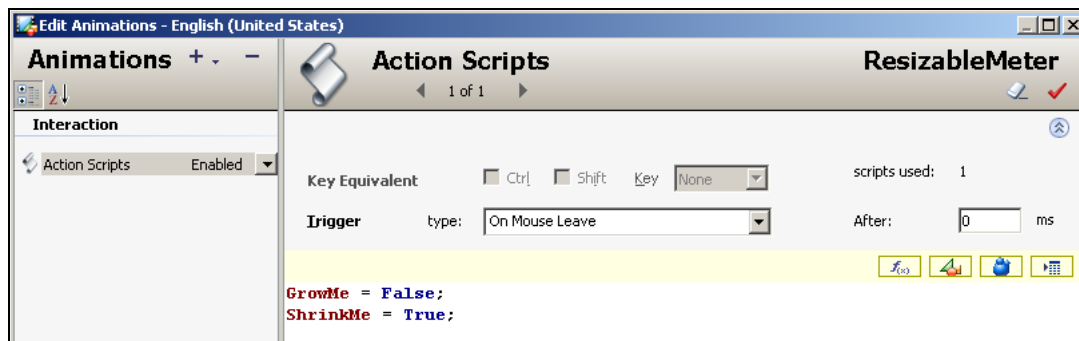
15. Double-click **ResizableMeter** to open it for editing.
16. Click **+** and select **Action Scripts**.



17. In the **Trigger Type** field, select **On Mouse Over**, and then enter the following script:  
**GrowMe = True;**  
**ShrinkMe = False;**



18. In the **Trigger Type** field, select **On Mouse Leave**, and then enter the following script:  
**GrowMe = False;**  
**ShrinkMe = True;**

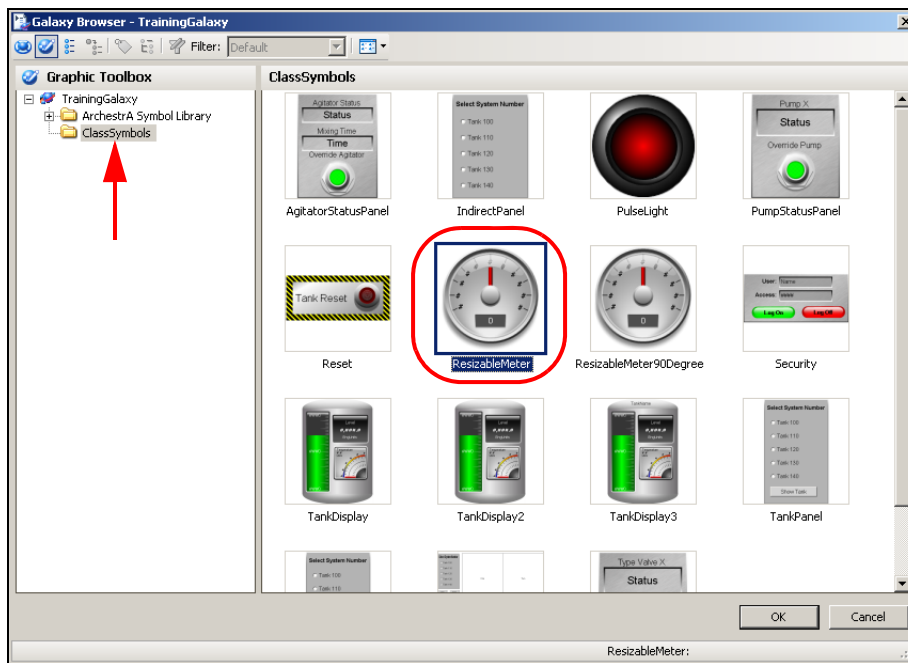


19. Click **OK**.
20. Click **Save and Close**, and then check in the symbol.

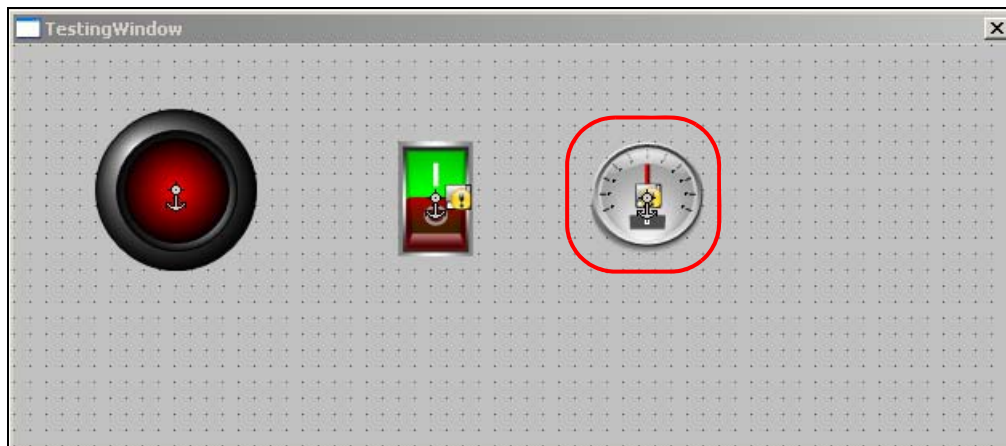
Next, you embed the symbol in the TestingWindow.

21. In WindowMaker, click **Embed ArchestrA Graphic**.

22. In **ClassSymbols**, double-click **ResizableMeter**.

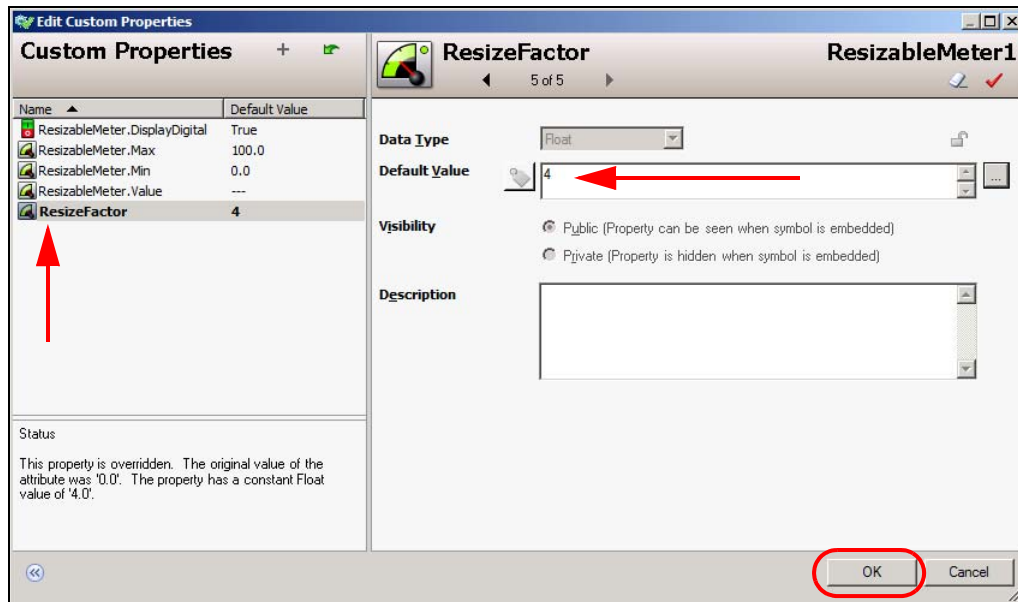


23. In **TestingWindow**, place the symbol and shrink it to a smaller size.



24. Double-click to access its **Custom Properties**.

25. Select the **ResizeFactor** custom property, change its **Default Value** to 4, and then click **OK**.



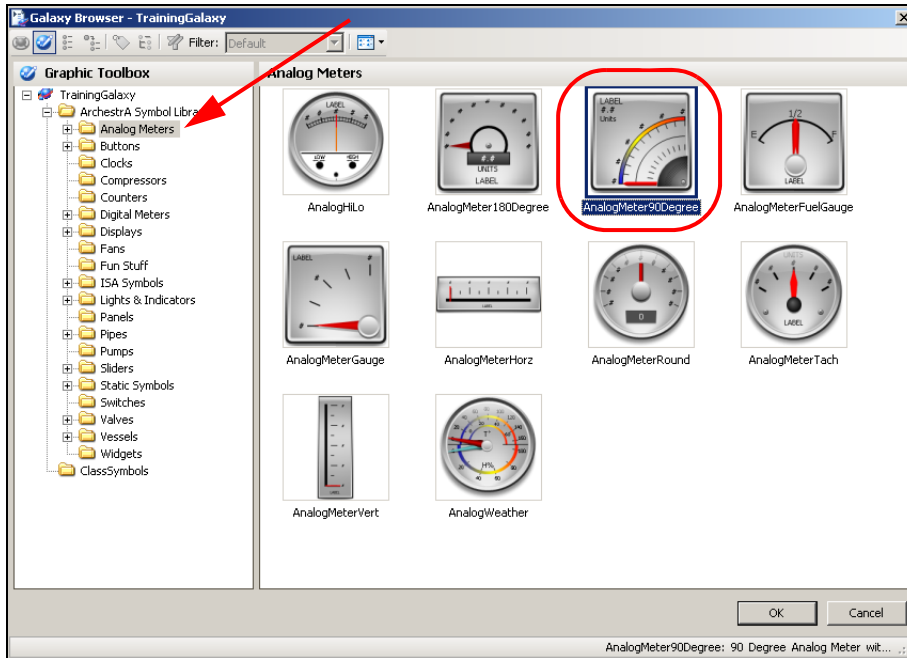
**Note:** Since you are only testing the symbol's re-sizability it is not necessary to link the remaining custom properties.

## Test What You Have Built

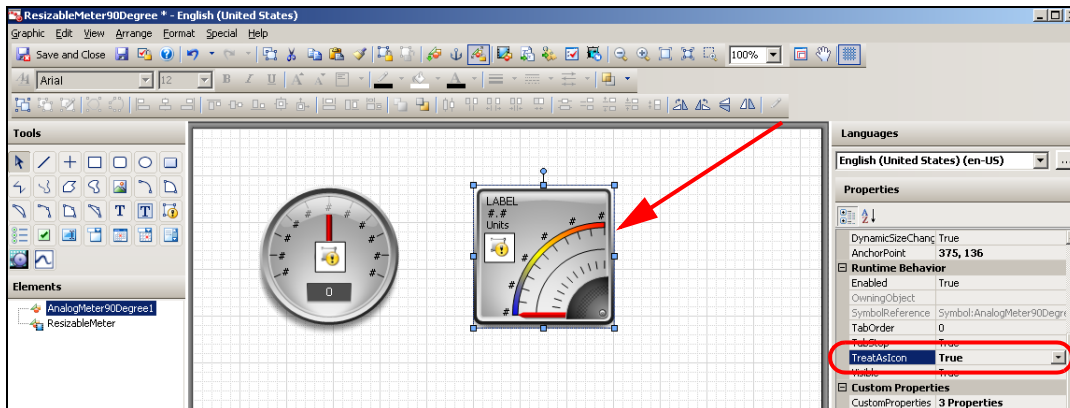
26. Click **Runtime**.
27. Mouse over the **ResizableMeter** and it grows.
28. Move your **Mouse** off of the **ResizableMeter** and it shrinks.
29. Click **Development!**.

## Create an Additional Re-Sizable Object

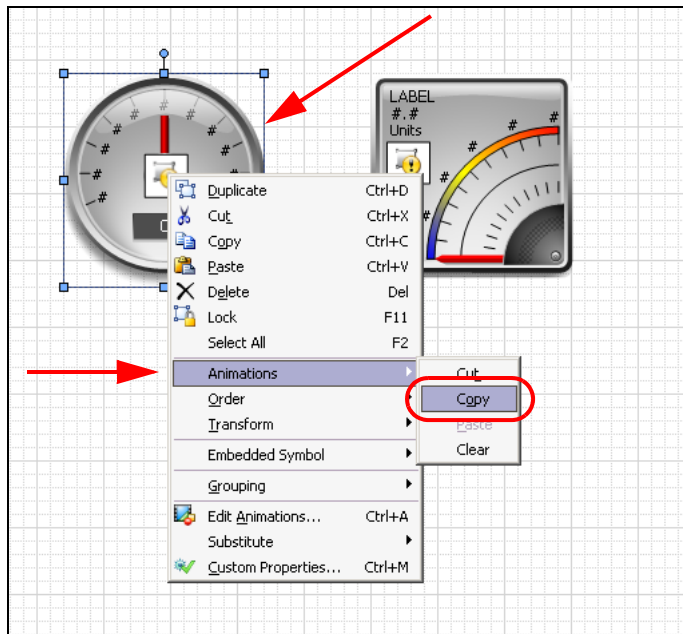
30. In Archestra IDE, duplicate the **ResizableMeter** symbol.
31. Name it **ResizableMeter90Degree**, and then double-click on it for editing.
32. Click **Embed Archestra Graphic**.
33. In **Archestra Symbol Library**, click **Analog Meters**, and then double-click **AnalogMeter90Degree**.



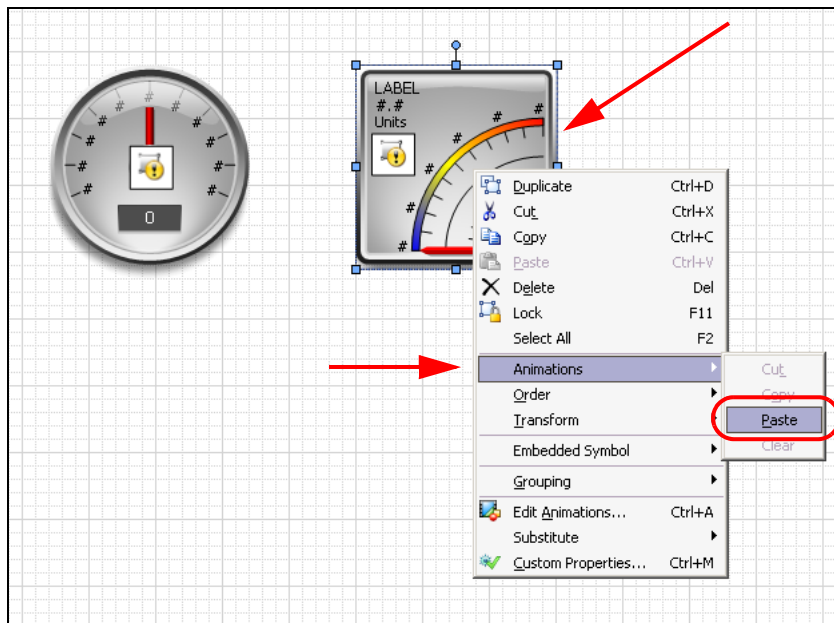
34. On the canvas, place the **AnalogMeter90Degree** element, and then set the **TreatAsIcon** property to **True**.



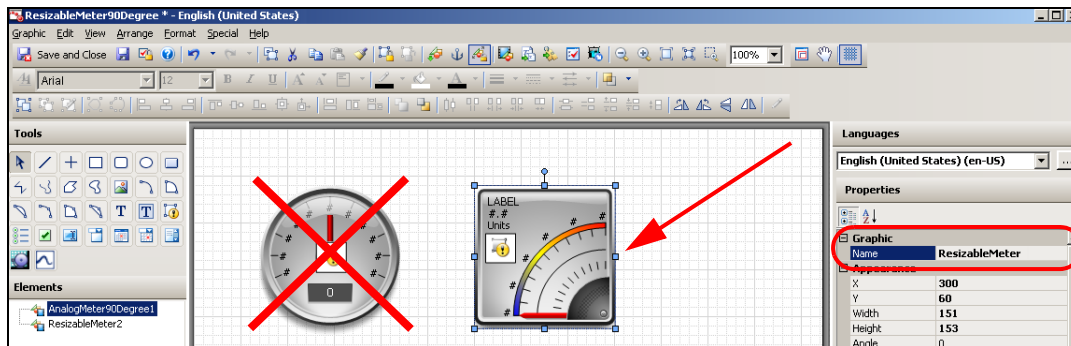
35. Right-click **ResizableMeter** and select **Animations | Copy**.



36. Right-click **AnalogMeter90Degree** and select **Animations | Paste**.

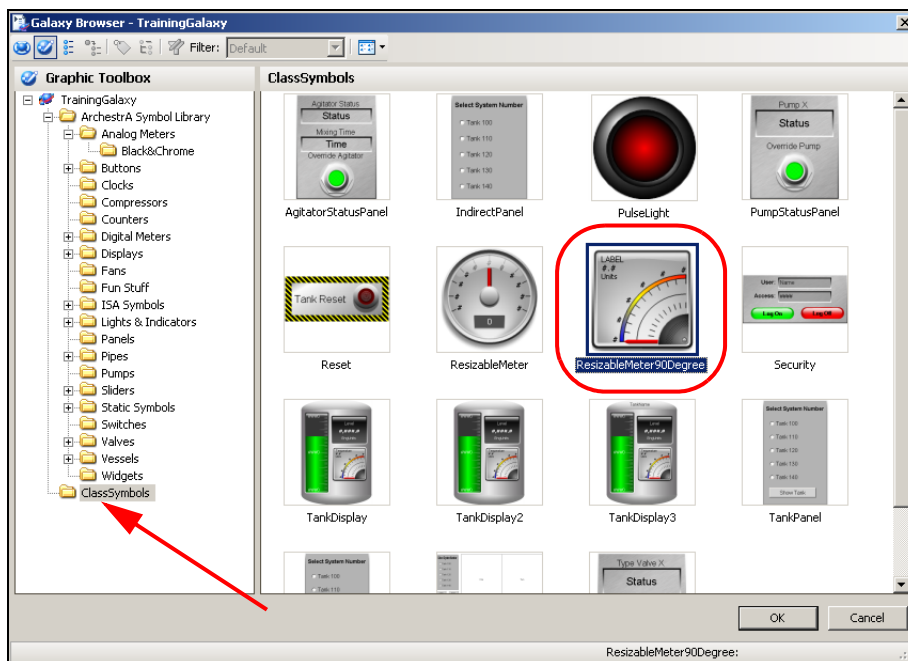


37. Delete the **ResizableMeter** symbol, and then rename **AnalogMeter90Degree** to **ResizableMeter**.

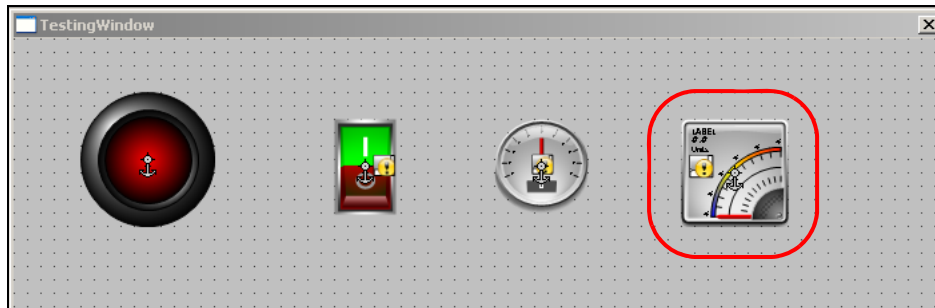


**Note:** This will allow the original scripting to work with the new symbol.

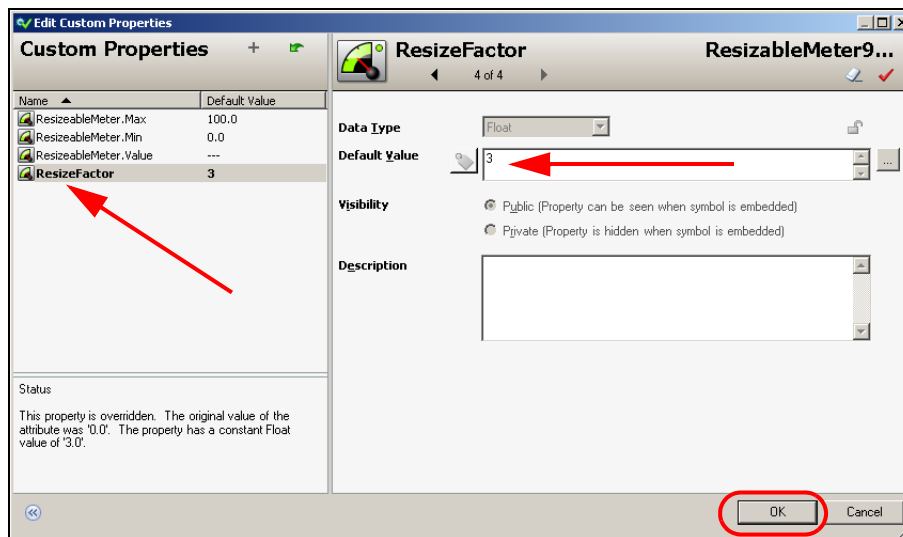
38. Click **Save and Close**, and then check in the symbol.
- Lastly, place ResizableMeter90Degree in the TestingWindow.
39. In WindowMaker, click **Embed ArchestrA Graphic**.
40. In **ClassSymbols**, double-click **ResizableMeter90Degree**.



41. In **TestingWindow**, place the symbol and shrink it down to a smaller size, and then double-click to access its **Custom Properties**.



42. Select the **ResizeFactor** custom property, change its **Default Value** to **3**, and then click **OK**.



## Test What You Have Built

43. Click **Runtime**.
44. Mouse over the **ResizableMeter** and it grows.
45. Move your mouse off of the **ResizableMeter** and it shrinks.
46. Click **Development!**.
47. If necessary, **Save and Close** and check in any symbols opened during this lab.





## Section 3 – Modify and Enhance Existing Symbols

This section describes the modification and enhancement of existing graphics and the issues associated with the modifications.

### Customized Symbols

Not every symbol in the library will fit every need, but they do provide a great starting point. With a little bit of modification, existing symbols can be enhanced beyond their default capabilities. These customized symbols can be integrated into an application, as well as other symbols.

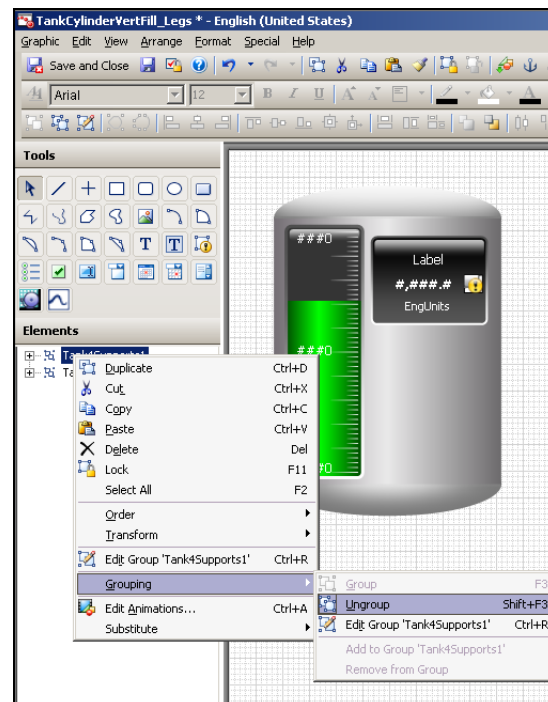
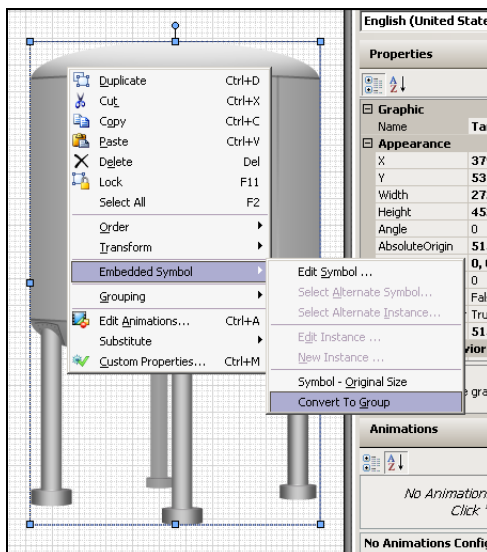
### Symbol Duplication

As a best practice, it is better to duplicate a symbol, copy the symbol, rename it, and make changes, rather than edit the original symbol.

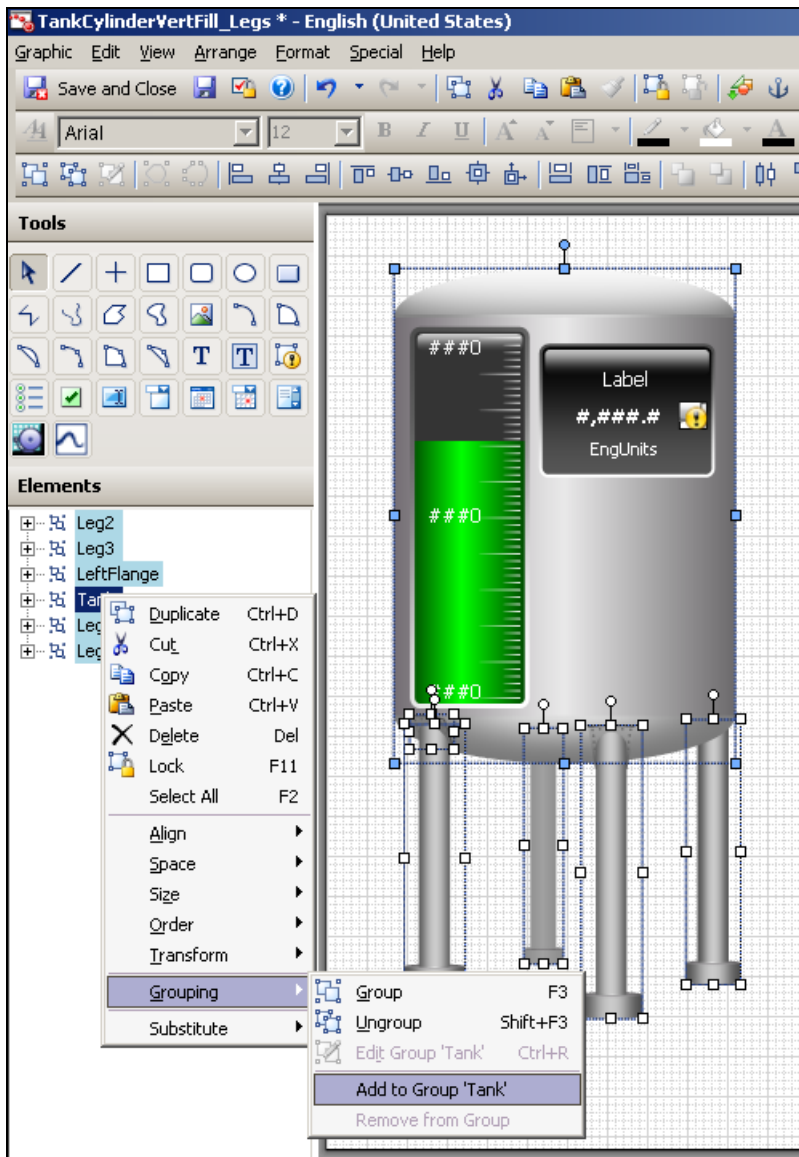
To copy a symbol or group of symbols, right-click and select **Duplicate**. The new symbol appears in the toolset in Edit mode, allowing you to rename the symbol.

### Grouping and Ungrouping

When combining elements of multiple symbols into a single symbol, first convert the embedded symbol into a group, then ungroup its elements. This allows components to be leveraged for the modified symbol being created.



Once the pieces are added to the existing symbol, delete the unnecessary parts that remain and regroup the remaining elements so that they act as a single entity.



**Note:** The group element that is selected last determines which group the elements will be added to.

# **Lab 9 – Modifying and Enhancing Existing Symbols**

## **Introduction**

While there are many symbols in ArchestrA Symbol Library, not all of them will meet your requirements all of the time. Rather than building a new symbol from scratch, many of the symbols in ArchestrA Symbol Library will fit your needs with a few modifications. In this lab, you will enhance the appearance and functionality of existing ArchestrA symbols by using them as a base for creating more advanced symbols in your applications.

## **Objectives**

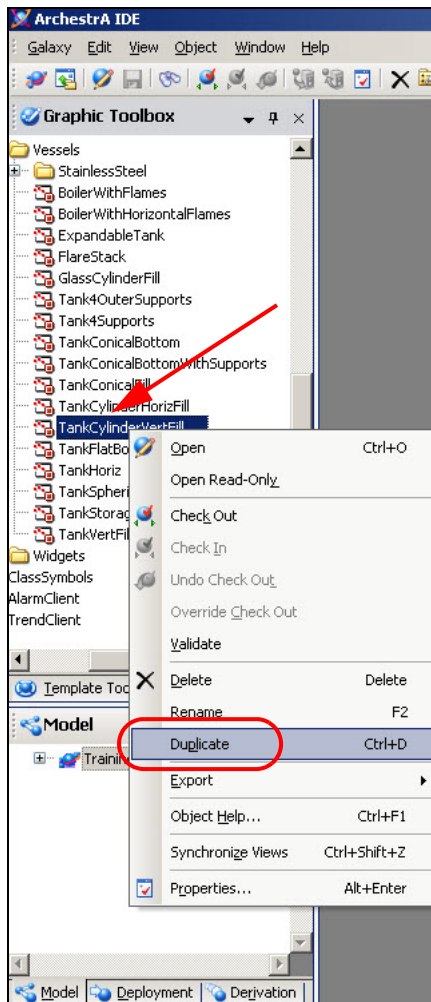
Upon completion of this lab, you will be able to:

- Convert an embedded symbol into a group
- Add additional animations to the existing symbols

## Modify an Existing Symbol's Graphics

For this portion of the lab, you will modify a vertical tank which contains most of the animations you will need, but lacks some of the graphical elements, like legs.

1. In ArchemstrA IDE, in **ArchemstrA Symbol Library**, expand **Vessels**, and then right-click **TankCylinderVertFill** and select **Duplicate**.



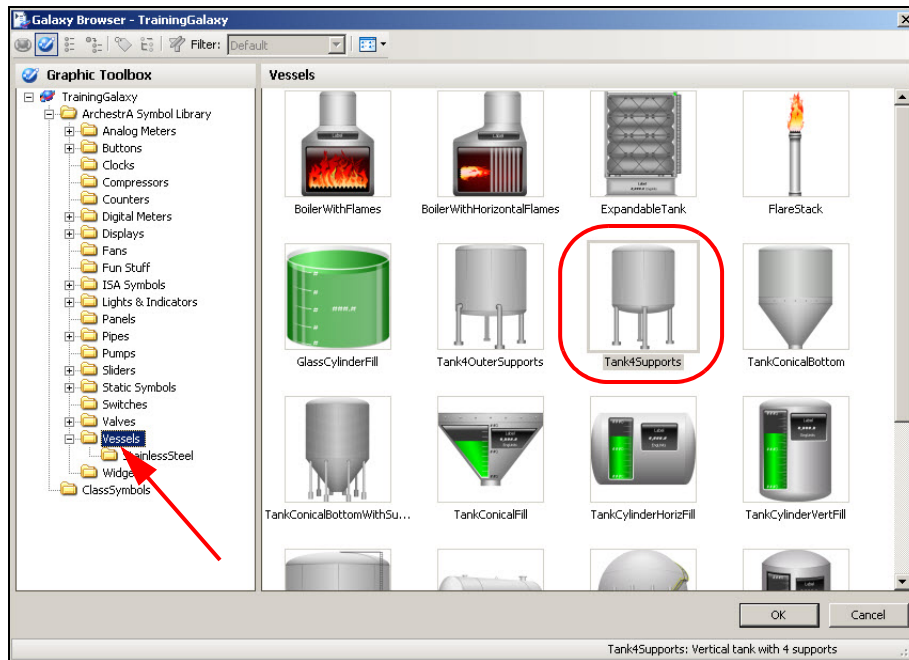
---

**Note:** It is important to create a duplicate of the symbol so that you do not modify the original, which could already be in use in other parts of the application.

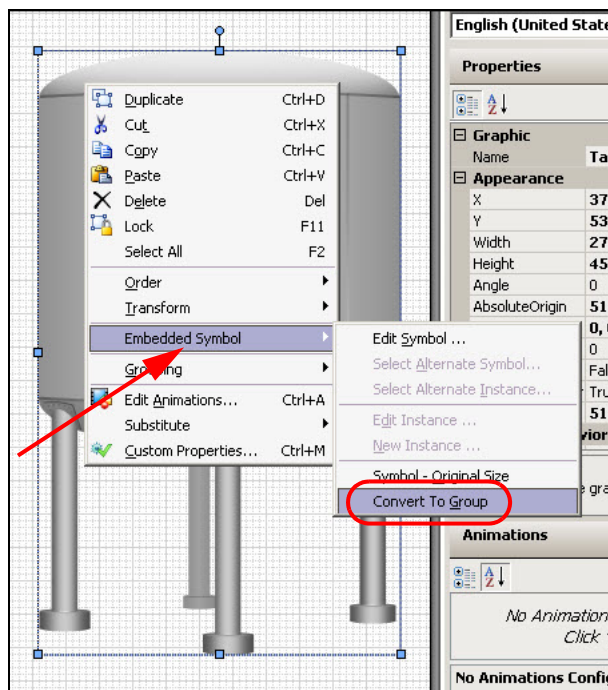
---

2. Name the new symbol **TankCylinderVertFill\_Legs**, and then double-click to open it for editing.
3. Click **Embed ArchemstrA Graphic**.

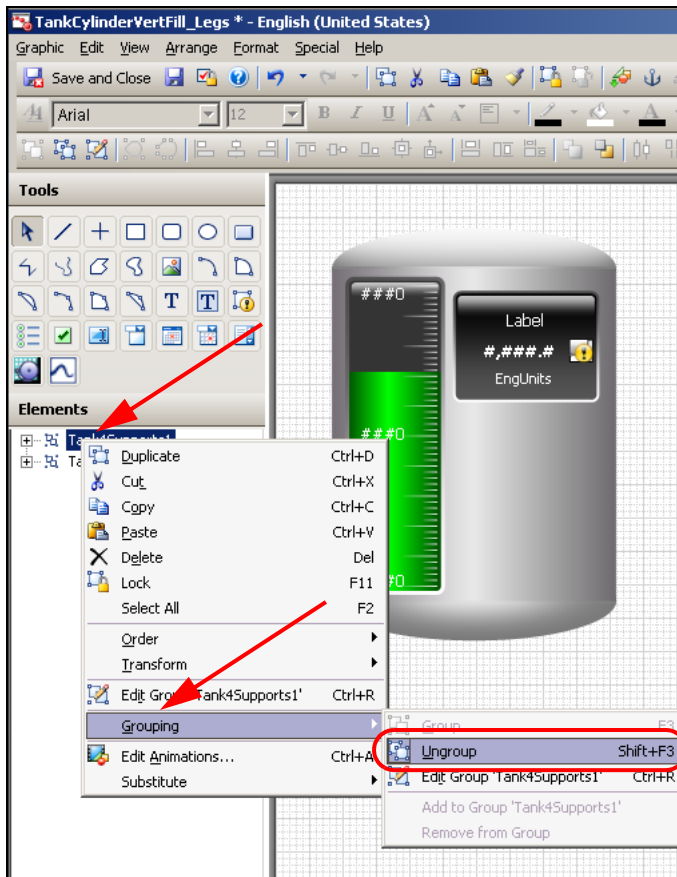
4. In **ArchestrA Symbol Library**, select **Vessels**, and then double-click **Tank4Supports**.



5. On the canvas, place **Tank4Supports** next to the tank.
6. Right-click **Tank4Supports** and select **Embedded Symbol | Convert To Group**.



7. In **Elements** pane, right-click **Tank4Supports1** and select **Grouping | Ungroup**.

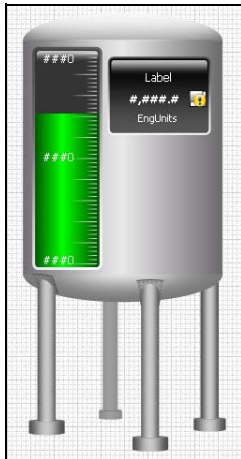


8. Delete the following elements:

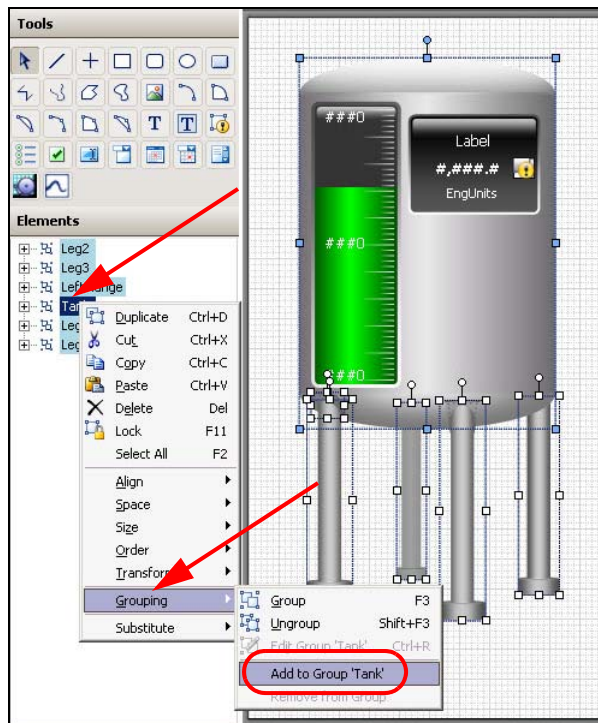
- **TankLines**
- **TankBody**
- **TopofTank**
- **BottomofTank**

9. Select the remaining leg elements and place them on the vertical tank.

You will need to organize the legs individually and correct the order of elements so that the legs are properly arranged on the tank as shown below.



10. In the **Elements** pane, while holding down the **Ctrl** key, select each of the elements.  
Make sure to select the **Tank** element last, so that it highlights in a darker blue color than the leg elements.
11. In **Elements** pane, right-click **Tank** and select **Grouping | Add to Group 'Tank'**.



**Note:** The group element that is selected last, determines which group the elements will be added to.

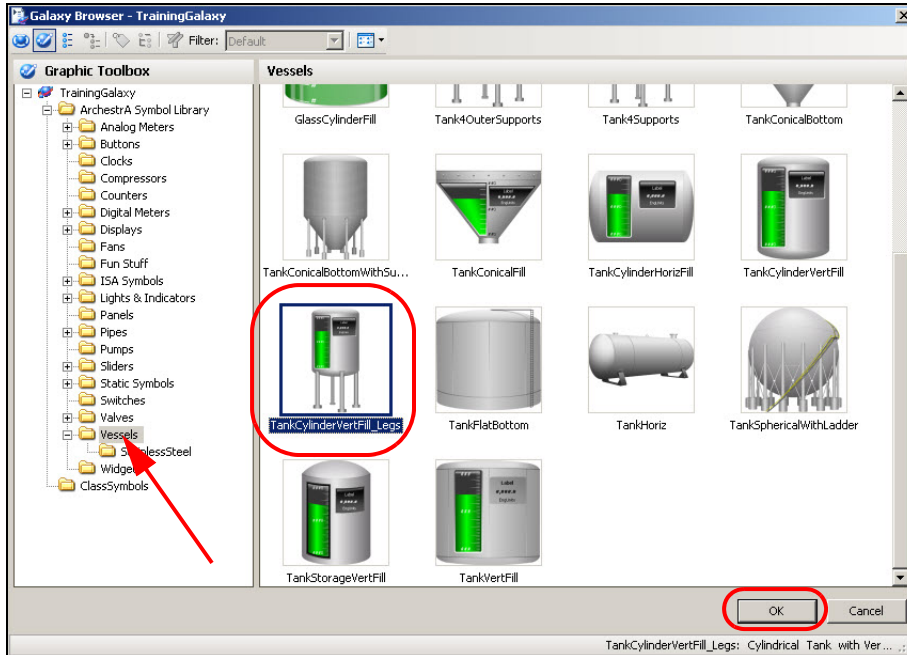
12. Click **Save and Close**, and then check in the symbol.



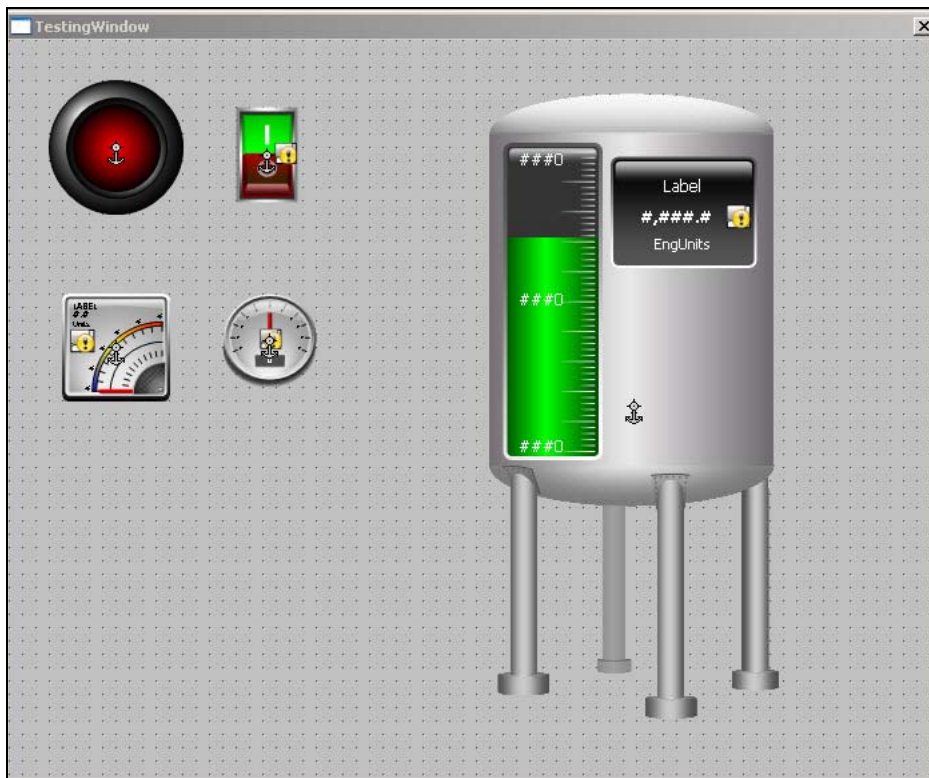
Next, you will embed the symbol in your TestingWindow in WindowMaker.

13. In WindowMaker, click **Embed ArchestrA Graphic**.

14. In **ArchestrA Symbol Library**, select **Vessels**, and then double-click **TankCylinderVertFill\_Legs**.



15. In **TestingWindow**, place the symbol.

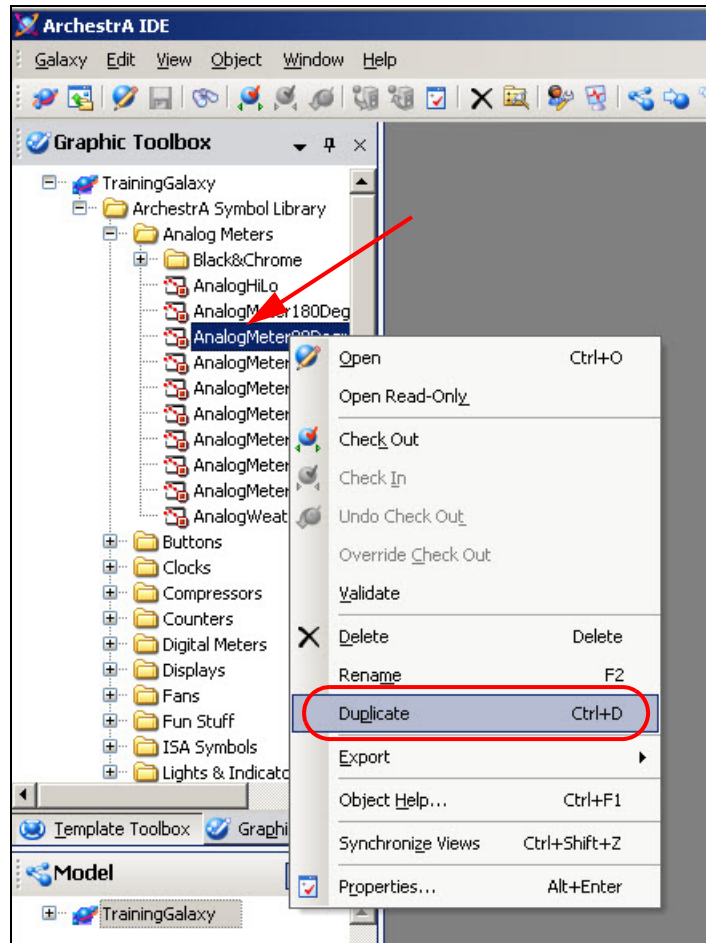




## Modify an Existing Symbol's Functionality

In the next portion of the lab, you will modify an analog meter that graphically contains all of the elements you need, but lacks an animation to show the units of measurement.

16. In Archestra IDE, in **Archestra Symbol Library**, expand **AnalogMeters**, right-click **AnalogMeter90Degree** and select **Duplicate**.



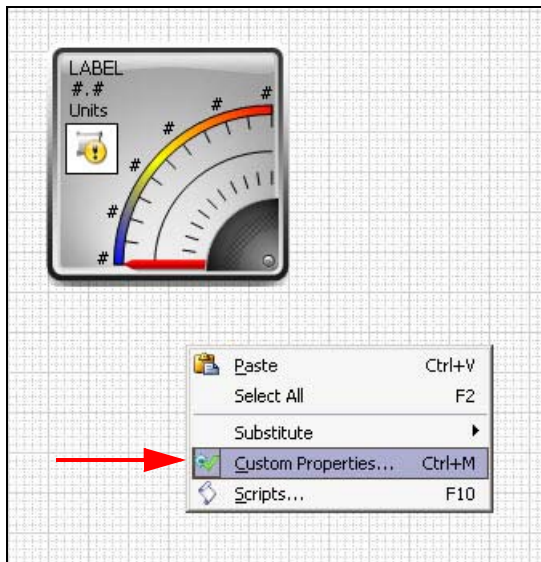
17. Name the new symbol **AnalogMeter90Degree\_<your initials>**, and then double-click to open it for editing.

---

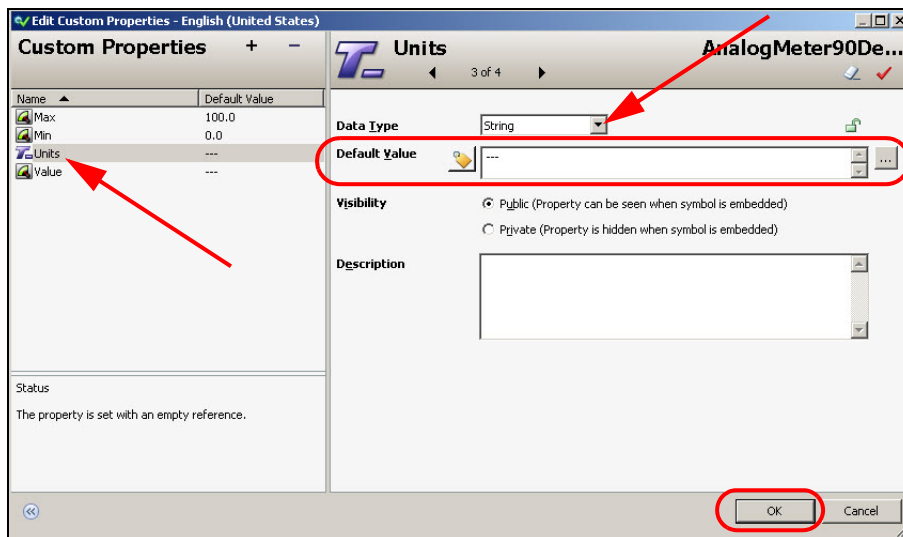
**Note:** For the examples shown in this training, you will use the initials DK.

---

18. On the canvas, right-click and select **Custom Properties**.



19. Click **+** and name the new custom property **Units**.
20. Select **String** as the **Data Type**, click **Text**  to switch to expression mode , set its **Default Value** to ---, and then click **OK**.

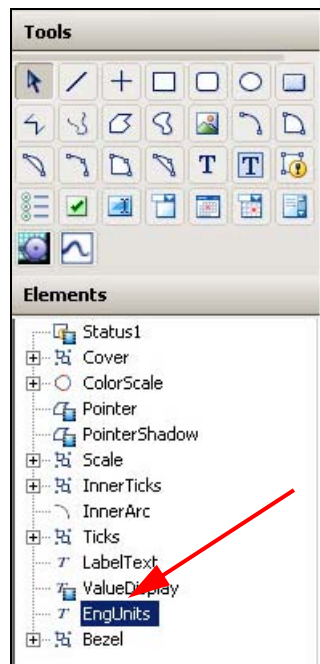


---

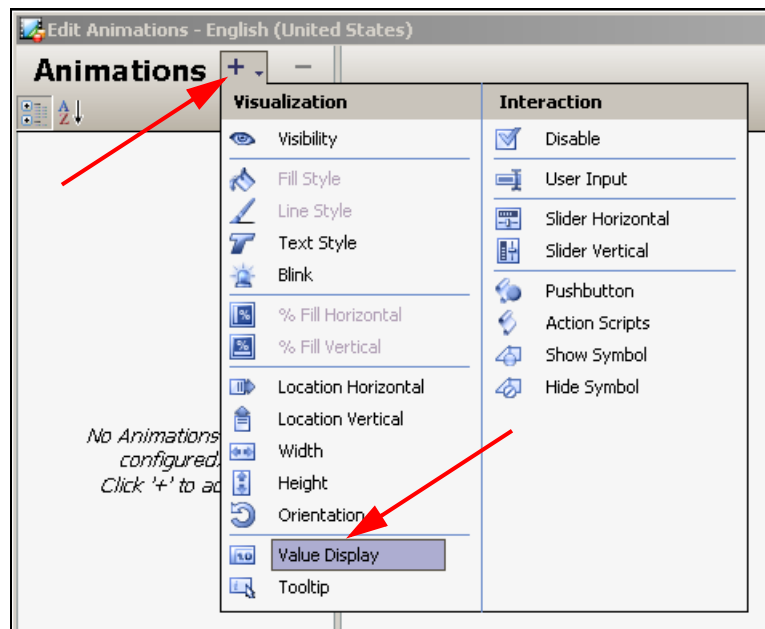
**Note:** Setting the **Default Value** to the placeholder "---" will display a blank value if a unit of measure is not specified.

---

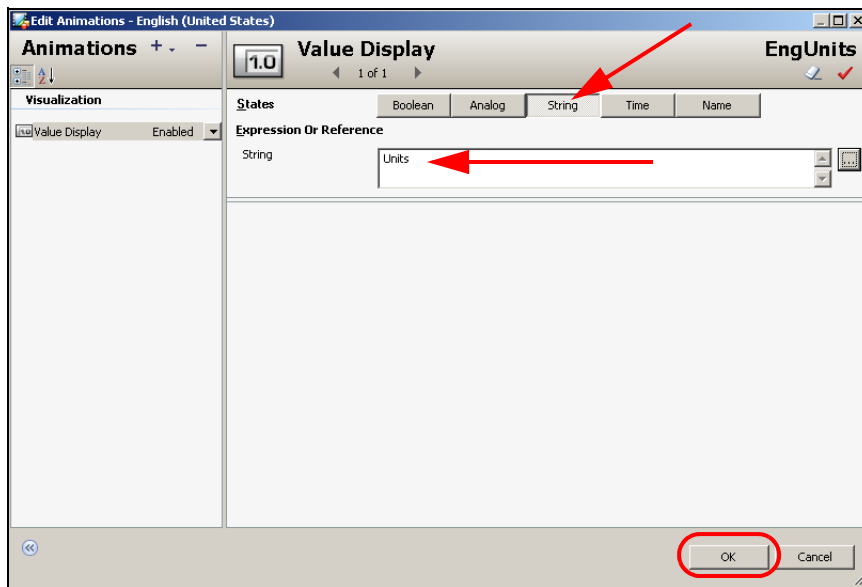
21. In the **Elements** pane, double-click **EngUnits**.



22. Click + and select **ValueDisplay**.



23. Click **String** and in the **String** field enter **Units**, and then click **OK**.

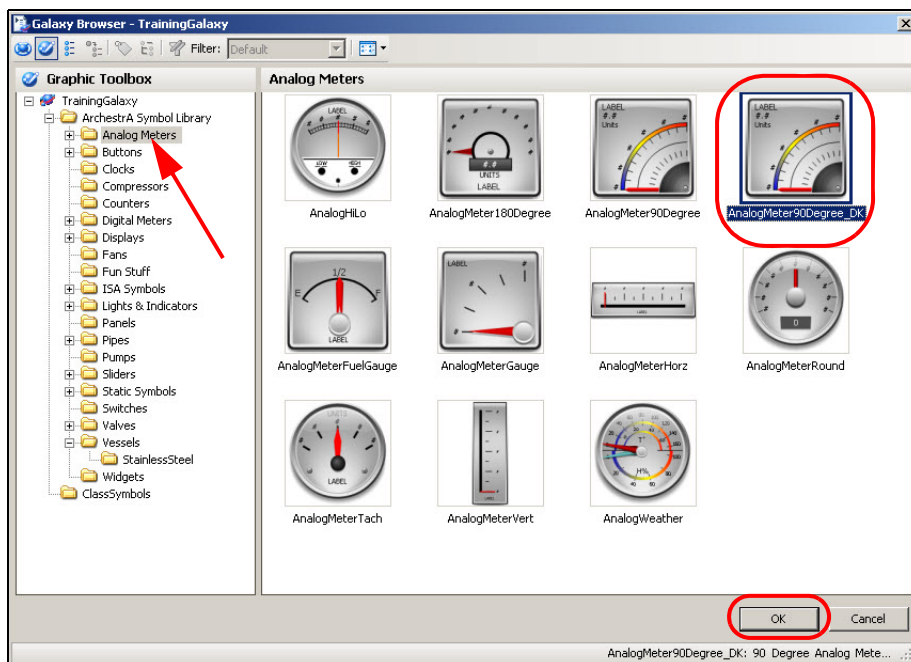


24. Click **Save and Close**, and then check in the symbol.

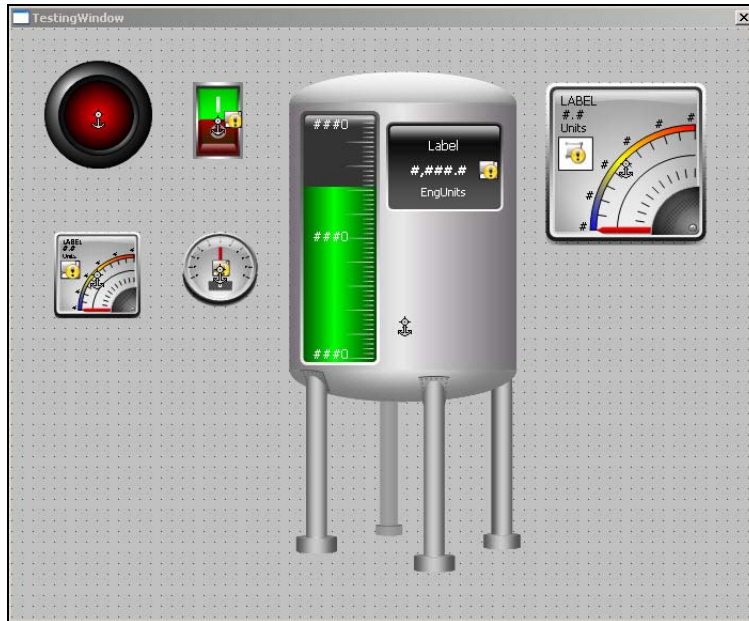
Next, in WindowMaker, you will embed the symbol in your TestingWindow.

25. In WindowMaker, click **Embed ArchestrA Graphic**.

26. In **ArchestrA Symbol Library**, select **AnalogMeters**, and then double-click **AnalogMeter90Degree\_<your initials>**.



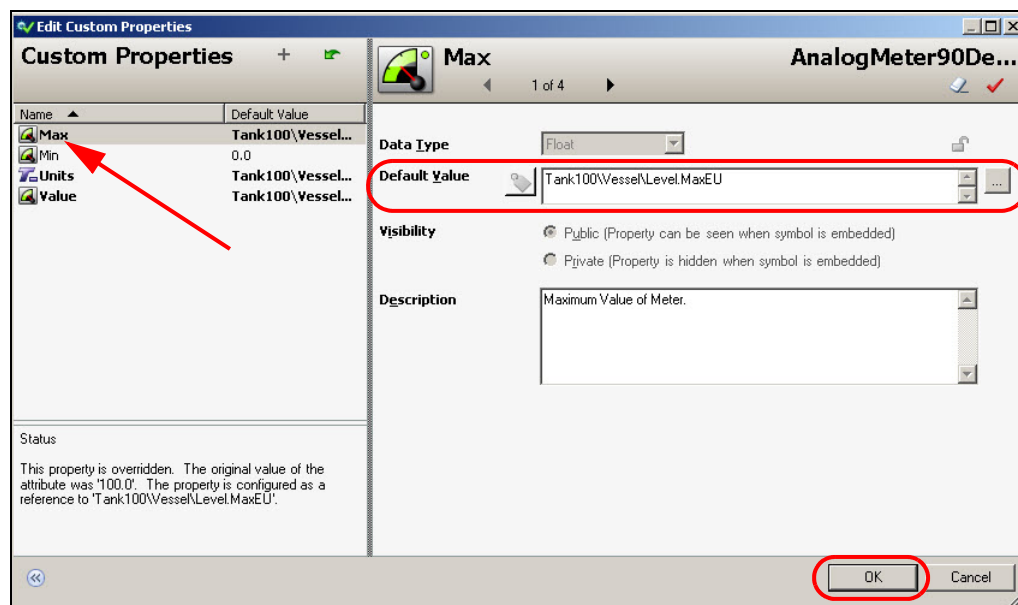
27. In **TestingWindow**, place the symbol, and then double-click it for editing.



28. Configure the **Custom Properties** as follows:

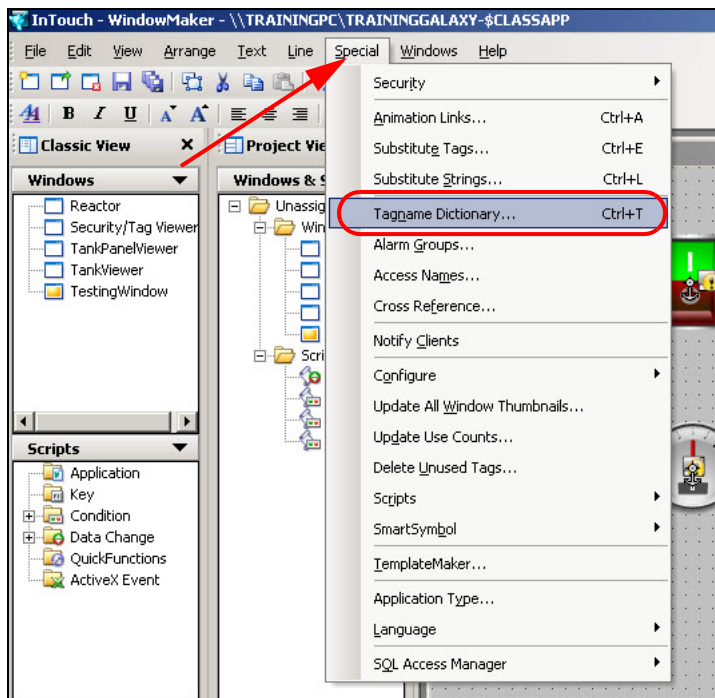
| Custom Property | Default Value                 |
|-----------------|-------------------------------|
| Value           | Tank100\Vessel\Level          |
| Units           | Tank100\Vessel\Level.EngUnits |
| Max             | Tank100\Vessel\Level.MaxEU    |

The custom property **Max** is shown as an example in the image below:

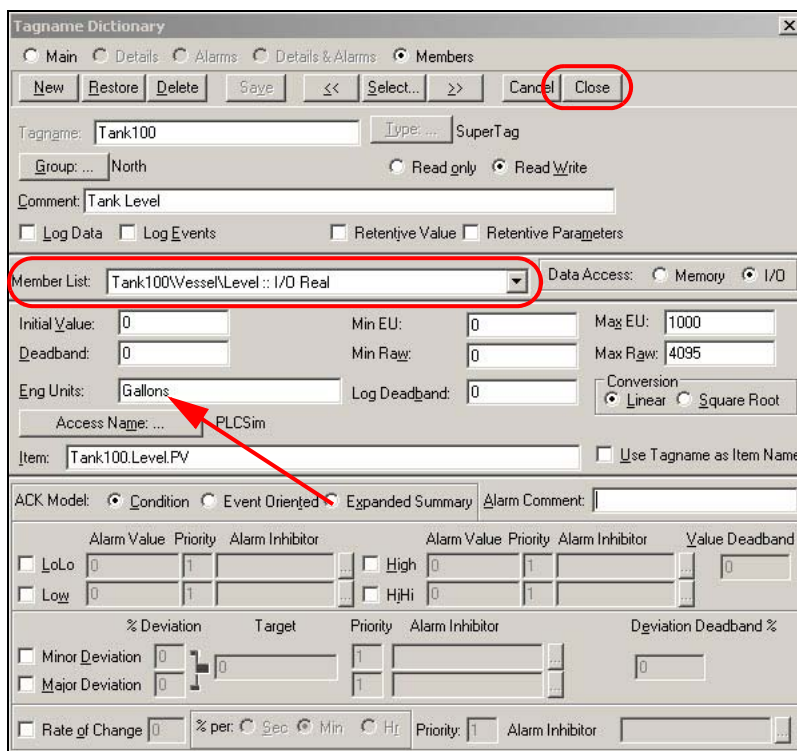


29. Click **OK**.

30. On the **Special** menu, click **Tagname Dictionary**.

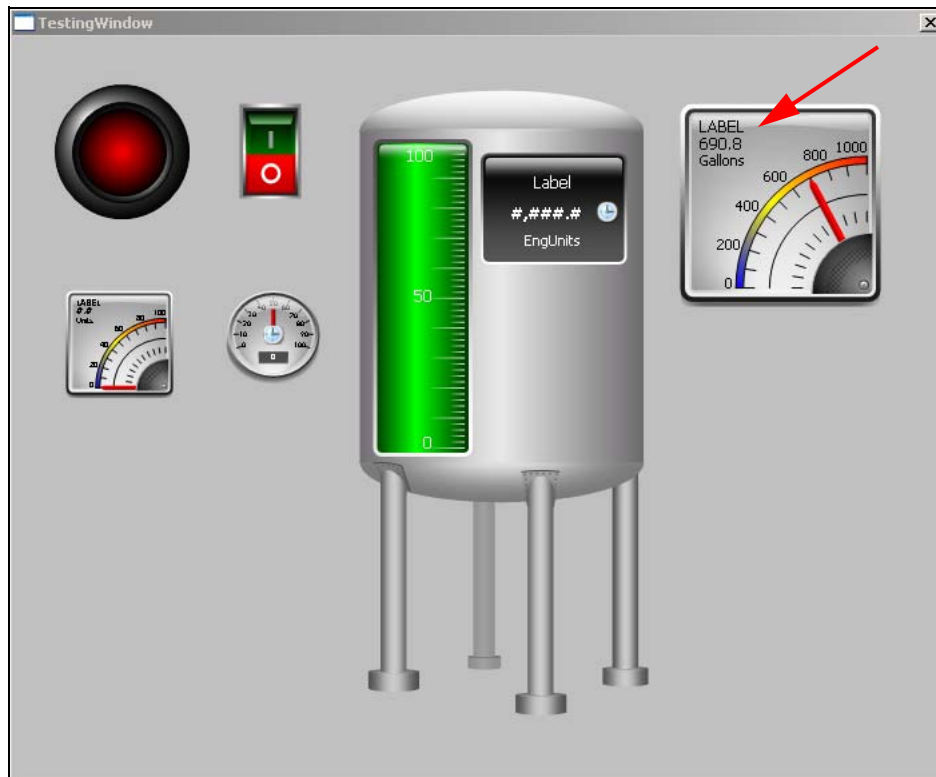


31. Click **Select**, and then double-click **Tank100**.
32. From the **Member List** drop-down select **Tank100\Vessel\Level :: I/O Real**, and in the **Eng Units** enter **Gallons**, and then click **Close**.



## Test What You Have Built

33. Click **Runtime**.
34. Ensure **Gallons** is listed as the unit of measure.



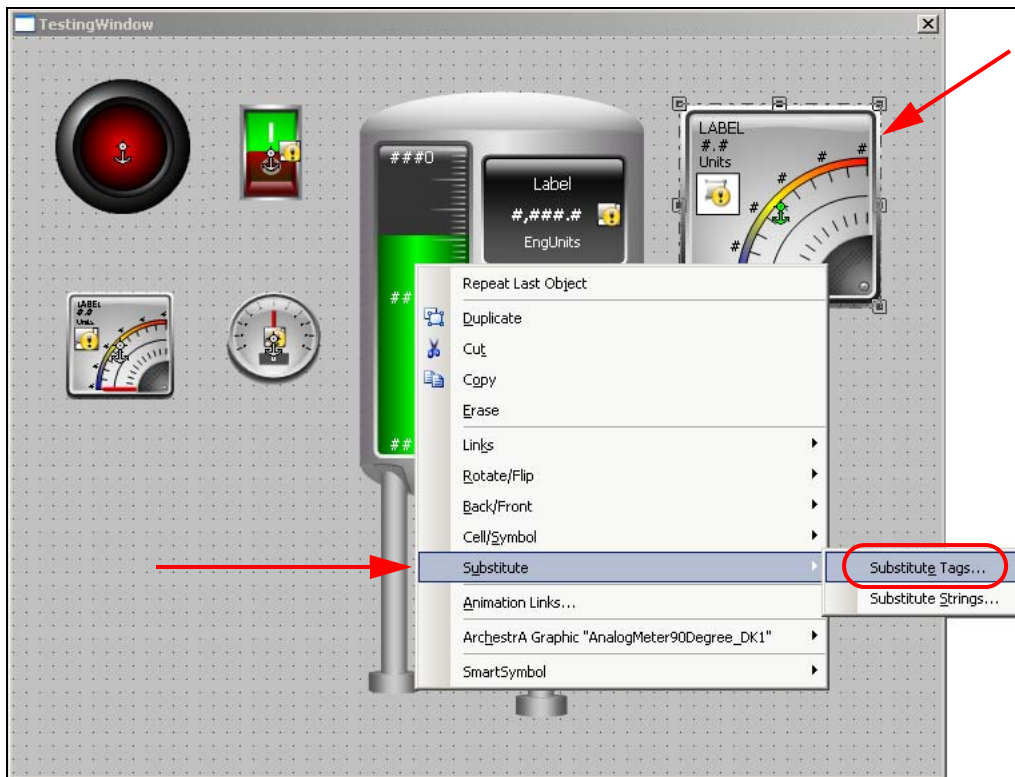
35. Click **Development!**.



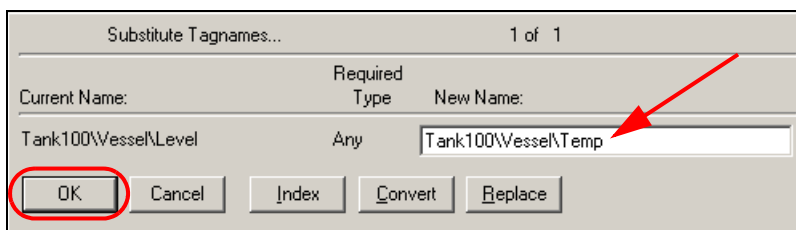
## Substitute Tags

Next, you will change the tag that your symbol is linked with.

36. Right-click **AnalogMeter90Degree\_<your initials>** and select **Substitute | Substitute Tags**.

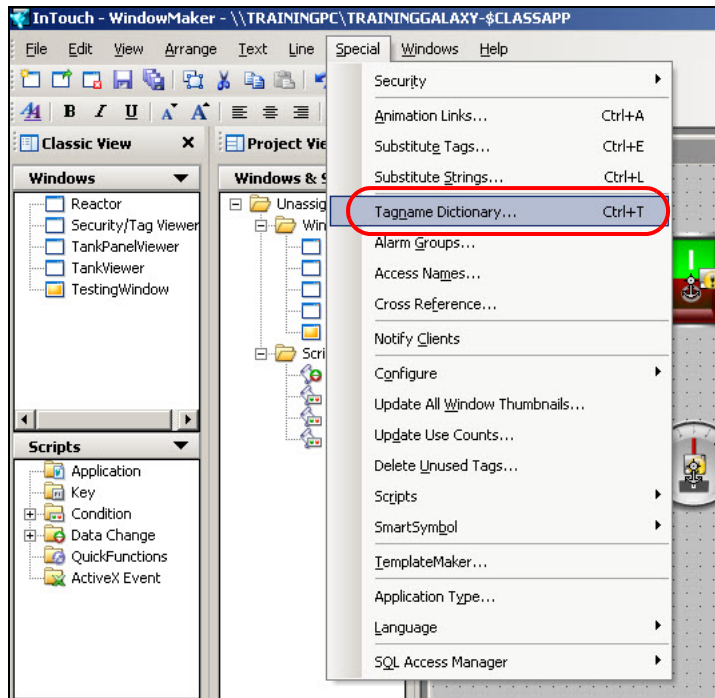


37. Change **Tank100\Vessel\Level** to **Tank100\Vessel\Temp**, and then click **OK**.

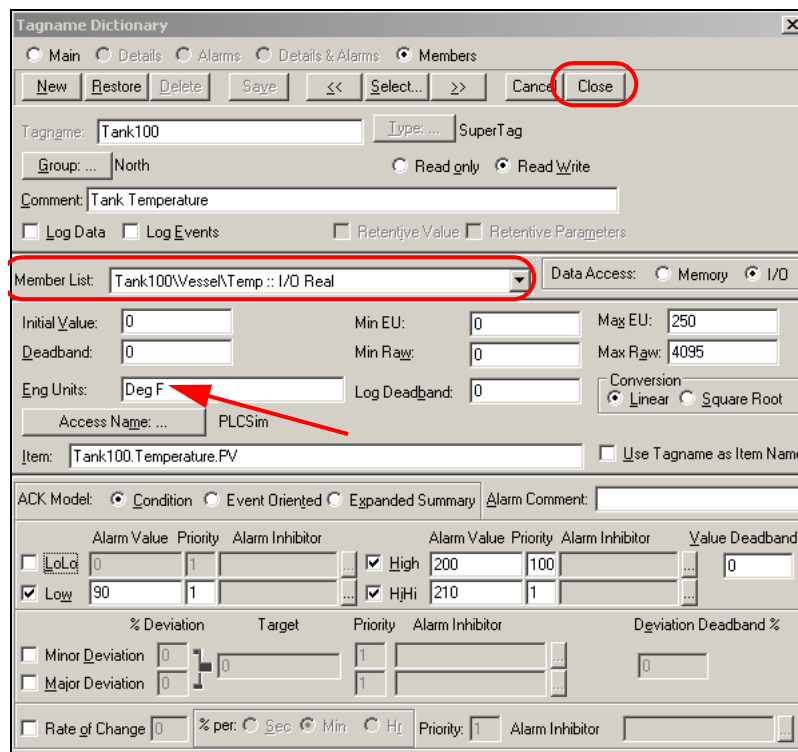




38. On the **Special** menu, click **Tagname Dictionary**.

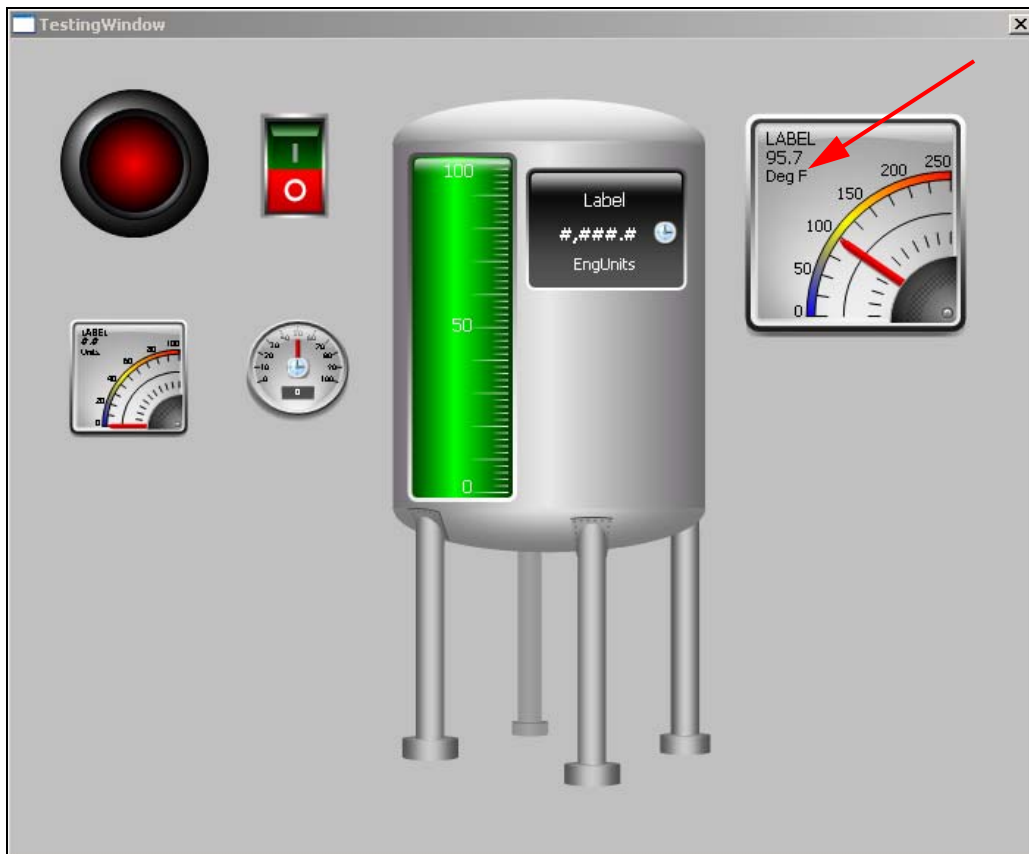


39. From the **Member List** drop-down select **Tank100\Vessel\Temp :: I/O Real**, verify that the **Eng Units** is **Deg F**, and then click **Close**.



## Test What You Have Built

40. Click **Runtime**.
41. Ensure **Deg F** is listed as the new unit of measurement.



42. Click **Development!**.
43. In Archestra IDE, if necessary, **Save** and check in any open symbols.

---

## Section 4 – ActiveX Controls

This section describes the installation, configuration, and application of ActiveX controls in InTouch applications.

### ActiveX Controls

ActiveX controls, originally known as OLE controls or OCXs, are standalone software components that perform specific functions in a standard way. ActiveX controls define standard interfaces for reusable components. ActiveX controls are not separate applications, instead, they are servers that are placed into a control container. To use ActiveX controls, they must be placed in an ActiveX container. InTouch is an ActiveX container. Microsoft VisualBasic and Internet browsers are also ActiveX containers.

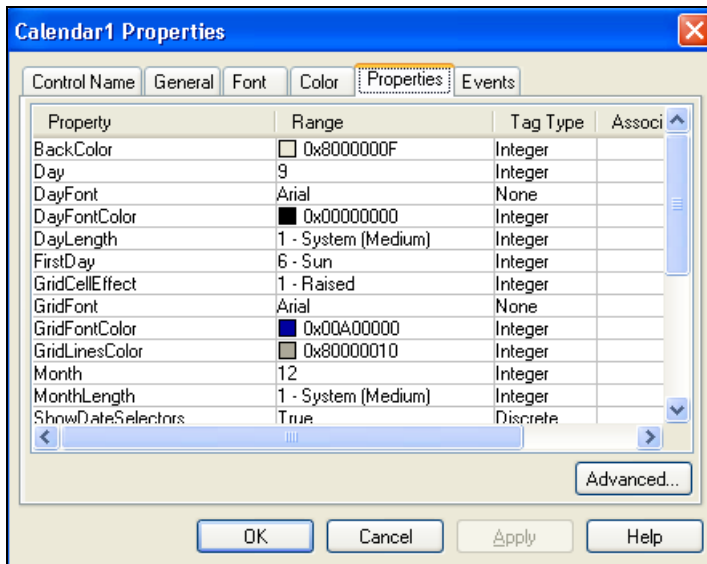
ActiveX controls behave exactly like InTouch Wizards, except they bring compelling new functionality to InTouch applications. ActiveX controls can be created using Visual Basic, Visual C, Delphi or other 3rd party development tools. ActiveX controls can also be bought from third-parties for specific functionality.

There are three main components of ActiveX controls:

- **Properties** are very similar to variables that can be modified (e.g., Calendar.day, Control.height)
- **Methods** are similar to script functions that can be called from the container. For example, Browser.Navigate("http://www.wonderware.com"), Engine.start()
- **Events** occur through the ActiveX container. For example, Control.click (shift), FileViewer.DoubleClick (name), etc.

## ActiveX Control Properties Configuration

The properties that can be configured for a particular ActiveX control are determined by the ActiveX control designer. Each ActiveX control's **Properties** property sheet displays three columns: **Property**, **Range**, and **Associated Tag**. The Property and Range columns are read-only. The Associated Tag column is used to associate InTouch tagnames with the respective property in the Property column.



## ActiveX Control Methods

ActiveX control methods are similar to ActiveX control properties. Methods are activated in runtime (WindowViewer). ActiveX control methods are accessed through ActiveX Event scripts and/or other InTouch QuickScripts.

---

**Note:** In order for an ActiveX Event script to function properly, the ActiveX control for which the script was created must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts or any other InTouch QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

---

## ActiveX Control Event Parameters

ActiveX control events can be executed in runtime (WindowViewer) by designing a particular action and associating it to the event. For example, if the ActiveX control has an error event handler, create an ActiveX Event script that displays a window with an error message when an error occurs. ActiveX Event scripts are provided to support event actions. Associate a named event script to each event.

---

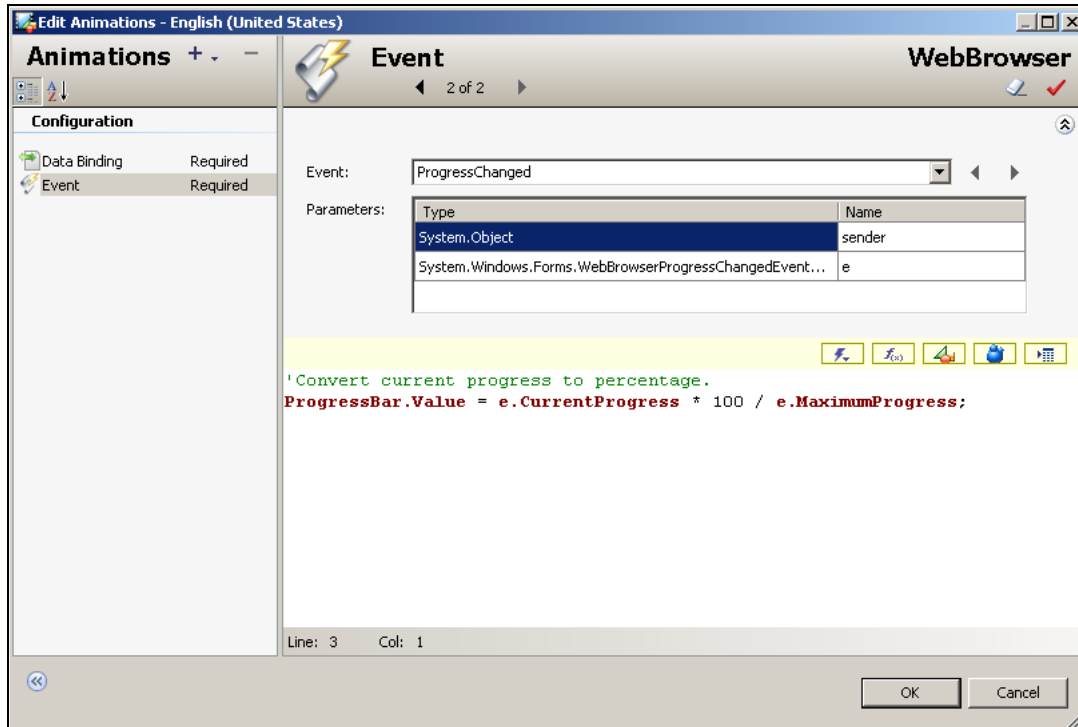
**Note:** In order for an ActiveX Event script to function properly, the ActiveX control for which the script was created must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts (or any other InTouch QuickScripts containing script functions associated with that ActiveX control) will not execute properly.

---

## ActiveX Event Scripts

Most ActiveX controls have events associated with them. For example, click, double-click, mouse down, and key press are typical events used in many ActiveX controls. InTouch ActiveX Event scripts are provided to support event actions. One ActiveX Event script can be associated with each event. ActiveX control events are executed in runtime (WindowViewer).

Use an Event animation to configure an ArchestrA script that is executed when a client control event occurs.

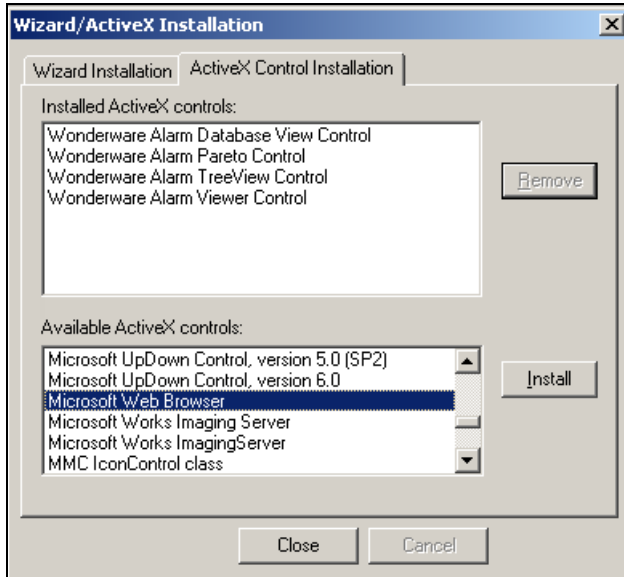


## Reuse ActiveX Event Scripts

ActiveX Event scripts can only be reused for the same event for the same kind of ActiveX control. For example, the mouse down event may be a stock event on hundreds of ActiveX controls. However, an ActiveX Event script written for mouse down on ActiveX ControlA cannot be reused for mouse down on ActiveX ControlB, unless the two controls are of the same type.

## Wizard/ActiveX Installation

One or more ActiveX controls can be used in an InTouch application. InTouch allows easy installation or removal of available ActiveX controls by using the Wizards/ActiveX Installation feature.



---

**Note:** ActiveX Event scripts can also be imported from one application to another.

---

## Importing ActiveX Event Scripts

Importing ActiveX Event scripts from another InTouch application to a current application can save considerable amounts of development time. When ActiveX Event scripts is moved from one InTouch application to another, the **File/Import** command must be used.

---

**Note:** When importing ActiveX Event scripts from one application to another, all ActiveX Events scripts are imported. Additionally, in order for an imported ActiveX Event script to function properly in the new application, the same ActiveX control and the same event for which the script was originally created must also be used in the new application, and it must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts or any other InTouch QuickScripts containing script functions associated with that ActiveX control will not execute properly.

---

# **Lab 10 – Using an ActiveX Control as a Universal Document Viewer**

## **Introduction**

In this lab, you will use the Microsoft Web Browser ActiveX Control inside an InTouch application as a way to view multiple document formats.

## **Objectives**

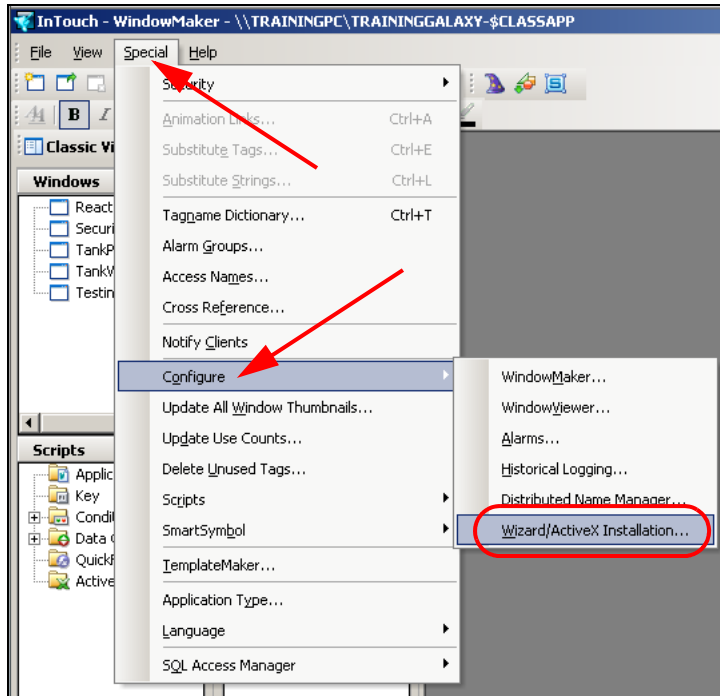
Upon completion of this lab, you will be able to:

- Load ActiveX controls into the application
- Add InTouch graphics and scripting to work with the ActiveX control
- Use various element types such as: PDF, hyperlink, JPG, and video to show how the ActiveX control works

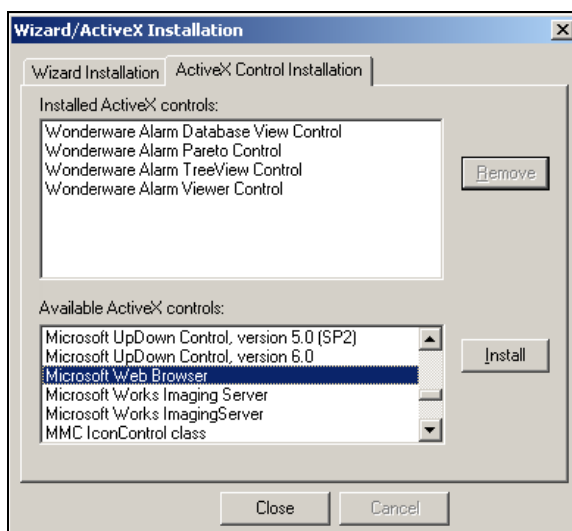
## Wizard/ActiveX Installation

The first step you take in order to utilize the Microsoft Web Browser ActiveX Control inside of your InTouch application, is to make the control available from within WindowMaker.

1. In WindowMaker, on the **Special** menu, click **Configure | Wizard/ActiveX Installation**.



2. In the **ActiveX Control Installation** tab, in the **Available ActiveX controls** pane, select **Microsoft Web Browser**, and then click **Install**.



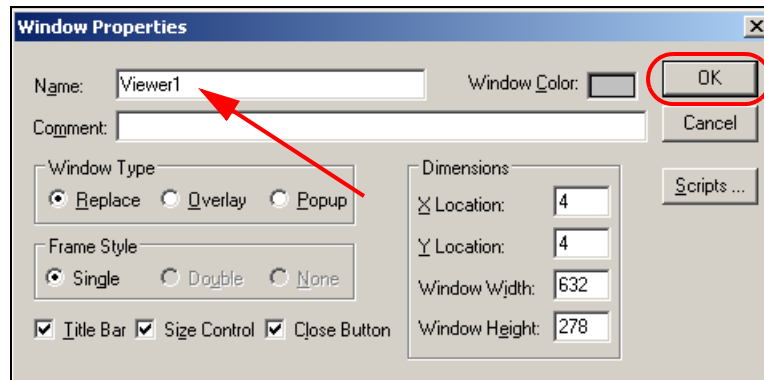
**Note:** This will move **Microsoft Web Browser** from the list of **Available ActiveX** controls to **Installed ActiveX** controls.

3. Click **Close**.

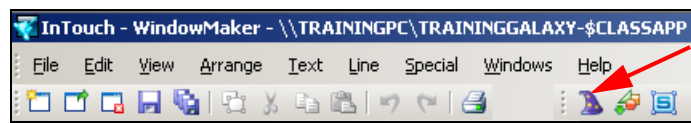


Next, create a new window in WindowMaker to place the ActiveX control.

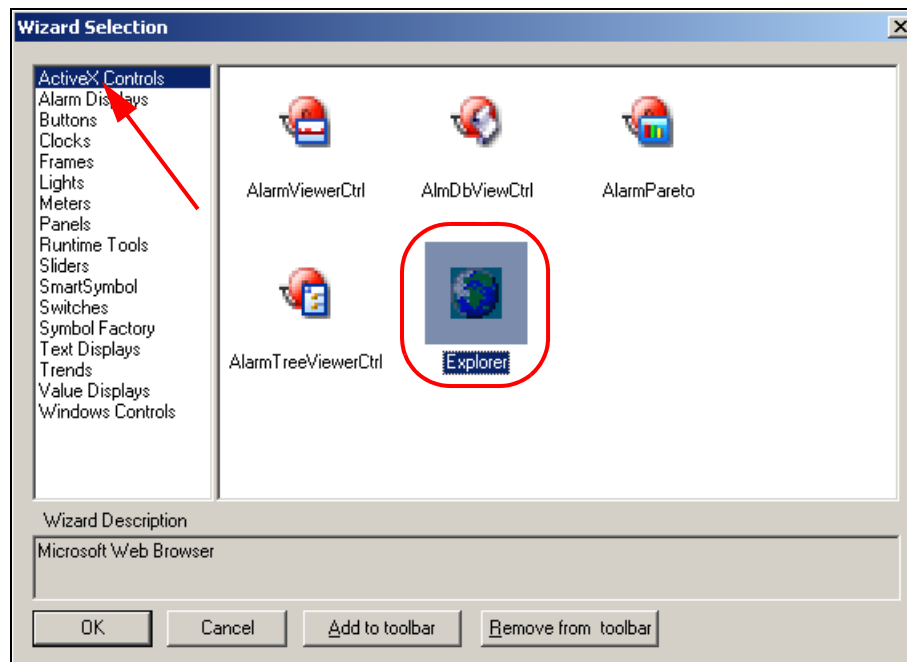
4. In WindowMaker, on the **File** menu, click **New Window**.
5. Name the window **Viewer1**, and then click **OK**.



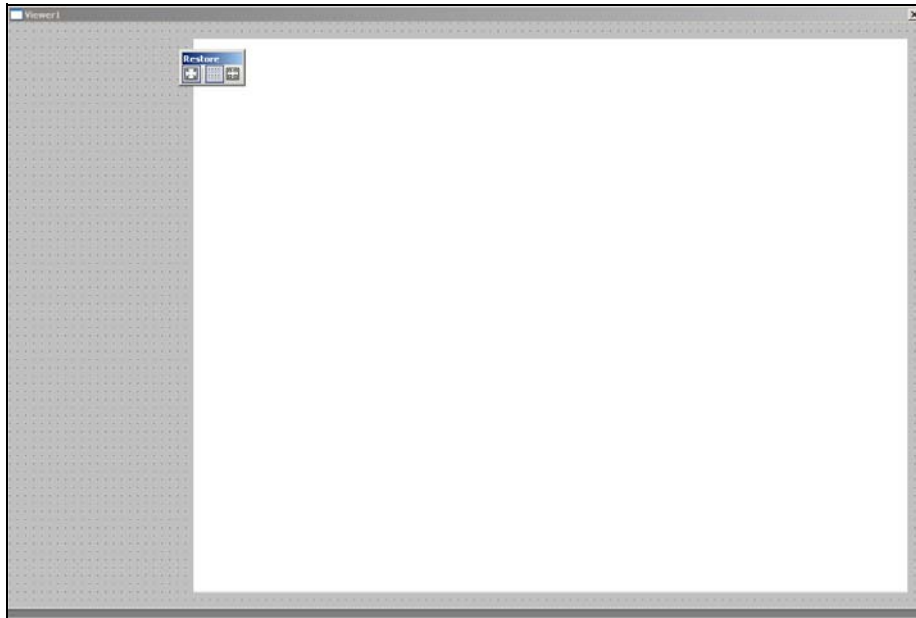
6. Close all other open windows.
7. Click **Fullscreen** and enlarge **Viewer1** to fill the viewable area.
8. Exit full screen mode.
9. Click **Wizards**.



10. Select **ActiveX Controls**, and then double-click **Explorer**.



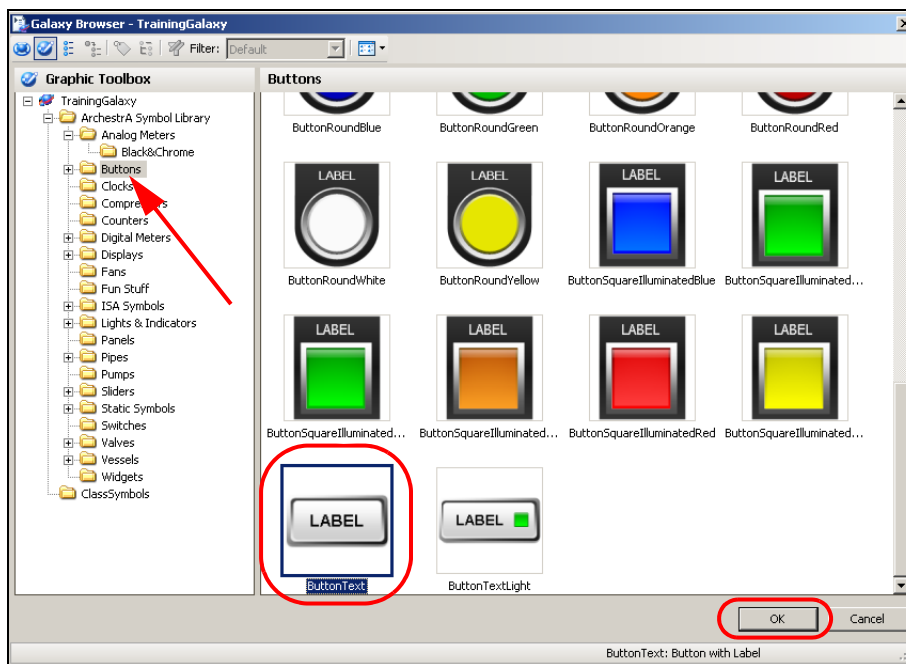
11. In the **Viewer1** window, place the **Explorer**, and then click **Fullscreen**.
12. Enlarge **Explorer** to fill most of the visible space, and then exit full screen mode.



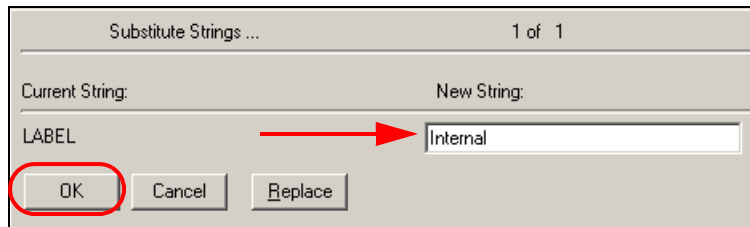
After placing **Explorer** in the **Viewer1** window, you will add five buttons to control which documents to view inside the **Explorer ActiveX control**.

### Internal

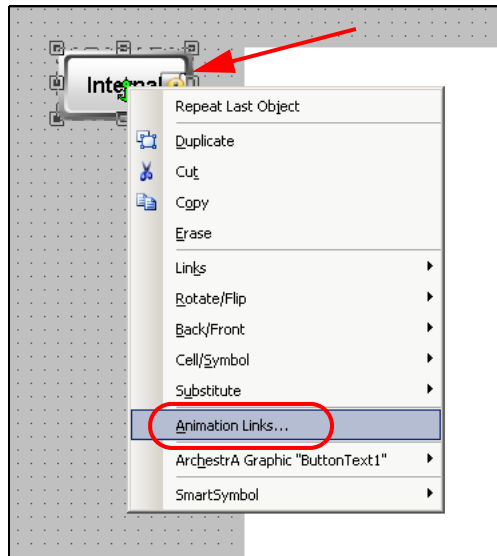
13. Click **Embed Archestra Graphic**.
14. In **Archestra Symbol Library**, select **Buttons**, and then double-click **ButtonText**.



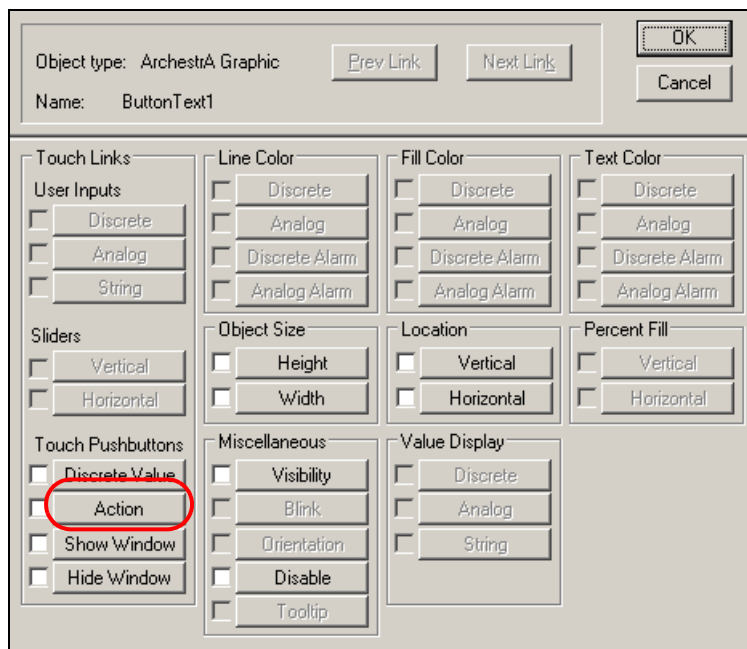
15. In the **Viewer1** window, to the left of **Explorer**, place **ButtonText** and size it if necessary.
16. Press **Ctrl+L** and change **LABEL** to **Internal**, and then click **OK**.



17. Right-click on the button and select **Animation Links**.



18. Click **Action**.

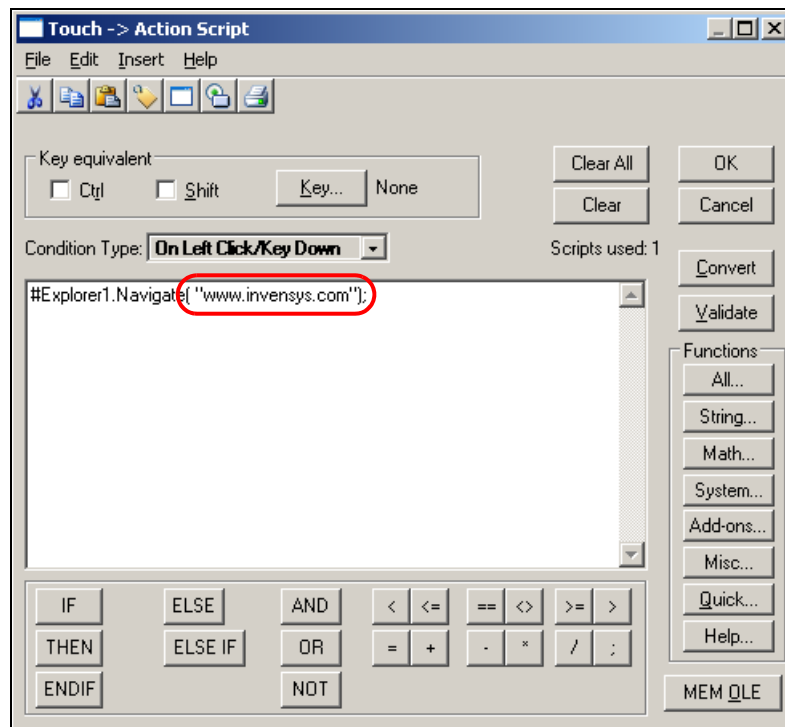


- 
- The screenshot shows the 'Touch -> Action Script' dialog box. At the top, there is a menu bar with 'File', 'Edit', 'Insert', and 'Help'. A red arrow points to the 'Help' menu. Below the menu bar is a toolbar with icons for various actions. The 'Key equivalent' section has checkboxes for 'Ctrl' and 'Shift', and a 'Key...' button. The 'Condition Type' is set to 'On Left Click/Key Down'. On the right, there are buttons for 'Clear All', 'Clear', 'OK', 'Cancel', 'Convert', 'Validate', 'Functions' (with sub-options: All..., String..., Math..., System..., Add-ons..., Misc..., Quick..., Help...), and 'MEM QLE'.

- 
- ActiveX Control Browser
- | Control Name | Method / Property                                                  |
|--------------|--------------------------------------------------------------------|
| Explorer1    | FullScreen                                                         |
|              | GetProperty( "String" )                                            |
|              | GoBack()                                                           |
|              | GoForward()                                                        |
|              | GoHome()                                                           |
|              | GoSearch()                                                         |
|              | Height                                                             |
|              | Hwnd                                                               |
|              | Left                                                               |
|              | LocationName                                                       |
|              | LocationURL                                                        |
|              | MenuBar                                                            |
|              | Name                                                               |
|              | Navigate( "String", "Pointer", "Pointer", "Pointer", "Pointer" )   |
|              | Navigate2( "Pointer", "Pointer", "Pointer", "Pointer", "Pointer" ) |
|              | Offline                                                            |
|              | Parent                                                             |
|              | Path                                                               |
|              | Quit()                                                             |
|              | ReadyState                                                         |
|              | Refresh()                                                          |
- OK Cancel

21. Modify the script as follows:

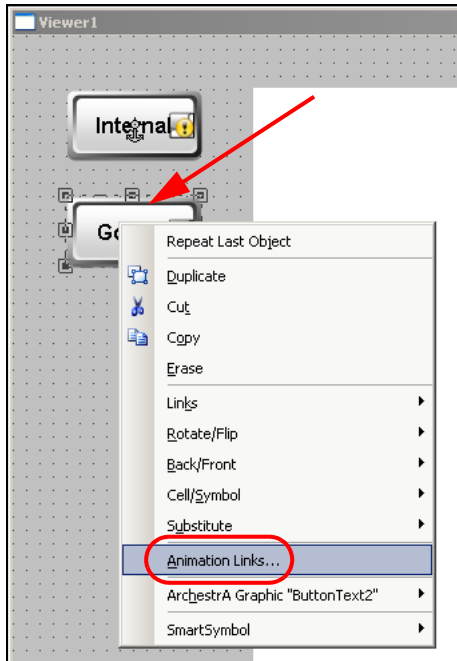
```
#Explorer1.Navigate("www.invensys.com");
```



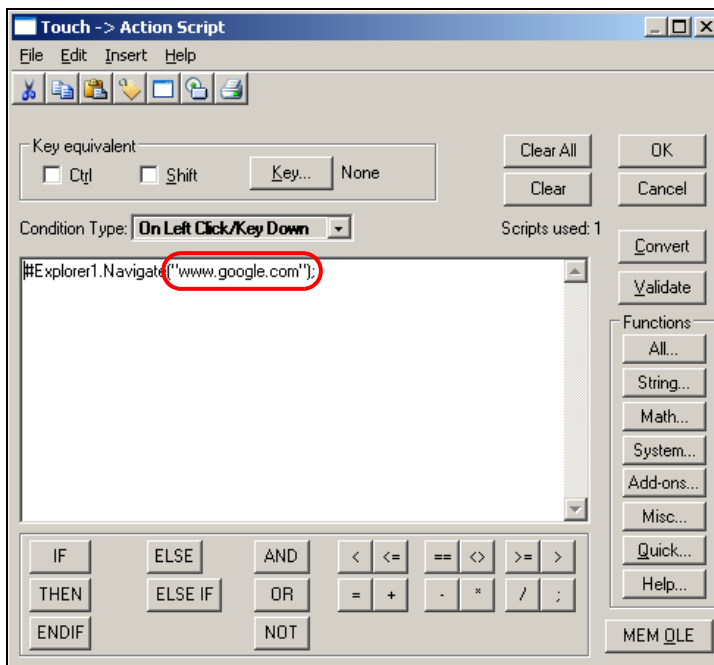
22. Click **OK** to exit the script editor, and then click **OK** to exit the animation links.

## Google

23. Duplicate **Internal**, and then place the new button directly below it.
24. Press **Ctrl+L** to change **Internal** to **Google**.
25. Right-click **Google** and select **Animation Links**.



26. Click **Action**.
27. Change **www.invensys.com** to **www.google.com**.



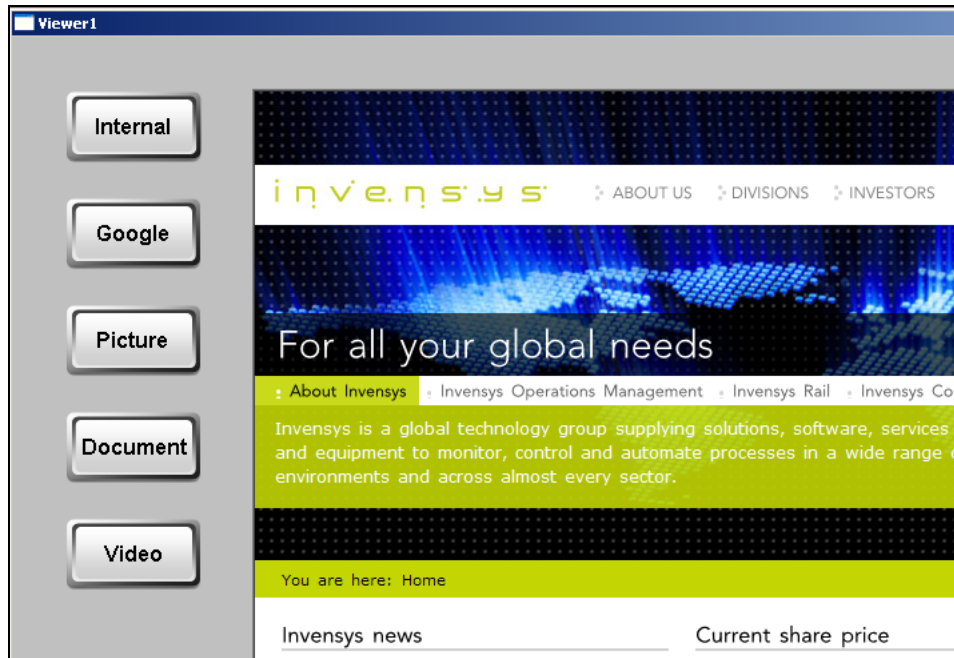
28. Click **OK** to exit the script editor, and then click **OK** to exit the animation links.

### Picture, Document, and Video Links

Now that you have created the first two buttons; Internal and Google, you will use the same process to create three more buttons; Picture, Document, and Video.

29. Duplicate the **Google** button three times, label them **Picture**, **Document**, and **Video**.

Arrange them as shown here:



30. Right-click each and select **Animation Links | Action**.

31. Configure as follows:

| Button   | Action Scripting                                        |
|----------|---------------------------------------------------------|
| Picture  | #Explorer1.Navigate("C:\training\intouch.bmp");         |
| Document | #Explorer1.Navigate("C:\training\ide.pdf");             |
| Video    | #Explorer1.Navigate("C:\training\video\wildlife.html"); |

### Test What You Have Built

32. Click **Runtime**.
33. Click each of the buttons to ensure they are functioning as expected.
34. Click **Development!**.





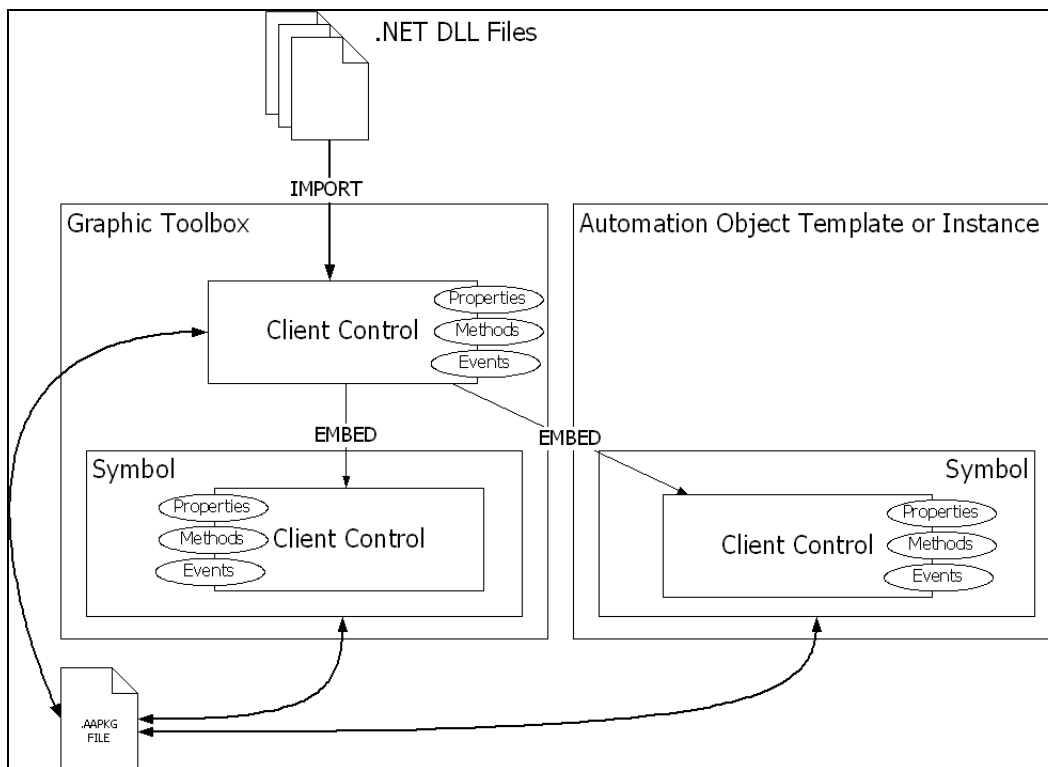
## Section 5 – .NET Controls

This section describes how to import, configure, and use .NET Client Controls in InTouch.

### About Client Controls

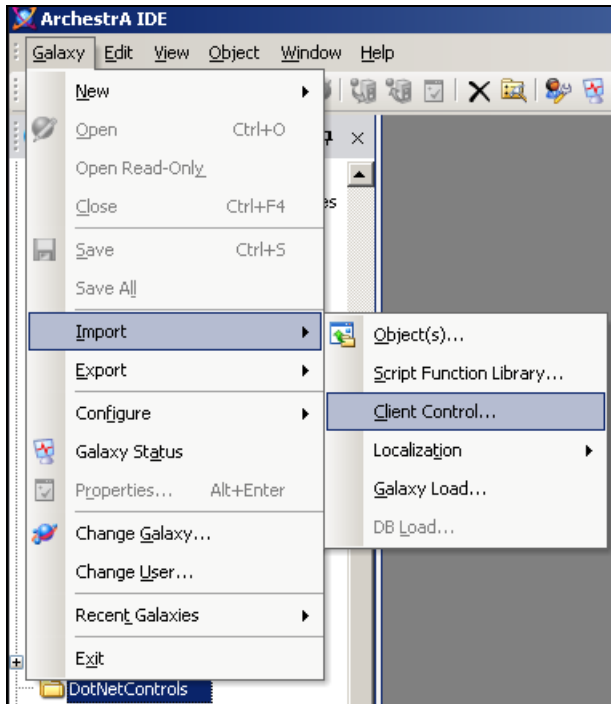
Client Controls provide functionality contained in third party .NET applications that are used in symbols. To use this functionality:

- Import the .DLL file associated with the client control. Any client control contained in the .DLL file is imported into the Graphic Toolbox.
- Browse and embed one or more of the client controls in a new or existing symbol. The client controls appear as elements.
- View and edit the exposed client control properties.
- Bind the client control properties to ArchestrA attributes, symbol custom properties, or InTouch tagnames. This can be done with the Data Binding animation.
- Configure scripts for client control events. This can be done with the Events animation.
- Optionally use the symbol containing the embedded client control in an IDE-managed InTouch application.



## Importing Client Controls

Import client controls into the Graphic Toolbox from .NET Dynamic Link Library (.DLL) files. After importing client controls, organize them in the Graphic Toolbox, as with ArchestrA symbols. Import client controls that have previously been exported in an ArchestrA package (.aaPKG) file.



## Importing Previously Exported Client Controls

Import one or more previously exported client controls from an ArchestrA package file (.aaPKG). Previously the client controls may have been:

- Exported on their own
- Embedded in a symbol and the symbol was exported
- Embedded in a symbol and contained in an Automation object instance or template and the Automation object was exported

To import a previously exported ArchestrA package containing one or more client controls, import the exported client control(s) the same way as importing an Automation object (.aaPKG).

## Organizing and Embedding Client Controls

Organize the client controls within the Graphic Toolbox the same way as with ArchestrA Graphics.

You can:

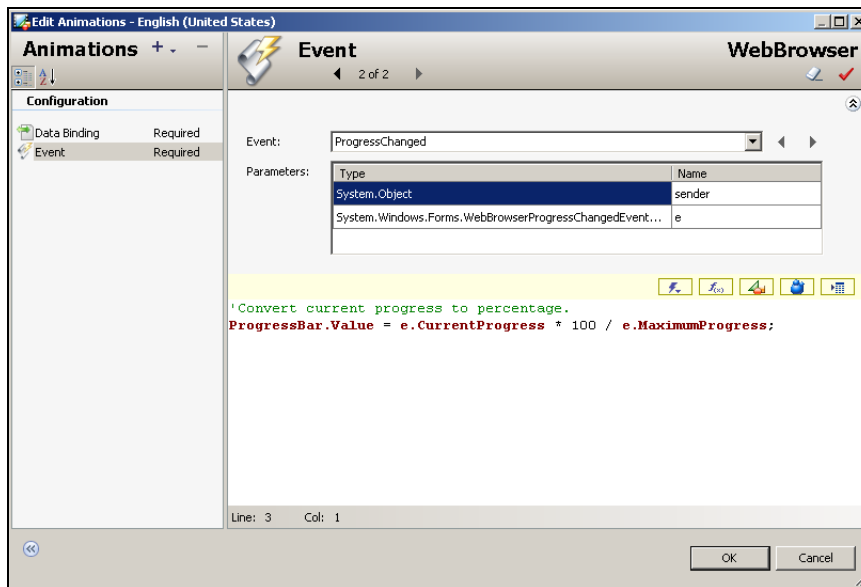
- Rename Client Controls
- Move Client Controls in and out of Graphic Toolsets
- Delete Client Controls

Embed an installed Client Control in a symbol as embedding a symbol within another symbol. However, it is recommended not to overlap client controls with other elements on the canvas, otherwise the client controls may not work correctly.

When embedding a client control into a symbol, the native properties of the client control are imported into the Properties Editor in the Misc group.

## Event Scripts

Lastly, by using an Event animation, you can configure an ArchestrA script that is executed when a client control event occurs.





# Lab 11 – Using .Net Controls

## Introduction

In this lab, you will be using a client control from the Microsoft .NET Framework within an ArchestrA symbol. With the help of custom properties, scripting, and animations, you will recreate the basic functionality of a Web Browser.

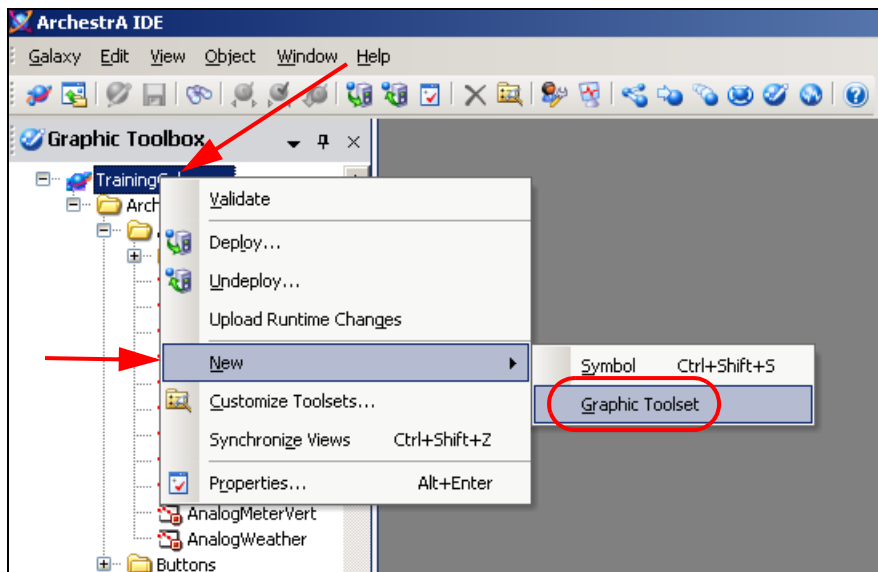
## Objectives

Upon completion of this lab, you will be able to:

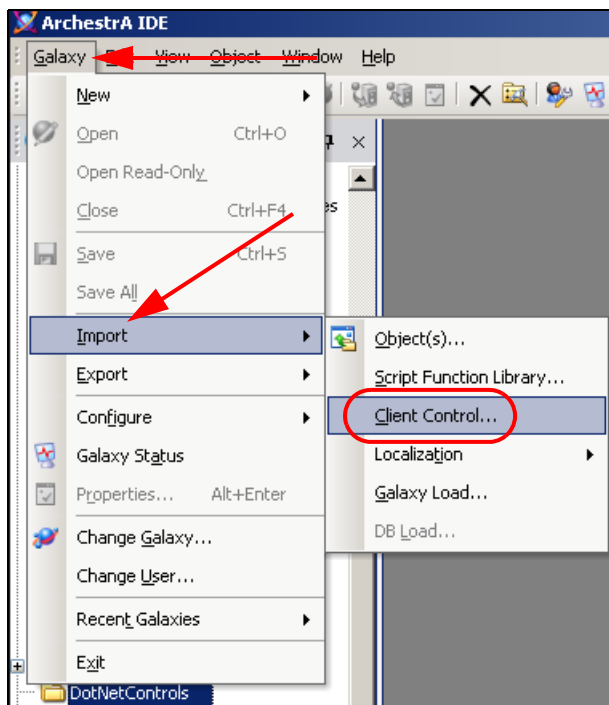
- Import .Net client controls
- Create a new symbol using two .Net controls
- Create web navigation controls
- Show interaction via both controls
- Document your scripts and custom properties

## Import the .NET Framework Controls

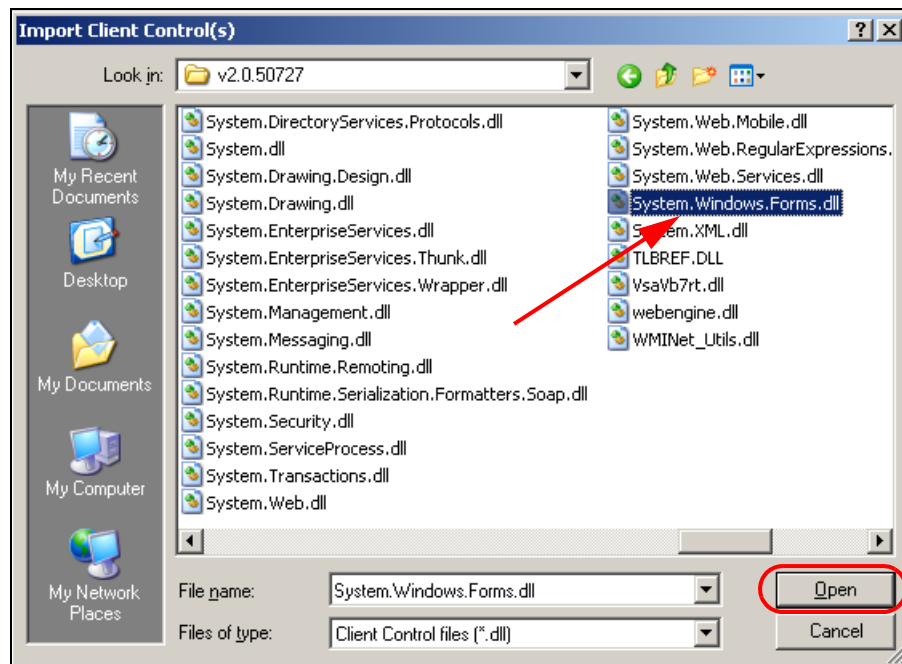
1. In the ArchestrA IDE **Graphic Toolbox**, right-click **TrainingGalaxy** and select **New | Graphic Toolset**.



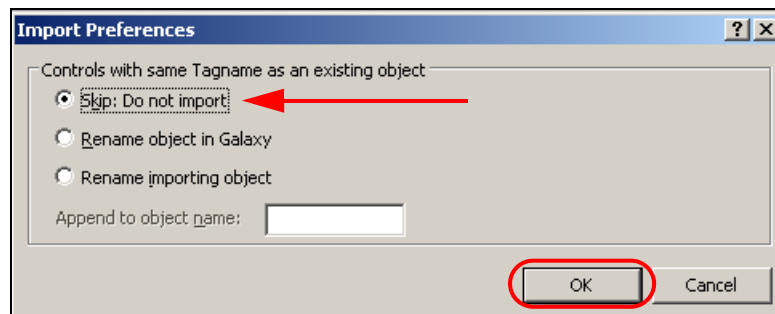
2. Name the new toolset **DotNetControls**.
3. On the **Galaxy** menu, select **Import | Client Control**.



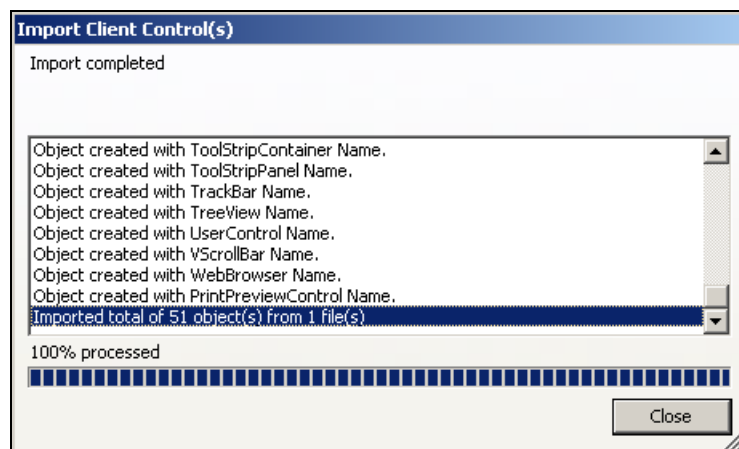
4. Open `C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.Windows.Forms.dll`.



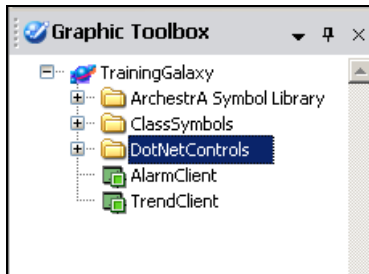
5. Ensure **Skip: Do not import** is selected, and then click **OK**.



6. When the import is complete, click **Close**.

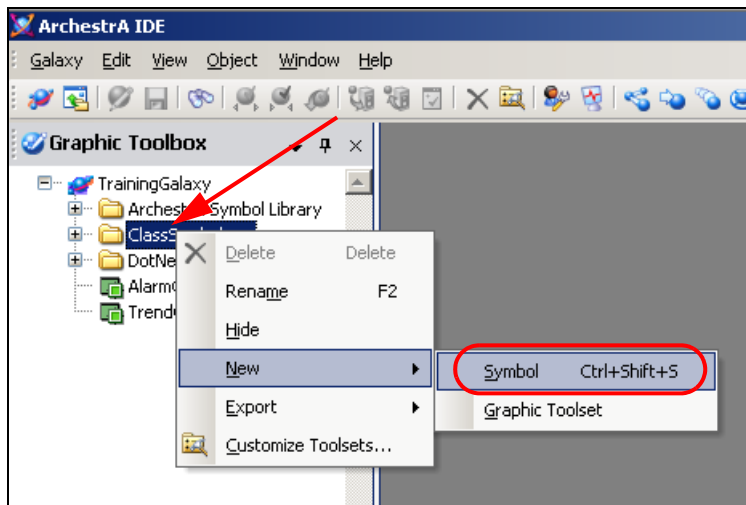


7. Move the **.NET Framework** controls except for **AlarmClient** and **TrendClient**, to the **DotNet Controls** graphic toolset.



## Create a Web Browser Symbol

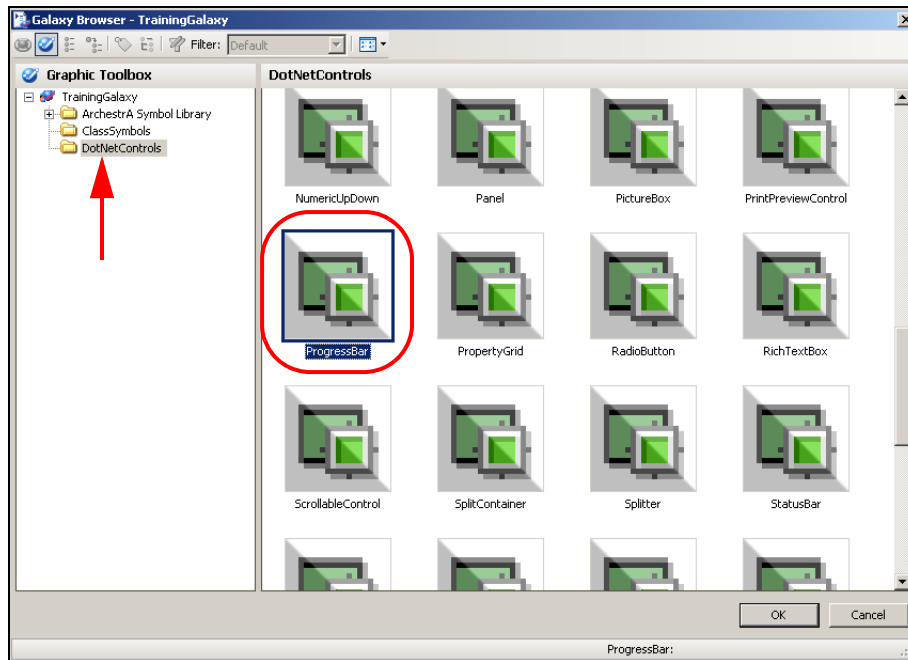
8. Right-click **ClassSymbols** and select **New | Symbol**.



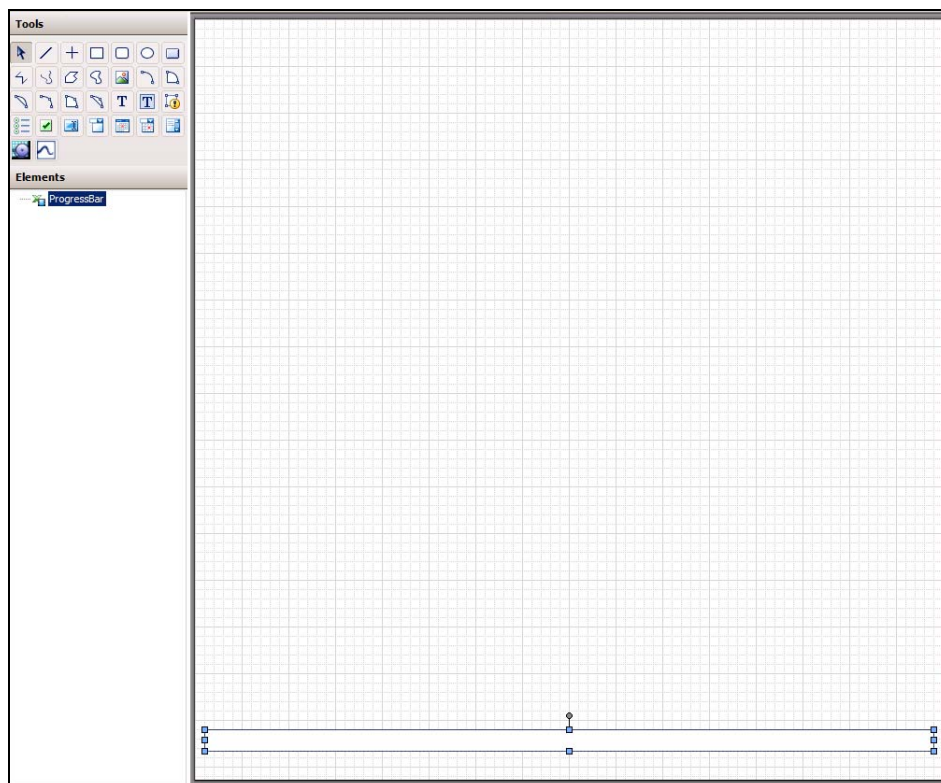
9. Name the symbol **WebBrowserMini**, and then double-click it for editing.
10. Click **Embed ArchestrA Graphic**.



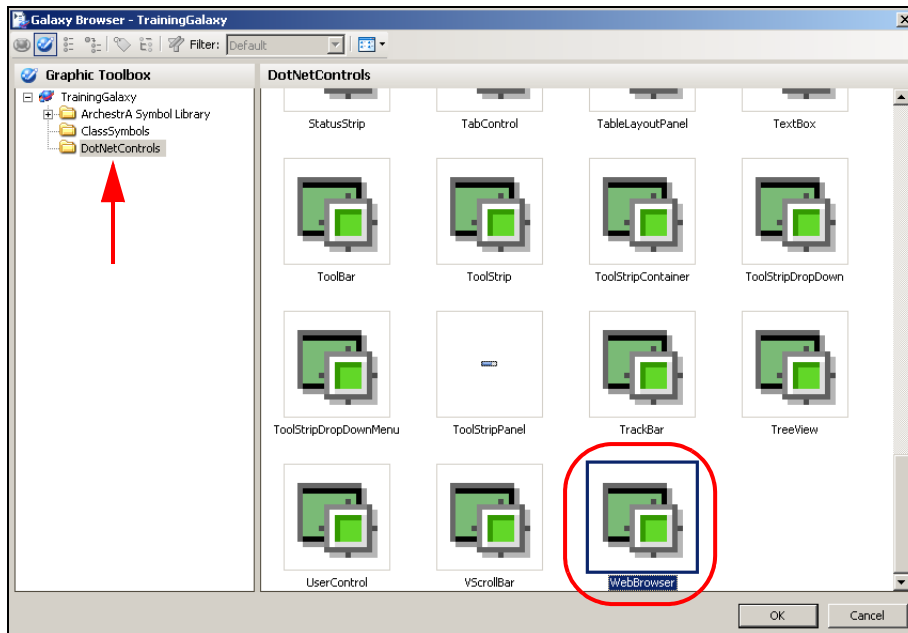
11. Select **DotNet Controls**, and then double-click **ProgressBar**.



12. At the bottom-center of the canvas, embed **ProgressBar**, and then resize as needed.
13. In the right pane, change the name to **ProgressBar**.

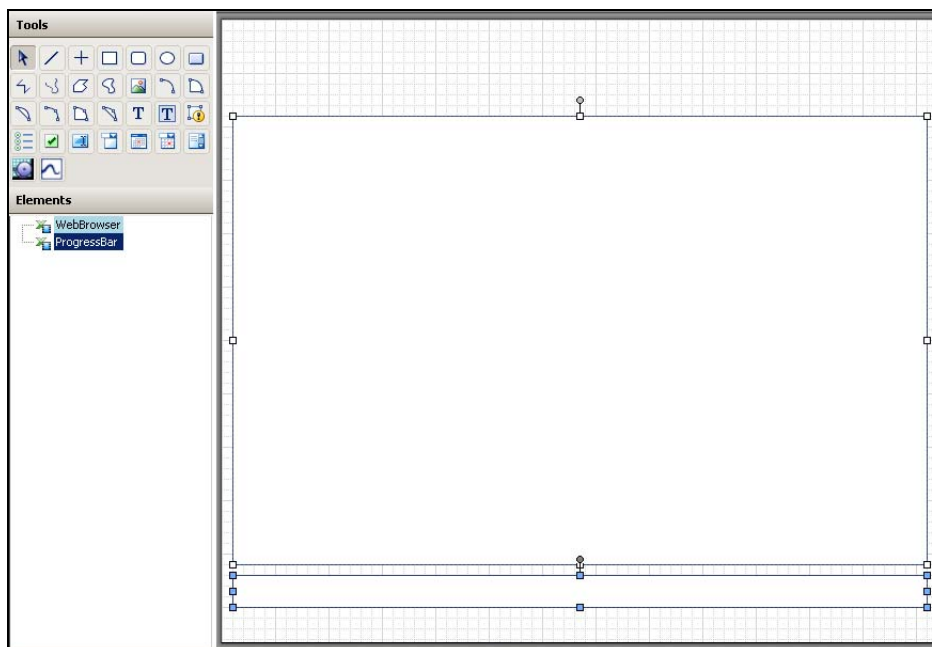


14. Click **Embed ArchestraA Graphic**.
15. Select **DotNet Controls**, and then double-click **WebBrowser**.

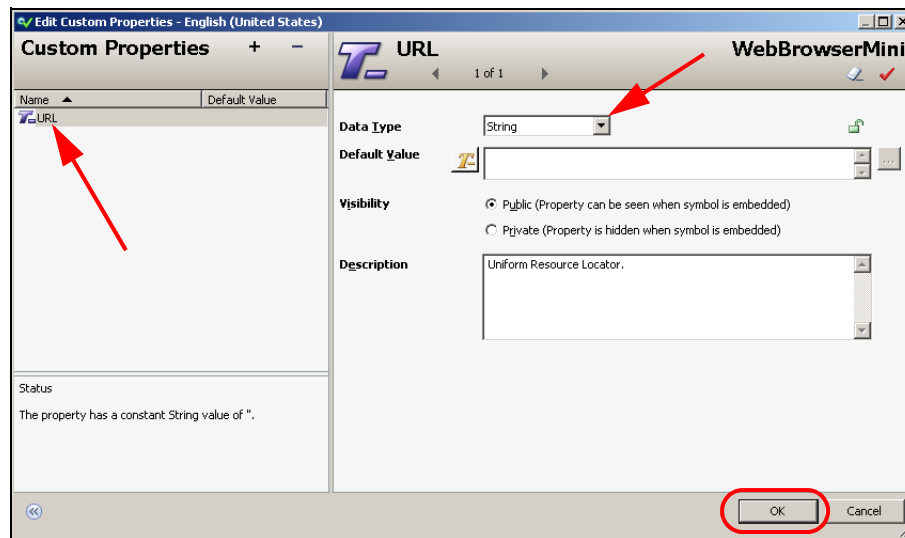


16. Directly above the **ProgressBar**, embed **WebBrowser**, and then resize as needed.
17. In the right pane, change the name to **WebBrowser**.

Your symbol should look as shown below:



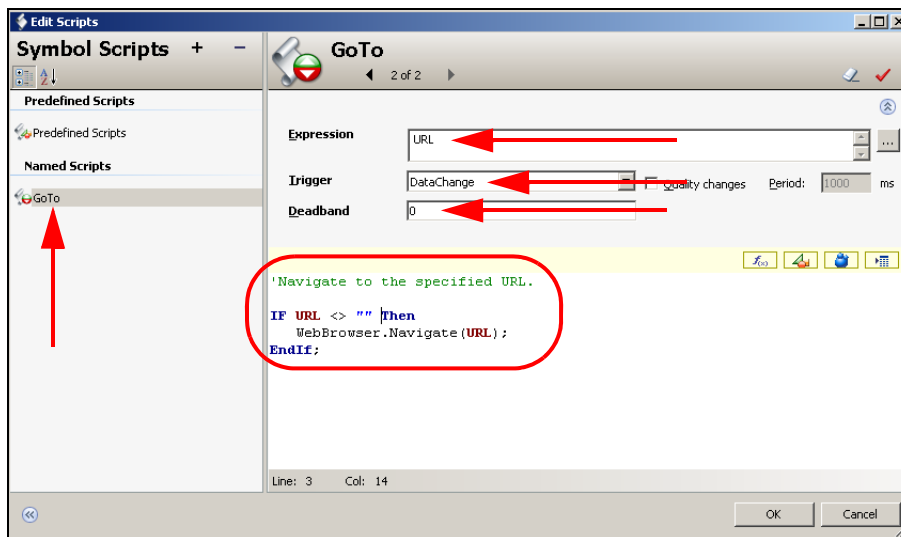
18. On the canvas, right-click and select **Custom Properties**.
19. Click **+** and name the new custom property **URL**.
20. In the **Data Type** field, select **String**, and then click **OK**.



## Add Scripts

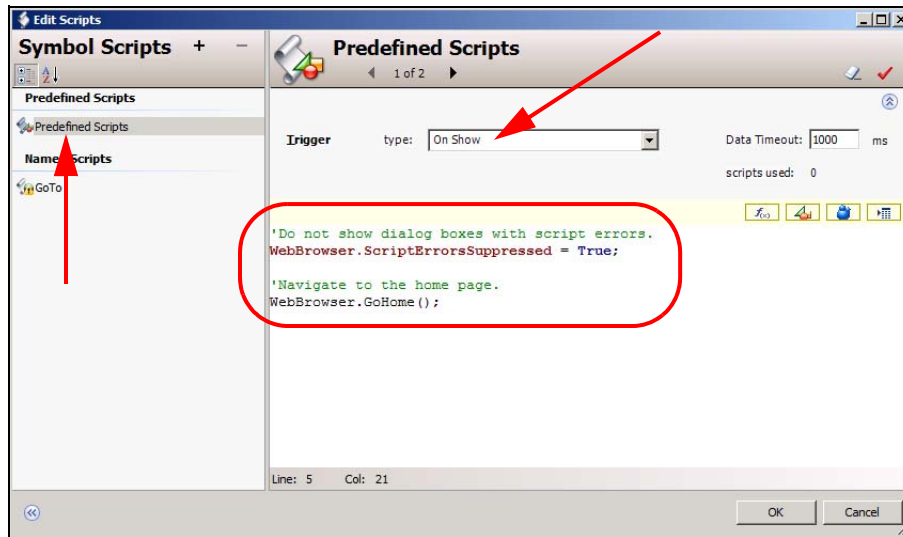
**Note:** All scripts for this lab are available to copy from the file **Lab 11 Scripts** in your **Training** directory.

21. Right-click on the canvas and select **Scripts**.
22. Click **+** and name the new script **GoTo**, in the **Expression** field enter **URL**, From the **Trigger** drop-down select **DataChange**, and then set the **Deadband** to **0**.
23. In your **Training** folder, open the **Lab 11 Scripts** file, and copy the section between **GoTo** **Named Scripting** and **On Show Predefined Scripting**:

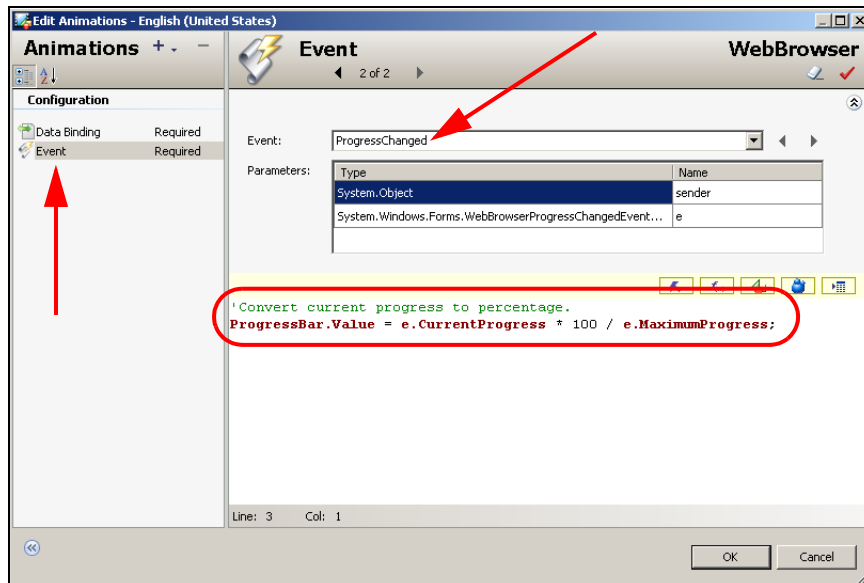


**Note:** For the purposes of this lab, comments have been added to the script to show an optional best practice when writing script. These comments do not alter the way the script behaves but provide a description as to what the script does. They are designated by a single quote mark.

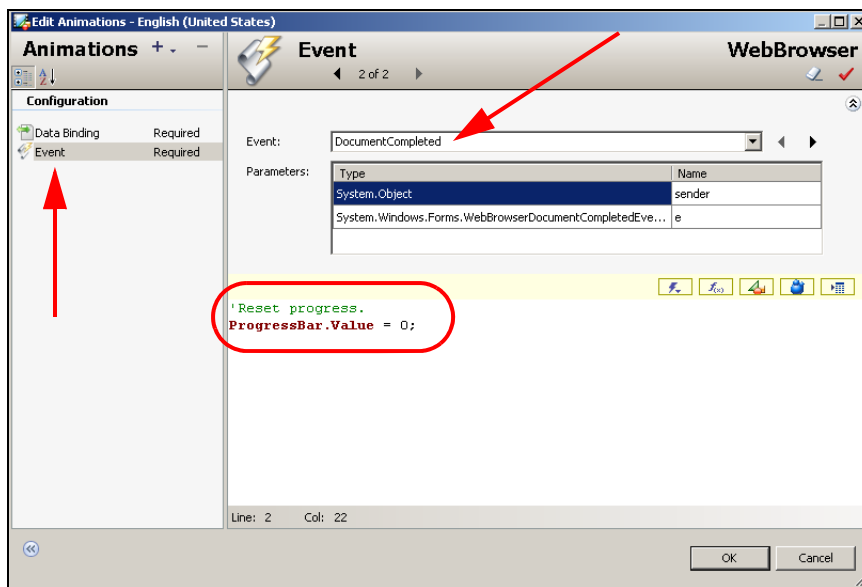
24. Click **Predefined Scripts**, and then from the **Trigger** drop-down select **On Show**.
25. In your **Training** folder, open the **Lab 11 Scripts** file, and copy the section between **On Show Predefined Scripting** and **ProgressChanged Event Scripting**:



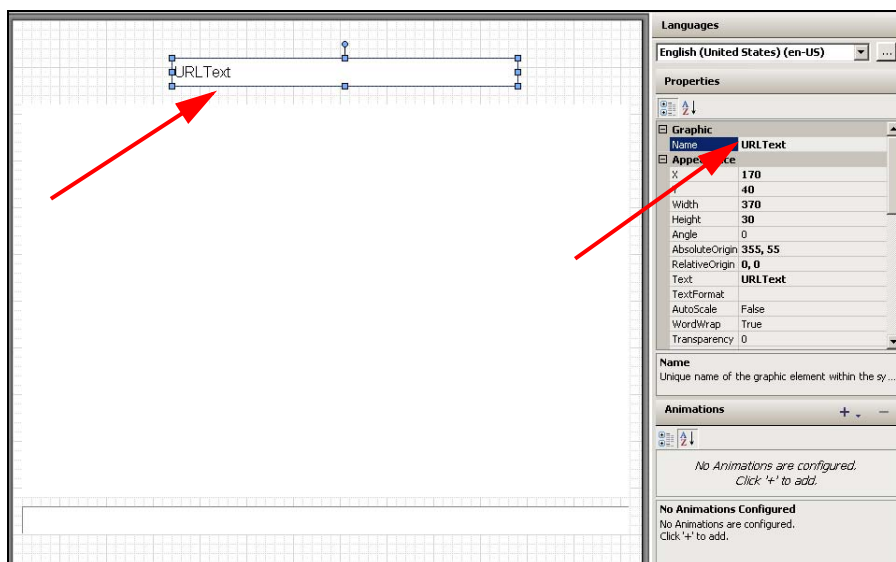
26. Click **OK**.
27. Double-click the **WebBrowser** element.
28. In the **Configuration** pane, select **Event**, and then from the **Event** drop-down select **ProgressChanged**.
29. In your **Training** folder, open the **Lab 11 Scripts** file, and copy the section between **ProgressChanged Event Scripting** and **DocumentCompleted Event Scripting**.



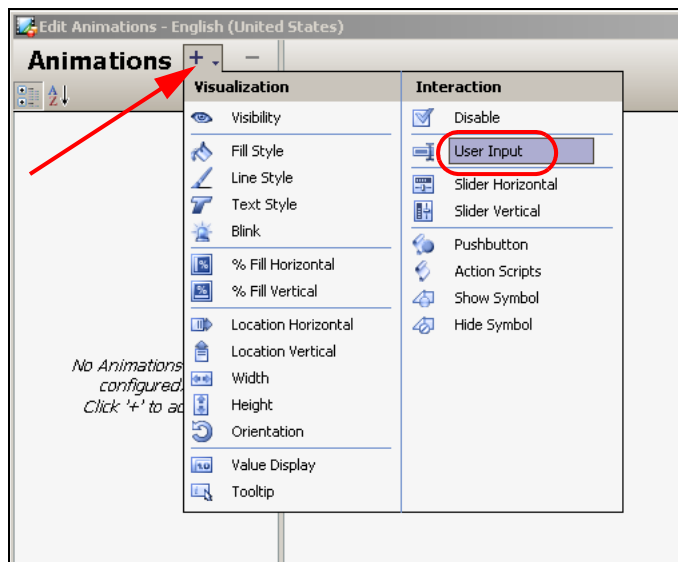
30. In the **Event** drop-down, select **DocumentCompleted**.
31. In your **Training** folder, open the **Lab 11 Scripts** file, and copy the section between **DocumentCompleted Event Scripting** and **BackButton Action Script**.



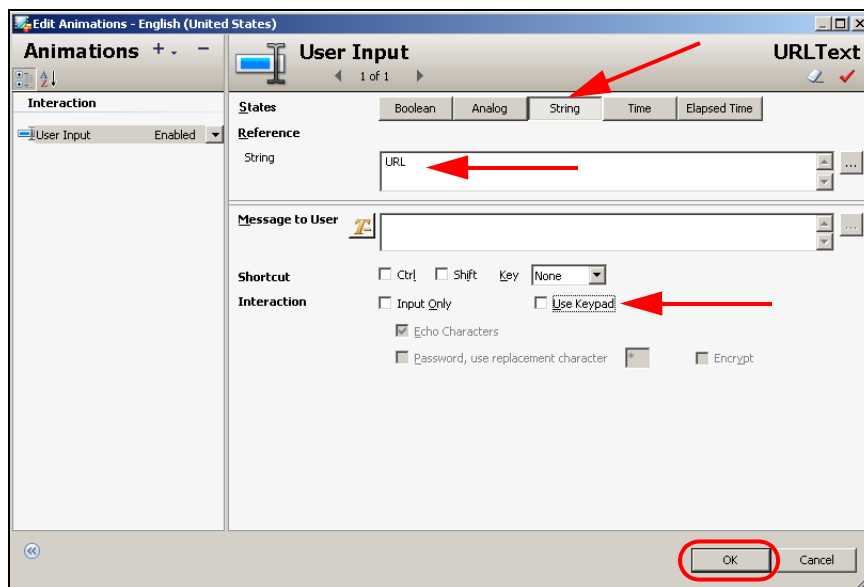
32. Click **OK**.
33. Add a **Text Box** containing the text **URL Text** across the top-center of the **WebBrowserMini** symbol.
34. Name the new text box **URLText**, and then double-click it for editing.



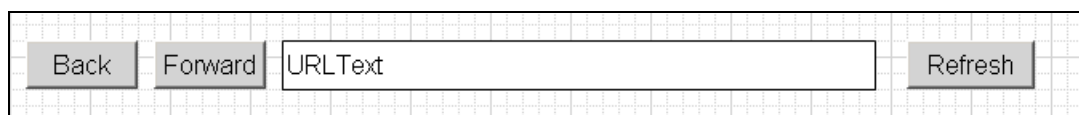
35. Click **+** and select **User Input**.



36. Click **String**, in the **String** field enter **URL**, uncheck **Use Keypad**, and then click **OK**.

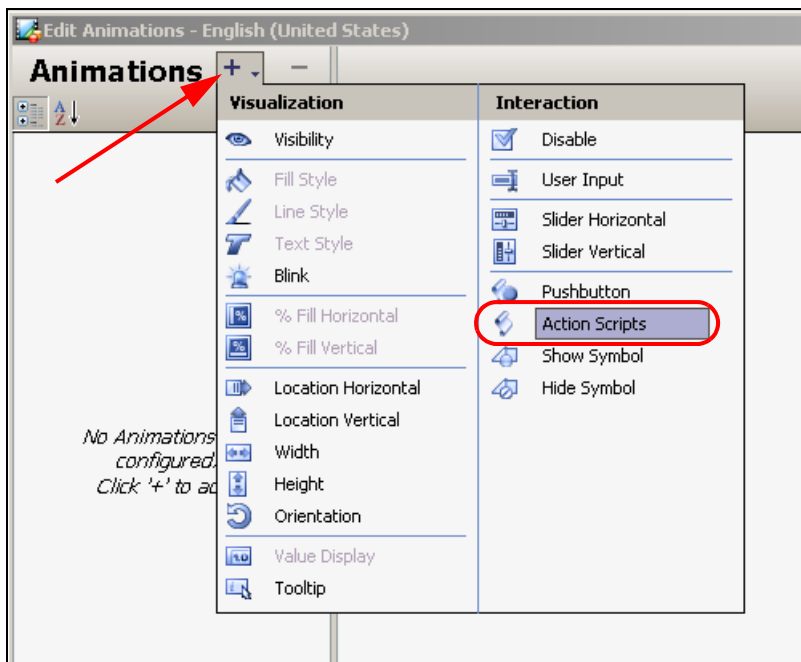


37. In the top-left, draw a button and label it **Back**, and then in the right pane name it **BackButton**.
38. To the right of **Back**, draw a button and label it **Forward**, and then in the right pane name it **ForwardButton**.
39. To the right of the **URL** box, draw a third button and label it **Refresh**, and then in the right pane name it **RefreshButton**.

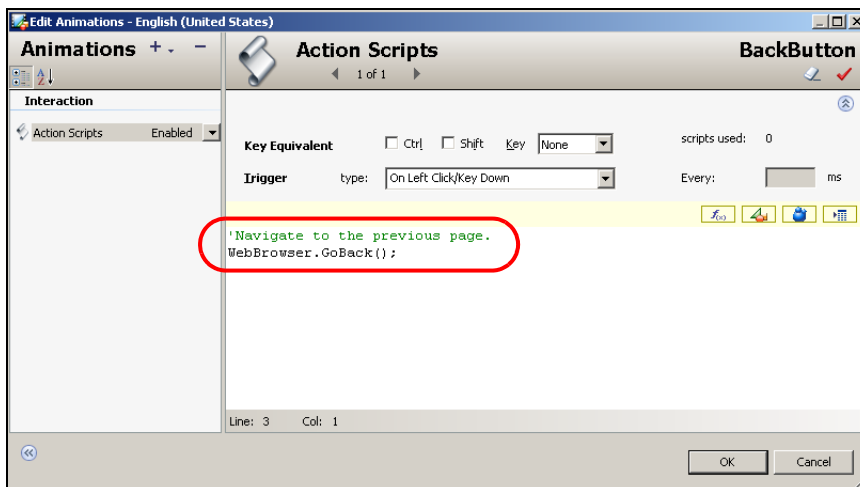


40. Double-click **BackButton** to open it for editing.

41. Click + and select **Action Scripts**.



42. In your **Training** folder, open the **Lab 11 Scripts** file, and copy the section between **BackButton Action Script** and **ForwardButton Action Script**:



43. Click **OK**.
44. Add **Action Script** animations to the remaining two buttons configured as follows (You can enter manually or cut and paste these scripts from the **Lab 11 Scripts** file):

| Element Name  | Trigger                | Script                                                 |
|---------------|------------------------|--------------------------------------------------------|
| ForwardButton | On Left Click/Key Down | 'Navigate to the next page.<br>WebBrowser.GoForward(); |
| RefreshButton | On Left Click/Key Down | 'Refresh the current page.<br>WebBrowser.Refresh();    |

45. Click **Save and Close**, and check in the symbol.

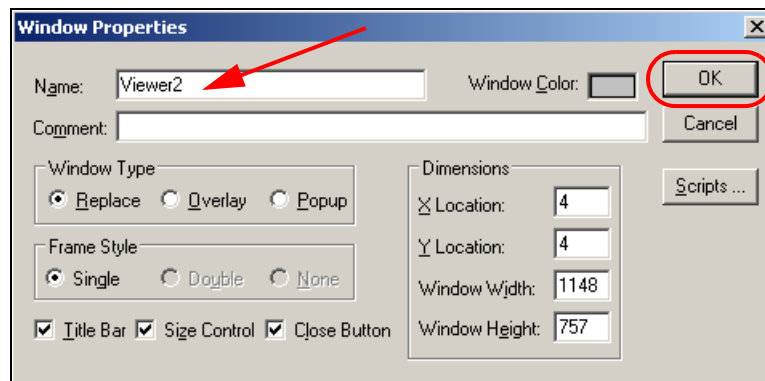


## Embed the Symbol in WindowMaker

Next, create a new window in WindowMaker to place the symbol you just created.

46. In WindowMaker, on the **File** menu, click **New Window**.

47. Name the window **Viewer2**, and then click **OK**.

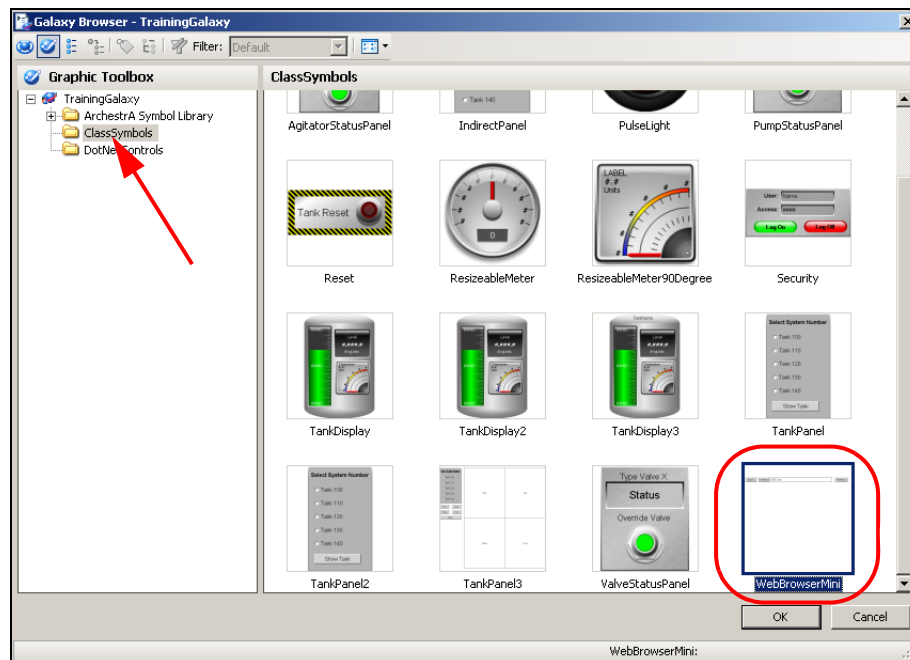


48. Click **Fullscreen** and enlarge **Viewer2** to fill the viewable area.

49. Exit full screen mode.

50. Click **Embed ArchestrA Graphic**.

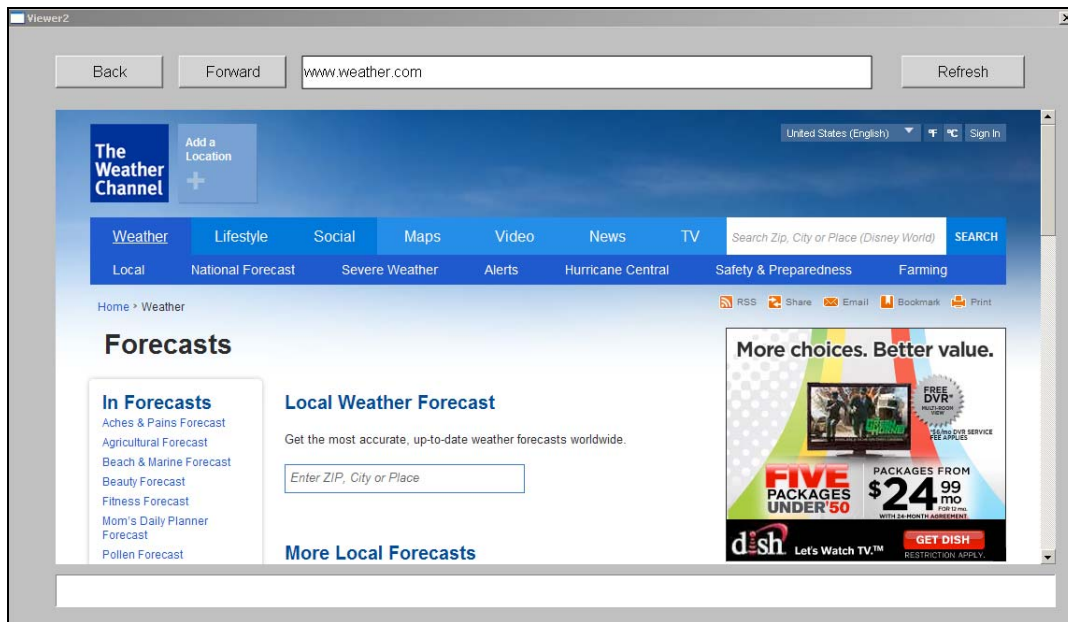
51. In **ClassSymbols**, double-click **WebBrowserMini**.



52. In the **Web** window, place the graphic and adjust the size as necessary.

## Test What You Have Built

53. Click **Runtime**.
54. Click each of the buttons to ensure they are functioning as expected.



55. Click **Development!**.
56. In Arcestra IDE, if necessary, click **Save & Close**, and then check in any open symbols.



## Module 5 – I/O Troubleshooting

---

|                                                   |      |
|---------------------------------------------------|------|
| Section 1 – Graphical Overrides                   | 5-3  |
| Lab 12 – Working with Graphical Overrides         | 5-5  |
| Section 2 – Visualization and Indirect Addressing | 5-9  |
| Lab 13 – Working with Failover                    | 5-11 |

**Module Objectives**

- Create a visual display to draw operator attention
- Configure I/O failover capabilities
- Implement graphical overrides

## Section 1 – Graphical Overrides

This section presents the methods for monitoring and graphically displaying the Quality and Status of data in an InTouch application.

### Status Visualization

Status feedback is configured in Archedra IDE. From the Galaxy menu, select **Configure | Quality and Status Display** to design the quality and status of data that is visualized. With flexible options, most aspects of the graphics can be overridden, and the indication of status is shown automatically in InTouch,

The image below displays the **Status Style Overrides** for Text, Fill, Line, Outline, and Status aspects of graphics.

| Status              | Ts | Fs | Ls | Ol | St |
|---------------------|----|----|----|----|----|
| Communication Error |    |    |    |    | X  |
| Configuration Error |    |    |    |    | X  |
| Pending             |    |    |    |    | X  |
| Operational Error   |    |    |    |    | X  |
| Software Error      |    |    |    |    | X  |
| Security Error      |    |    |    |    | X  |
| Warning             |    |    |    |    | X  |
| Bad                 |    |    |    | X  | X  |
| Uncertain           |    |    |    |    | X  |
| Initializing        |    |    |    |    | X  |

**Note:** In the example above, the Quality and Status display has been configured to place a blinking red dashed box around “Bad” data.



# Lab 12 – Working with Graphical Overrides

## Introduction

In this lab, you will add a graphical override to visually display a blinking dashed red line around all items displaying bad data. This helps draw attention to problem areas requiring the operator's attention.

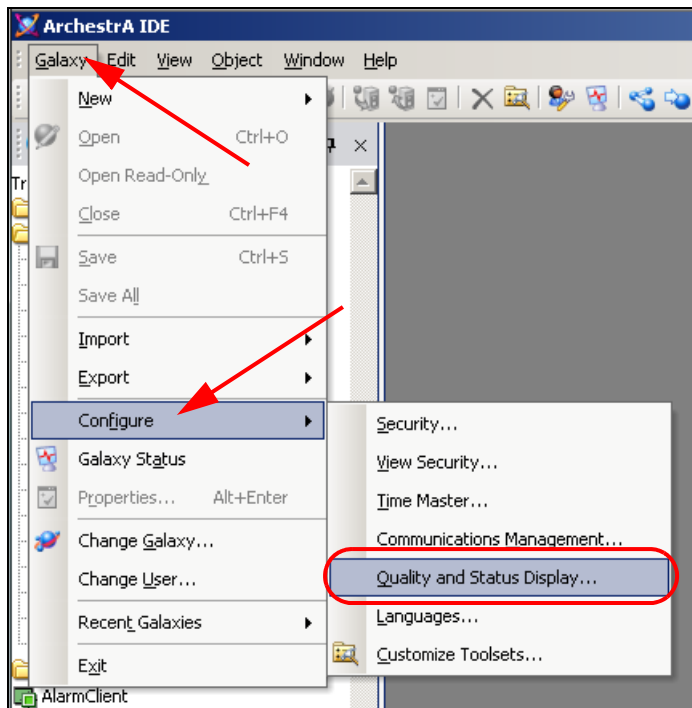
## Objectives

Upon completion of this lab, you will be able to:

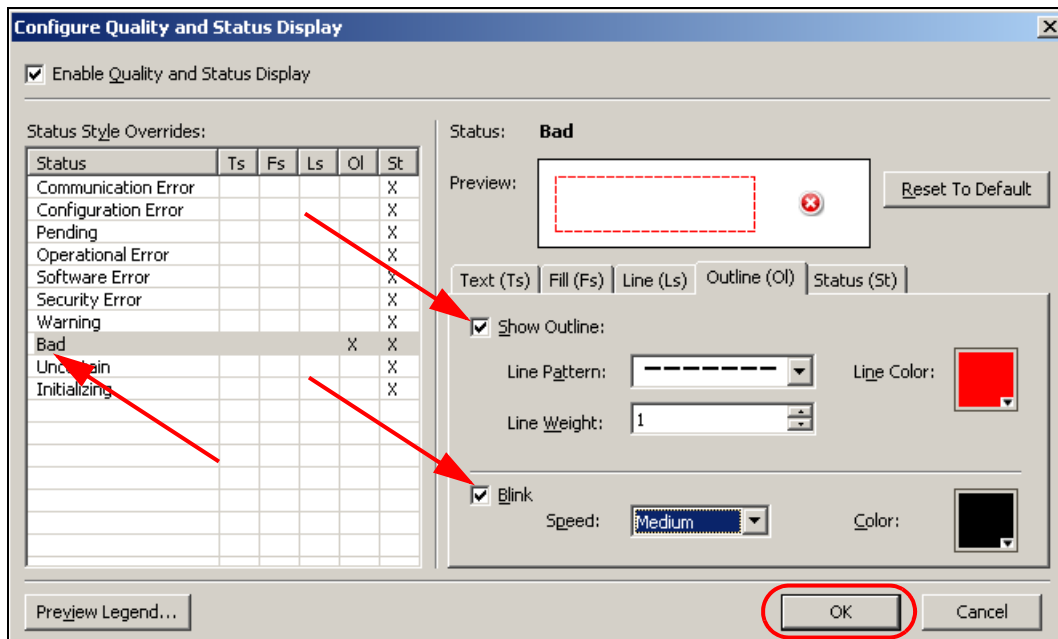
- Configure graphical overrides for your Galaxy

## Add a Graphical Override

1. In Archestra IDE, on the **Galaxy** menu, click **Configure | Quality and Status Display**.



2. Select **Bad**, click the **Outline (Ol)** tab, check **Show Outline**, and then select dashed line in bright red.
3. Check **Blink** and set speed to **Medium**, and then click **OK**.



4. Close WindowViewer to implement the changes.



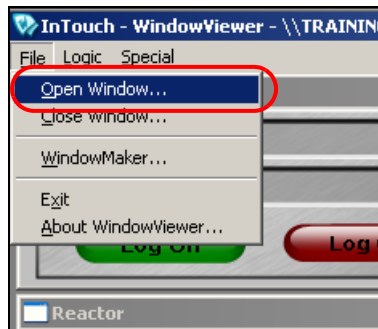
- In WindowMaker, reopen the **Reactor** and the **Security/Tag Viewer** windows, and then close all other windows.

## Test What You Have Built

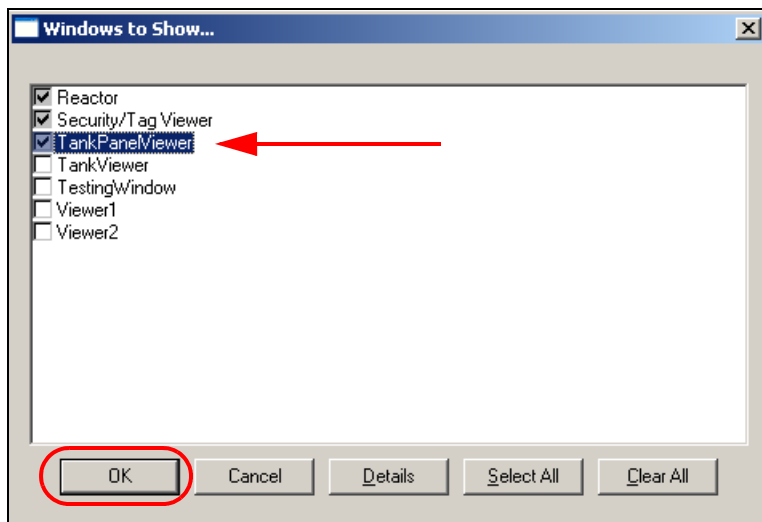
- Click **Runtime** and after your instructor terminates the I/O source you should see blinking red dashed lines around all objects displaying I/O.



- On the **File** menu, click **Open Window**.



8. Check **TankPanelViewer**, and then click **OK**.



9. Select **Tank 100**, and then click **One**.

You should see blinking red dashed lines around all symbols displaying I/O.

Once your instructor restarts the I/O source the blinking red dashed lines will go away.

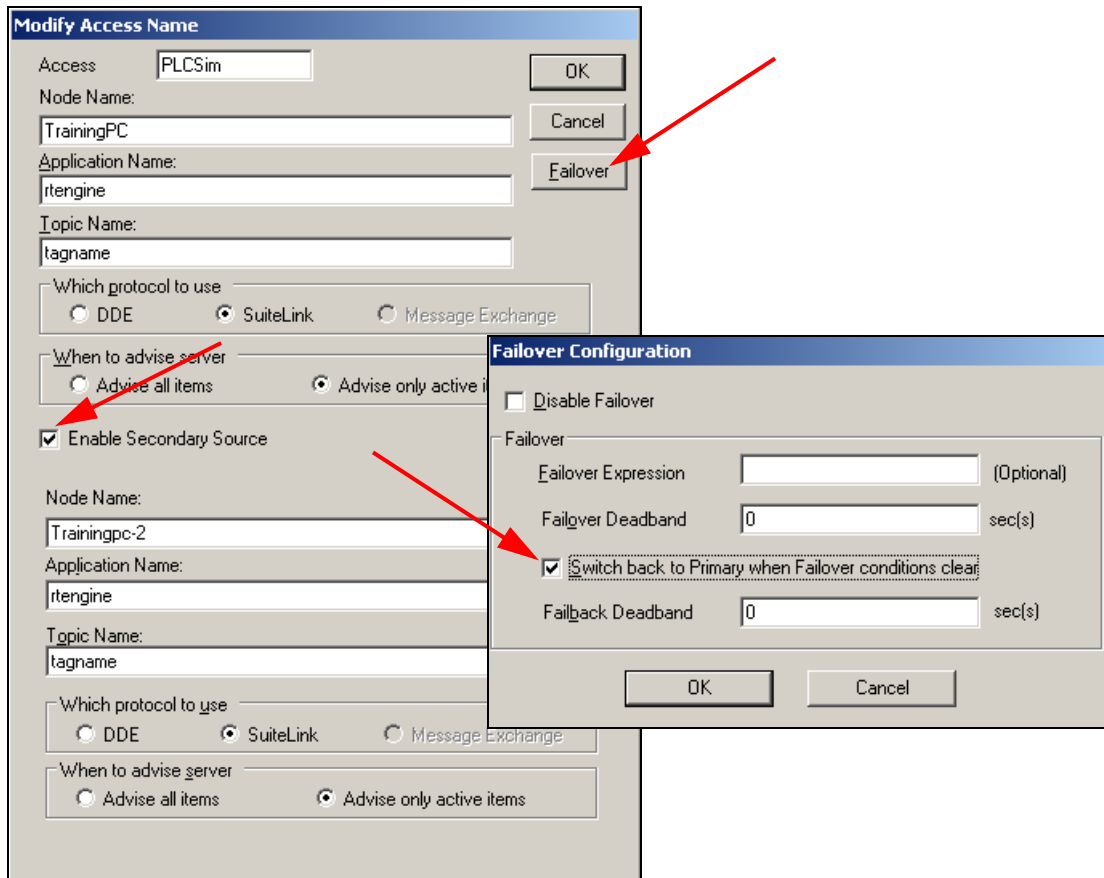
10. Click **Development!**.

## Section 2 – Visualization and Indirect Addressing

This section discusses the configuration and application of I/O Failover, which is a process that preserves the integrity of data even when there is a network communication error.

### Configuring and Monitoring I/O Failover

This feature provides the ability to easily configure I/O failover using dialog boxes and options in the user interface. I/O failover allows one to switch between access names when an I/O communication fails, and back again when the connection recovers.



The **Failover** button is available for accessing optional configuration settings:

- **Disable Failover** is for enabling and disabling failover for this access name
- **Failover Expression** is the discrete tag can be used to trigger the failover
- **Failover Deadband** is the delay before switching to the secondary source from the primary source, when I/O communication fails. The default value is 0 seconds
- **Switch back to Primary when Failover conditions clear** specifies whether InTouch should switch back to the primary source when I/O communication returns
- **Fail-back Deadband** is the delay before switching back to the primary source from the secondary source when I/O communication returns. The default value is 0 seconds

**Note:** With default value of 0 seconds, failover and fail-back are immediate.

Script functions for I/O failover are also provided. These can be used in any InTouch script and are based on triggers related to Access Names.

| Script Function                        | Description                                                                                           |
|----------------------------------------|-------------------------------------------------------------------------------------------------------|
| IOForceFailover(AccessName)            | Forces failover for an access name                                                                    |
| IODisableFailover(Accessname,Option)   | Enables and disables failover (specifying the "Option" as 0 enables failover and 1 disables failover) |
| IOGetActiveSourceName(AccessName)      | Returns the I/O Source of the currently active source of an Access Name                               |
| IOGetAccessNameStatus(AccessName,Mode) | Returns the status of the primary, secondary or active I/O Source of an Access Name.                  |

The modes defined below describe the IOGetAccessNameStatus Script Function above.

| Mode | Description                                  |
|------|----------------------------------------------|
| 0    | Status of the Active Access Name I/O Source  |
| 1    | Status of the Access Name Primary I/O Source |
| 2    | Status of the Access Name Secondary Source   |

The example below indicates the script syntax using the IOGetAccessNameStatus ScriptFunction:

Syntax: `Result = IOGetAccessNameStatus(AccessName, Mode);`

The Result tag returns the status of the Active, Primary, and Secondary nodes; however, this will only work if there is an active data point using the access name.

# Lab 13 – Working with Failover

## Introduction

In this lab, you will configure InTouch to have an alternate or secondary data source location for an access name, as well as monitor the status of the primary and secondary data sources.

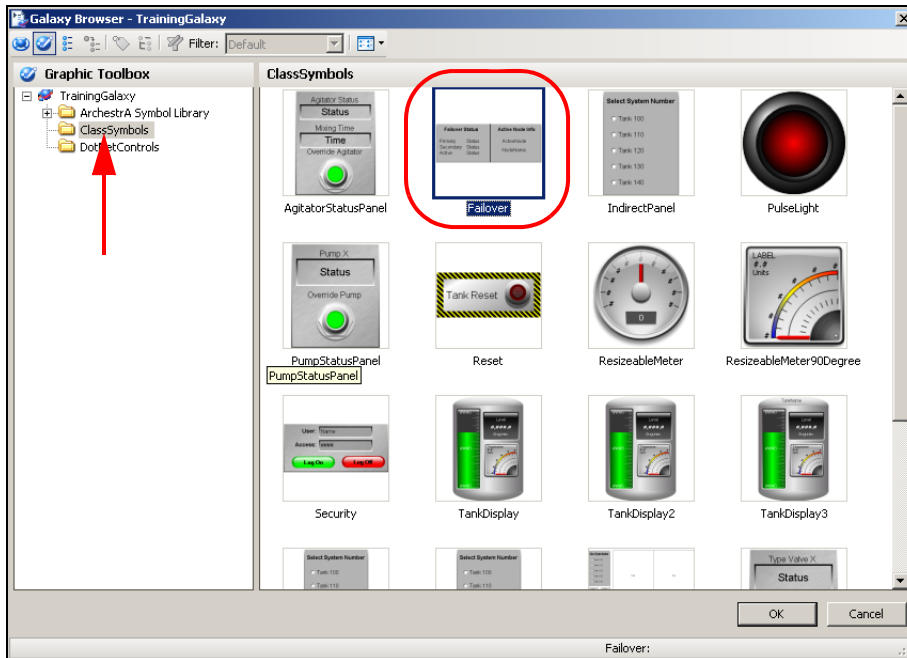
## Objectives

Upon completion of this lab, you will be able to:

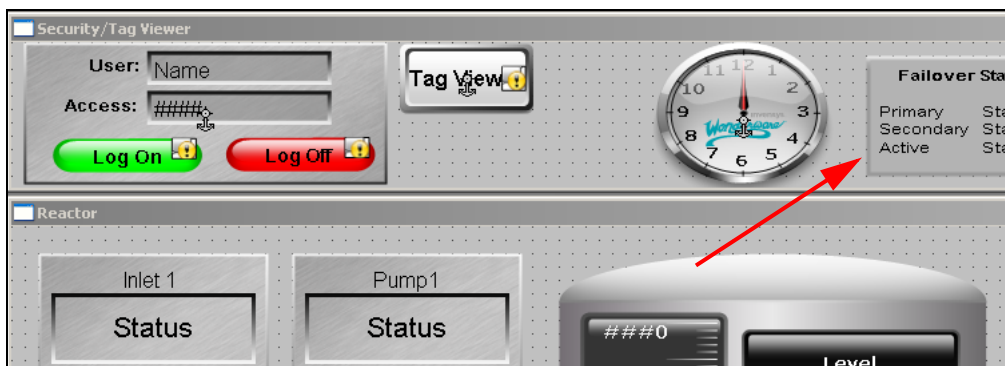
- Configure I/O failover
- Monitor I/O communications via scripting
- Monitor the failover status via scripting

## Enable Failover

1. In the ArchestrA IDE, from **C:\Training\Failover.aaPKG**, import the **Failover** symbol.
2. In WindowMaker, open **Security/Tag Viewer** and **Reactor** windows, and then close all others.
3. Click **Embed ArchestrA Graphic**.
4. Select **ClassSymbols**, and then double-click **Failover**.



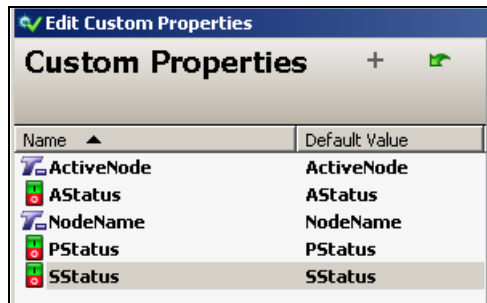
5. In the **Security/Tag Viewer** window, place it to the right of the clock, and then adjust its size as necessary.



6. Double-click **Failover** to open it for configuration.

Configures as follows:

| Name       | Default Value |
|------------|---------------|
| ActiveNode | ActiveNode    |
| AStatus    | AStatus       |
| NodeName   | NodeName      |
| PStatus    | PStatus       |
| SStatus    | SStatus       |



7. Click **OK**.

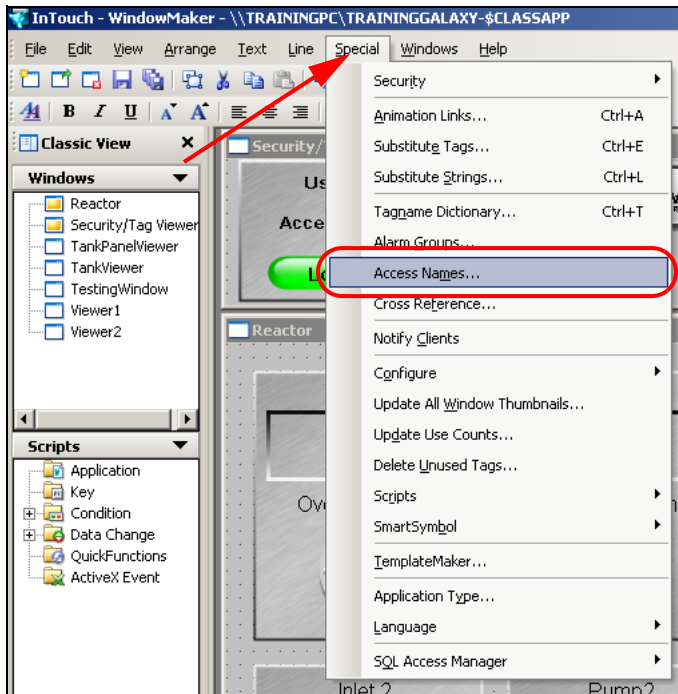
You will be promoted to define each **Tagname** since they have not been created yet.

8. Click **OK** for each, set the **Type** as shown in the table below, and then click **Close**.

| Name       | Type            |
|------------|-----------------|
| ActiveNode | Memory Message  |
| AStatus    | Memory Discrete |
| NodeName   | Memory Message  |
| PStatus    | Memory Discrete |
| SStatus    | Memory Discrete |

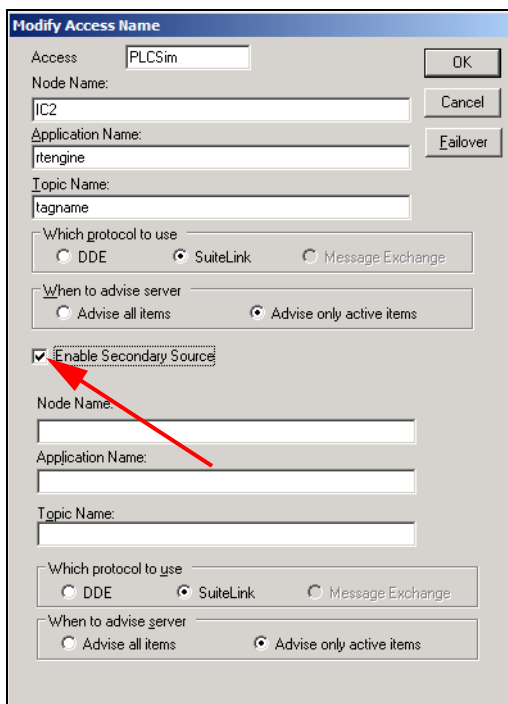
Next, you will configure the AccessName for Failover parameters.

9. On the **Special** menu, click **Access Names**.



10. Select **PLCSim**, and then click **Modify**.

11. Check **Enable Secondary Source**.



---

**Note:** Your instructor will provide the required information for the remaining fields.

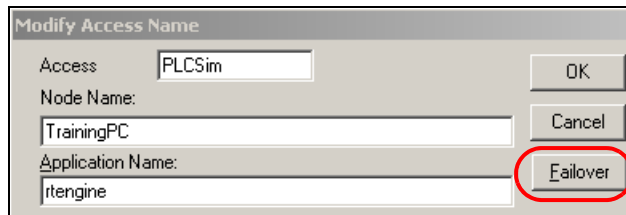
---



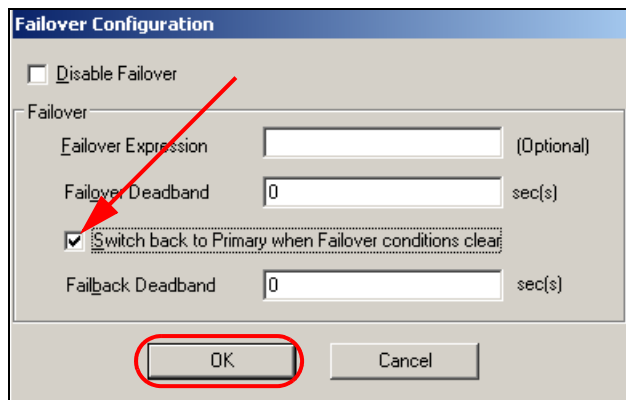
12. Click **Failover**.

If prompted to shut down the WindowViewer, do so, and again click **Failover**.

The **Modify Access Name** dialog box appears.



13. Check **Switch back to Primary when Failover conditions clear**, and then click **OK** to close the **Failover Configuration** dialog box.

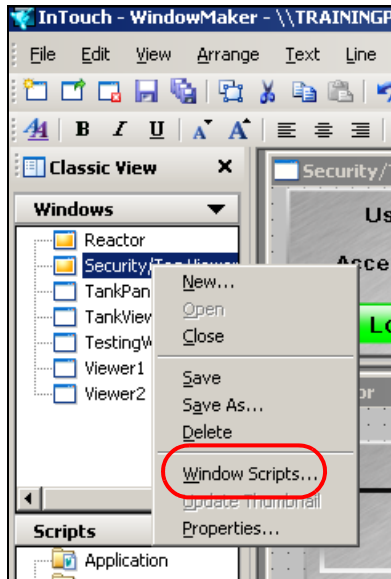


14. Click **OK** to close the **Modify Access Name** dialog box.
15. Click **Close** to close the **Access Names** dialog box.

Next, you will add the script.

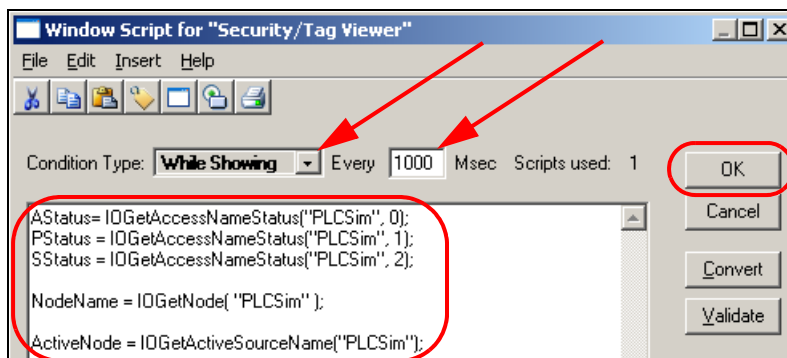
**Note:** All scripts for this lab are available to copy from the file **Lab 13 Script** in your **Training** directory.

16. On the **Security/Tag Viewer** window, right-click and select **Window Scripts**.



17. Select **While Showing every 1000 Msec**, and then enter the following script:

```
AStatus= IOGetAccessNameStatus("PLCSim", 0);
PStatus = IOGetAccessNameStatus("PLCSim", 1);
SStatus = IOGetAccessNameStatus("PLCSim", 2);
NodeName = IOGetNode("PLCSim");
ActiveNode = IOGetActiveSourceName("PLCSim");
```



18. Click **OK**.

## Test What You Have Built

19. Click **Runtime**, and then from the **Panel** on the right select one of the tanks.
20. Your instructor will stop the primary data source. Ensure the **Failover Status** for **Primary** says **Failed** in blinking red text, and the **Active Node** info lists the **Secondary** source.

| Failover Status |         | Active Node Info |  |
|-----------------|---------|------------------|--|
| Primary         | Failed  | Secondary        |  |
| Secondary       | Working | Trainingpc-2     |  |
| Active          | Working |                  |  |

21. Once your instructor restores the primary data source, ensure that the **Primary** is now listed as **Working**, and the **Active Node** info lists the **Primary** source.

| Failover Status |         | Active Node Info |  |
|-----------------|---------|------------------|--|
| Primary         | Working | Primary          |  |
| Secondary       | Working | TrainingPC       |  |
| Active          | Working |                  |  |

22. Click **Development!**.





## Module 6 – Alarm Tools

---

|                                                          |             |
|----------------------------------------------------------|-------------|
| <b>Section 1 – Basic Review of Alarming and Controls</b> | <b>6-3</b>  |
| <b>Lab 14 – Working with the Alarm Viewer</b>            | <b>6-7</b>  |
| <b>Lab 15 – Working with the Alarm Tree Viewer</b>       | <b>6-25</b> |
| <b>Section 2 – Alarm Databasing</b>                      | <b>6-31</b> |
| <b>Lab 16 – Working with the Alarm DB Manager</b>        | <b>6-35</b> |
| <b>Lab 17 – Working with the Alarm Pareto Control</b>    | <b>6-55</b> |
| <b>Section 3 – Alarm Client</b>                          | <b>6-63</b> |
| <b>Lab 18 – Working with the Alarm Client</b>            | <b>6-65</b> |
| <b>Lab 19 – Creating Pop-Up Alarm Displays</b>           | <b>6-75</b> |

**Module Objectives**

- Monitor and navigate alarms using the ActiveX alarm control and Alarm Tree Viewer
- Utilize alarm databasing options
- Configure the alarm Pareto
- Implement the Alarm Client

## Section 1 – Basic Review of Alarming and Controls

This section provides a basic summary of alarm concepts and discusses how to work with the ActiveX alarm control and alarm tree viewer to navigate alarms.

### Alarms

Alarms represent warnings of process conditions that could cause problems, and require an operator response.

A typical alarm is triggered when a process value exceeds a user-defined limit, such as an analog value exceeding a hi-limit threshold. This triggers an unacknowledged alarm state that can be used to notify the operator of a problem. Once the operator acknowledges the alarm, the system returns to an acknowledged state.

### Main Alarm States

| Alarm State | Condition                                           |
|-------------|-----------------------------------------------------|
| ACK         | Alarm was acknowledged.                             |
| ALM         | Alarm has occurred.                                 |
| RTN         | Tag returned from an alarm state to a normal state. |

### Alarm Concepts

There are a number of terms and concepts that generally apply to alarms, regardless of the particular system in which they are used, or how they are implemented.

### Alarms

In general, an alarm is condition-based; in particular, an alarm is an abnormal condition. The intention of an alarm is to signal that something has gone wrong, or that a particular stage of processing has been reached. For example, an alarm might indicate that a boiler has exceeded a safe temperature limit.

### Priority

A level of priority is associated with an alarm to indicate how severe the situation or condition is. In the case of the boiler temperature limit, it might be very severe, requiring immediate attention to save life and property. By contrast, in other (less dire) situations, the severity might be minimal. The severity of an alarm usually depends upon the circumstances – the factory application, the nature of the equipment, safety, availability of backup systems, or potential costs of damage or downtime.

### Sub-states

An alarm condition may also have sub-conditions, in which case, it is called a multi-state alarm. For example, an analog alarm typically has several limits, with High and Low bounding the normal operating range, and HiHi and LoLo marking the extreme departures from normal. The boiler temperature level mentioned above could be in the alarmed condition for any one of these sub-states. It could also transit between any two sub-states while continuing to remain in the overall alarmed condition.

## **Events**

An event is defined as a detectable occurrence, which may or may not be associated with an alarm. A transition into or out of an alarmed state constitutes one kind of event. An event might also mark an operator action, a change to the system configuration or some kind of system error. It is important to make a distinction between an alarm condition and an event. A condition may persist for minutes, hours, days or weeks. An event is momentary; it takes place and is immediately over. An alarm is a condition; an alarm notification is an event.

## **Acknowledgment**

One major purpose of an alarm is to alert someone about the condition. The person or system should then acknowledge the alarm, indicating that it has been seen. Acknowledgment is separate from the issue of taking corrective action, which might not happen right away. It is also separate from the issue of whether the alarm condition returns to normal, which it might do on its own even without any external intervention. Acknowledgment merely indicates that someone has noticed the occurrence of the alarm. A high or medium priority alarm usually requires acknowledgment, while a very low priority alarm might not. Although the condition that generated the alarm may go away (for example, a temperature rises too high and then becomes low again), the alarm itself is not considered handled until it is acknowledged.

## **Groups**

Alarms may be organized into groups to facilitate tracking and management. These groups might represent different areas of a factory, pieces of equipment, or operator responsibility of a specific functionality. Groups may also be organized into a hierarchy of parent groups and subgroups.

## **Alarm Viewer ActiveX Control**

InTouch provides an Alarm Viewer Control with built-in scroll bars, sizable display columns, multiple alarm selections, an update status bar, dynamic display types, and display colors based on alarm priority. The Alarm Viewer ActiveX Control is the preferred control for viewing alarms.

## **Guidelines**

Design the appearance of the Alarm Viewer Control and the data displayed by specifying the following attributes:

- Context sensitive menu features
- Display mode
- List control options
- Colors for different properties
- Font type, style and size
- Alarm selection (filters)
- Query filters
- Column management
- Sorting



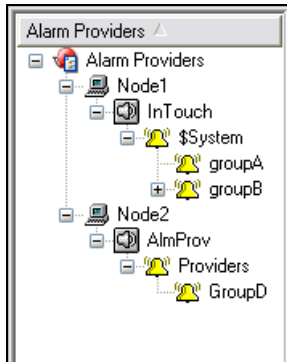
Once the control format has been designed, a user may have the ability to make the following adjustments to manipulate the data they are viewing:

- Sort the information by column
- Update the display
- Perform a query
- Resize the width of a column

The Alarm Viewer Control can be dropped into WindowMaker, resized, and positioned. Configuration of the Alarm Viewer Control is done using the Property sheets. The alarm data can then be viewed in the Alarm Viewer Control view window.

## Alarm Tree View Control

The Alarm Tree View Control provides an expandable/collapsible tree structure for displaying and interacting with the hierarchy of Alarm Providers and Alarm Groups, as shown below:



The Alarm Tree View can show multiple nodes and Alarm Providers. In the tree structure:

- Nodes are represented by a computer icon.
- Alarm Providers are represented by a speaker icon.
- Alarm Groups are represented by a bell icon.

With one or more Alarm Groups selected, generate queries for alarms that can be used in the Alarm Viewer and the Alarm DB View Controls. To select multiple Alarm Groups, hold down the Shift key while clicking on a Group. To cancel the selection of Groups, click on an empty area.

Configure the Alarm Tree View Control to display any of the following commands on the Context menu when a user right-clicks on the control in Runtime:

- **Refresh:** Forces a manual update of the alarms
- **Freeze:** Stops the alarms from updating
- **Query Favorites:** Opens the Alarm Query dialog box to select an alarm query from a list of previously defined alarm queries
- **Create Query Favorites:** Opens the Add Query dialog box with a query string entered based on the selected Groups (if any)
- **Sort:** Opens the Sort dialog box to sort Alarm Groups in ascending or descending order
- **Statistics:** Opens the Alarm Statistics dialog box for viewing the percent of Alarm Providers retrieved

# Lab 14 – Working with the Alarm Viewer

## Introduction

In this lab, you will add an Alarm Viewer display and then enable and disable its functionality based on the user's security access.

## Objectives

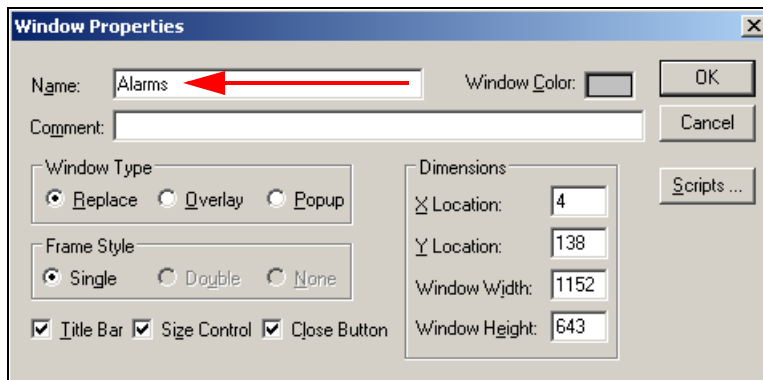
Upon completion of this lab, you will be able to:

- Create a display using the Alarm Viewer ActiveX control
- Control Alarm Viewer functionality based on security access and scripting

## Work with Alarm Viewer

First create a new InTouch window to house the Alarm Viewer control.

1. In WindowMaker, create a new window called **Alarms**.



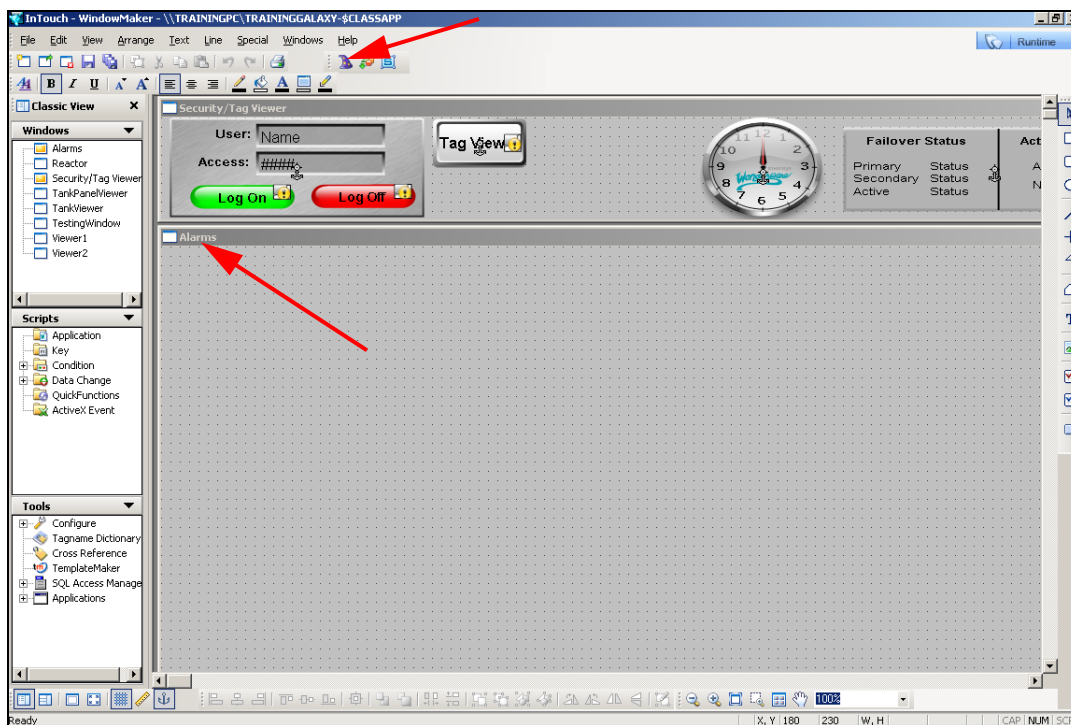
---

**Note:** Your window dimensions may vary depending on your resolution.

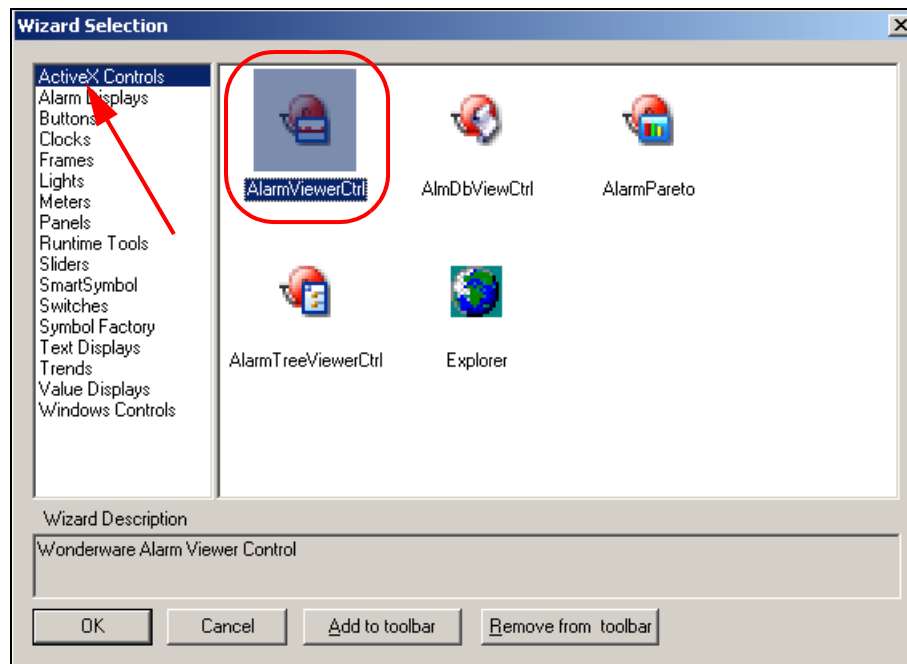
---

**Alarms** window should fit just below the **Security/Tag Viewer** window and take up the remainder of the visible area.

2. Close all other windows except **Security/Tag Viewer** and **Alarms**.
3. Click **Wizards**.

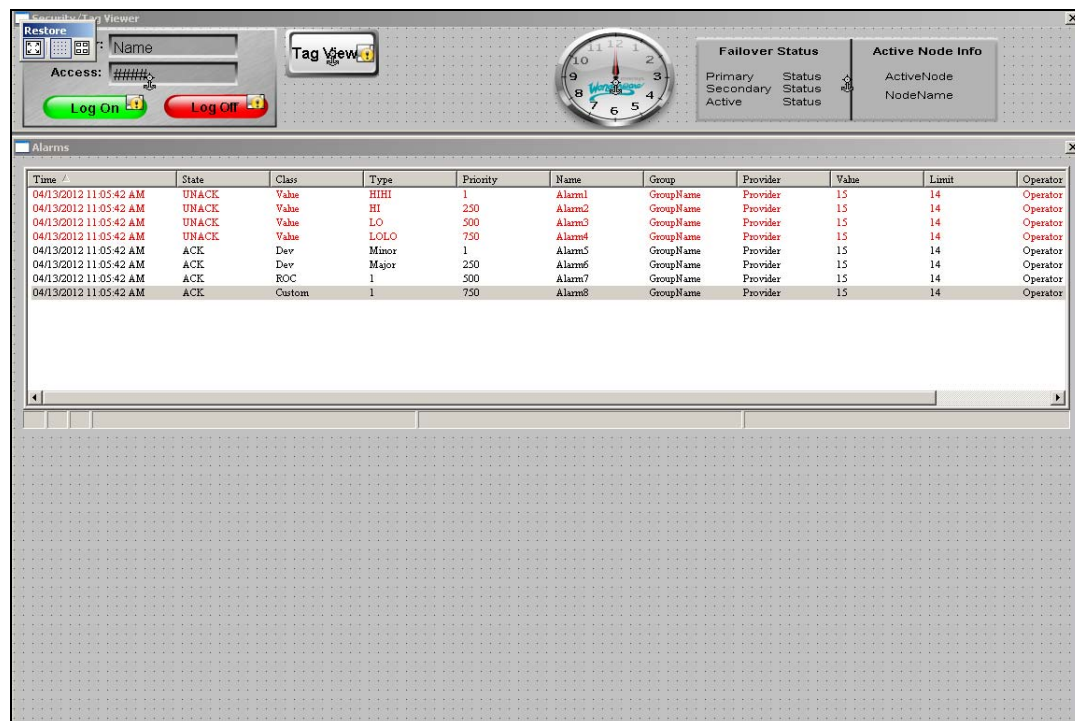


4. Select **ActiveX Controls**, and then double-click **AlarmViewerCtrl**.

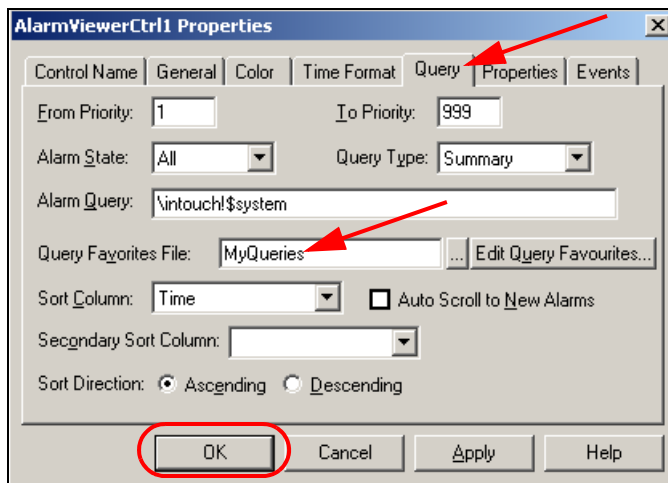


5. In the **Alarms** window, place **AlarmViewerCtrl**, and then size it so that it fills the width and half the height of the window.

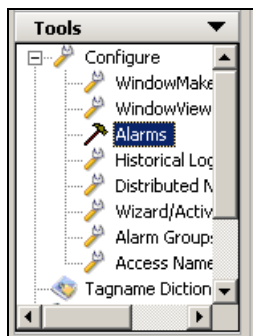
You may need to enter full screen mode to accomplish this.



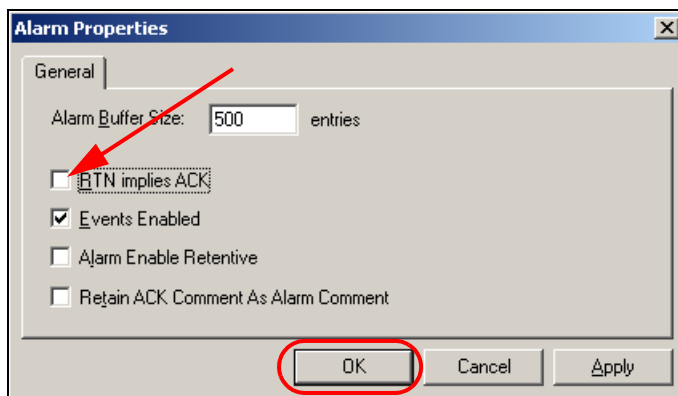
6. Double-click **AlarmViewerCtrl** to open it for editing.
7. Click the **Query** tab, in the **Query Favorites File** field enter **MyQueries**, and then click **OK**.



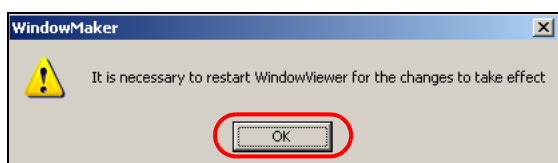
8. In **Tools**, expand **Configure**, and then double-click **Alarms**.



9. Uncheck **RTN implies ACK**, and then click **OK**.



10. Click **OK**, and then shut down WindowViewer.

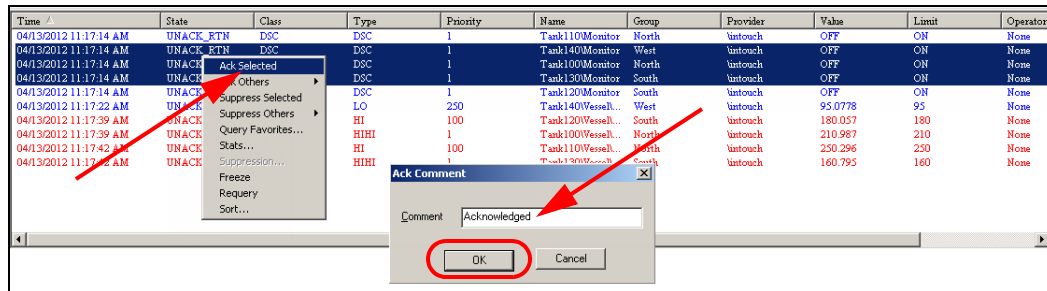


## Test What You Have Built

### 11. Click **Runtime**.

The tags in your application have already been configured for alarming, so you should see various alarms on your display.

### 12. Select one or more alarms, right-click them and select **Ack Selected**, enter a **Comment**, and then click **OK**.

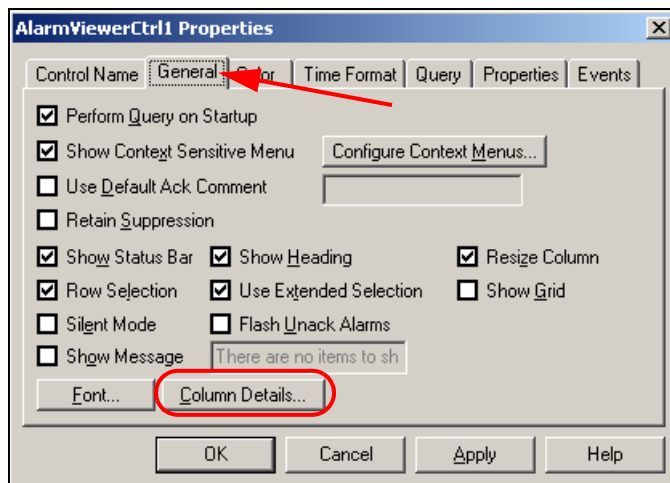


### 13. Click **Development!**.

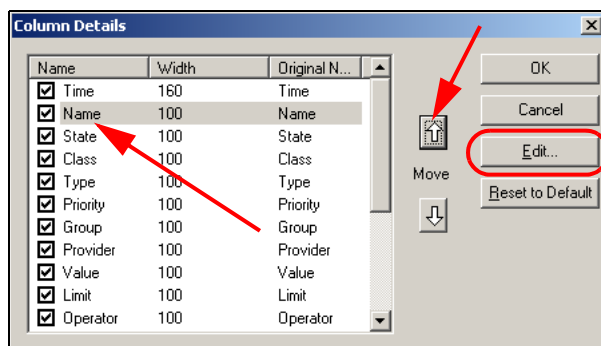
Next, you will customize the display.

### 14. Double-click **AlarmViewerCtrl**.

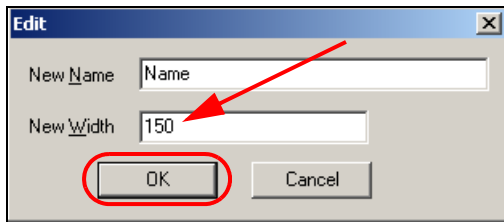
### 15. Select the **General** tab, and then click **Column Details**.



### 16. Select **Name**, using the arrow keys move it up to just below **Time**, and then click **Edit**.



17. In the **New Width** field, enter **150**, and then click **OK**.



18. Click **OK** to exit configuration.
19. Click **OK** to exit the **Properties** dialog box.
20. Click **Runtime**.

The **Name** column displays next to **Time**, providing the full tagname.

The image shows a window titled 'Alarms'. It contains a table with five columns: 'Time', 'Name', 'State', 'Class', and 'Type'. The 'Name' column is circled in red. The table lists 10 alarms with their respective timestamps, names, states, classes, and types.

| Time                   | Name              | State     | Class | Type |
|------------------------|-------------------|-----------|-------|------|
| 04/13/2012 11:29:44 AM | Tank130VesselTemp | UNACK     | VALUE | HIHI |
| 04/13/2012 11:29:45 AM | Tank110VesselTemp | UNACK_RTN | VALUE | LO   |
| 04/13/2012 11:29:56 AM | Tank130Monitor    | UNACK_RTN | DSC   | DSC  |
| 04/13/2012 11:29:56 AM | Tank120Monitor    | UNACK_RTN | DSC   | DSC  |
| 04/13/2012 11:29:56 AM | Tank140Monitor    | UNACK_RTN | DSC   | DSC  |
| 04/13/2012 11:29:56 AM | Tank100Monitor    | UNACK_RTN | DSC   | DSC  |
| 04/13/2012 11:29:56 AM | Tank110Monitor    | UNACK_RTN | DSC   | DSC  |
| 04/13/2012 11:30:07 AM | Tank120VesselTemp | UNACK_RTN | VALUE | LO   |
| 04/13/2012 11:30:09 AM | Tank140VesselTemp | UNACK     | VALUE | HI   |
| 04/13/2012 11:30:10 AM | Tank100VesselTemp | UNACK_RTN | VALUE | HI   |

At the bottom of the window, there is a status bar that says 'Displaying 1 to 10 of 10 alarms.' and a button labeled 'Default Query'.

21. Click **Development!**.

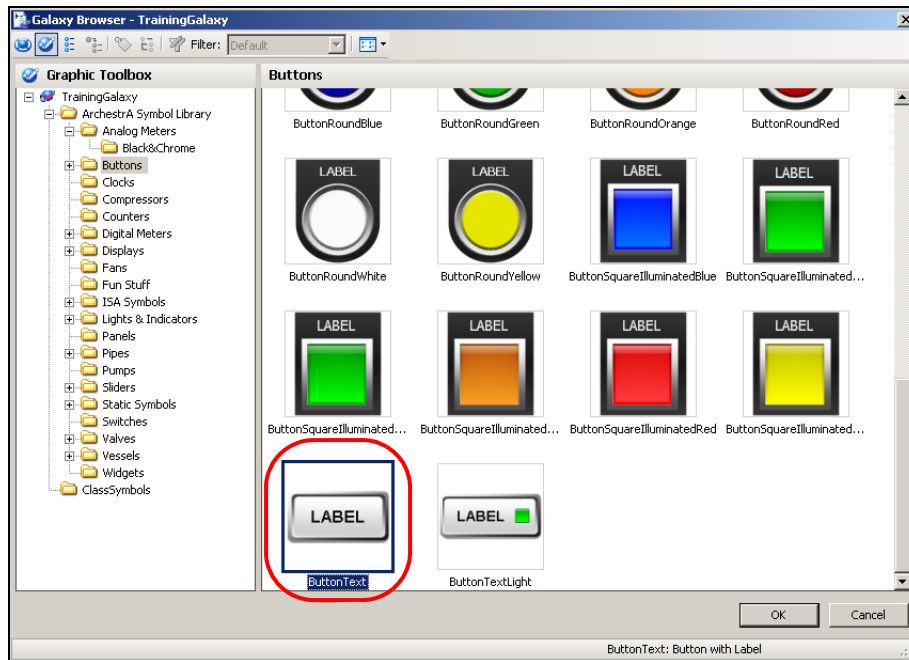


## Add a Runtime Interface

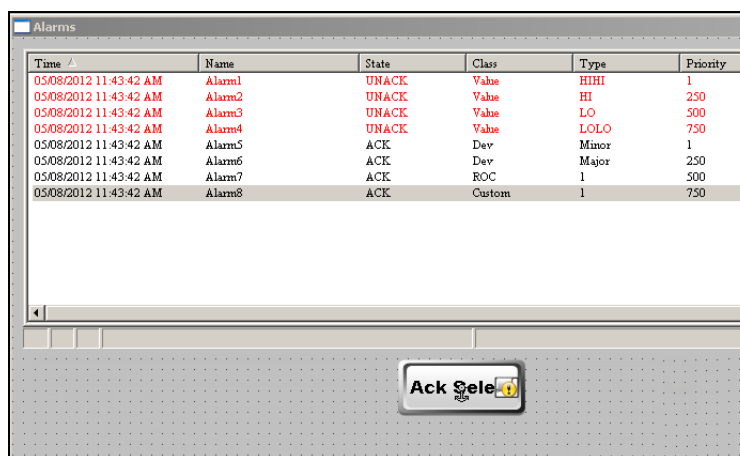
Add a runtime interface to acknowledge alarms and change the display from summary to historical mode.

22. Click **Embed ArchestrA Graphic**.

23. From **ArchestrA Symbol Library**, select **Buttons**, and then double-click **ButtonText**.



24. In the **Alarms** window, embed the symbol directly below the **Alarm Display** and beneath the **State** and **Class** columns.

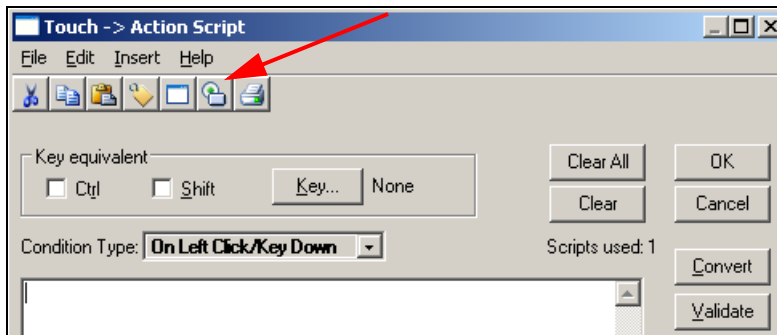


25. Press **Ctrl+L** and change the label to **Ack Select**.

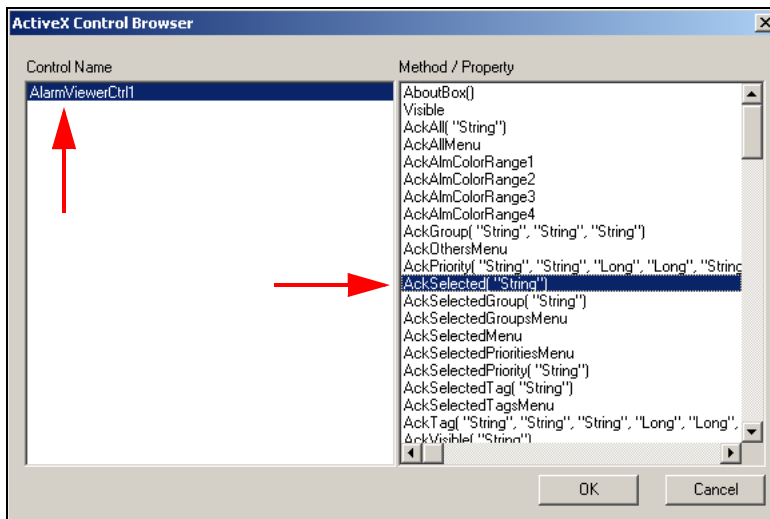
26. Right-click the **Ack Select** button and select **Animation Links**.

27. Click **Action**.

28. Clear any existing script, and then click **Insert ActiveX Control**.



29. Click **AlarmViewerCtrl1**, and then double-click **AckSelected("String")**.




---

**Note:** Your list of **Control Name** may vary slightly from the example, depending on what you have built in previous labs. For example, your Control Name list may also contain Explorer1.

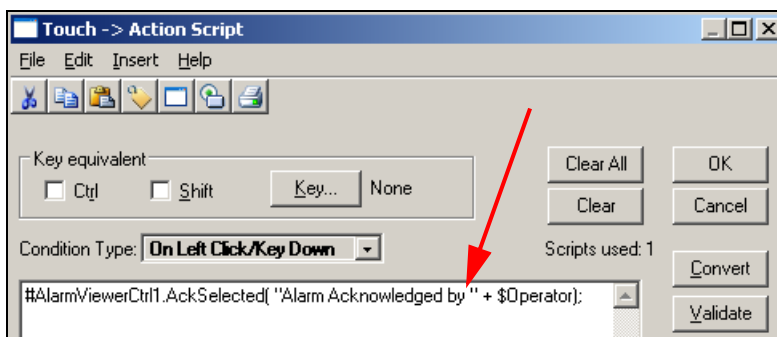
---

30. Alter the script to read as follows:  
**#AlarmViewerCtrl1.AckSelected("Alarm Acknowledged by " + \$Operator);**

---

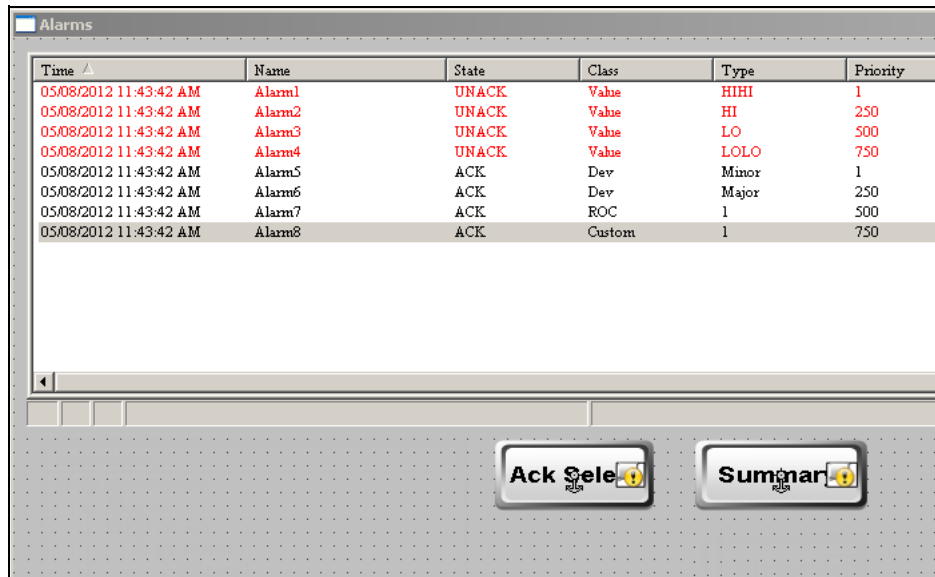
**Note:** Make sure to add a space after the word **by**. This is to improve readability of the alarm comment.

---

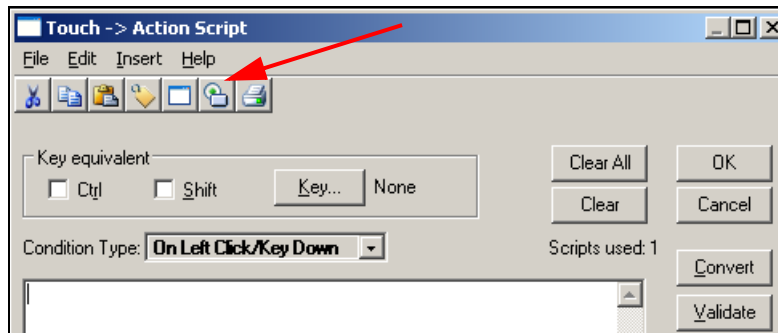


31. Click **OK** to exit the **Action Script** dialog box.

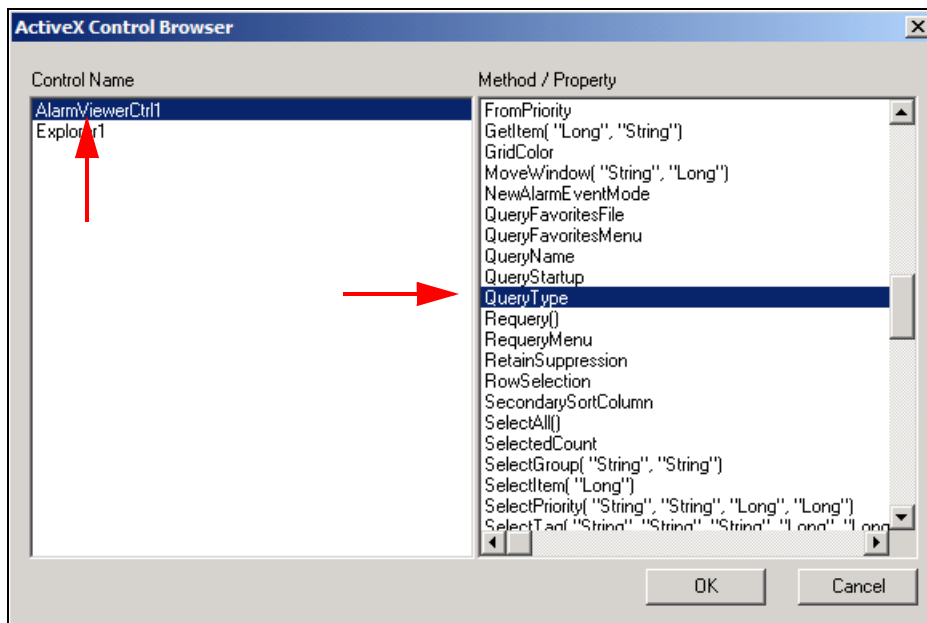
32. Click **OK** to exit the **Animation** dialog box.
33. Duplicate the **Ack Select** button, and then press **Ctrl+L** and change the label to **Summary**.
34. Place the **Summary** button to the right of the **Ack Select** button.



35. Right-click **Summary** and select **Animation Links**.
36. Click **Action**.
37. Clear any existing script, and then click **Insert ActiveX Control**.



38. Click **AlarmViewerCtrl1**, and then double-click **QueryType**.



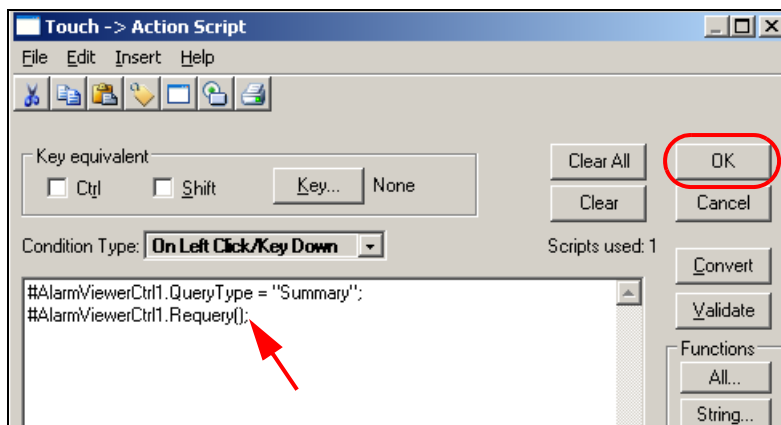
39. Alter the script so it looks as follows:

```
#AlarmViewerCtrl1.QueryType = "Summary";
```

40. Press **Enter** to add a new line to the script.

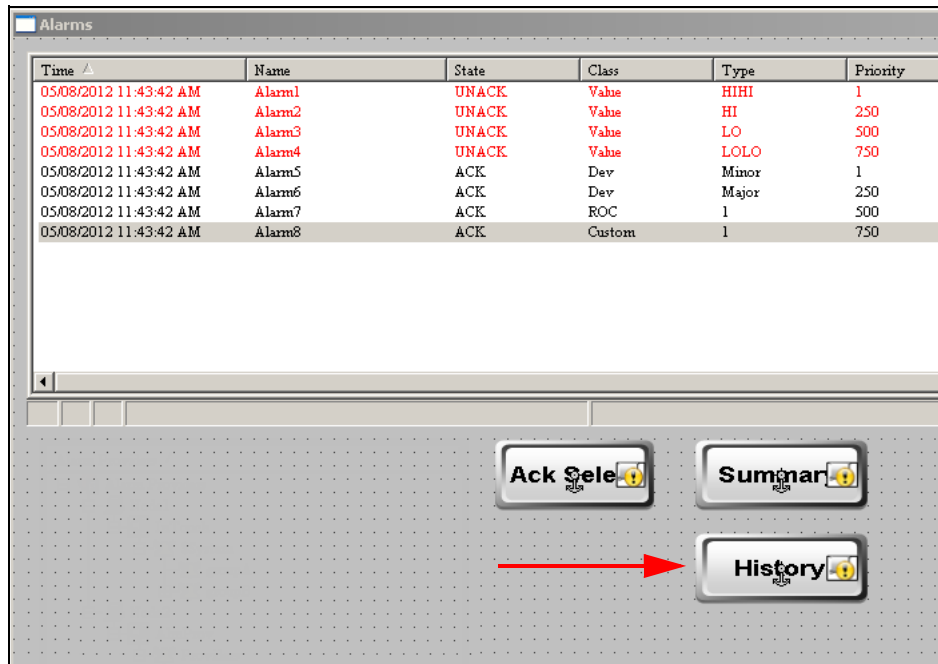
41. Click **Insert ActiveX Control**, select **AlarmViewerCtrl1**, and then double-click **Requery()**.

42. Add a semicolon to the end of the placed script, and then click **OK**.

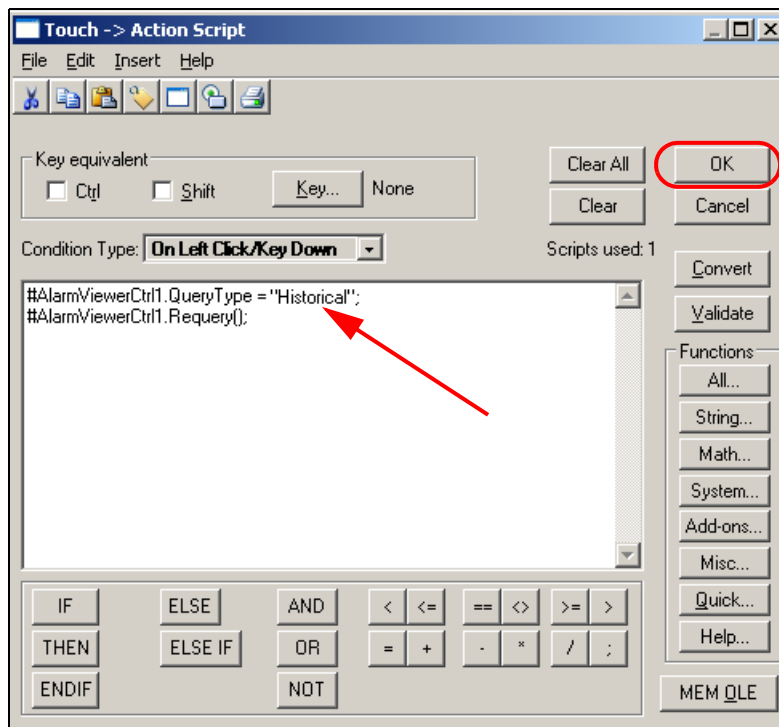


43. Click **OK**.

44. Duplicate the **Summary** button, and then change the label to **History**.
45. Place the **History** button directly below the **Summary** button.



46. Right-click **History** and select **Animation Links**.
47. Click **Action**.
48. In the first line of the script, change **Summary** to **Historical**, and then click **OK**.



49. Click **OK**.

## Test What You Have Built

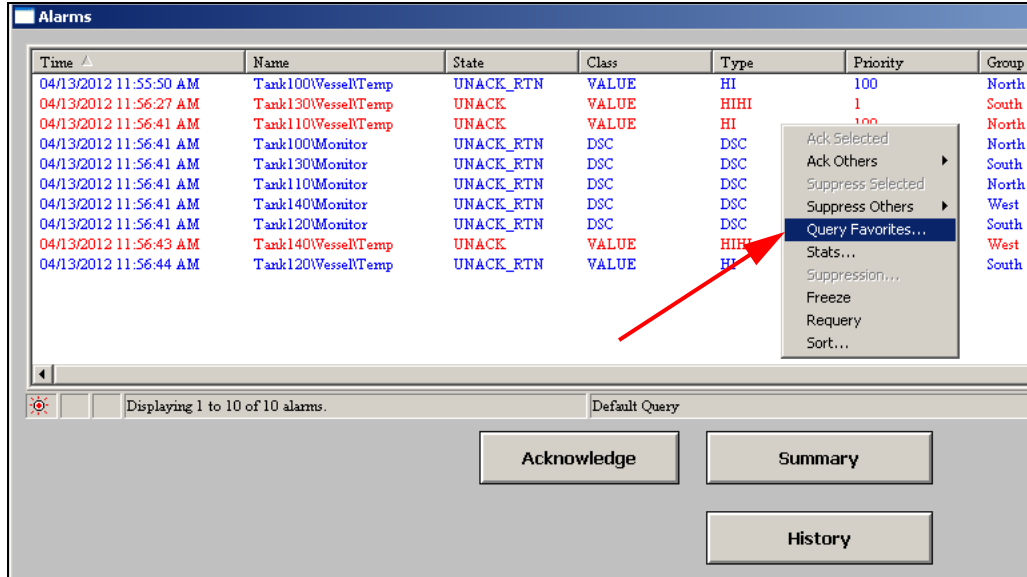
50. Click **Runtime**.

51. Click **History** to see all alarms that have occurred since the most recent restart of InTouch.

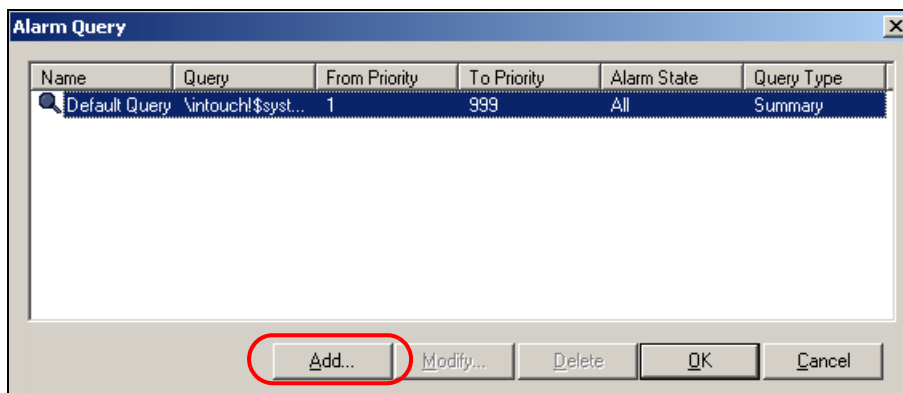
52. Click **Summary** to switch back to viewing only the active alarms.

You can also adjust the query type of the display by right-clicking and selecting **Query Favorites**. This will allow you to add additional queries.

53. On the display, right-click and select **Query Favorites**.



54. Select **Default Query**, and then click **Add**.



55. Enter the following parameters:

**Name:** North

**Query:** \InTouch!North

The 'Add Query' dialog box is shown with the following fields and values:

- Name:** North
- Query:** \InTouch!North
- From Priority:** 1
- To Priority:** 999
- Display Type:** Summary (selected), Historical
- Alarm State:** All
- Buttons:** OK (circled in red), Cancel

56. Click **OK**.

57. Click **Add**.

58. Enter the following parameters:

**Name:** South

**Query:** \InTouch!South

The 'Add Query' dialog box is shown with the following fields and values:

- Name:** South
- Query:** \InTouch!South
- From Priority:** 1
- To Priority:** 999
- Display Type:** Summary (selected), Historical
- Alarm State:** All
- Buttons:** OK (circled in red), Cancel

59. Click **OK**.

60. Double-click **North** to view it.

You will only see the alarm group **North**.

| Time                   | Name              | State     | Class | Type | Priority | Group | Provider  |
|------------------------|-------------------|-----------|-------|------|----------|-------|-----------|
| 04/13/2012 12:03:23 PM | Tank100Monitor    | UNACK_RTN | DSC   | DSC  | 1        | North | \InTouch! |
| 04/13/2012 12:03:23 PM | Tank110Monitor    | UNACK_RTN | DSC   | DSC  | 1        | North | \InTouch! |
| 04/13/2012 12:03:38 PM | Tank100VesselTemp | UNACK     | VALUE | HIHI | 1        | North | \InTouch! |
| 04/13/2012 12:03:42 PM | Tank110VesselTemp | UNACK     | VALUE | HI   | 100      | North | \InTouch! |

61. On the display, right-click and select **Query Favorites**.

62. Double-click **South** to view it.

You will only see the alarm group **South**.

| Time                   | Name              | State     | Class | Type | Priority | Group |
|------------------------|-------------------|-----------|-------|------|----------|-------|
| 04/13/2012 12:04:26 PM | Tank120VesselTemp | UNACK_RTN | VALUE | LO   | 250      | South |
| 04/13/2012 12:04:27 PM | Tank130VesselTemp | UNACK_RTN | VALUE | HI   | 100      | South |
| 04/13/2012 12:04:33 PM | Tank130Monitor    | UNACK     | DSC   | DSC  | 1        | South |
| 04/13/2012 12:04:33 PM | Tank120Monitor    | UNACK     | DSC   | DSC  | 1        | South |

63. On the display, right-click and select **Query Favorites**.

64. Double-click **Default Query** to return to seeing all groups.

| Time                   | Name              | State     | Class | Type | Priority | Group |
|------------------------|-------------------|-----------|-------|------|----------|-------|
| 04/13/2012 12:05:00 PM | Tank140VesselTemp | UNACK_RTN | VALUE | HI   | 100      | West  |
| 04/13/2012 12:05:19 PM | Tank100VesselTemp | UNACK_RTN | VALUE | HI   | 100      | North |
| 04/13/2012 12:05:22 PM | Tank120VesselTemp | UNACK_RTN | VALUE | HI   | 100      | South |
| 04/13/2012 12:05:23 PM | Tank130VesselTemp | UNACK     | VALUE | HIHI | 1        | South |
| 04/13/2012 12:05:23 PM | Tank120Monitor    | UNACK_RTN | DSC   | DSC  | 1        | South |
| 04/13/2012 12:05:23 PM | Tank130Monitor    | UNACK_RTN | DSC   | DSC  | 1        | South |
| 04/13/2012 12:05:23 PM | Tank140Monitor    | UNACK_RTN | DSC   | DSC  | 1        | West  |
| 04/13/2012 12:05:23 PM | Tank110Monitor    | UNACK_RTN | DSC   | DSC  | 1        | North |
| 04/13/2012 12:05:23 PM | Tank100Monitor    | UNACK_RTN | DSC   | DSC  | 1        | North |

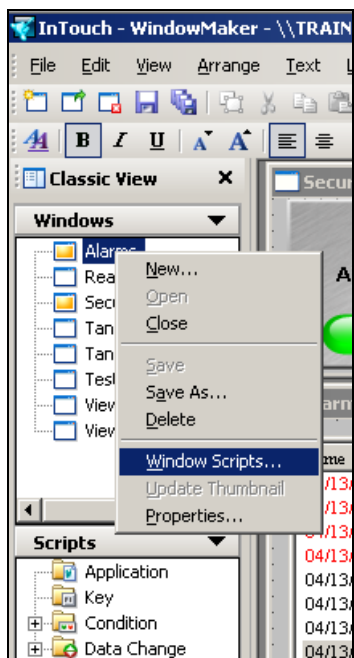
65. Click **Development!**.

## Add Security

For this portion of the lab, you will enable and disable functionality based on the user's security access.

**Note:** All scripts for this lab are available to copy from the file **Lab 14 Script** in your **Training** directory.

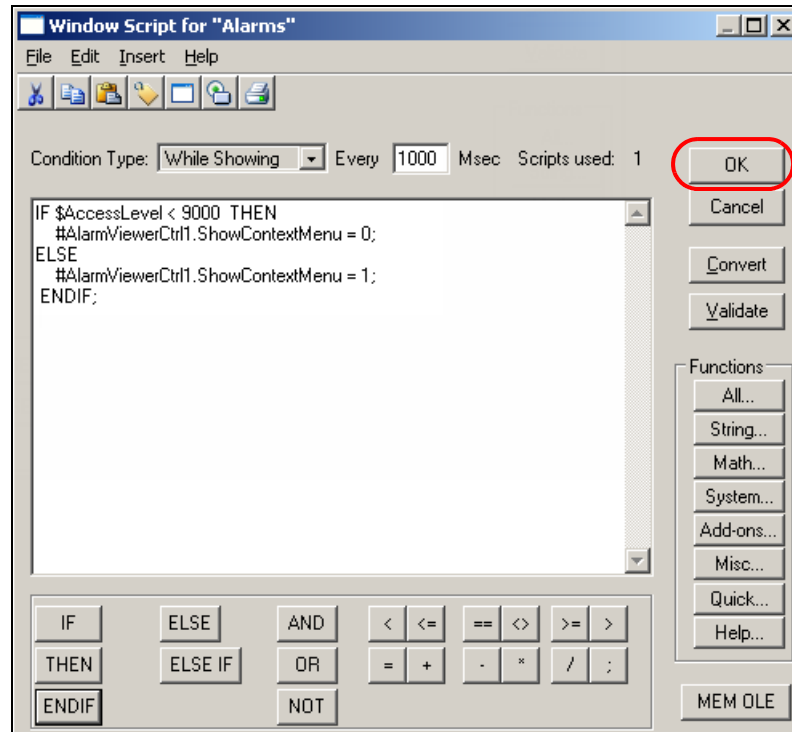
66. Right-click the **Alarms** window and select **Window Scripts**.





67. Add the following script to disable right-click functionality for access levels below 9000:

| Condition Type | How Often (Every) | Scripting                                                                                                                                    |
|----------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| While Showing  | 1000 Msec         | <pre>IF \$AccessLevel &lt; 9000 THEN     #AlarmViewerCtrl1.ShowContextMenu = 0; ELSE     #AlarmViewerCtrl1.ShowContextMenu = 1; ENDIF;</pre> |



68. Click **OK**.

## Test What You Have Built

69. Click **Runtime**.

70. In the **Alarms** window, right-click the **Alarm Viewer** and nothing happens.

71. Click **Log On** as an **Administrator** (password is **Wonderware**) or another user with sufficient access level.

72. Right-click and you have access to the context menu.

73. Click **Development!**.

## Set up Security

To disable access to the **Ack Select** based on security level you will add a disable animation.

74. Right-click the **Ack Select** button and select **Animation Links**.

75. Click **Disable**.

Object type: Button

Prev Link Next Link

OK Cancel

Touch Links

User Inputs

☐ Discrete

☐ Analog

☐ String

Sliders

☐ Vertical

☐ Horizontal

Touch Pushbuttons

☐ Discrete Value

☒ Action

☐ Show Window

☐ Hide Window

Line Color

☐ Discrete

☐ Analog

☐ Discrete Alarm

☐ Analog Alarm

Fill Color

☐ Discrete

☐ Analog

☐ Discrete Alarm

☐ Analog Alarm

Text Color

☐ Discrete

☐ Analog

☐ Discrete Alarm

☐ Analog Alarm

Object Size

☐ Height

☐ Width

Location

☐ Vertical

☐ Horizontal

Percent Fill

☐ Vertical

☐ Horizontal

Miscellaneous

☐ Visibility

☐ Blink

☐ Orientation

☒ Disable

☐ Tooltip

Value Display

☐ Discrete

☐ Analog

☐ String

76. Enter the following expression:

**\$AccessLevel < 9000**

77. Set the **Disabled State** to **On**, and then click the top **OK**.

Object type: Button

Prev Link Next Link

OK Cancel

Object Disabled -> Discrete Value

Expression:

OK Cancel

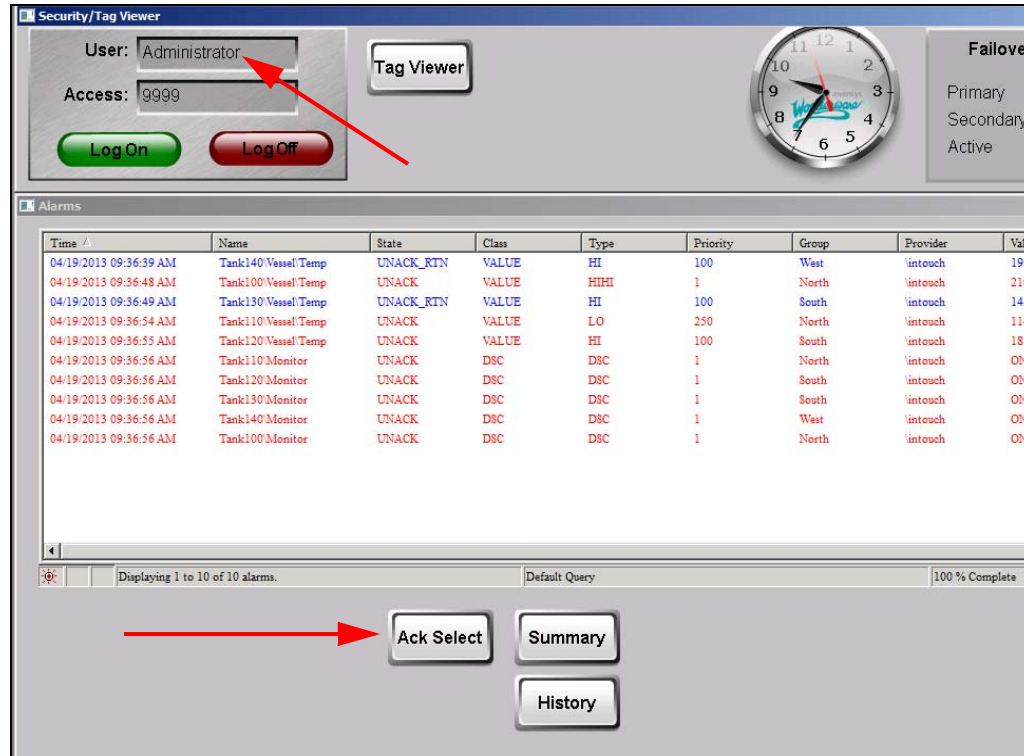
Disabled State

☒ On ☐ Off

Clear

## Test What You Have Built

78. Click **Runtime**.
79. While logged on as a user with the proper access level, the **Ack Select** button functions normally.



80. Log off and the **Ack Select** button will no longer function.
81. Click **Development!**.



# Lab 15 – Working with the Alarm Tree Viewer

## Introduction

In this lab, you will add the Alarm Tree Viewer ActiveX control to your alarm display to aid in the navigation of the Alarm Viewer ActiveX control.

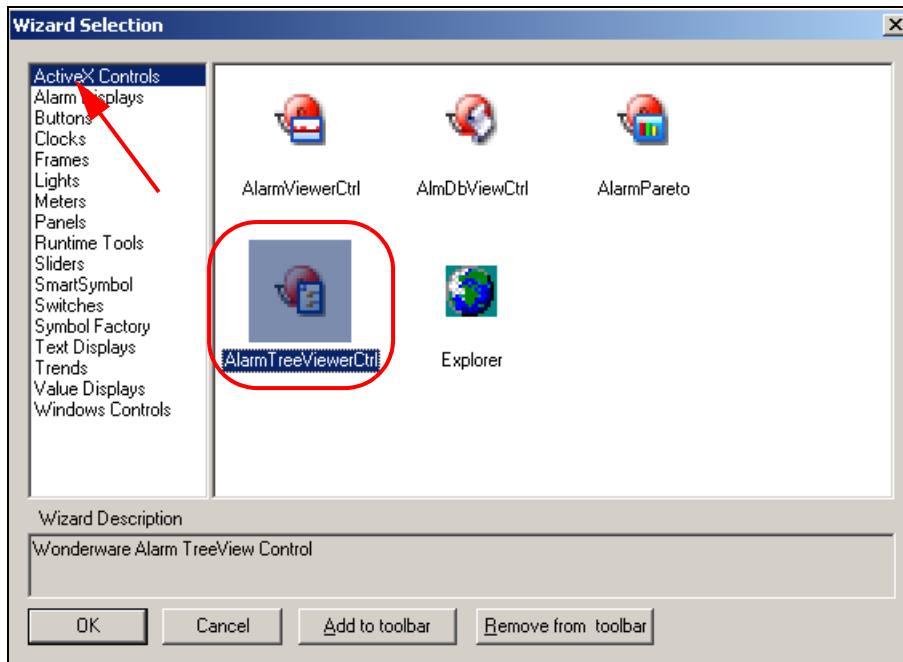
## Objectives

Upon completion of this lab, you will be able to:

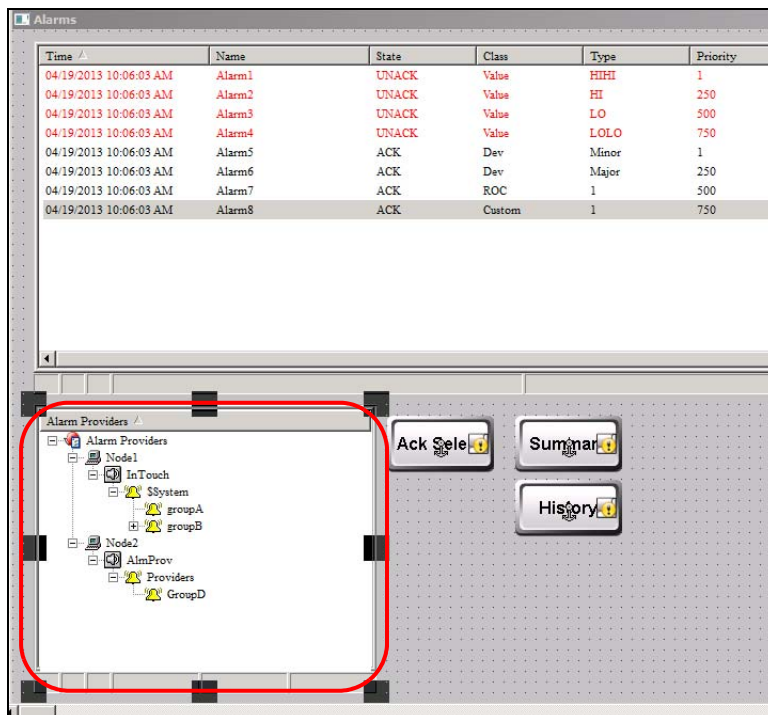
- Change the Query of the Alarm Viewer using the Alarm Tree Viewer

## Work with the Alarm Tree Viewer

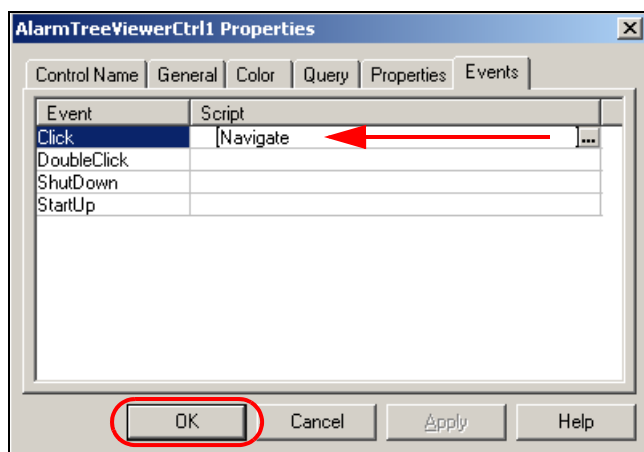
1. In WindowMaker, open the **Alarm** and **Security/Tag Viewer** windows; close all other windows.
2. Click **Wizards**.
3. Select **ActiveX Controls**, and then double-click **AlarmTreeViewCtrl**.



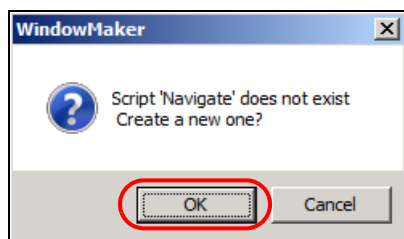
4. Place it below the **Alarm Viewer** to the left of the **Ack Select** button and size as necessary.



5. Double-click **Alarm Tree Viewer** and select the **Events** tab.
6. In the **Event** column select **Click**, in the **Script** column enter **Navigate** as the script name, and then click **OK**.



7. When prompted to create a new script, click **OK**.



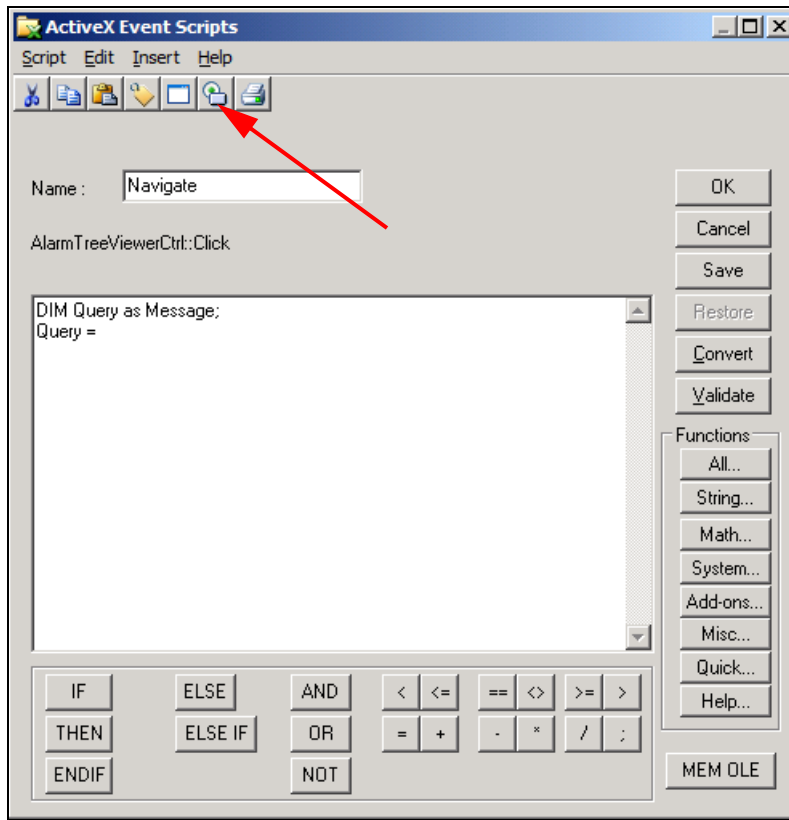
---

**Note:** The script for this lab is available to copy from the file **Lab 15 Script** in your **Training** directory.

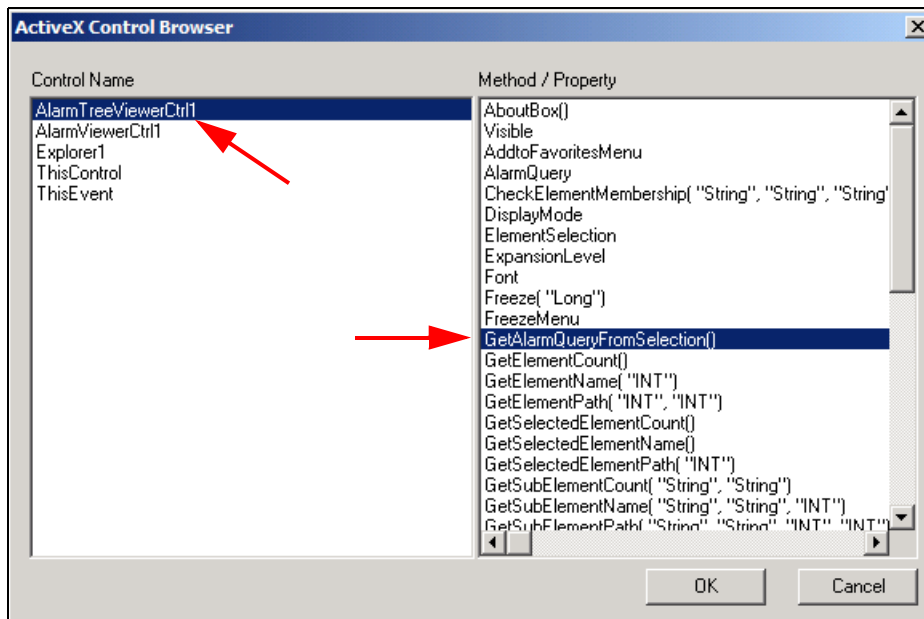
---

8. Enter the following script:  
**DIM Query as Message;**  
**Query =**

9. Click **Insert ActiveX Control**.

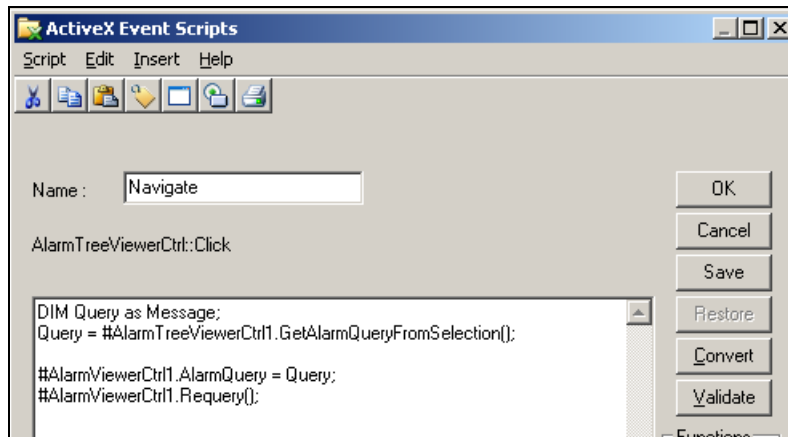


10. Click **AlarmTreeViewCtrl1**, and then double-click **GetAlarmQueryFromSelection()**.





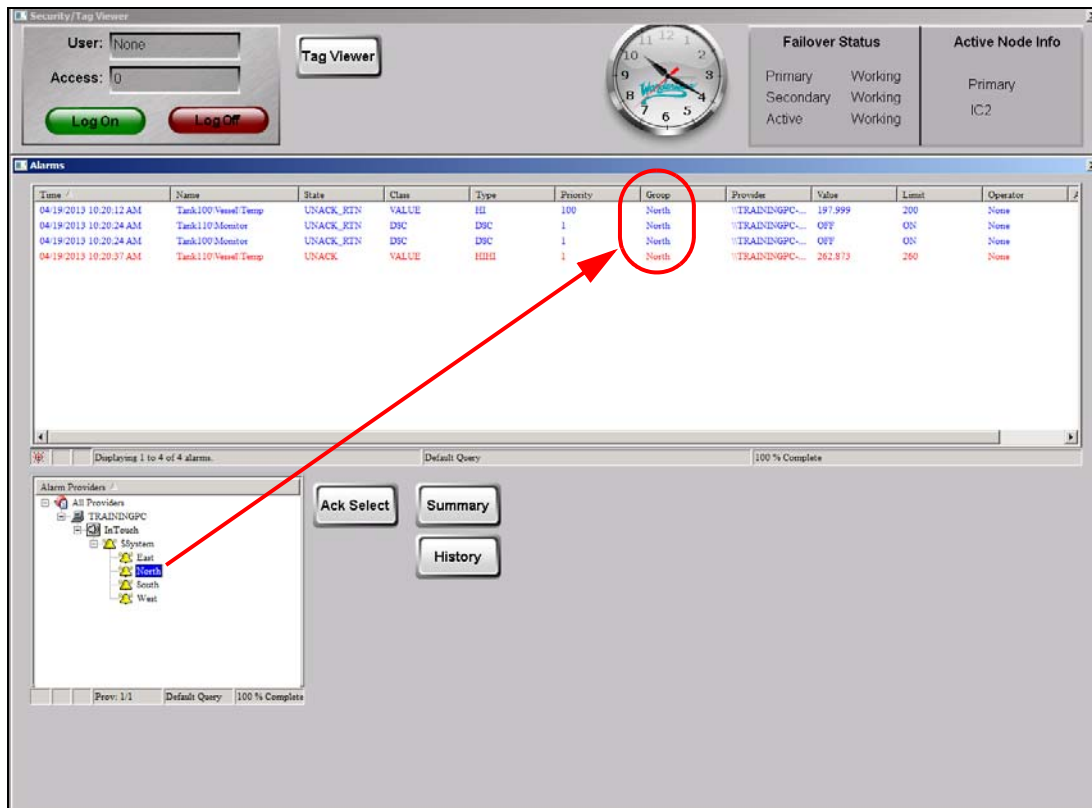
11. Add a semicolon to the end of the line:  
`Query = #AlarmTreeViewCtrl1.GetAlarmQueryFromSelection();`
12. Press **Enter**.
13. Click **Insert ActiveX Control**.
14. Click **AlarmViewerCtrl1**, double-click **AlarmQuery**, and then add `= Query;` to the end of the line:  
`#AlarmViewerCtrl1.AlarmQuery = Query;`
15. Press **Enter**, and then click **Insert ActiveX Control**.
16. Click **AlarmViewerCtrl1**, double-click **Requery**, and then add a semicolon to the end of the line:  
`#AlarmViewerCtrl1.Requery();`



17. Click **OK** to close the **ActiveX Event Script** dialog box.
18. Click **OK** to close the **Properties** dialog box of **AlarmTreeViewCtrl1**.

## Test What You Have Built

19. Click **Runtime**.
20. In the **Alarm Providers** pane, click **North** to view only alarms from the group **North**.



21. Do the same for **South**, **West**, and **East**.

**Note:** If there are no alarms to display, it will be blank.

22. Click **System** to display all of the alarms across groups.
23. Click **Development!**.

## Section 2 – Alarm Databasing

This section discusses how InTouch is used to configure alarm databasing options, and work with the alarm database view and the alarm pareto.

### Alarm DB Logger Manager Utility

The InTouch Distributed Alarm system includes the Alarm DB Logger Manager utility that logs alarms and events to Microsoft SQL Server and/or Microsoft Data Engine (MSDE).

Alarm DB Logger is an Alarm Consumer. Configure it with an alarm query that defines which alarms are to be logged. Use the Alarm DB Logger to specify alarm queries and to log the resultant alarm records. These alarm queries are sent via the Alarm Consumer interface of the Distributed Alarm System.

The Alarm DB Logger also has the ability to auto-reconnect. When the connection to the database is lost, the logger continually checks for the database connection at regular intervals. When the connection is re-established, logging proceeds. The Alarm DB Logger reports all errors whether running as a service or a normal application to the Wonderware Logger.

The Alarm DB Logger consists of the following two components:

#### Alarm DB Logger Manager Utility

This utility is a separate executable that solely takes care of starting and stopping the logging operations. It is launched and starts working either as a service or a normal application (depending upon the running mode selected in the Alarm DB Logger Manager). The logging utility retrieves the setting information from the registry and performs the logging.

#### Alarm DB Logger Configuration Utility

This utility takes care of user input and database configuration. The Alarm DB Logger Manager allows selection of the mode in which the Alarm DB Logger will run (either as a windows service or a normal application).

The Alarm DB Logger Configuration Utility provides the ability to:

- Run the application as a Windows Service or as a normal application
- Select the database connection type – SQL Server or MSDE
- Create the necessary SQL tables in the database
- Specify the alarm query that will be a part of the logging instance
- Select the logging mode – Detailed or Consolidated
- Check/Uncheck logging of events
- Set performance tuning parameter – The auto reconnect rate is not the same as the performance tuning parameter. It depends on the time out for a connection attempt associated with SQL Server
- Store the setting in the registry

### SQL Database Logging

The Alarm DB Logger logs alarm data into the database. If the OLEDB Provider is SQL Server, specify the SQL Server machine in the Alarm DB Logger Manager. Alarm DB Logger automatically creates the necessary data structures, if they do not already exist in the database.

## Alarm DB Logger Configuration

Before using the Alarm DB Logger, configure a few items, such as the database connection, the query list, and the logging mode. This information is configured via the Alarm DB Logger Manager.

The first Alarm DB Logger Manager Utility dialog box deals with the database connection. Select either SQL Server or Microsoft Data Engine (MSDE) for storage of data. The Alarm DB Logger Configuration Utility also allows the creation of necessary data structures (tables, views, and stored procedures), and to test the database connection.

### Query Configuration

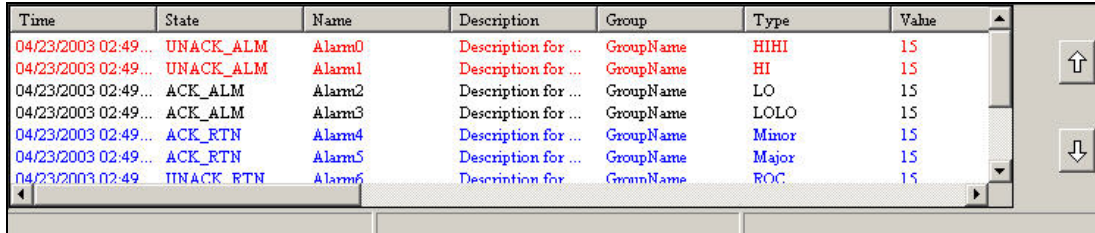
The second Alarm DB Logger Manager dialog box refers to the query details that need to be selected for alarm logging. It provides a multi-line text entry box to key in the query. It also allows the “from” and “to” alarm priority values to be selected.

### Advanced Setting Configuration

The third Alarm DB Logger Manager dialog box refers to the advanced query settings, such as the running mode (either as a Windows Service or a normal application). It also provides the option for logging events. Tune the performance of the alarm logging by setting the frequency at which the alarms are to be flushed from the memory buffer to the database.

## Alarm DB Viewer Control

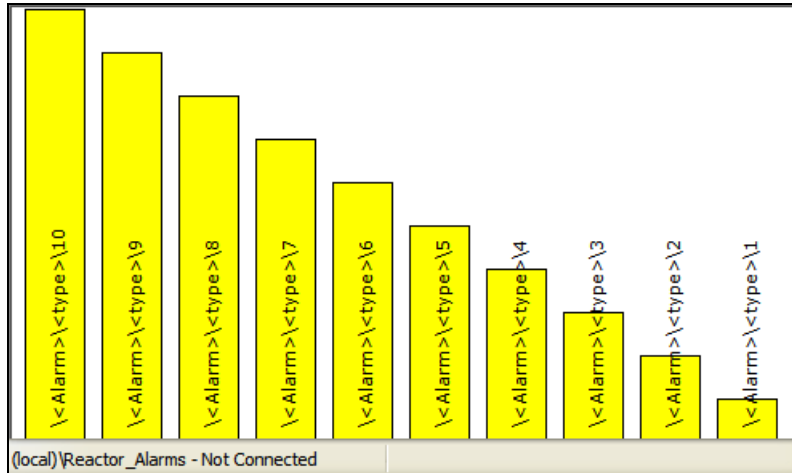
InTouch provides an Alarm DB View ActiveX Control that allows the visualization of the alarm data from the Alarm DB Logger. This control is used to view all alarm and event information.



| Time                | State     | Name   | Description         | Group     | Type  | Value |
|---------------------|-----------|--------|---------------------|-----------|-------|-------|
| 04/23/2003 02:49... | UNACK_ALM | Alarm0 | Description for ... | GroupName | HIHI  | 15    |
| 04/23/2003 02:49... | UNACK_ALM | Alarm1 | Description for ... | GroupName | HI    | 15    |
| 04/23/2003 02:49... | ACK_ALM   | Alarm2 | Description for ... | GroupName | LO    | 15    |
| 04/23/2003 02:49... | ACK_ALM   | Alarm3 | Description for ... | GroupName | LOLO  | 15    |
| 04/23/2003 02:49... | ACK_RTN   | Alarm4 | Description for ... | GroupName | Minor | 15    |
| 04/23/2003 02:49... | ACK_RTN   | Alarm5 | Description for ... | GroupName | Major | 15    |
| 04/23/2003 02:49... | UNACK_RTN | Alarm6 | Description for ... | GroupName | ROC   | 15    |

## Alarm Pareto Control

The Alarm Pareto Control provides a bar graph showing the frequencies of alarms throughout a period of time based on records from the alarm database. The bars are placed in order of frequency, either from left-to-right or top-to-bottom in descending order.



The Alarm Pareto Control supports three basic ways of measuring alarms:

- **Count:** Counts the occurrences of items
- **Alarm Time:** Adds the total time spent in alarm
- **Ack Time:** Adds the total time from onset of alarm to acknowledgment



# Lab 16 – Working with the Alarm DB Manager

## Introduction

In this lab, you will create and configure the Alarm Database (DB) logger manager to connect and log InTouch alarms to the Microsoft SQL server database. Then in InTouch create an interface to interact with the Alarm Database.

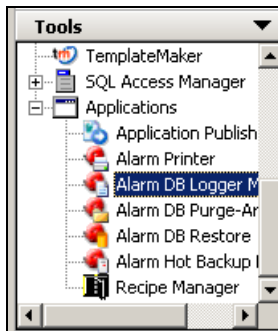
## Objectives

Upon completion of this lab, you will be able to:

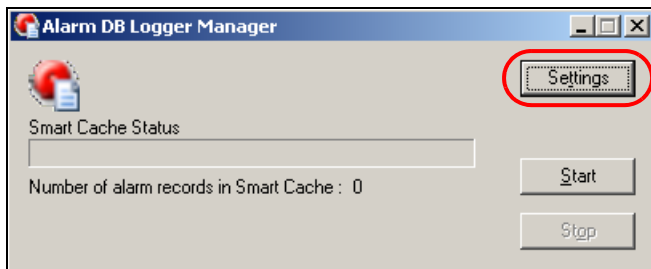
- Log alarms to a database
- Add the Alarm DB view control
- Apply data filters
- Add an interface to change query duration

## Configure the Alarm Database

1. In WindowMaker, in **Tools** pane, expand **Applications**, and then double-click **Alarm DB Logger Manager**.



2. Click **Settings**.



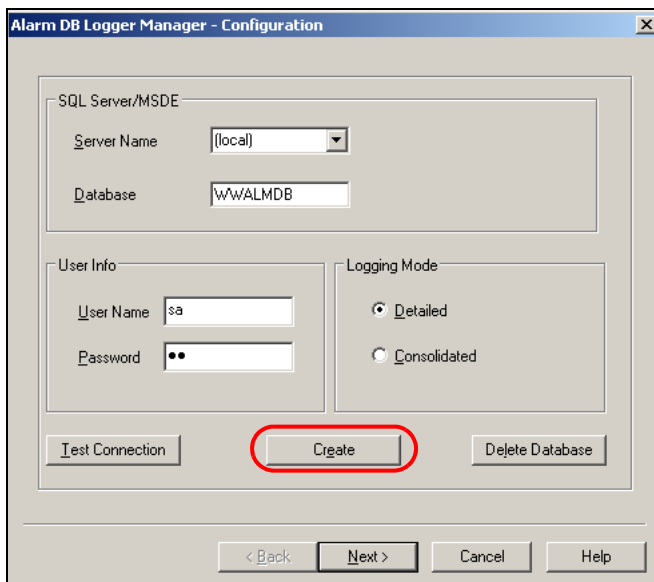
It may take a few moments for the **Alarm DB Logger Manager** dialog box to appear.

3. Enter the **SQL Server** and user information.

---

**Note:** Your instructor will provide the necessary connection information.

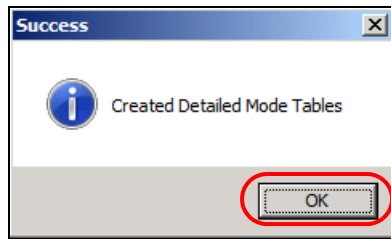
---



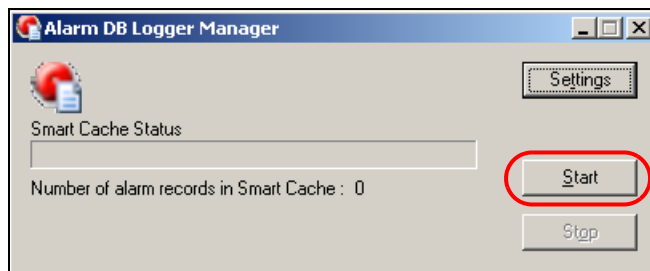
4. Click **Create**.



5. Click **OK**.



6. In the **Alarm DB Logger Manager – Configuration** dialog box, click **Next**.
7. In the **Alarm DB Logger Manager – Query Selection** dialog box, click **Next** to accept the default query of `\\InTouch!$System`.
8. In the **Alarm DB Logger Manager – Advanced Setting** dialog box, click **Finish**.
9. Click **Start**.



You may observe the count dropping as each of the **Alarm** records in the **Smart Cache** are written to the **WWALMDB** database.

10. When count reaches to 0 and stops, click **X** to close the **Alarm DB Logger Manager** dialog box.

---

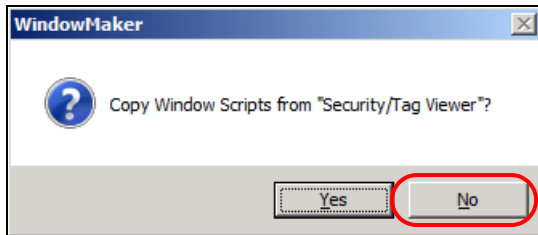
**Note:** Do not click **Stop** as this will kill the process.

---

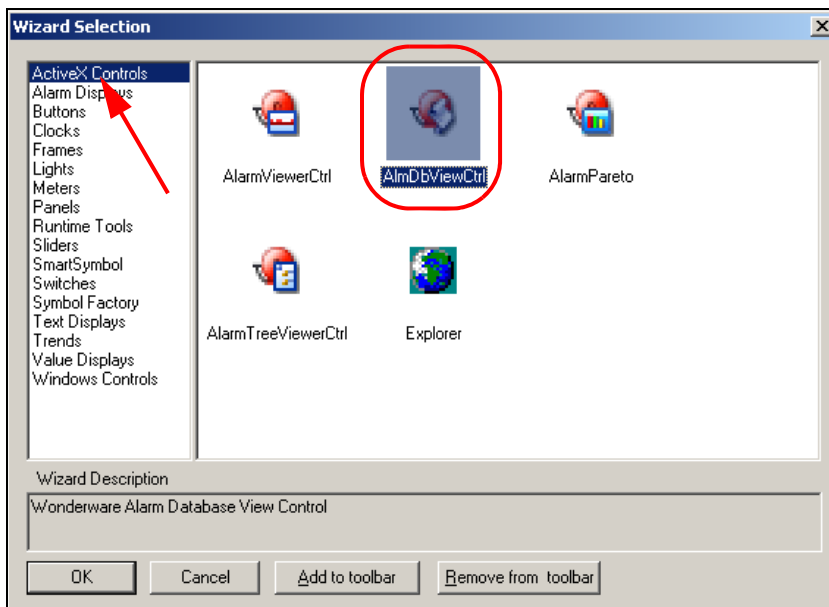
11. In the **Alarms** window, click in a blank area to set focus to the window.

This will ensure the dimensions and properties of the new window you will create are the same as the **Alarms** window.

12. On the **File** menu, click **New Window**.  
13. Click **No** if prompted.

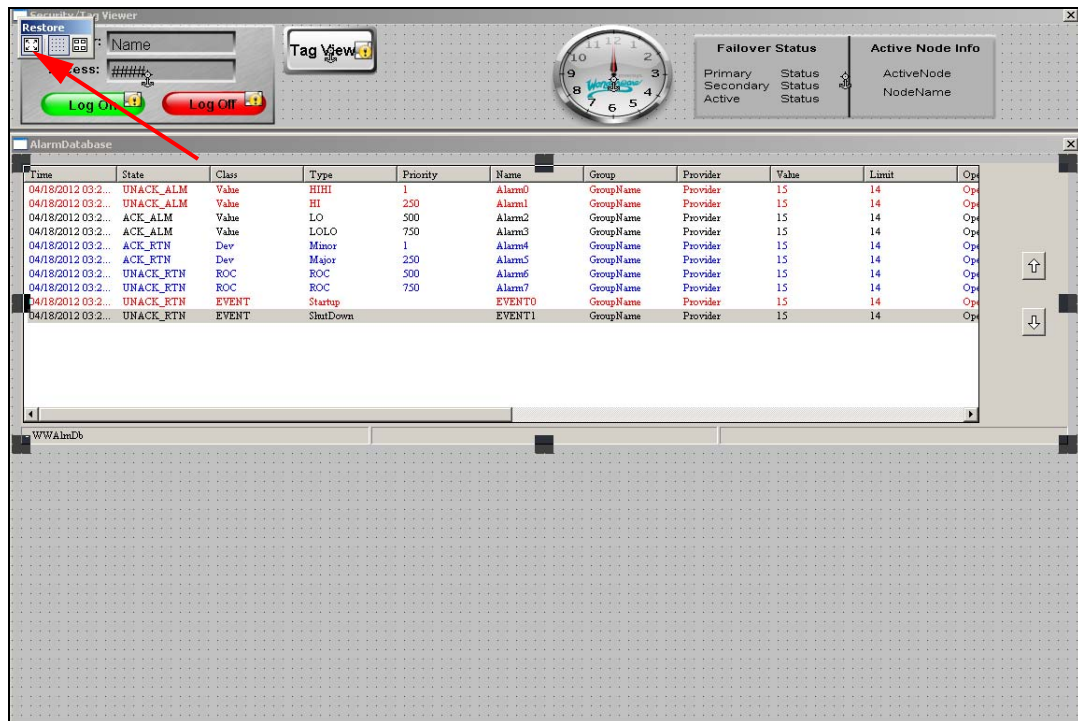


14. Enter **AlarmDatabase** as the window name.  
The window dimensions should be the same as the **Alarms** window dimension.  
15. Click **OK**.  
16. Close all other windows except for **AlarmDatabase** and **Security/Tag Viewer**.  
17. Click **Wizards**.  
18. Select **ActiveX Controls**, and then double-click **AlmDbViewCtrl**.

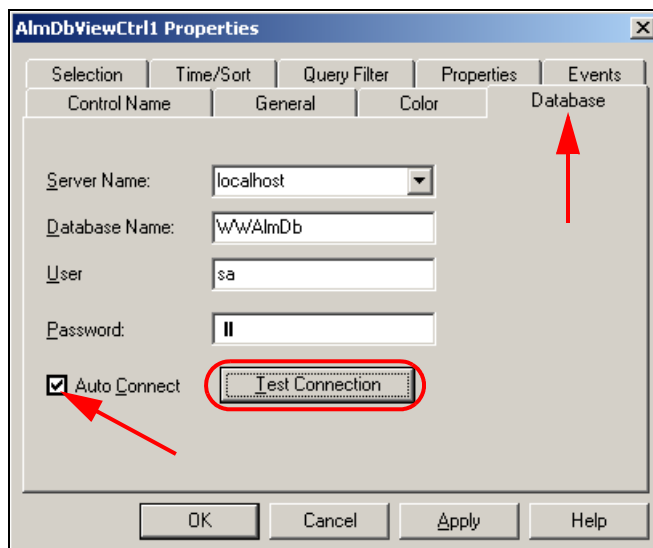


19. In the **AlarmDatabase** window, place the symbol.

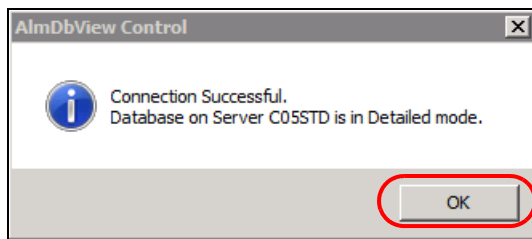
20. Switch to full screen mode and size the **AlmDbViewCtrl** object so that it takes up half the height and the full width of the **AlarmDatabase** window.



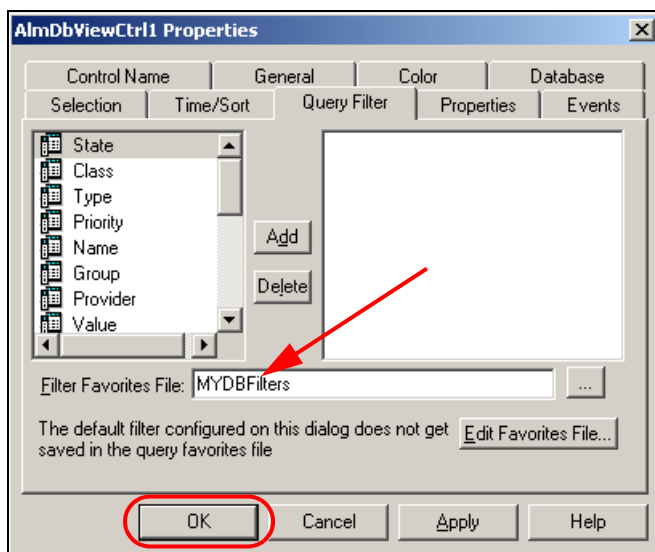
21. Exit full screen mode.
22. Double-click **AlmDbViewCtrl** to configure.
23. Click the **Database** tab and enter the information provided by your instructor.
- It may take a few moments for the **Properties** dialog box to appear.
24. Check **Auto Connect**, and then click **Test Connection**.



25. Click **OK**.



26. Select the **Query Filter** tab, in the **Filter Favorites File** field enter **MYDBFilters**, and then click **OK**.

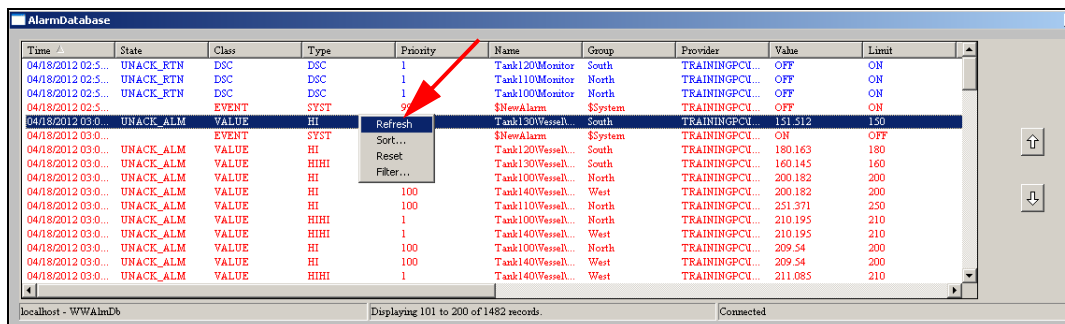


## Test What You Have Built

27. Click **Runtime**.

28. Use the down arrow to show additional alarms.

29. On the display, right-click and select **Refresh** to show any new alarms that have occurred since the last time the **Alarm DB** refreshed.

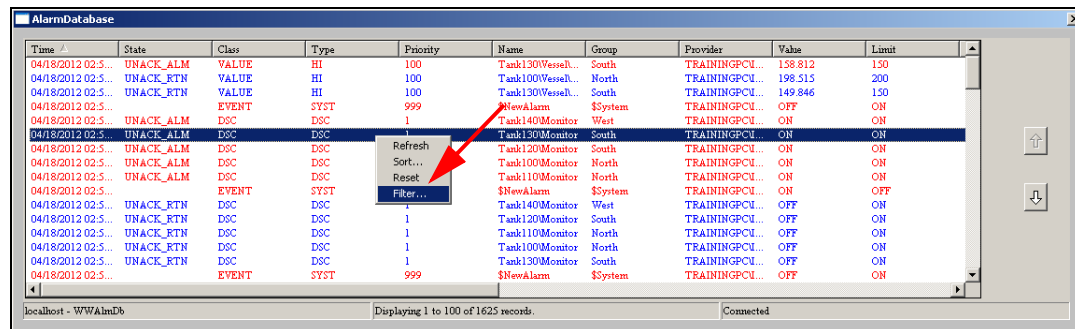


Next you will work with the filtering functionality.

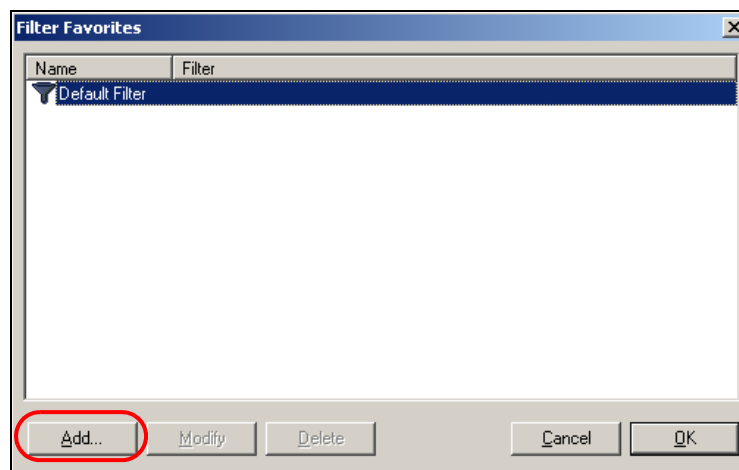
## Build Filters

First, you will build a filter to include only the Unacknowledged Return to Normal state alarms.

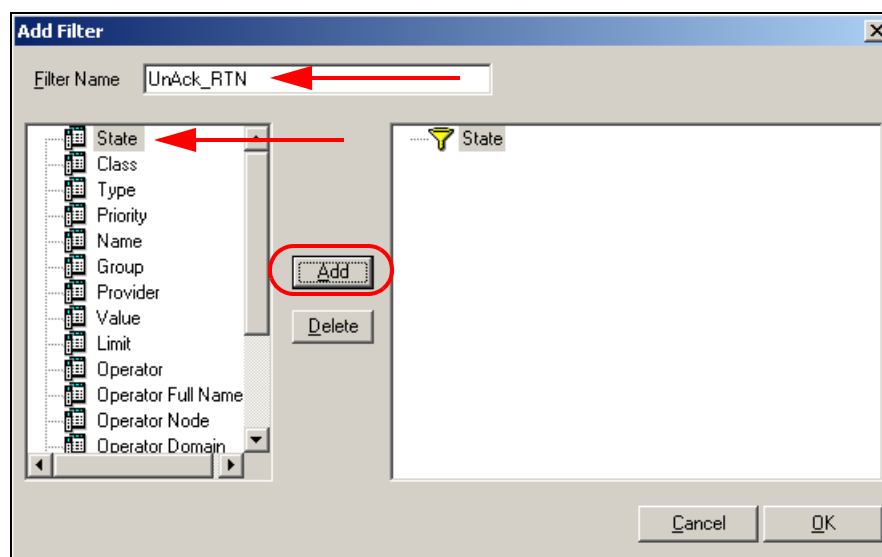
30. On the display, right-click and select **Filter**.



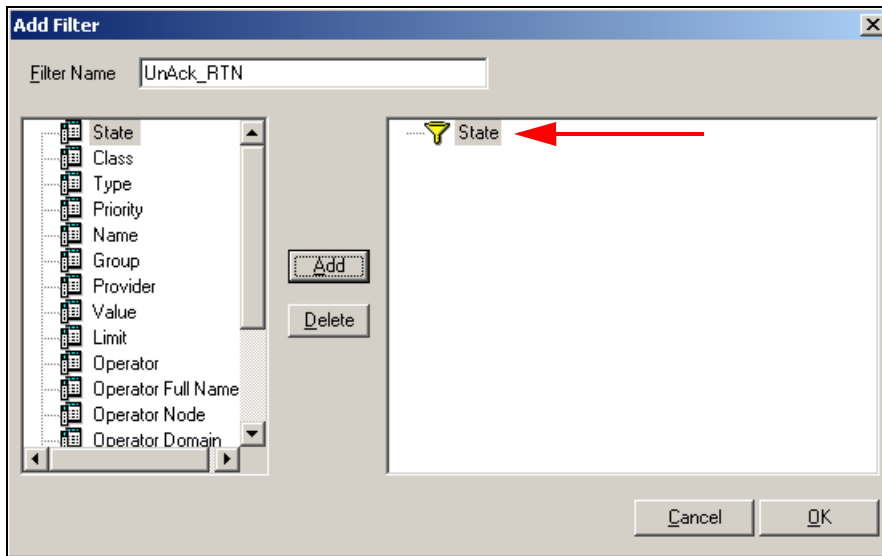
31. Click **Add**.



32. In the **Filter Name** field enter **UnAck\_RTN**, select **State**, and then click **Add**.



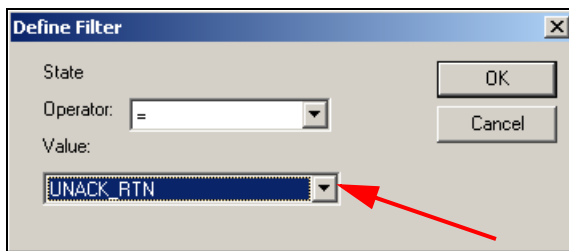
33. In the right pane, double-click **State**.



34. Configure as follows:

**Operator:**     =

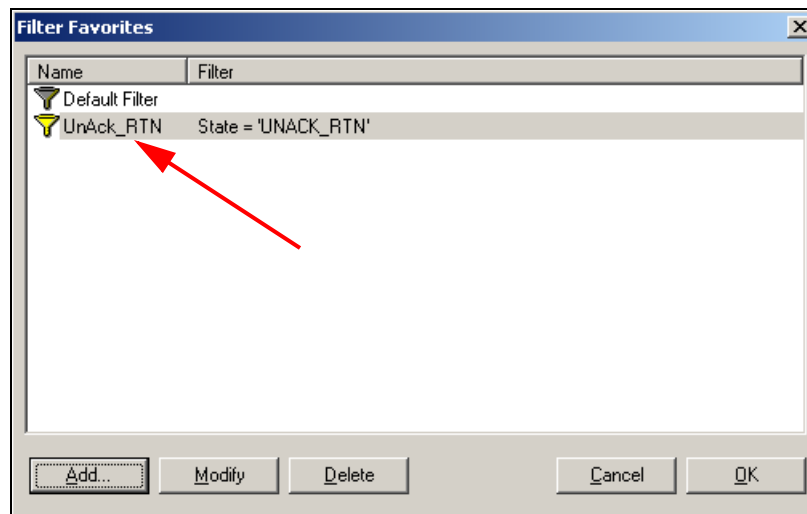
**Value:**        UNACK\_RTN



35. Click **OK**.

36. In the **Add Filter** dialog box, click **OK**.

37. Double-click **UnAck\_RTN** to apply the filter.



Now you are viewing only the unacknowledged return to normal state alarms.

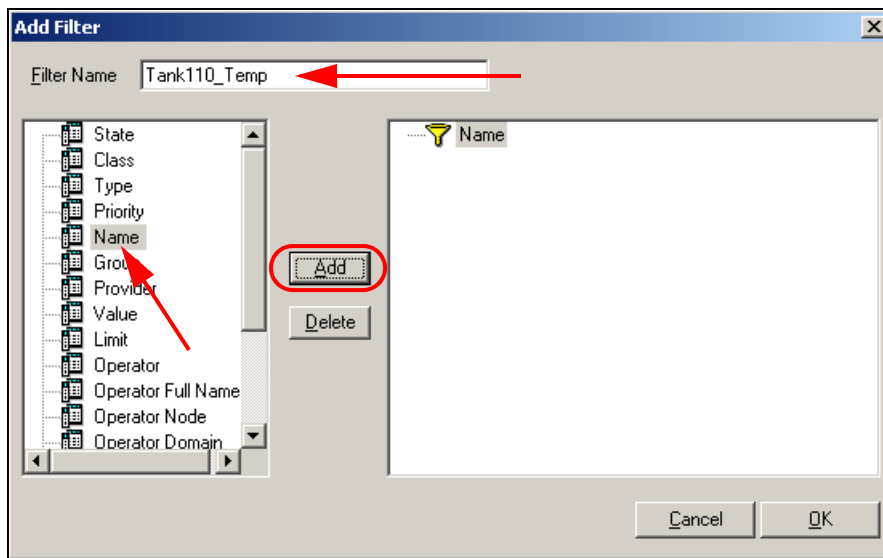
The image shows the 'AlarmDatabase' window. It contains a table with columns: Time, State, Class, Type, Priority, and Name. The 'State' column is circled in red. The table displays 16 rows of alarm data, all with 'UNACK\_RTN' in the State column. The status bar at the bottom indicates 'localhost - WWAlmDb' and 'Displaying 1 to 100 of 868 records'.

| Time               | State     | Class | Type | Priority | Name |
|--------------------|-----------|-------|------|----------|------|
| 04/18/2012 02:5... | UNACK_RTN | VALUE | HI   | 100      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | HI   | 100      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | VALUE | LO   | 250      | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |
| 04/18/2012 02:5... | UNACK_RTN | DSC   | DSC  | 1        | Ta   |

Next, you will add a new filter to include only Tank110 temperature alarms.

38. On the display, right-click and select **Filter**, and then click **Add**.

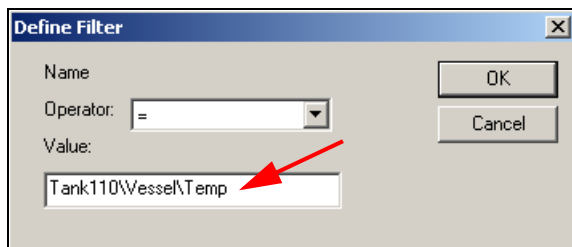
39. In the **Filter Name** field enter **Tank110\_Temp**, select **Name**, and then click **Add**.



40. In the right pane, double-click **Name** and configure as follows:

**Operator:**     =

**Value:**        Tank110\Vessel\Temp



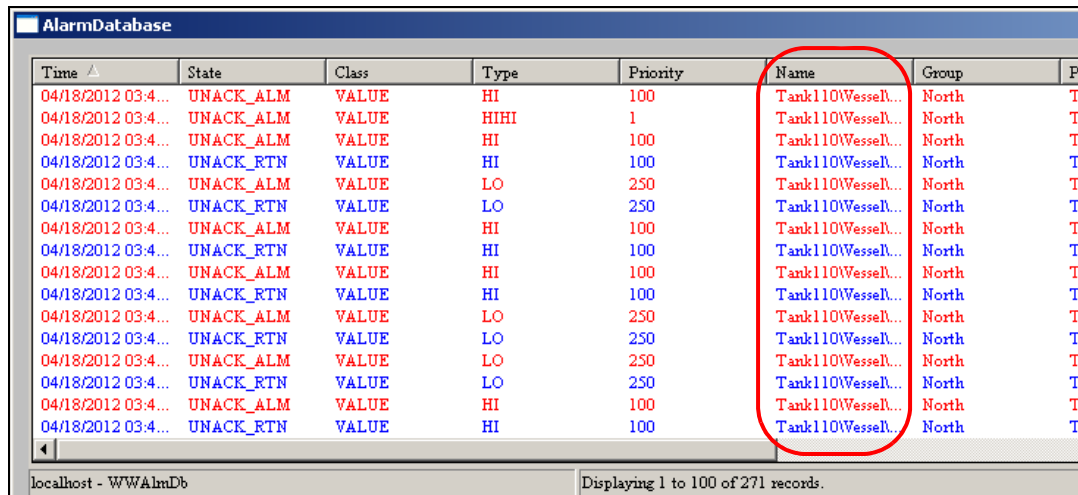
41. Click **OK**.

42. In the **Add Filter** dialog box, click **OK**.



43. Double-click **Tank110\_Temp** to apply the filter.

Now you are viewing only alarms pertaining to **Tank110** temperature alarms.

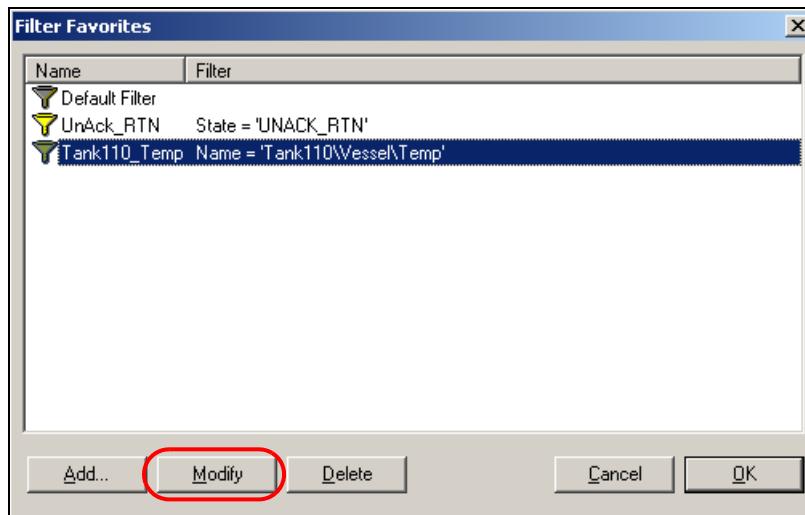


| Time               | State     | Class | Type | Priority | Name             | Group | P |
|--------------------|-----------|-------|------|----------|------------------|-------|---|
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HIHI | 1        | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | LO   | 250      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_ALM | VALUE | HI   | 100      | Tank110Vessel... | North | T |
| 04/18/2012 03:4... | UNACK_RTN | VALUE | HI   | 100      | Tank110Vessel... | North | T |

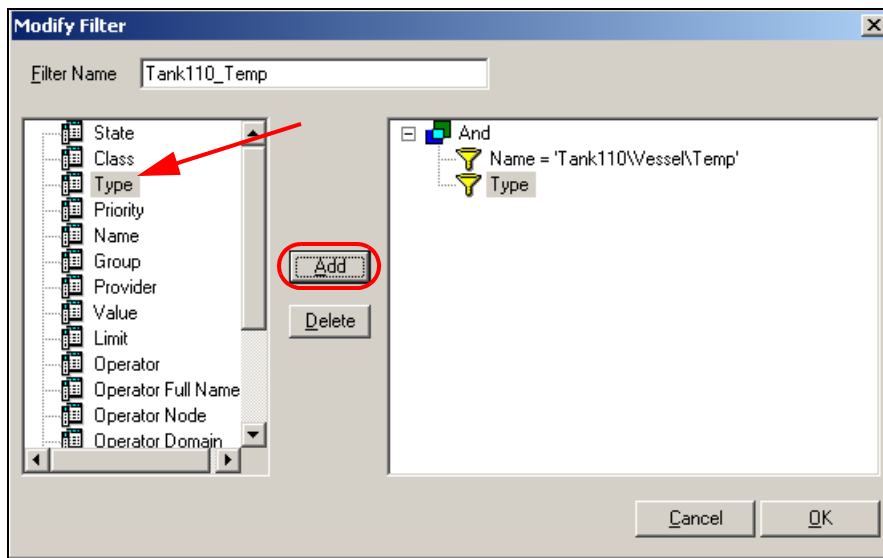
localhost - WWAlmDb      Displaying 1 to 100 of 271 records.

Lastly, you will modify the Tank110\_Temp filter.

44. On the display, right-click and select **Filter**, and then click **Modify**.

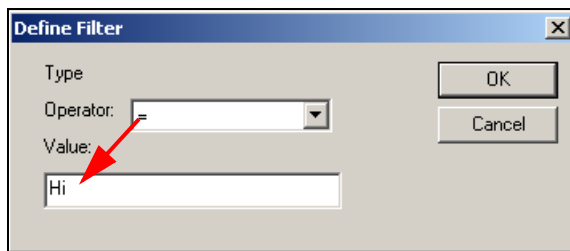


45. Select **Type**, and then click **Add**.

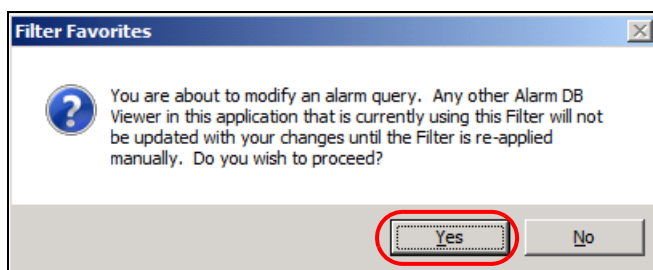


46. In the right pane, double-click **Type** and configure as follows:

**Operator:**     =  
**Value:**        Hi



47. Click **OK**.  
48. In the **Add Filter** dialog box, click **OK**.  
49. In the **Filter Favorites** dialog box, click **Yes**.



50. Double-click **Tank110\_Temp** to apply the filter.

Now you are viewing only the alarms pertaining to **Tank110** temperature alarms that have a **HI** as the type of alarm.

| AlarmDatabase      |           |       |      |          |                   |       |
|--------------------|-----------|-------|------|----------|-------------------|-------|
| Time ▲             | State     | Class | Type | Priority | Name              | Group |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_ALM | VALUE | HI   | 100      | Tank110\Vessel... | North |
| 04/18/2012 03:5... | UNACK_RTN | VALUE | HI   | 100      | Tank110\Vessel... | North |

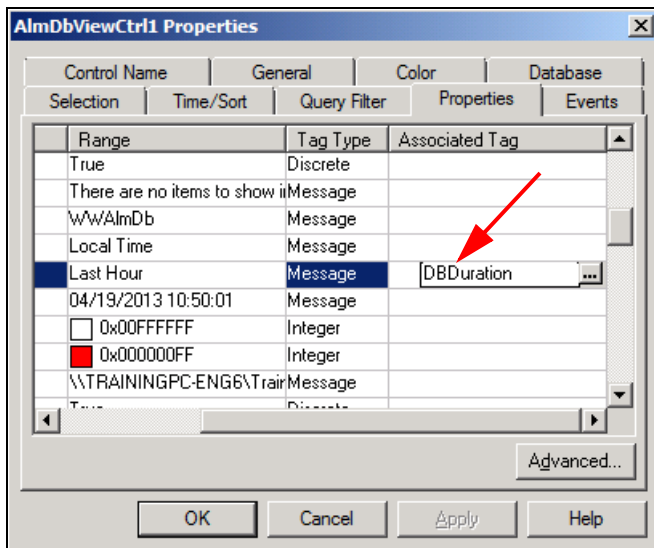
localhost - WWAlmDb      Displaying 1 to 100 of 139 records.

51. Click **Development!**.

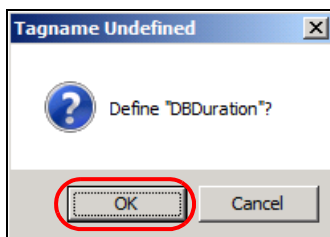
## Time Selections

The right-click menu found on the display during runtime does not give any access to time information. To modify and allow different time selections, you will need to build the functionality into your alarm display.

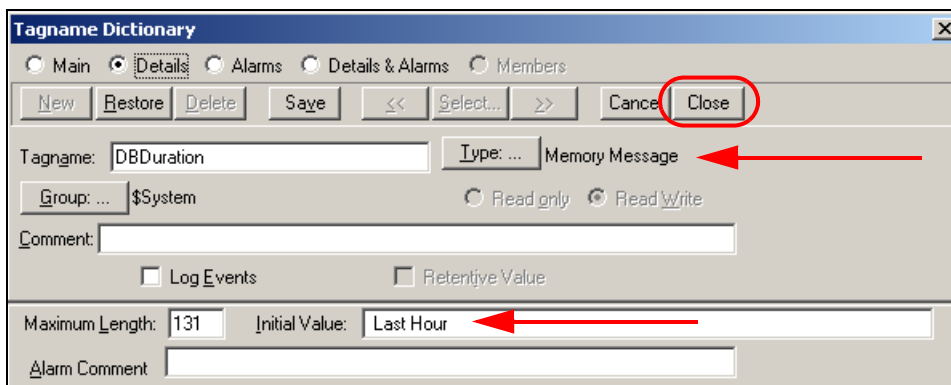
52. Double-click on the alarm display.
53. Click the **Properties** tab, scroll down to **Duration**, and scroll to the right until you see the **Associated Tag** column, enter **DBDuration**, and then press **Enter**.



54. In the **Tagname Undefined** dialog box, click **OK**.

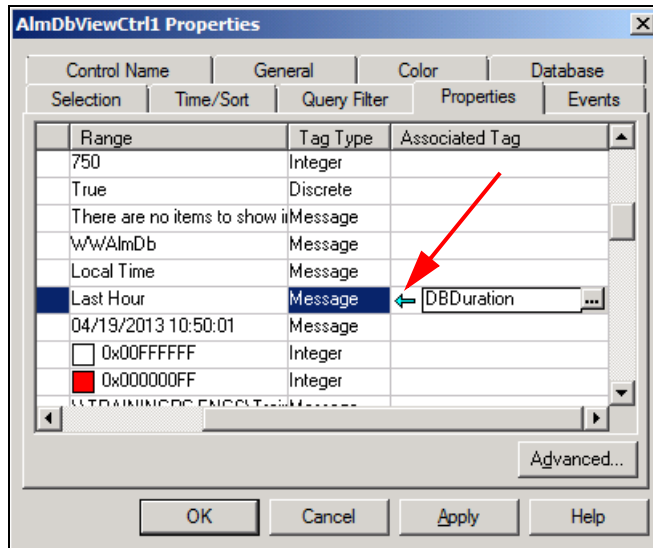


55. Ensure the **Type** is set to **Memory Message**, in the **Initial Value** field enter **Last Hour**, and then click **Close**.

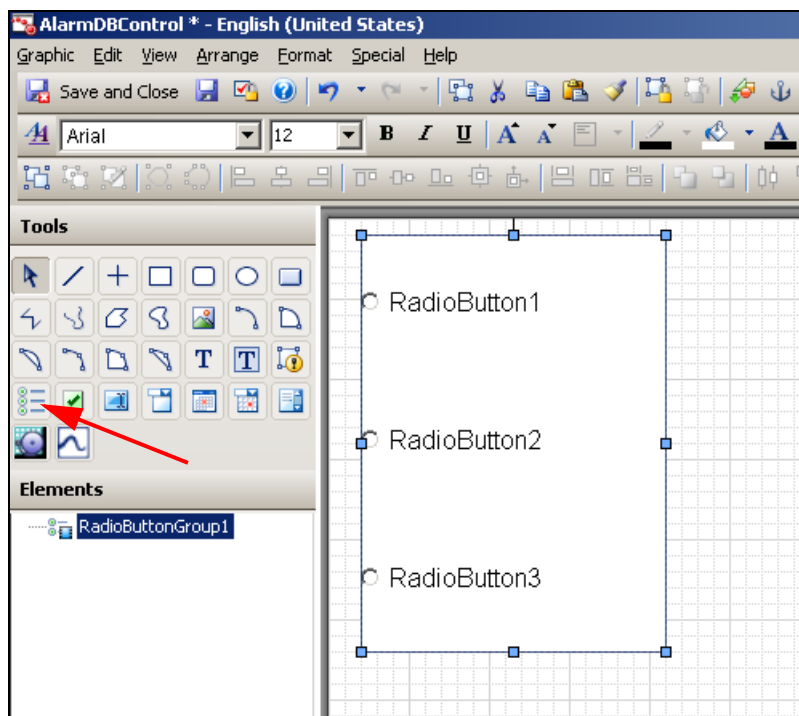


56. Click **OK**.

57. Double-click on the alarm display.
58. Click the **Properties** tab, scroll down to **Duration**, and then scroll to the right until you see the **Associated Tag** column.
59. Double-click the double-arrow until it becomes a left pointing arrow (read-only) and click **OK**.

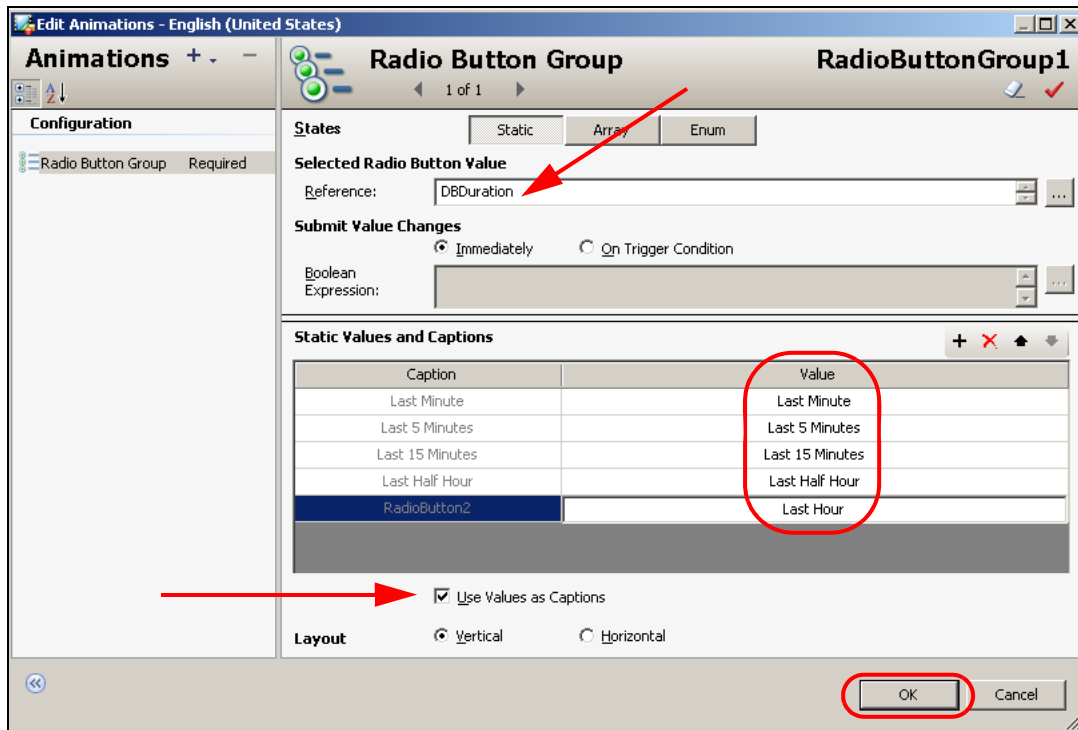


60. Close WindowViewer so that the initial value will be implemented.
- Next, you will provide a method to change the time value.
61. In Archestra IDE, right-click **ClassSymbols** and select **New | Symbol**.
62. Name it **AlarmDBControl**, and then double-click to open it.
63. On the canvas, place a radio button group, and then double-click to open it for editing.



64. Check **Use Values as Captions**, and then configure as follows:

**Selected Radio Button Value:** DBDuration  
**Static Values and Captions (5 values):** Last Minute  
Last 5 Minutes  
Last 15 Minutes  
Last Half Hour  
Last Hour

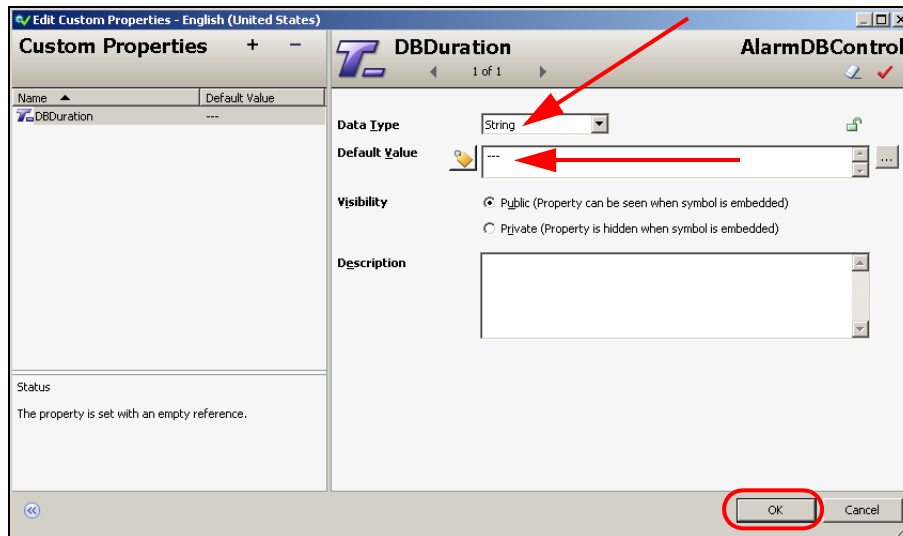


65. Click **OK**.

66. On the canvas, right-click and select **Custom Properties**.

67. Click **+** and enter **DBDuration** as the name.

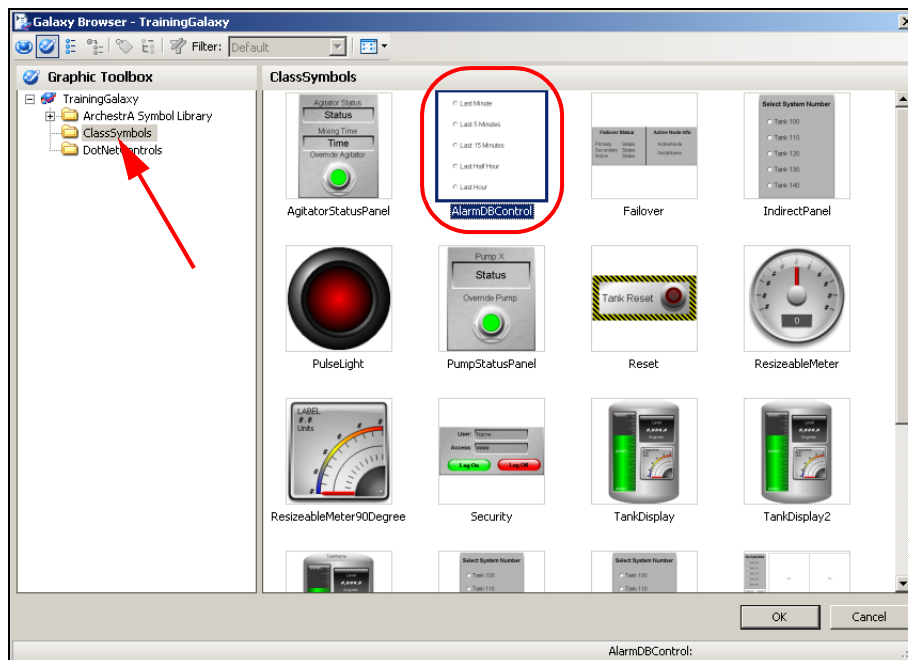
68. Set the **Data Type** to **String**, click the static text button  to switch to expression mode , enter **---** as the **Default Value**, and then click **OK**.



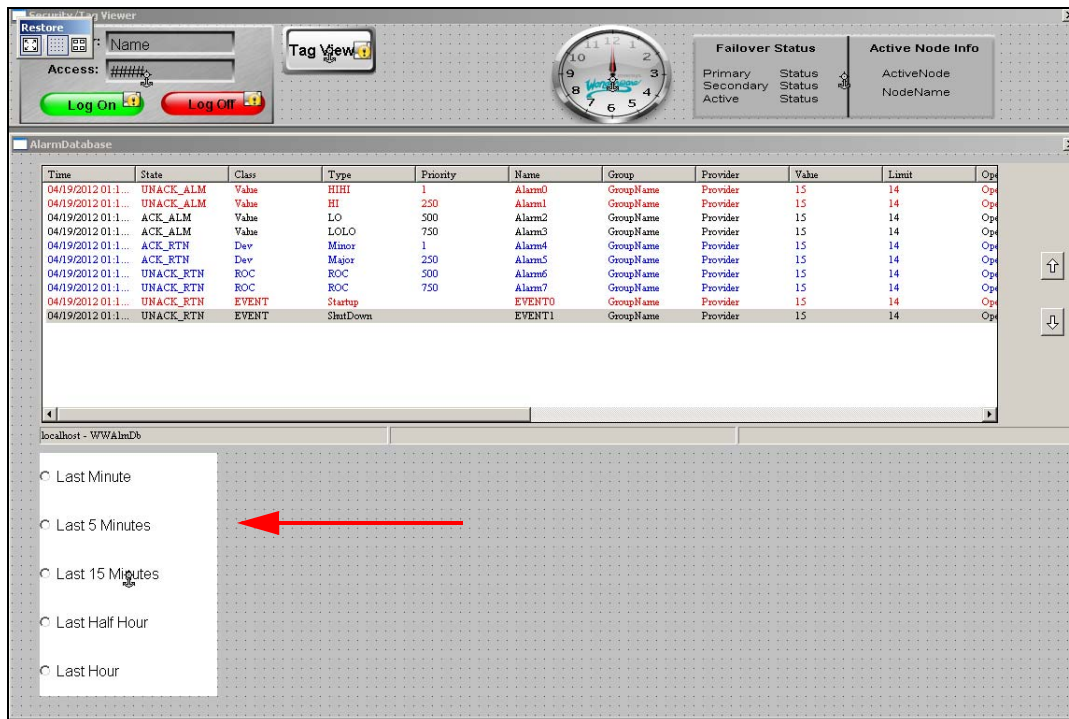
69. Click **Save and Close**, and then check in the symbol.

70. In WindowMaker, click **Embed ArchestrA Graphic**.

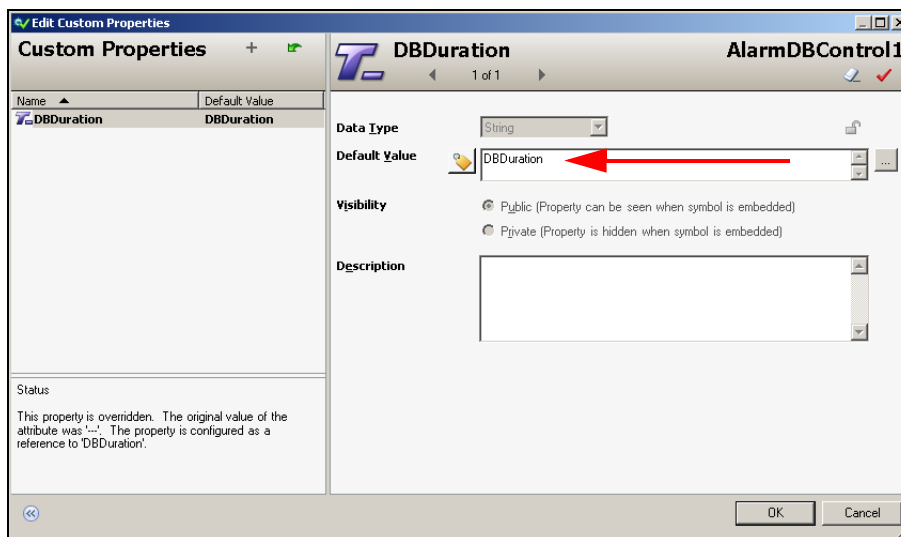
71. Select **ClassSymbols**, and then double-click **AlarmDBControl**.



72. In the **AlarmDatabase** window, place **AlarmDBControl** on the left and directly below the **Alarm** display.



73. Double-click to configure the custom properties and set the **Default Value** to **DBDuration**.



74. Click **OK**.



The last thing you will need is scripting to trigger a refresh based off of the DBDuration.

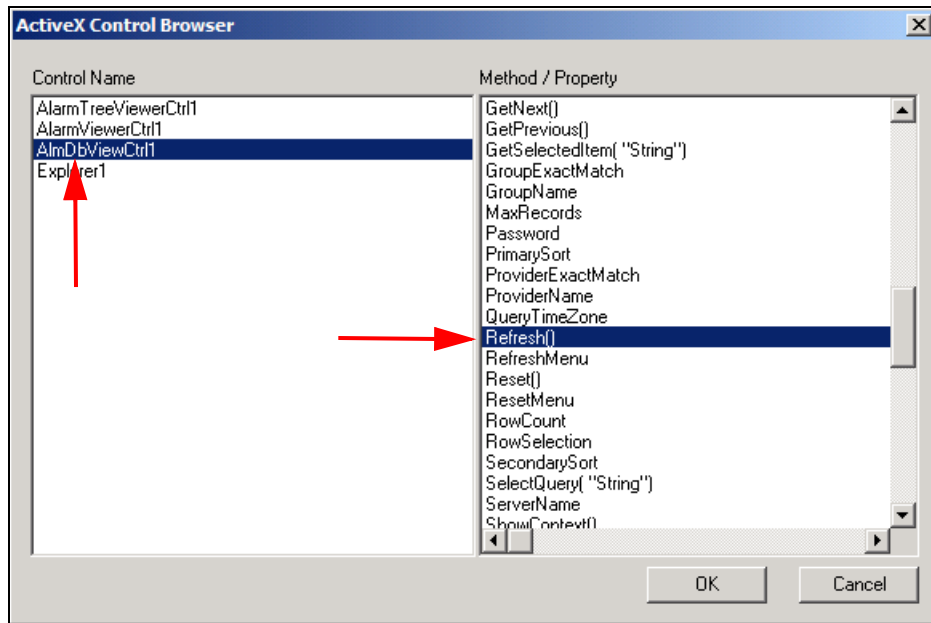
75. On the **Special** menu, click **Scripts | Data Change Scripts**.

76. Click **Script | New**.

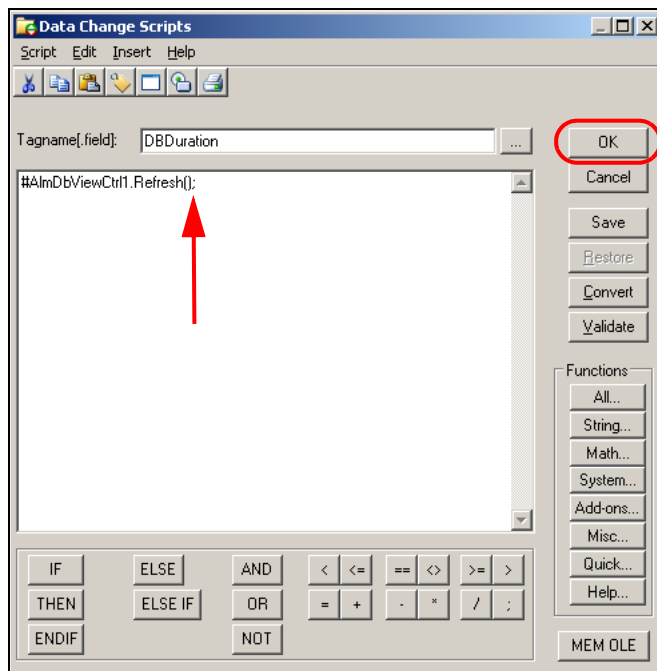
77. In **Tagname[.field]**, enter **DBDuration**.

78. Clear out any existing scripts, and then click **Insert ActiveX Control**.

79. Select **AlmDbViewCtrl1**, and then double-click **Refresh**.



80. Add a semicolon to the end of the script, and then click **OK**.



## **Test What You Have Built**

81. Click **Runtime**.
82. Select each of the radio buttons to see the number of records change. Test it along with changing the filter.
83. Click **Development!**.

# Lab 17 – Working with the Alarm Pareto Control

## Introduction

In this lab, you will add and configure the Alarm Pareto control. This provides a bar chart type view of data based alarms, showing the most frequent alarms in terms of count or time in state.

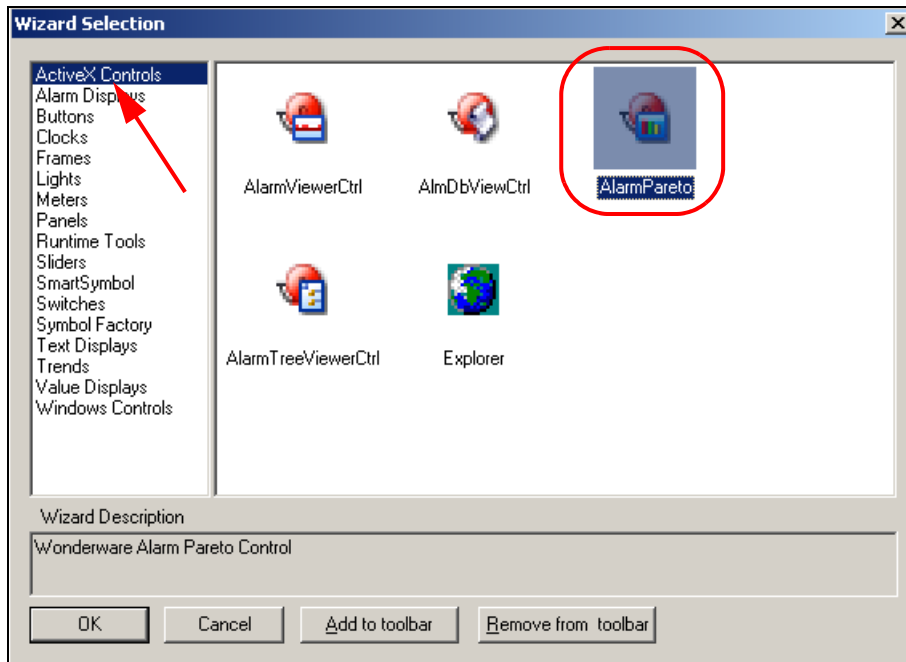
## Objectives

Upon completion of this lab, you will be able to:

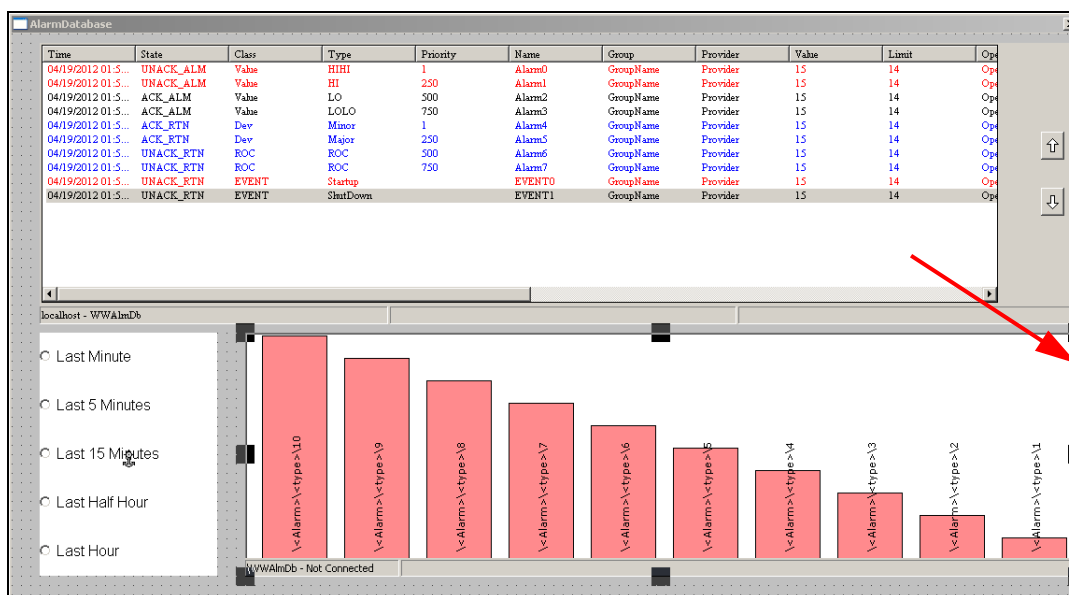
- Use the Alarm Pareto control for both counts and time in state
- Modify existing scripts to work in conjunction with the control

## Add the Alarm Pareto Control

1. In WindowMaker, ensure the **AlarmDatabase** and **Security/Tag Viewer** windows are open, and then close all others.
2. Click **Wizards**.
3. Select **ActiveX Controls**, and then double-click **AlarmPareto**.



4. In the **AlarmDatabase** window, place it to the right of the radio buttons, and then extend it to fit the remaining width and height of the window.

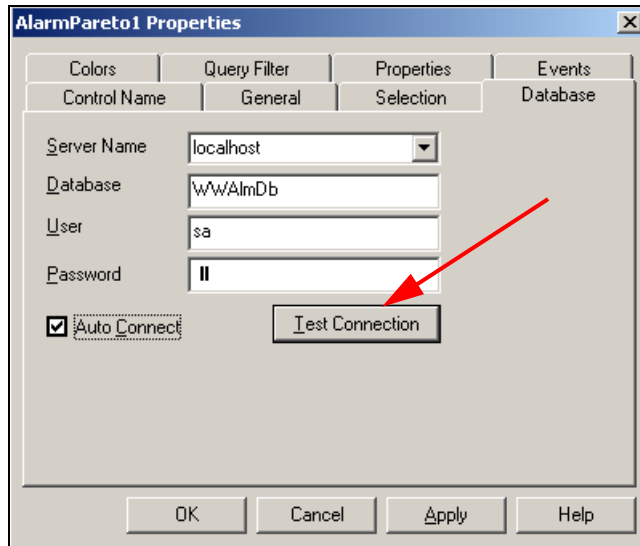


**Note:** You may need to switch to full screen mode to accomplish this.

5. Double-click **AlarmPareto** to configure.
6. Select the **Database** tab and enter the information provided by your instructor in the previous lab.

It may take a few moments for the **Database** tab to display.

7. Check **Auto Connect**, and then click **Test Connection**.

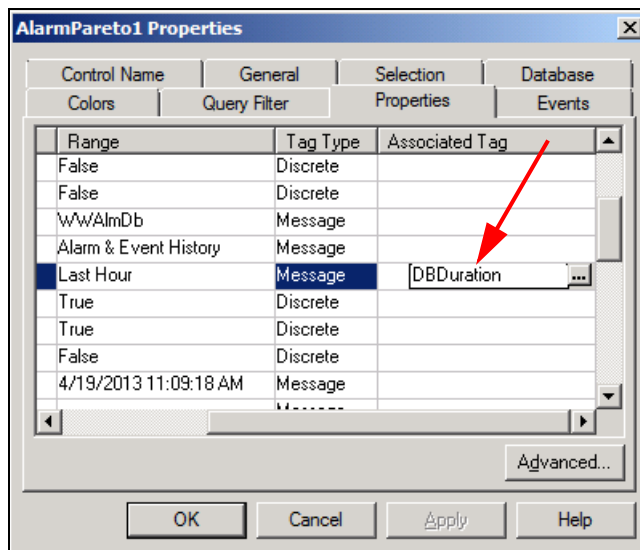


**AlarmPareto1 Properties**

Colors | Query Filter | Properties | Events  
Control Name | General | Selection | Database

Server Name: localhost  
Database: WwAlmDb  
User: sa  
Password:   
☒ Auto Connect  
Test Connection  
OK Cancel Apply Help

8. Click the **Properties** tab, scroll down to **Duration**, scroll to the right until you see the **Associated Tag** column, enter **DBDuration** and then press **Enter**.



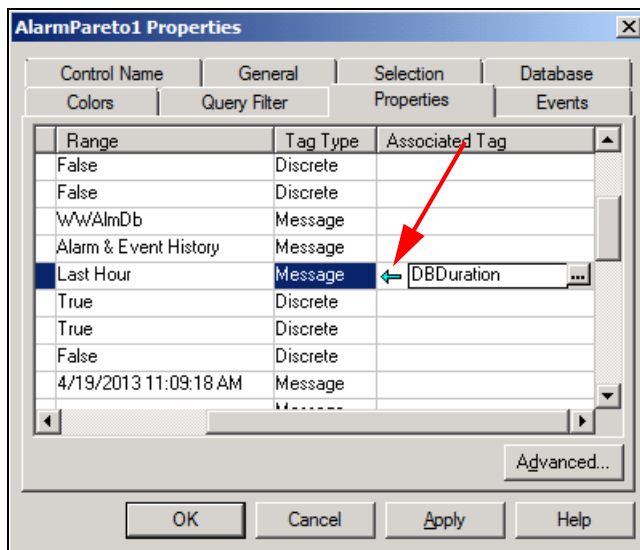
**AlarmPareto1 Properties**

Control Name | General | Selection | Database  
Colors | Query Filter | Properties | Events

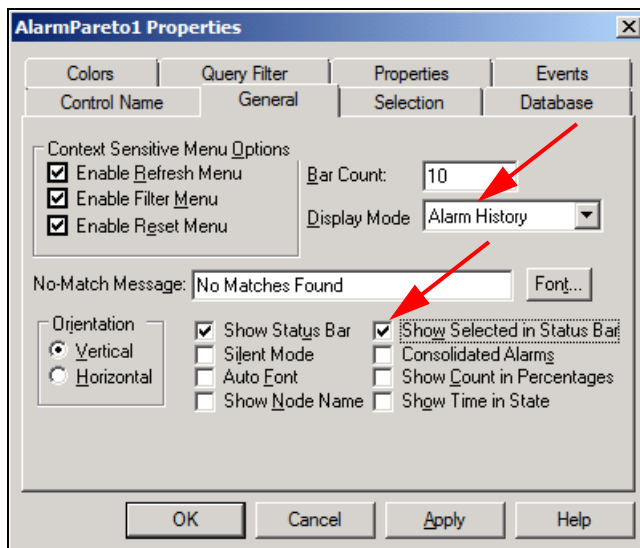
| Range                 | Tag Type | Associated Tag |
|-----------------------|----------|----------------|
| False                 | Discrete |                |
| False                 | Discrete |                |
| WwAlmDb               | Message  |                |
| Alarm & Event History | Message  |                |
| Last Hour             | Message  | DBDuration     |
| True                  | Discrete |                |
| True                  | Discrete |                |
| False                 | Discrete |                |
| 4/19/2013 11:09:18 AM | Message  |                |

Advanced...  
OK Cancel Apply Help

9. Double-click the double-arrow until it becomes a left pointing arrow (read-only).



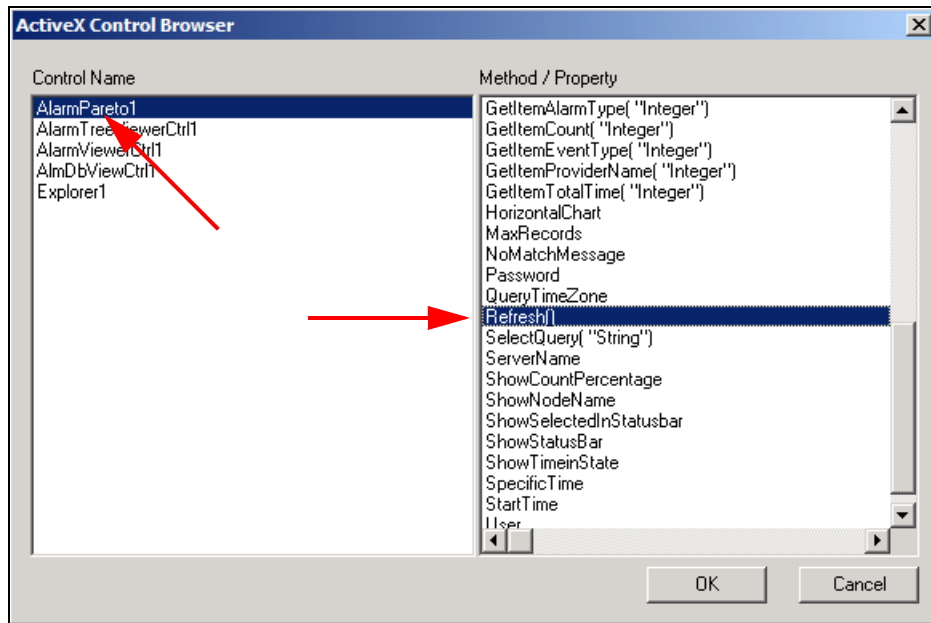
10. Click the **General** tab, set the **Display Mode** to **Alarm History**.
11. Check **Show Selected in Status Bar**.



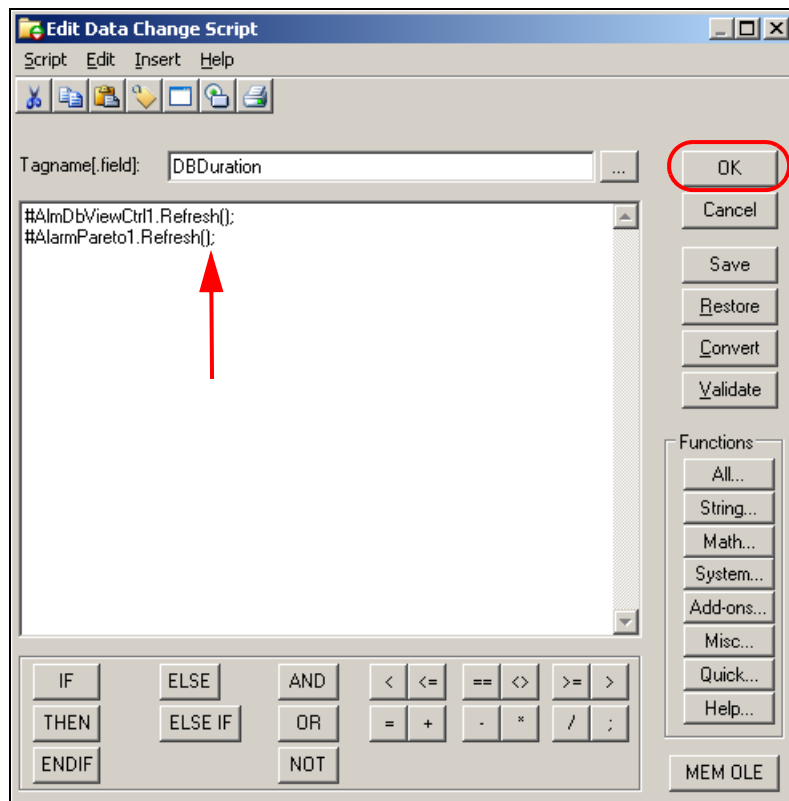
12. Click **OK**.

The last thing you will need to do is modify the DBDuration script to include the Pareto control.

13. On the **Special** menu, click **Scripts | Data Change Scripts**.
14. At the end of the existing script, press **Enter** to add a new line.
15. Click **Insert ActiveX Control**.
16. Select **AlarmPareto1**, and then double-click **Refresh**.



17. Add a semicolon to the end of the script, and then click **OK**.



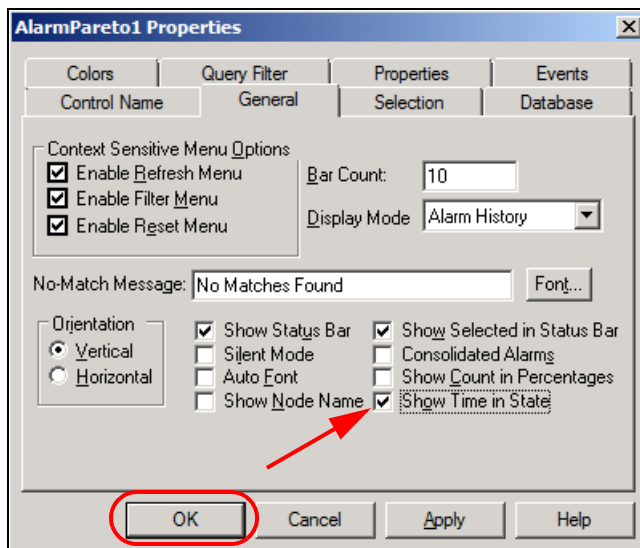




## Configure for Time in State

With additional configuration you can view time in state data as well.

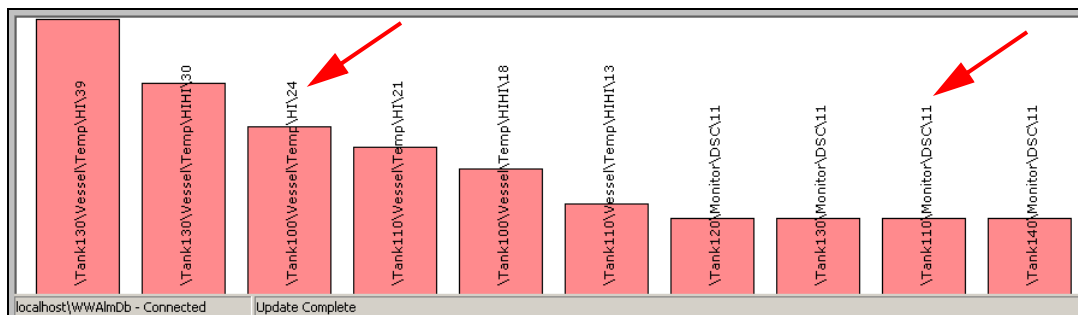
21. Double-click the **AlarmPareto1** control.
22. Click the **General** tab, and then check **Show Time in State**.



23. Click **OK**.

## Test What You Have Built

24. Click **Runtime**.
25. Notice that the top 10 alarms in terms of time in state are listed.



26. Click **Development!**.

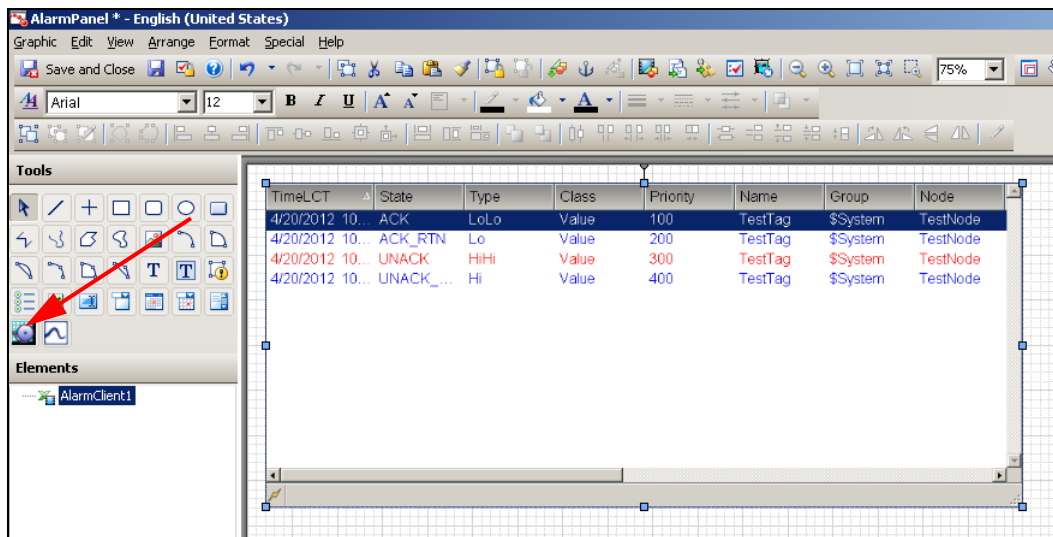
## Section 3 – Alarm Client

This section discusses the ArchestrA Graphics Alarm Client.

### ArchestrA Alarm Control

ArchestrA Alarm Control is a graphical element within the ArchestrA symbols that shows current and historical alarms and events. The ArchestrA Alarm Control replaces the Alarm Viewer control and Alarm DB View control in the InTouch HMI and extends alarm visualization to the ArchestrA Graphics environment.

ArchestrA Alarm Control is placed directly from the Tools pane in ArchestrA Symbol Editor onto the canvas and can be customized by adding further graphics, interactions, and scripts.



The Alarm Client provides the capability of selecting the alarm data and showing it from WindowViewer.

### Client Modes

The Alarm Control supports five different client modes, which can be grouped depending on their data source.

The Alarm Manager manages the following items:

- Current Alarms
- Recent Alarms and Events

The Alarm Database stores the following items:

- Historical Alarms
- Historical Events
- Historical Alarms and Events

The Alarm Control, in one of the current client modes, continuously shows the current value and quality of a tag in alarm state from InTouch running on the local computer.

| State | Type | Name      | Value    | Limit | CurrentValue | Quality |
|-------|------|-----------|----------|-------|--------------|---------|
| UNACK | HIHI | tanklevel | 553.2711 | 950   | 971.1335     | Good    |

---

**Note:** Current value and quality data from InTouch tags running on a remote computer are not displayed.

---

The client mode and many other features are controlled by properties and methods.

## Script Methods

Alarm Client Controls are elements that can be configured with script methods. Action Scripts have triggers such as Click, Double-Click, or Key Equivalent. Use the Attribute Browser, the Element Browser tab, to select the element and the property required for the control.

---

**Note:** Script methods not available from the Attribute Browser, can be obtained in the Help file, which contains all of the additional properties, methods, and events for Alarm Control.

---

## Alarm Acknowledgment

Create an Alarm Control to require an alarm to be acknowledged, even if the condition causing the alarm has passed. This would make an operator aware of events that caused a temporary alarm state but have returned to normal. Configure the Action Script to include the alarm, its property, and who the alarm is acknowledged by.

## Switch Between Client Modes

To change the client mode at runtime, create an ArcestrA script to interact with the Alarm Control Client Mode property at design time. Configure the Action Script to include the alarm and Client Mode to equal one of the Client Mode Numbers indicated below:

| Client Mode Name             | Client Mode Number |
|------------------------------|--------------------|
| Current Alarms               | 1                  |
| Recent Alarms and Events     | 2                  |
| Historical Alarms            | 3                  |
| Historical Events            | 4                  |
| Historical Alarms and Events | 5                  |

# Lab 18 – Working with the Alarm Client

## Introduction

In this lab, you will be working with the ArchestrA Alarm Client to view alarms coming from the InTouch application.

## Objectives

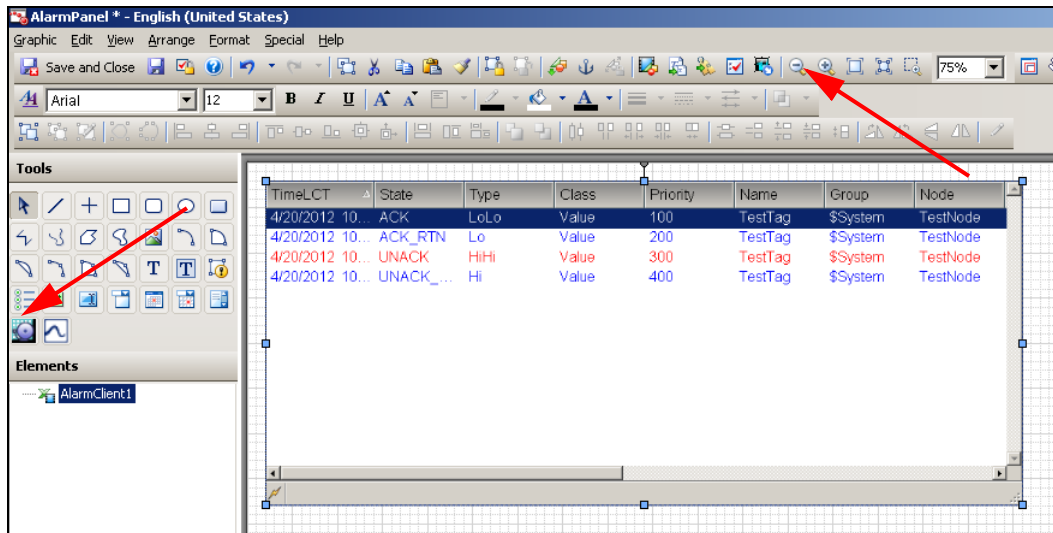
Upon completion of this lab, you will be able to:

- Implement the Alarm Client
- Write scripts to interact with the Alarm Client

## Work with the Alarm Client

1. In Archestra IDE, right-click **ClassSymbols** and select **New | Symbol**.
2. Name the symbol **AlarmPanel**, and then double-click to open it for editing.
3. In **Tools** pane, click **Alarm Client**, and then place it on the top left of the canvas and increase its width.

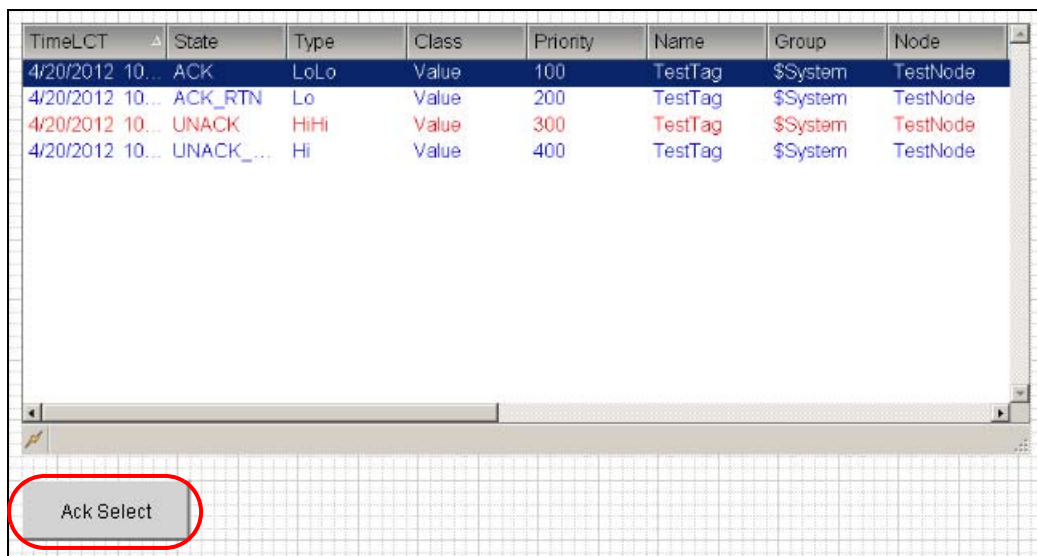
You may need to zoom out.



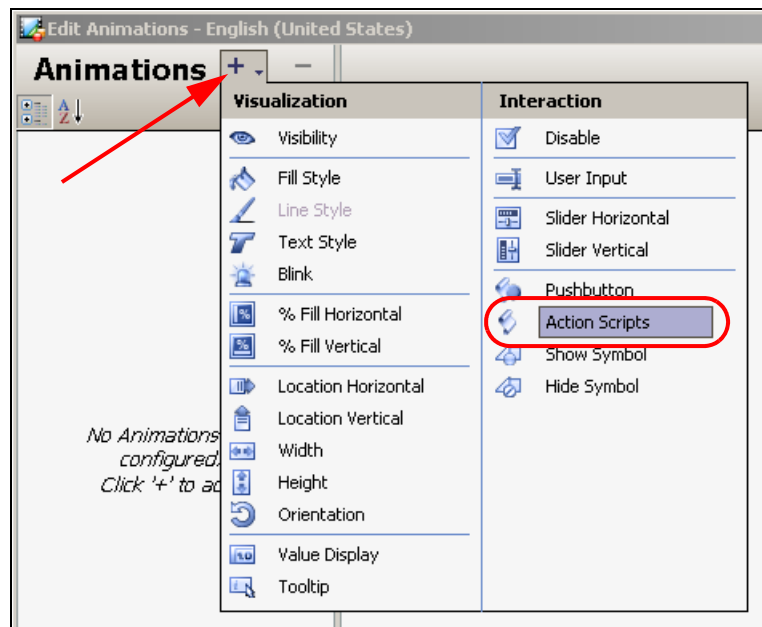
Next, you will create the buttons necessary to control the **Alarm Client**.

## Configure Ack Select

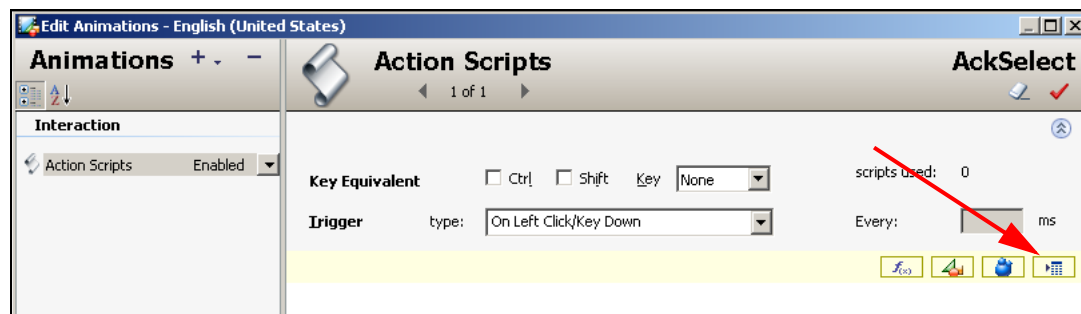
4. Below the **Alarm Client**, draw a button and label it **Ack Select**, in the right pane name it **AckSelect**, and then double-click to add animations.



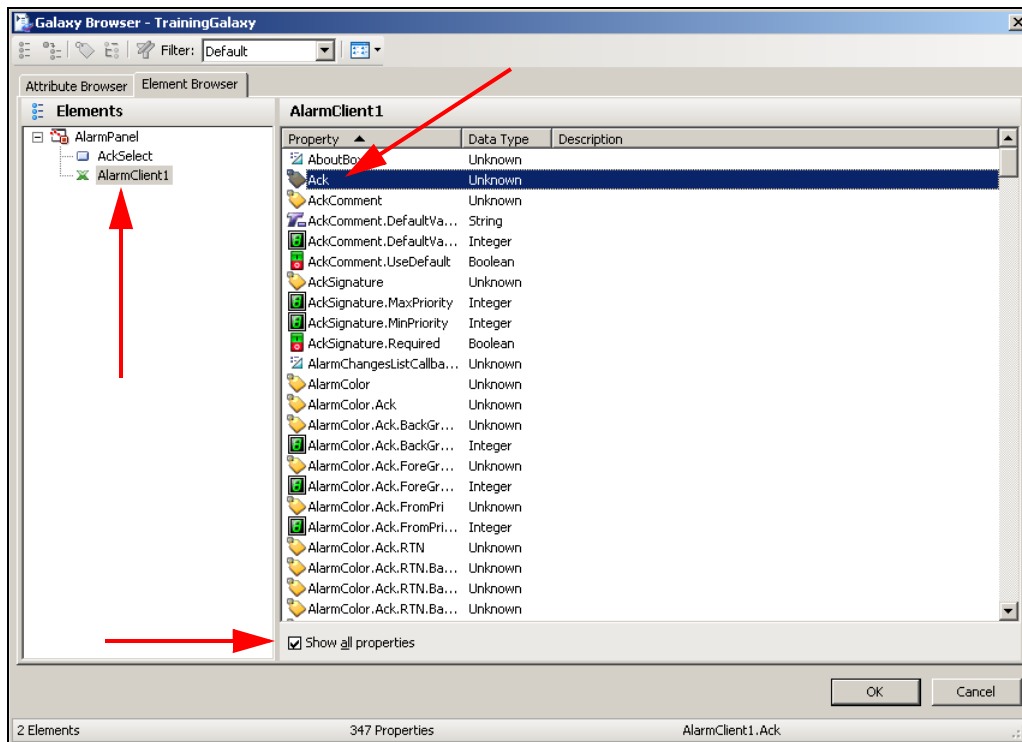
5. Click + and select **Action Scripts**.



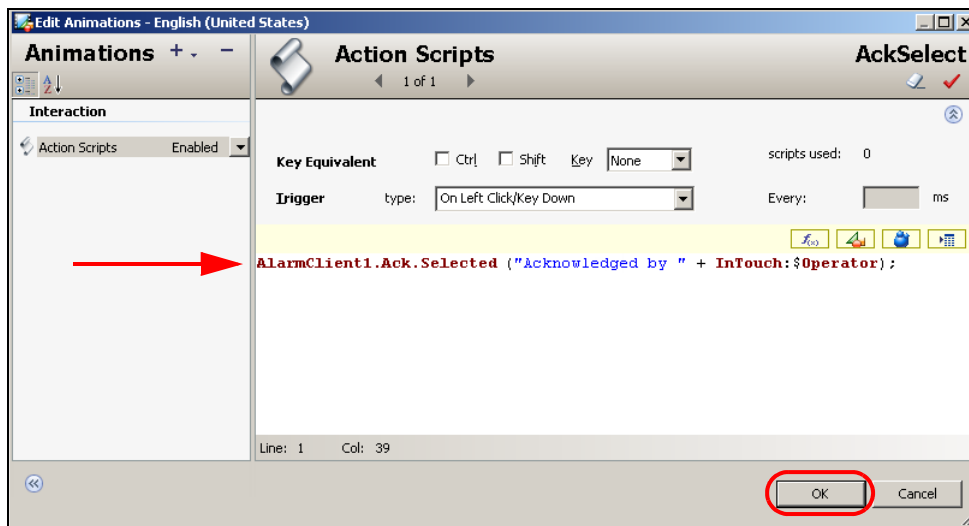
6. Click **Display Attribute Browser**.



7. In the **Element Browser** tab, select the **AlarmClient1** element, check **Show all Properties**, and then double-click **Ack**.



8. Configure the script so it looks as follows:  
**AlarmClient1.Ack.Selected ("Acknowledged by " + InTouch:\$Operator);**



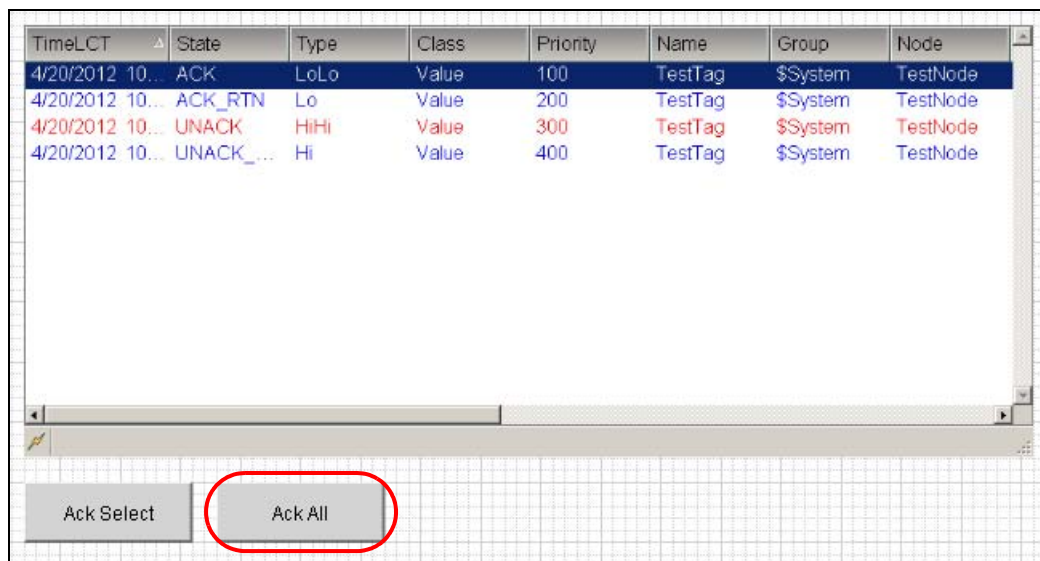
**Note:** As in the previous lab, make sure to put a space between the word **by** and the end quote.

9. Click **OK**.

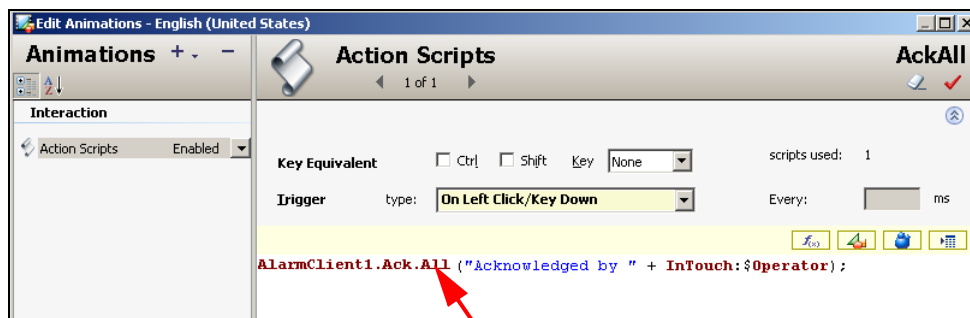


## Configure Ack All

10. Duplicate the **Ack Select** button, and then place the new button to right of the original.
11. Change the label to **Ack All** and the name to **AckAll**.

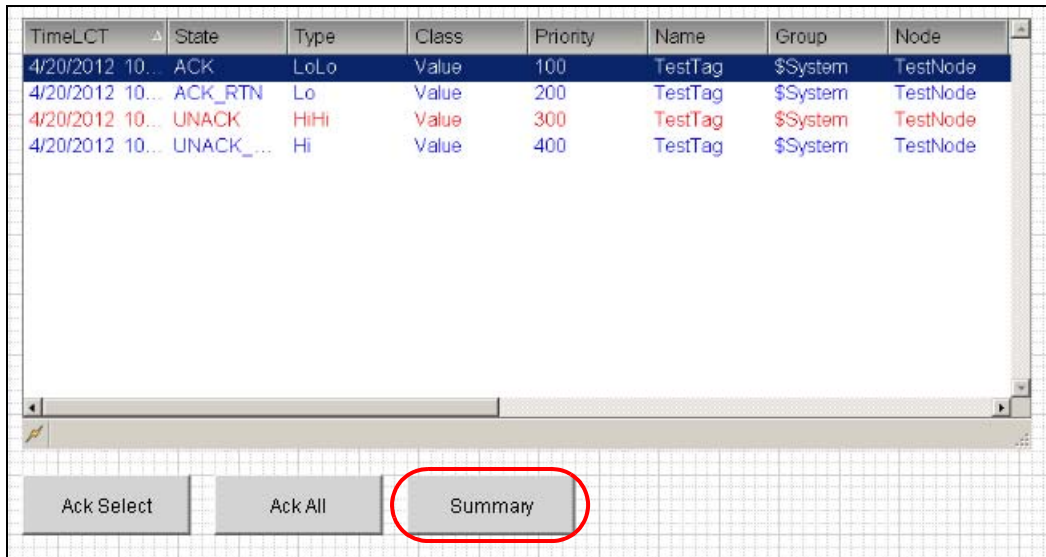


12. Double-click **AckAll** to open for editing.
13. In the script, change **Selected** to **All**, and then click **OK**.



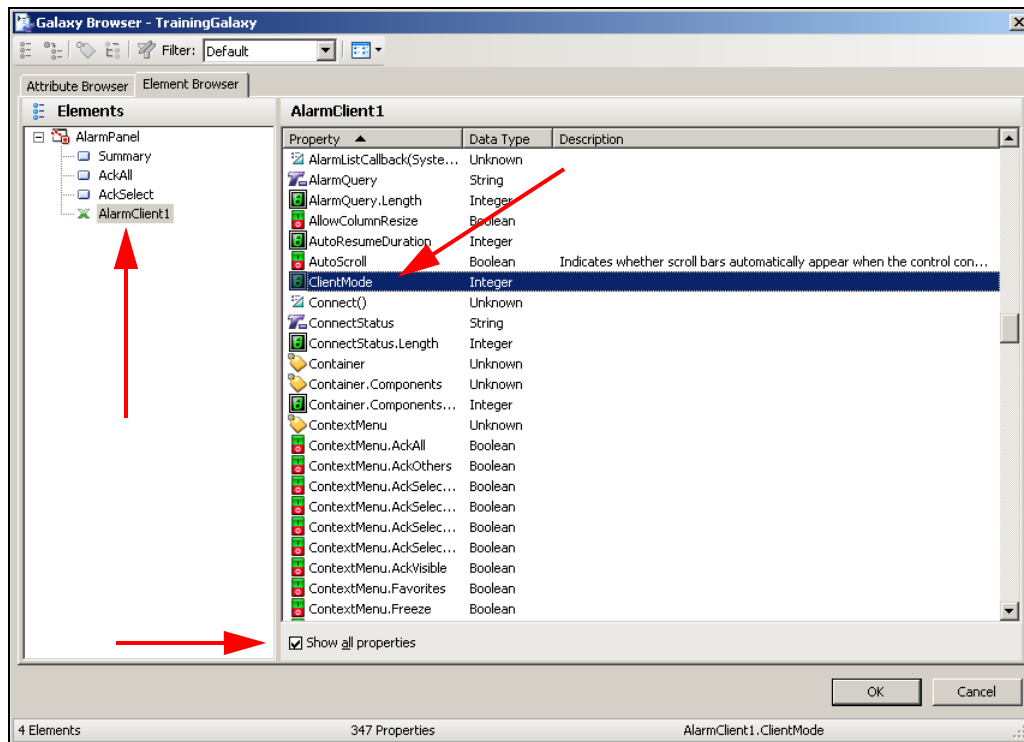
## Configure Summary

14. Duplicate the **Ack All** button, and then place the new button to right of the original.
15. Change the label and name to **Summary**.

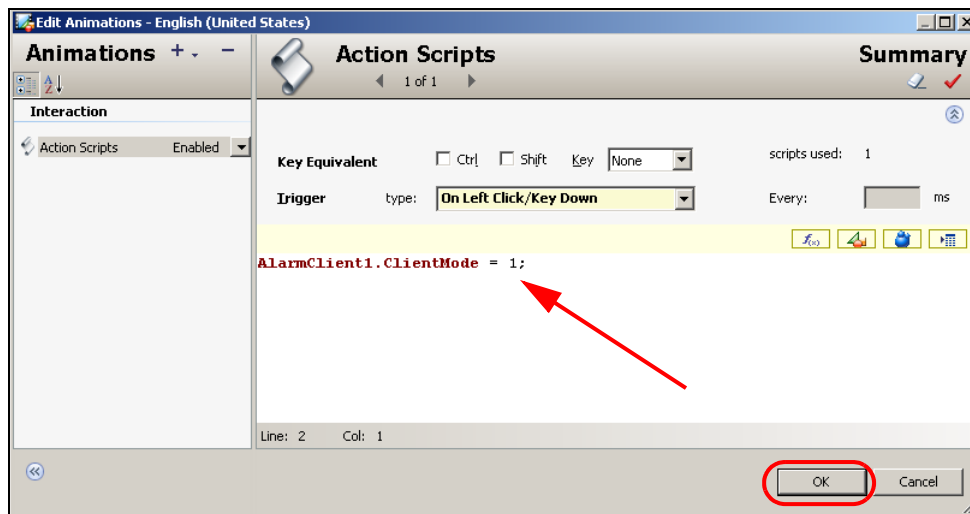


16. Double-click **Summary** to open for editing.
17. Remove the existing script.
18. Click **Display Attribute Browser**.

19. From the **Element Browser** tab, select the **AlarmClient1** element, check **Show all Properties**, and then double-click **ClientMode**.



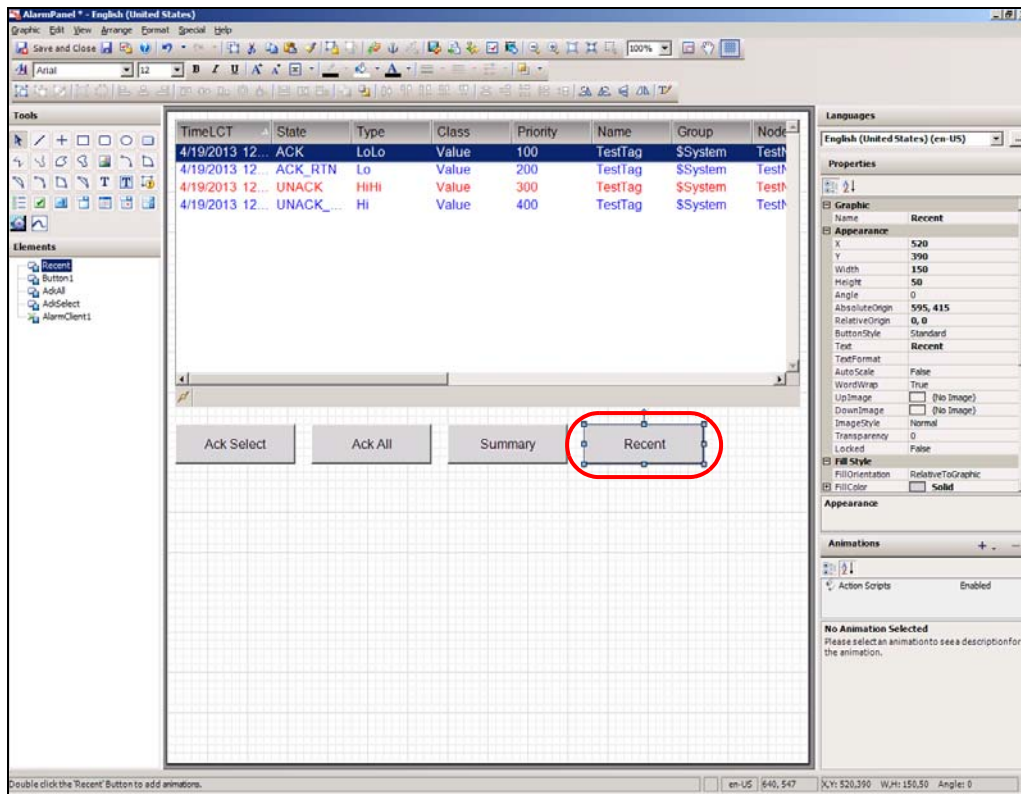
20. Configure the script so it looks as follows:  
**AlarmClient1.ClientMode = 1;**



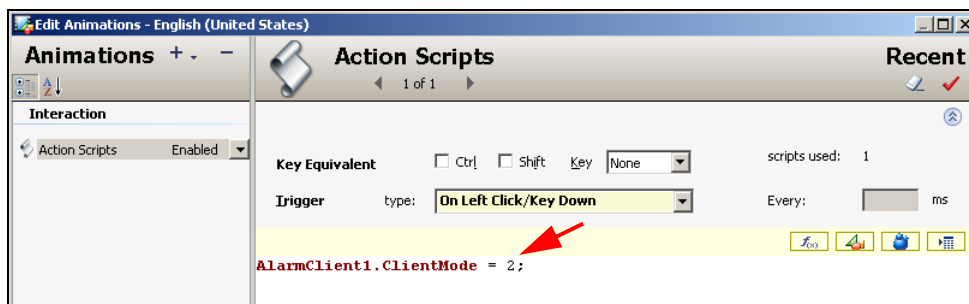
21. Click **OK**.

## Configure Recent

22. Duplicate the **Summary** button, and then place the new button to right of the original.
23. Change the label and name to **Recent**, and then double-click it for editing.



24. Configure the scripting so it looks as follows:  
**AlarmClient1.ClientMode = 2;**

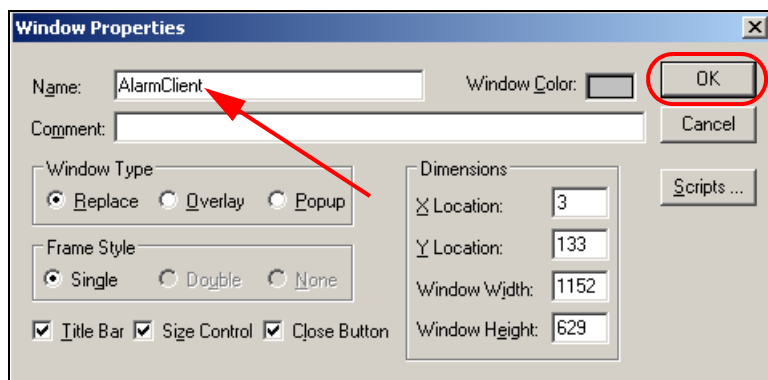


25. Click **OK**.
26. Click **Save and Close**, and then check in the symbol.

Now you will need to create a new window in WindowMaker.

27. In WindowMaker, on the **File** menu, click **New Window**.

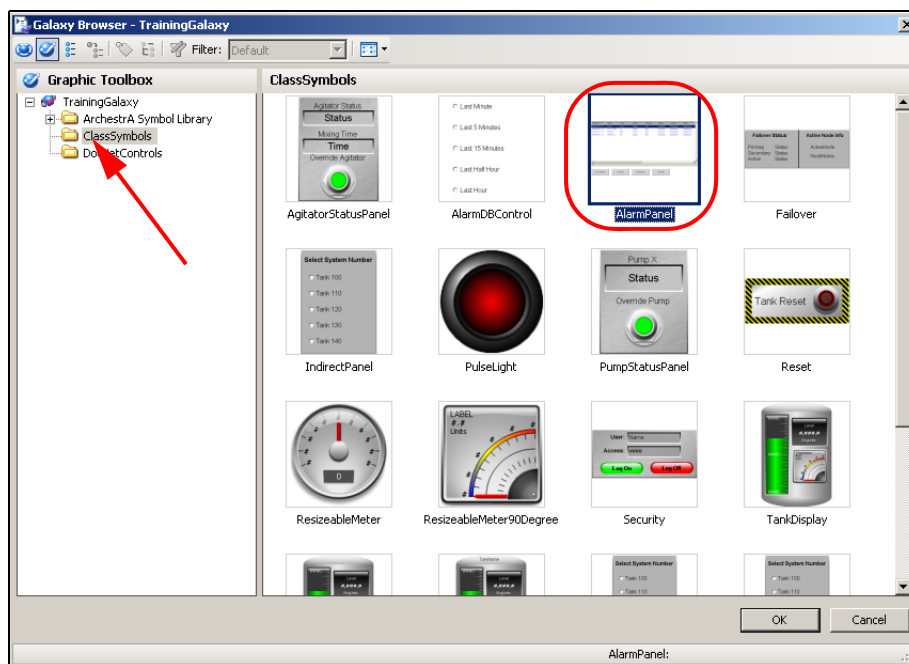
28. Name it **AlarmClient**, and then click **OK**.



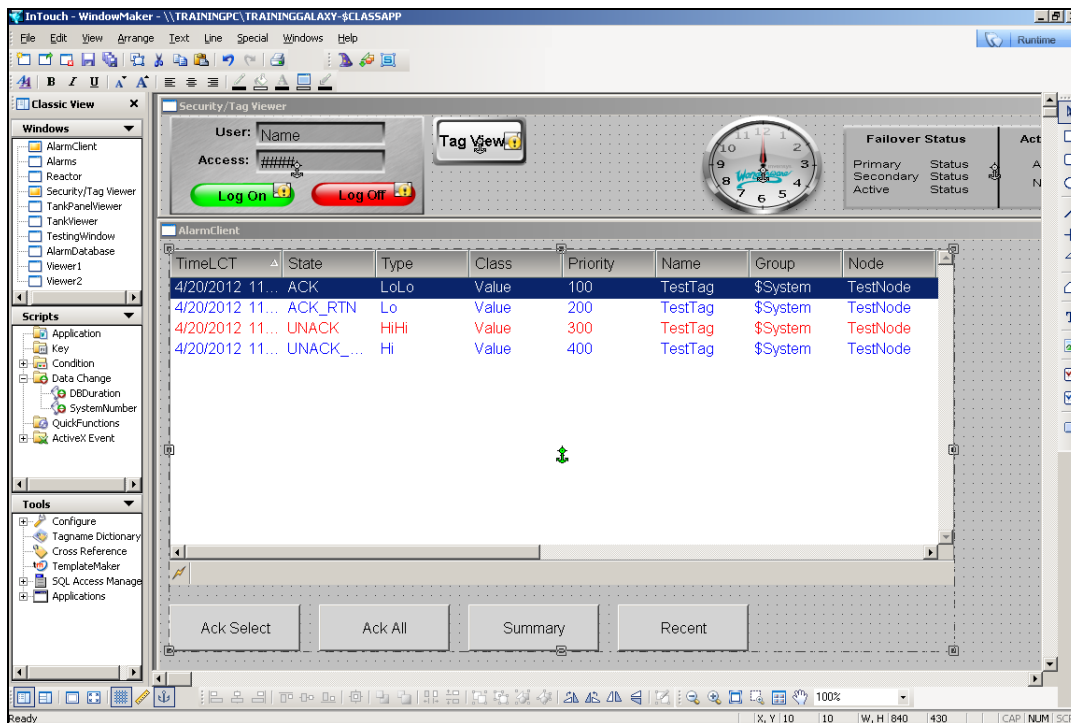
29. Close all other windows except for **AlarmClient** and **Security/Tag Viewer**.

30. Click **Embed ArchestrA Graphic**.

31. Select **ClassSymbols**, and then double-click **AlarmPanel**.



32. In the **AlarmClient** window, place **AlarmPanel**.



## Test What You Have Built

33. Click **Runtime**.
34. Click each of the buttons to see that they are functioning as desired.
35. Click **Development!**.

# Lab 19 – Creating Pop-Up Alarm Displays

## Introduction

In this lab, you will modify the InTouch application along with the associated Arcestra symbols to create context sensitive alarm displays. This will be done first by modifying a panel display from a previous lab that will allow four tanks to be displayed simultaneously, and then by modifying the single tank display from a previous lab.

## Objectives

Upon completion of this lab, you will be able to:

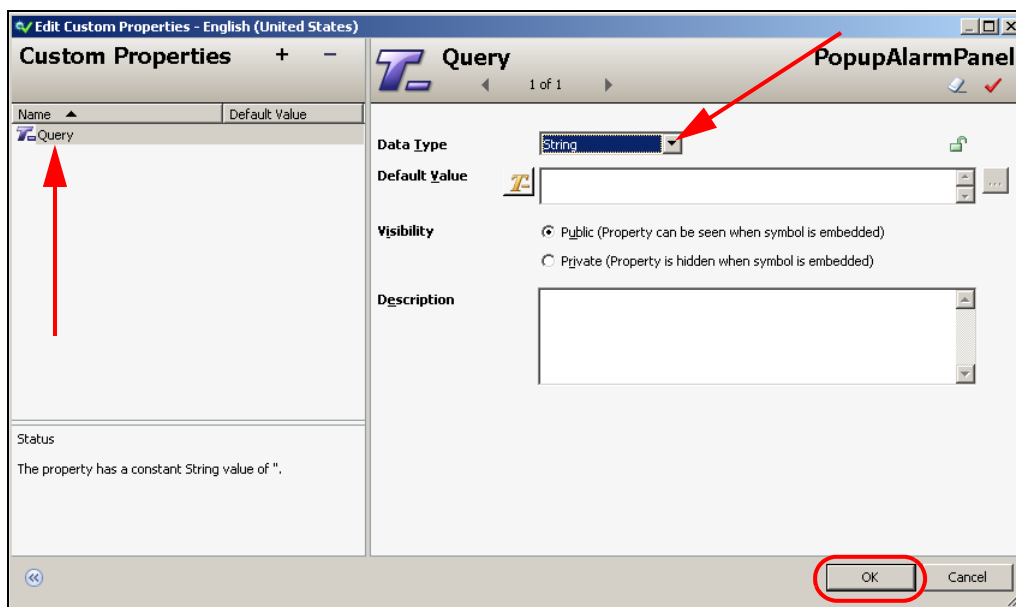
- Use the ShowGraphic scripting in conjunction with the Alarm Client
- Modify existing graphics for context sensitive alarming

## Use Pop-Up Displays for a Multiple Tank Display

This example will work with the graphics found in the TankPanelViewer. Similar to the previous example, there will be a pop-up display of alarm data, but in this portion of the lab multiple tanks will be displayed in a panel view.

### Modify the Alarm Client Display

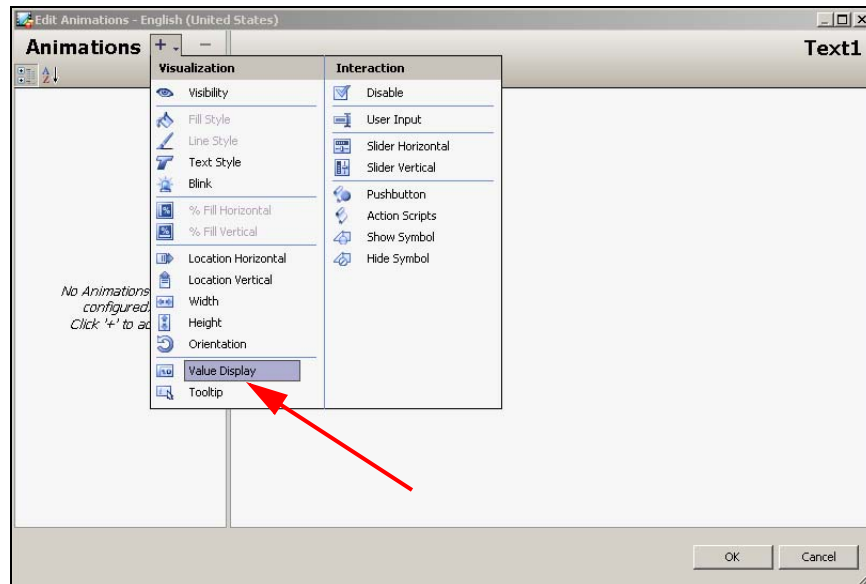
1. In ArchestrA IDE, right-click **AlarmPanel** and select **Duplicate**.
2. Name the symbol **PopupAlarmPanel**, and then double-click to open it for editing.
3. On the **Special** menu, click **Custom Properties**.
4. Click **+** and name the custom property **Query**, in the **Data Type** drop-down list select **String**, and then click **OK**.



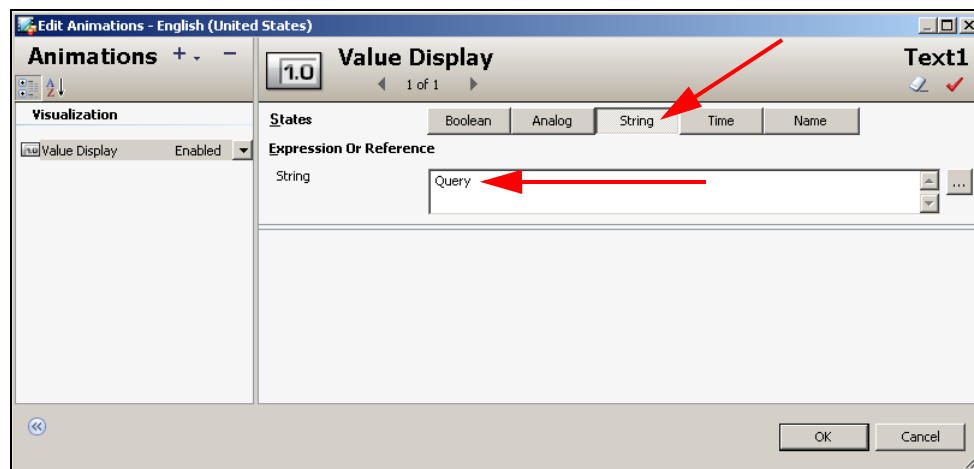
5. Using the **Text** tool, click below the **Ack Select** button, and enter **Query**.
6. Left-justify the text, and then in the right pane name it **QueryText**.



7. Double-click **QueryText** to open the **Edit Animations** dialog box.
8. Click **+** and select **Value Display**.

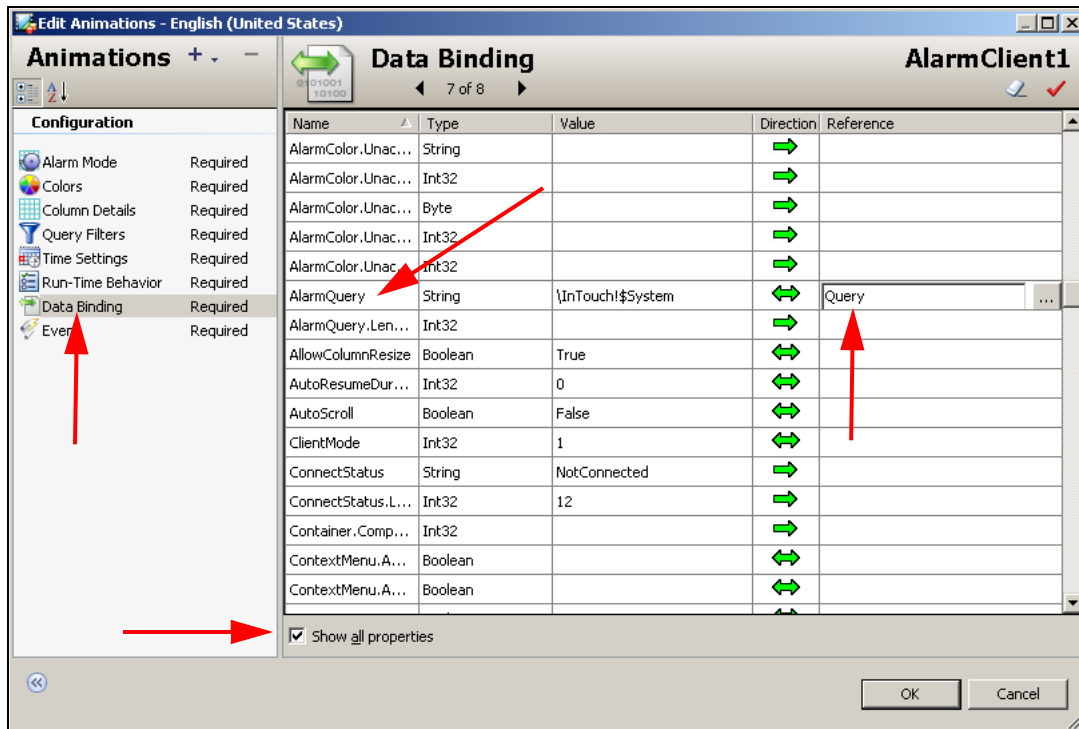


9. Click **String** and in the **String** field enter **Query**.



10. Click **OK**.

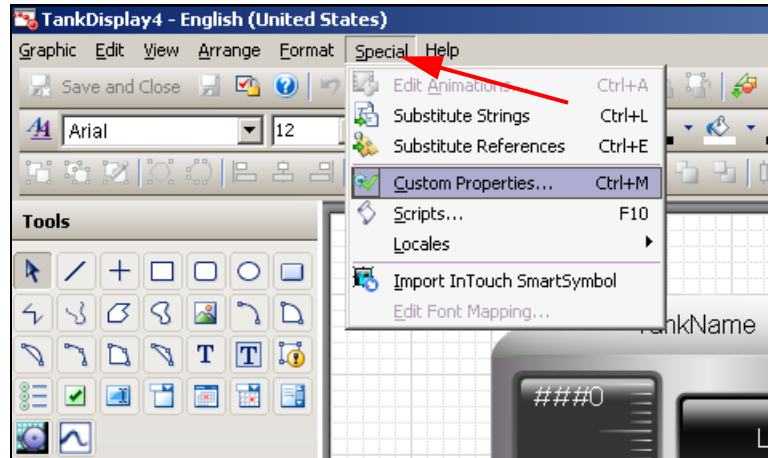
11. Double-click **AlarmClient1** and select **Data Binding**.
12. Check **Show all properties**, and in the **Name** column select **AlarmQuery**, click in the **Reference** column and enter **Query**.



13. Click **OK**.
14. Click **Save and Close**, and then check in the symbol.

### Modify the TankDisplay

15. In Archestra IDE, right-click **TankDisplay3** and select **Duplicate**.
16. Name the symbol **TankDisplay4**, and then double-click to open it for editing.
17. On the **Special** menu, click **Custom Properties**.



18. Create the following custom properties:

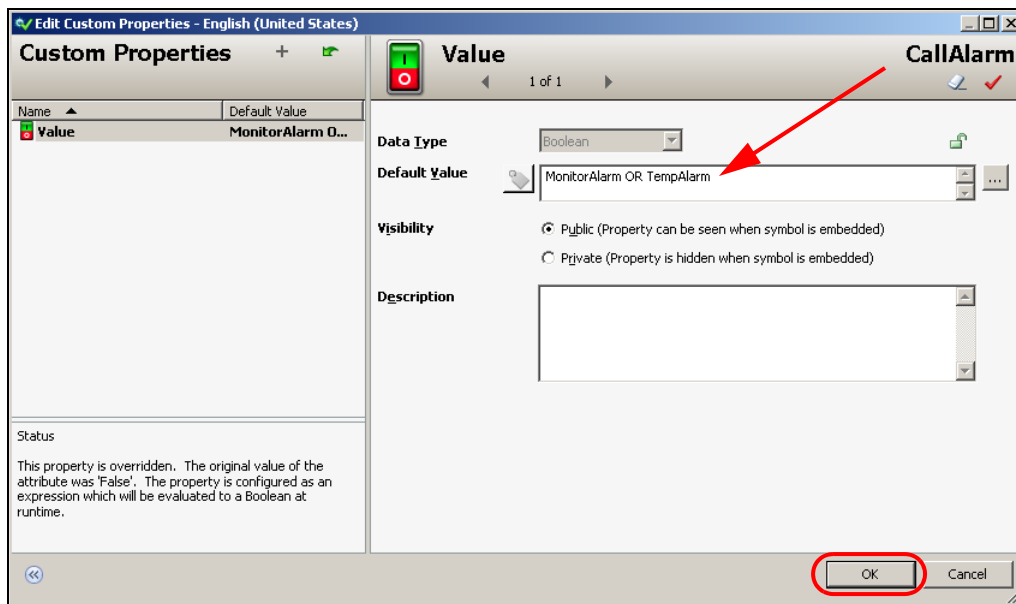
| Name         | Data Type |
|--------------|-----------|
| MonitorAlarm | Boolean   |
| TempAlarm    | Boolean   |

19. Click **OK** to close the **Edit Custom Properties** dialog box.
20. Click **Embed Archestra Graphic**.
21. Click **ClassSymbols**, and then double-click **PulseLight**.
22. Place **PulseLight** at the bottom of the tank, resize and name it **CallAlarm**.

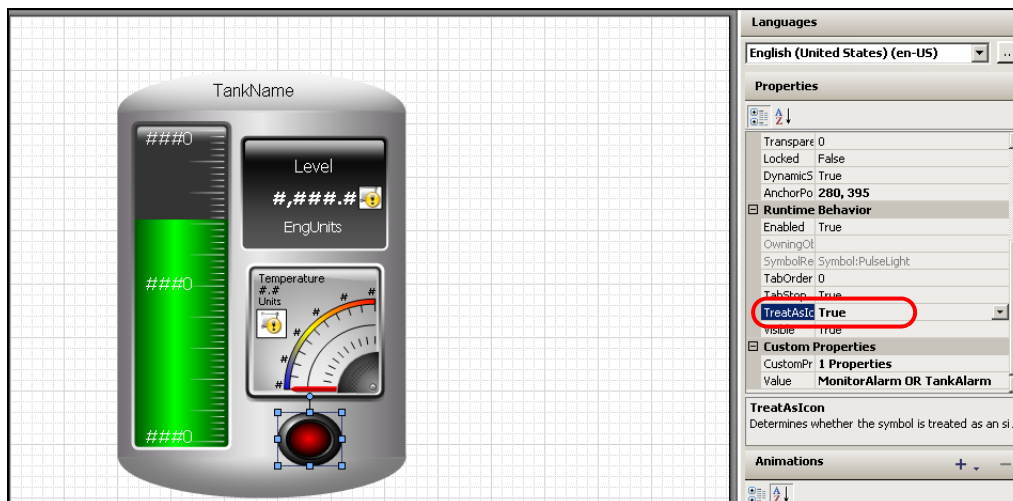


23. Right-click **CallAlarm** and select **Custom Properties**.
24. Select the **Value** custom property, and then in the **Default Value** field enter:

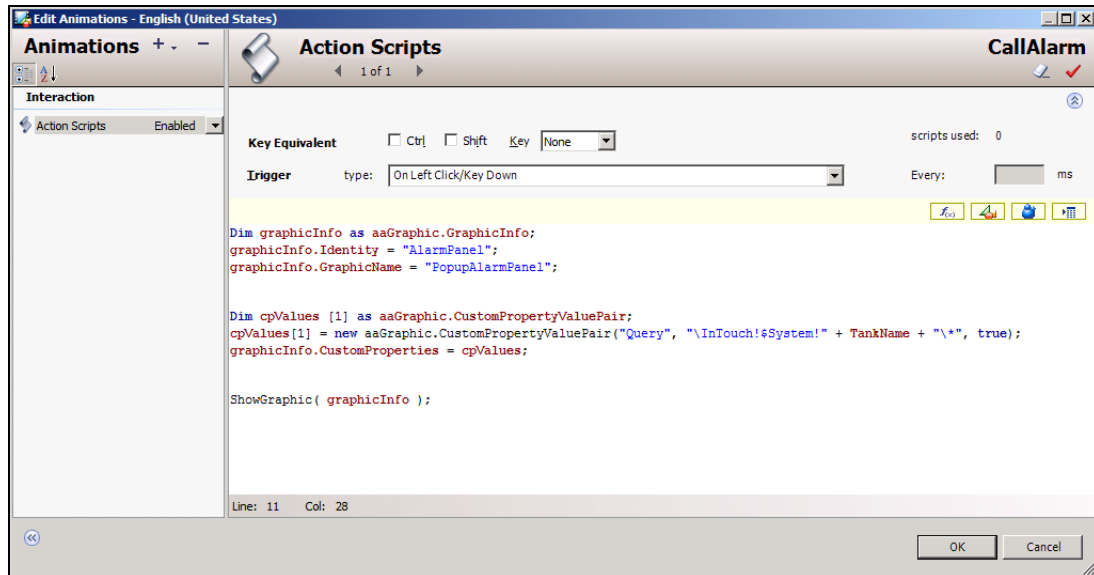
**MonitorAlarm OR TempAlarm**



25. Click **OK**.
26. Select **CallAlarm**, and then set the **TreatAsIcon** value to **True**.



27. Double-click **CallAlarm**.
28. Click **+** and select **ActionScripts**.
29. In your **Training** folder, open the **Lab 19 Scripts** file and copy the section between **Script for Calling the PopupAlarmPanel** and **Modified Script for Calling Popup Tank Alarms**.
30. In the script area for the **Action Scripts** animation, paste the script.



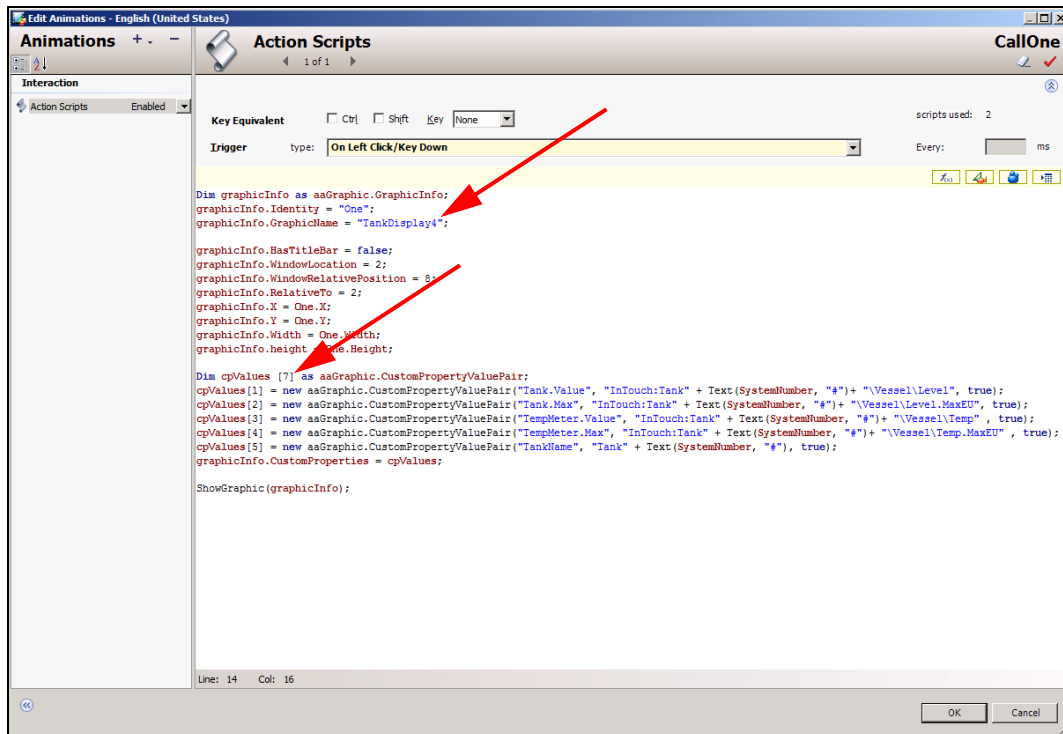
31. Click **OK**.
32. Click **Save and Close**, and then check in the symbol.

**Modify the TankPanel**

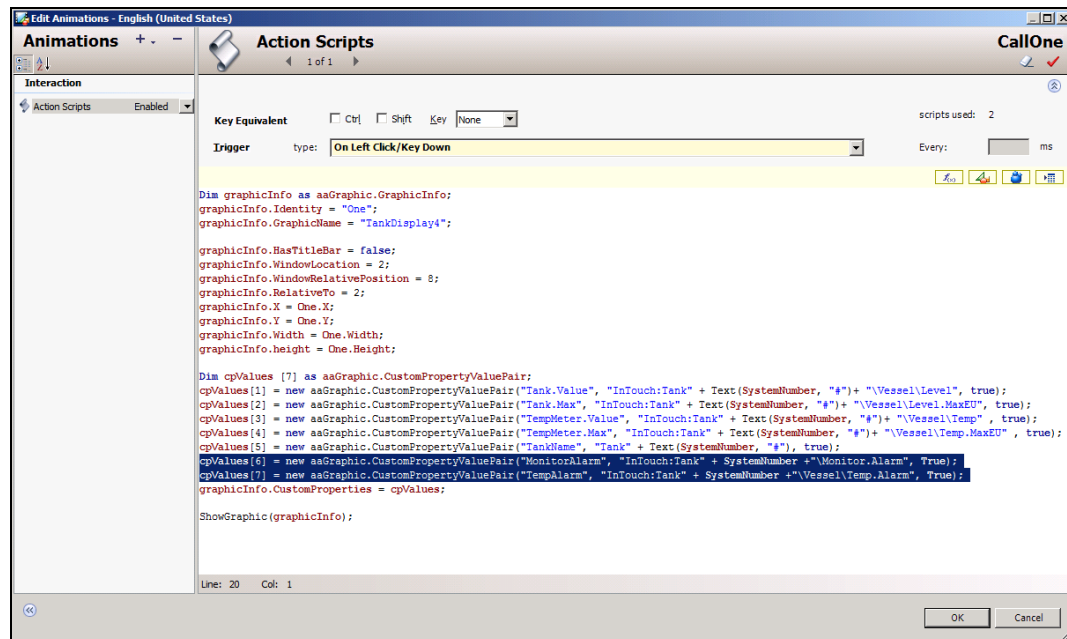
33. In Archestra IDE, right-click **TankPanel3** and select **Duplicate**.
34. Name the symbol **TankPanel4**, and then double-click to open it for editing.
35. Double-click button **One** to open it for editing, and then make the following modification to the script:

**Line 3:** Change **TankDisplay3** to **TankDisplay4**

**Line 14:** Change the **cpValues** from 5 to 7



36. From the **Lab19 Scripts** file, under the **Modified Script for Calling Popup Tank Alarms** section, copy **cpValues** [6] and [7].
37. In the script area for the **Action Scripts** animation, paste the script between **cpValues**[5] and **graphicInfo.CustomProperties = cpValues;**



38. Click **OK**.
39. Make the same script changes to buttons **Two**, **Three**, **Four**, and **Main**, remembering to:
- Change the reference from **TankDisplay3** to **TankDisplay4**
  - Increase the number of **cpValues** from **5** to **7**
  - Add the two additional lines of script

---

**Note:** Do not copy and paste the scripting from button **One** to the remaining buttons, you will lose the specific button references built into each button from the previous lab.

---

40. **Save and Close**, and then check in the symbol.
41. In WindowMaker, delete the contents of the **TankPanelViewer** window, and then using **Embed Archestra Graphics** embed **TankPanel4**.

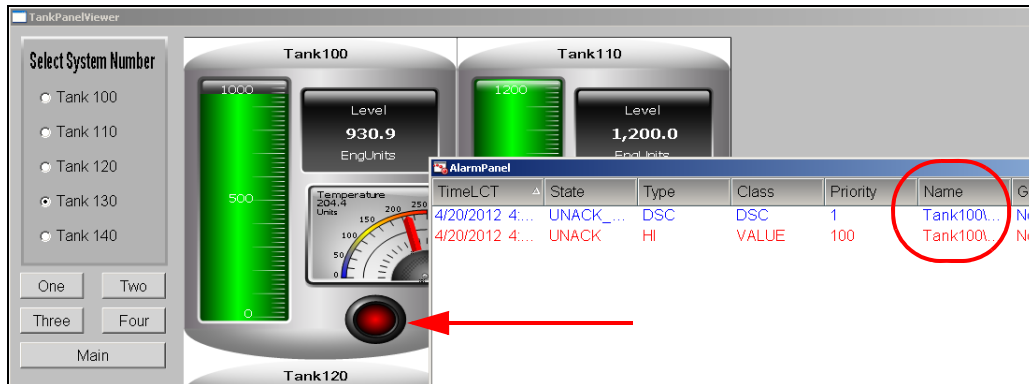
---

**Note:** You may need to switch to full screen mode to size it properly.

---

## Test What You Have Built

42. Click **Runtime**.
43. Place **Tank100** at position **One**, **Tank110** at position **Two**, **Tank120** at position **Three**, and **Tank130** at position **Four**.
44. Click on any of the **Pulsing Lights** and the **AlarmPanel** should display showing the alarms for the selected tank.



45. Repeat with a different **Pulsating Light** and see that the **AlarmPanel** displays showing the alarms for that specific tank.
46. Close **AlarmPanel**.
47. Click **Development!**.

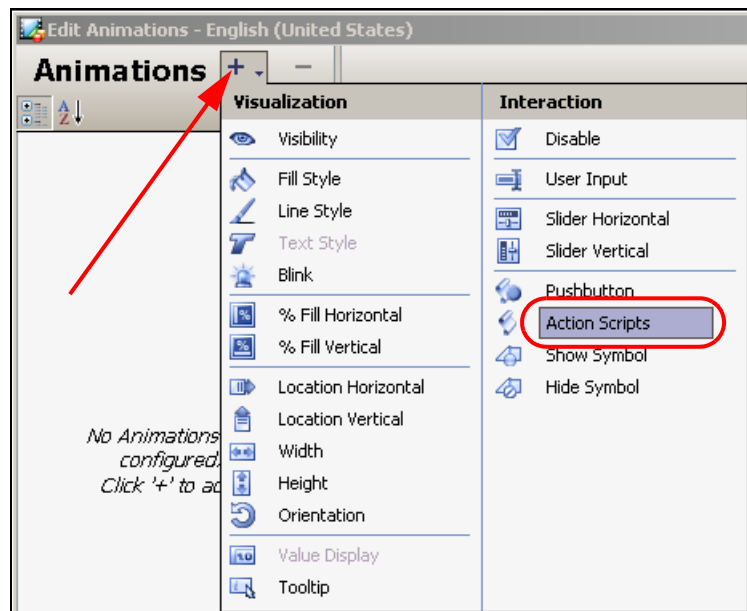


## Use Pop-Up Displays for a Single Tank Display

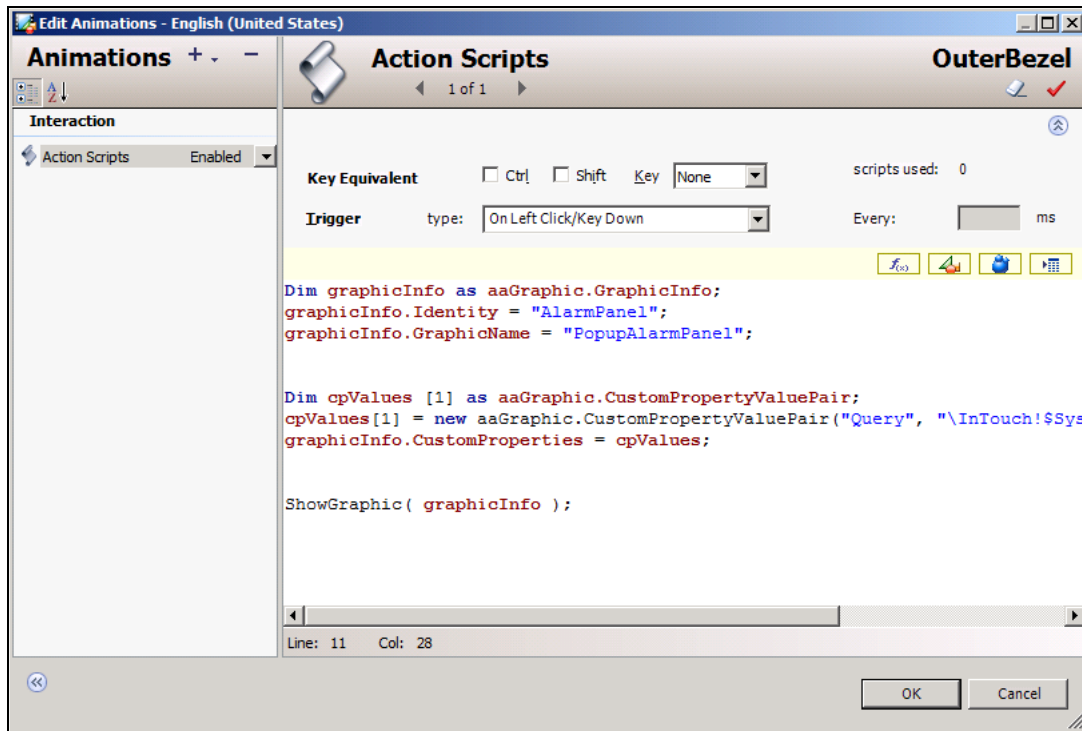
Now, you create a pop-up display for triggered alarms by making changes to the AlarmDisplay and PulseLight symbols.

### Modify PulseLight

48. In ArchestrA IDE, right-click **PulseLight** and select **Duplicate**.
49. Name the symbol **PulseLight\_CallAlarm**, and then double-click to open it for editing.
50. On the **Special** menu, click **Custom Properties**.
51. Click **+** and name the custom property **TankName**, in the **Data Type** drop-down list select **String**, and then click **OK**.
52. In the **Elements** pane, double-click **OuterBezel** to add animation.
53. Click **+** and select **Action Scripts**.



54. In your **Training** folder, open the **Lab 19 Scripts** file, and copy the section between **Script for Calling the PopupAlarmPanel** and **Modified Script for Calling Popup Tank Alarms**.
55. In the script area for the **Action Scripts** animation, paste the script.



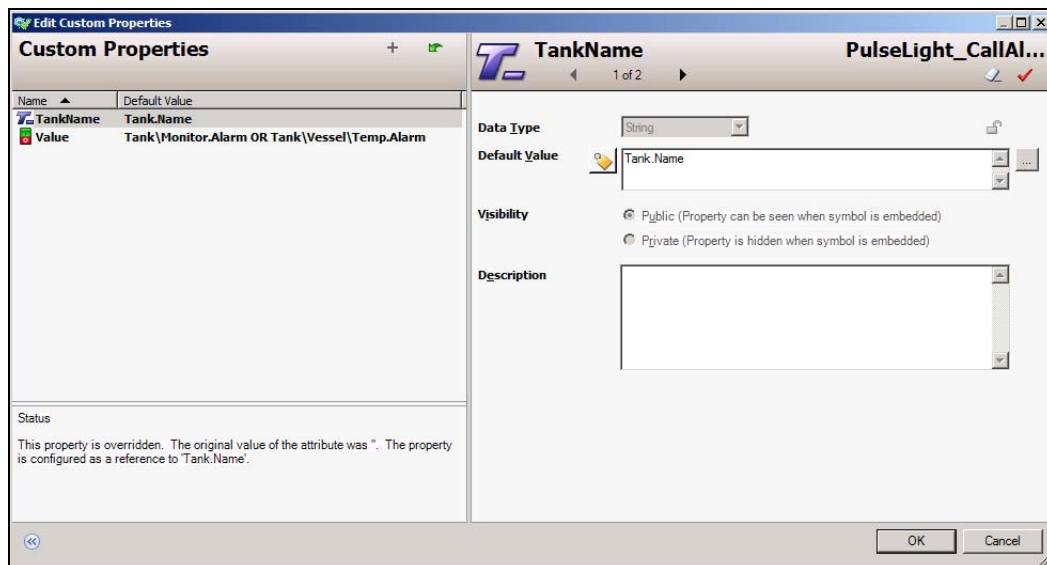
56. Click **OK**.
57. Click **Save and Close**, and check in the symbol.

Now you will add this symbol to the Reactor window in WindowMaker.

58. In WindowMaker, open the **Reactor** and **Security/Tag Viewer** windows, and then close all others.
59. Click **Embed Archestra Graphic**.
60. Select **ClassSymbols**, and then double-click **PulseLight\_CallAlarm**.
61. In the **Reactor** window, place it on top of your **Tank** symbol.  
You may need to shrink it to a more appropriate size to make it fit.



62. Double-click **PulseLight\_CallAlarm** to open the **Edit Custom Properties** dialog box.
63. Select **Value**, and then in the **Default Value** field enter the following:  
**Tank\Monitor.Alarm OR Tank\Vessel\Temp.Alarm**
64. Select **TankName**, switch to **Reference** mode , and then in the **Default Value** field enter  
**Tank.Name**.



65. Click **OK**.

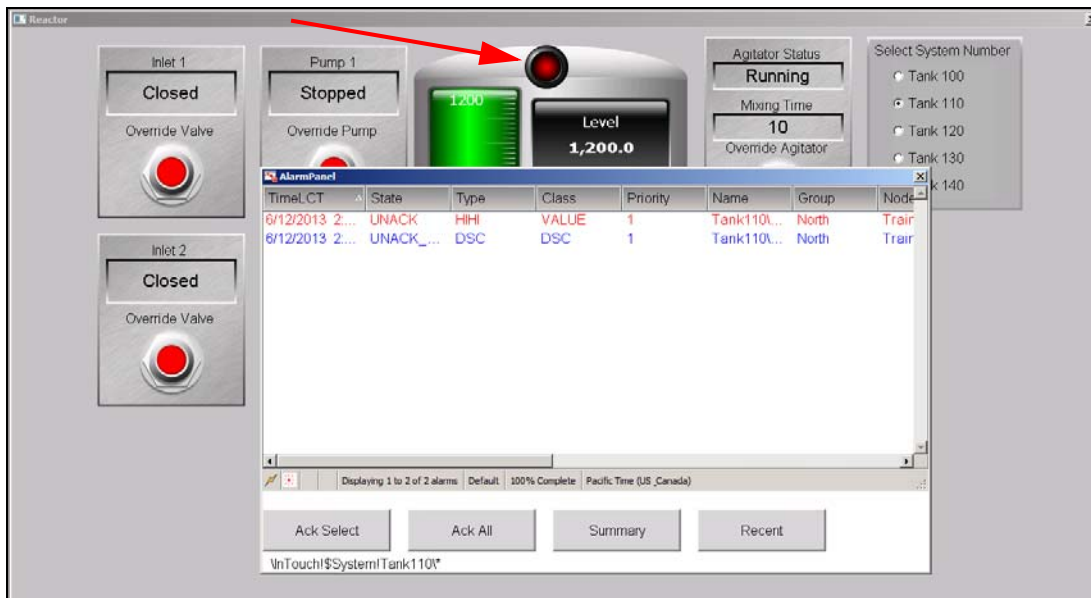
## Test What You Have Built

66. Click **Runtime**.

67. Select each of the tanks until you find one with an alarm being triggered.



68. Click on the pulsing light to display the **PopupAlarmPanel**, which will list the alarms for the selected tanks.



69. Close the **PopupAlarmPanel**.

70. Click **Development!**.



## Module 7 – Tag Servers

---

|                                                |             |
|------------------------------------------------|-------------|
| <b>Section 1 – Tag Server and Tag Provider</b> | <b>7-3</b>  |
| <b>Lab 20 – Working with a Tag Server</b>      | <b>7-7</b>  |
| <b>Section 2 – Alarm Providers</b>             | <b>7-17</b> |
| <b>Lab 21 – Distributed Alarming</b>           | <b>7-19</b> |
| <b>Section 3 – Access Tag Server History</b>   | <b>7-29</b> |
| <b>Lab 22 – Distributed History</b>            | <b>7-31</b> |

**Module Objectives**

- Receive and acknowledge alarms from a remote provider
- Retrieve historical data from a remote InTouch application
- Connect to a remote InTouch application as a tag source

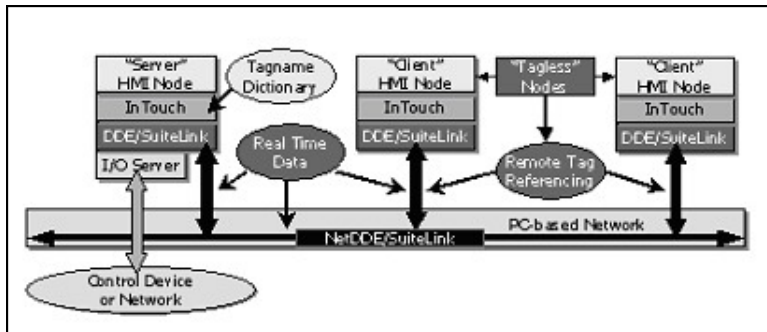
## Section 1 – Tag Server and Tag Provider

This section discusses how InTouch is used to configure the local application to browse the tag server's tagname dictionary and retrieve data from the tag server application.

### Remote Tag Referencing

Remote Tagname Referencing allows the InTouch developer to create true client/server MMI applications. With Remote Tagname Referencing, a single copy of the Tagname Dictionary may be stored on a server node and referenced across the network by any number of InTouch client nodes. It is unnecessary for client nodes to maintain copies of the Tagname Dictionary on their local hard drives. Remote Tagname Referencing allows client nodes to view and change all applicable data over the network.

The following diagram illustrates a three-node InTouch application; however, the two client nodes are "tagless". They do not have corresponding entries in their local Tagname Dictionaries for the data being obtained from the server. Remote referencing, in this case, reduces the number of configuration databases to be maintained from three to one.



### Remote Tagname Referencing Usage

InTouch allows true client-server architecture for factory automation applications. Client applications can be designed without using any tagnames in the local Tagname Dictionary with Remote Tagname Referencing.

For example:

In a window or QuickScript, use the syntax of `AccessName:"item"` to reference a remote tagname. It is not necessary to define the remote tagname in the local Tagname Dictionary.

Remote references can also access data from any I/O data source, such as a Wonderware I/O Server or Microsoft Excel. When using remote tagname references and importing a window or QuickScript, just convert the placeholder tagnames to remote tagname references. The tagnames do not need to be defined in the local Tagname Dictionary.

Whenever a remote reference (`accessname:"item"`) is used, InTouch validates the specified Access Name. If it determines that the Access Name is not defined, a prompt to define it appears. The Access Name is also validated when the remote tagname is activated. If errors are encountered, they will be written to the SMC Log Viewer. An Access Name that is used by a remote reference can be deleted as long as it is not being used by a local tagname.

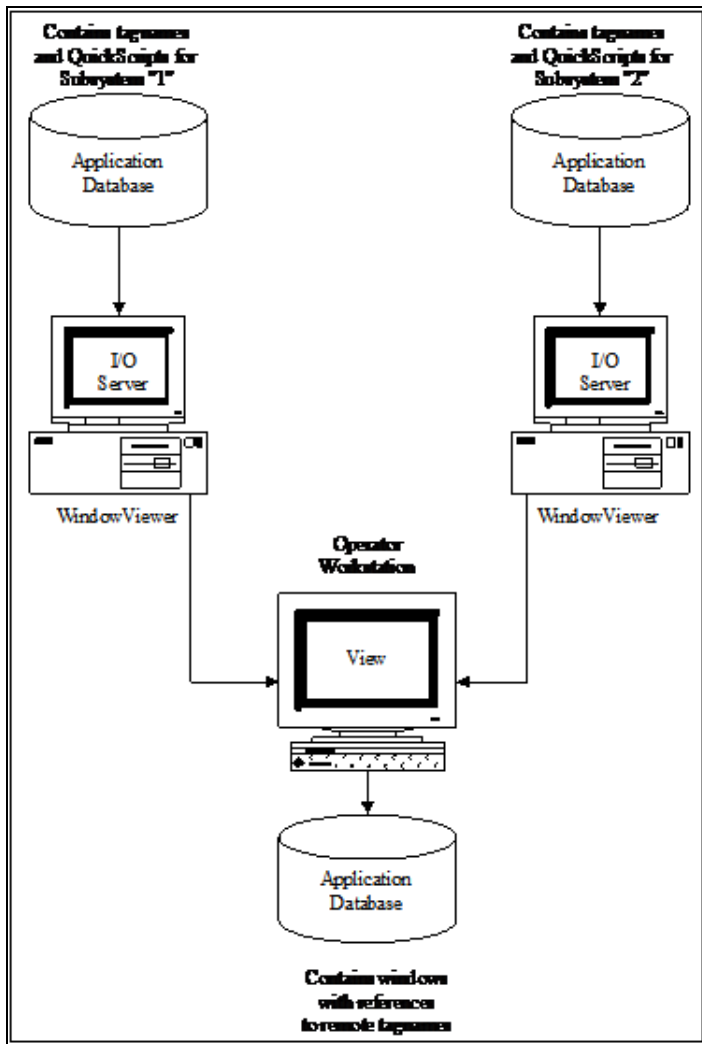
## Tag Sources Definition

Tag sources are defined for viewing in the Tag Browser. When a tag source definition is added or edited, information such as the local network Access Name to associate with the tag source's tagnames, a user-defined application name, and the data source for the tag source need to be entered.

## Tag Server Application Creation

By creating an application that contains only InTouch QuickScripts and tagnames, an instance of WindowViewer that functions as a tagname server can be established. Another application containing only windows and memory tagnames for window logic processing can be created. If these windows contain only remote tagname references, this application can serve as a repository for all the process windows for a facility. In this case, the remote tagname references are to tagnames in other WindowViewer instances that function as tagname servers. An instance of WindowViewer that connects to this database functions as an operator workstation.

This WindowViewer instance can open any window and view data from anywhere on the plant floor. For example:





Remote references are valid wherever an InTouch tag can be used

- InTouch Animation Links
- Arcestra Symbols
- Wizards
- Data Change Scripts: "Tagname[.field]"
- ActiveX events, properties, and methods
- All InTouch QuickScript types

Remote references are not valid for Historical Trend objects:

- Implementation of remote tagname references does not require conversion of applications that were created with earlier versions of InTouch (not supporting this feature). However, once implemented, the applications will not be backwards compatible with the earlier versions
- WindowViewer supports 32767 references to local tagnames and x references to active remote references, where  $x = 61,402$  minus the number of tagnames defined in the local Tagname Dictionary
- Remote references can be a maximum of 95 characters
- The **IOSetAccessName** function is supported for remote references and works the same as it does for local tagnames



# Lab 20 – Working with a Tag Server

## Introduction

In this lab, you will use a remote InTouch application as a tag server for your application. This is done by connecting to the remote application and using its tags as the data source for the animations and symbols in the local application.

## Objectives

Upon completion of this lab, you will be able to:

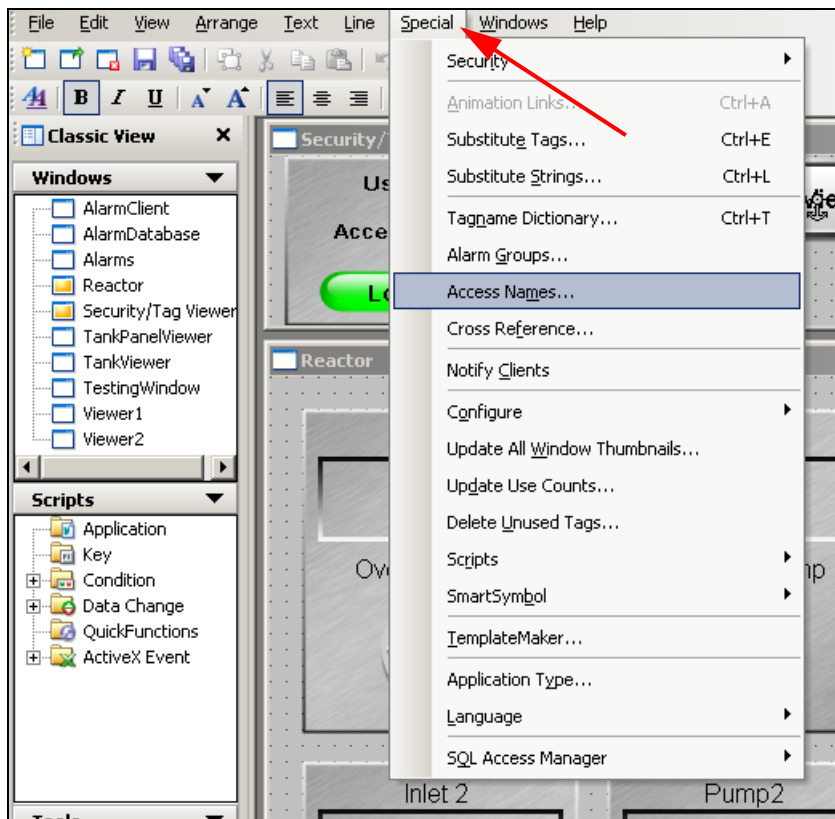
- Connect to the remote application
- Configure the tag provider
- Link to the tag server tags
- Build an interface to the tag server application

## Work with the Tag Server

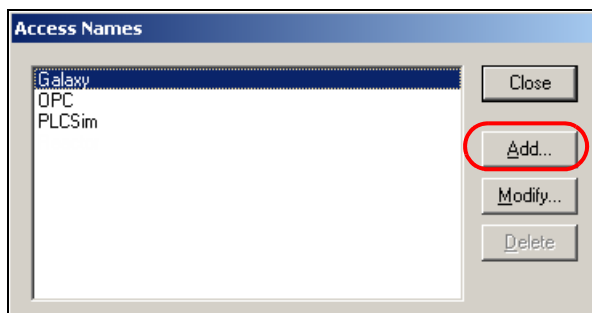
The first thing that must be done to have the remote application work as a tag server for your local application, is to share its application directory. This has already been done by your instructor who will provide you the proper node information.

### Define the Data Source

1. In WindowMaker, on the **Special** menu, click **Access Names**.



2. Click **Add**.



3. Configure as follows:

**Access:** Reactor  
**Node Name:** provided by your instructor  
**Application Name:** View  
**Topic Name:** Tagname

The screenshot shows the 'Add Access Name' dialog box with the following configuration:

- Access:** Reactor
- Node Name:** Trainingpc-2
- Application Name:** View
- Topic Name:** Tagname
- Which protocol to use:** SuiteLink (selected)
- When to advise server:** Advise only active items (selected)
- Enable Secondary Source:** unchecked

The 'OK' button is circled in red, indicating it should be clicked next.

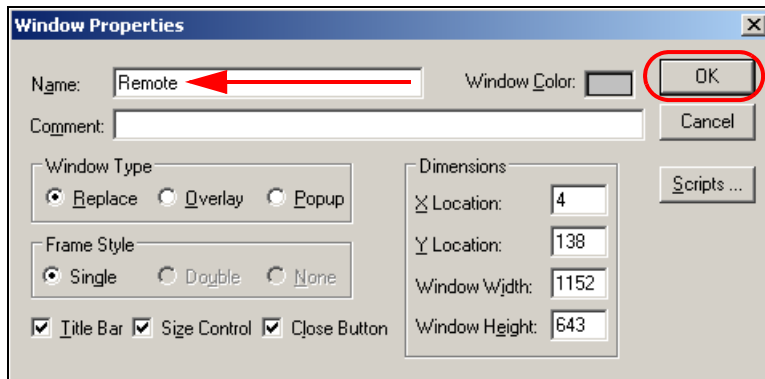
4. Click **OK**.

5. Click **Close**.

## Create a New InTouch Window

Create a new InTouch window using the same dimensions as your Reactor window.

- On the **File** menu, click **New Window**.
- Name the window **Remote**, and then click **OK**.

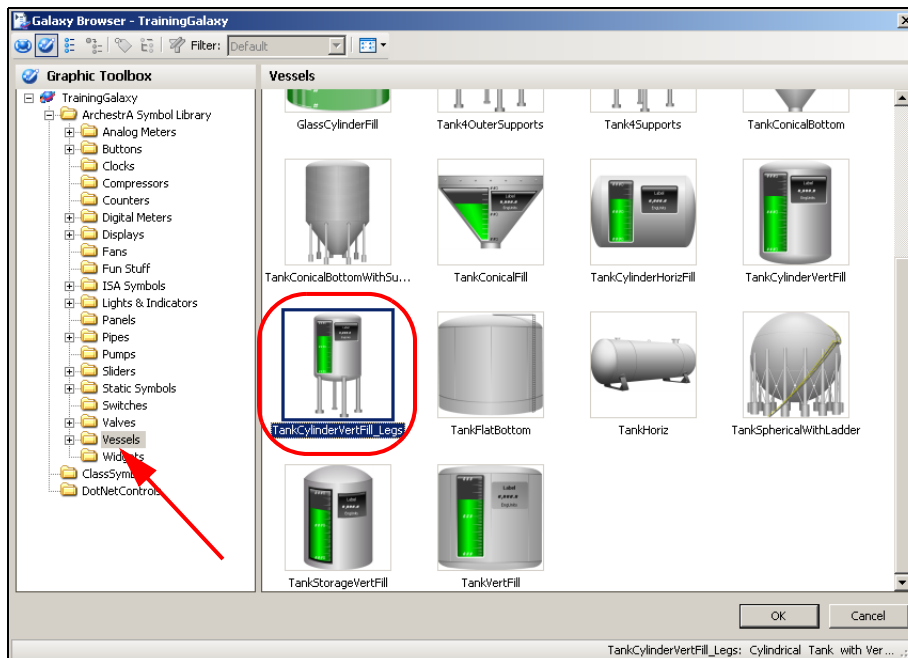


**Note:** Your window dimensions may vary based upon your resolution.

- Close all windows except for **Remote** and **Security/Tag Viewer**.

## Access Data

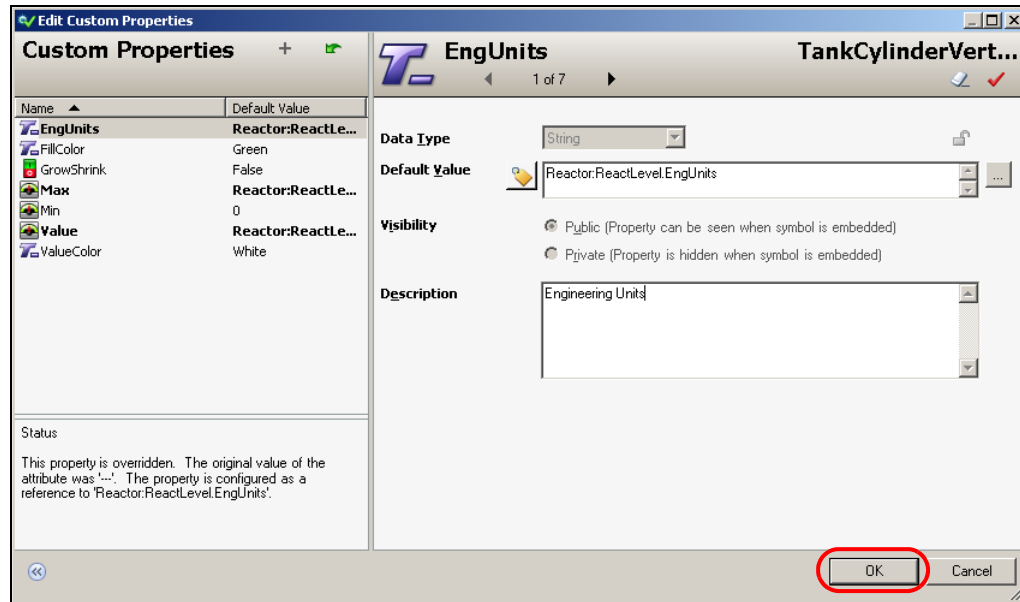
- Click **Embed ArchestrA Graphic**.
- Select **ArchestrA Symbol Library/Vessels**, and then double-click **TankCylinderVertFill\_Legs**.



11. In the **Remote** window, place the symbol.
12. Double-click on the symbol to open it for editing, and then configure the custom properties as follows:

| Custom Property | New Default Value           |
|-----------------|-----------------------------|
| Value           | Reactor:ReactLevel          |
| Max             | Reactor:ReactLevel.MaxEU    |
| EngUnits        | Reactor:ReactLevel.EngUnits |

13. Click **OK**.



14. Change the label to **Level**.

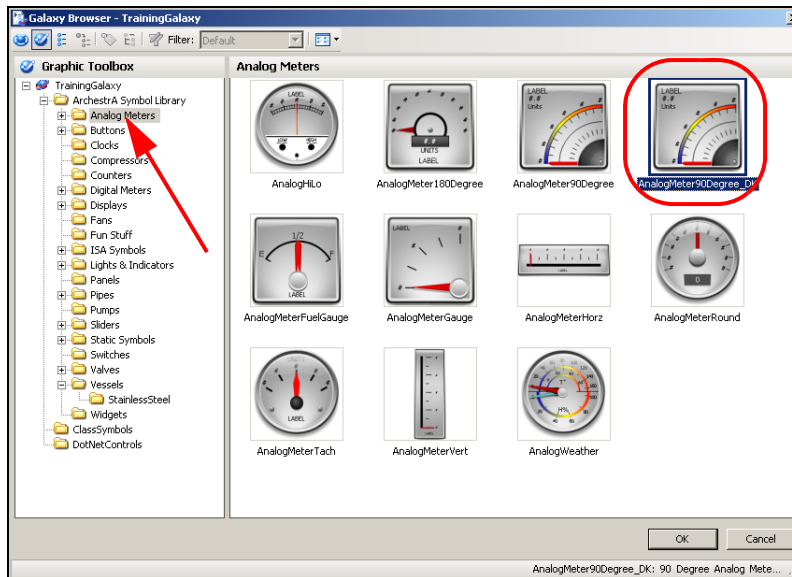
## Test What You Have Built

15. Click **Runtime**.
16. Ensure the animations are updating.
17. Click **Development!**.

## Define a Tag Provider

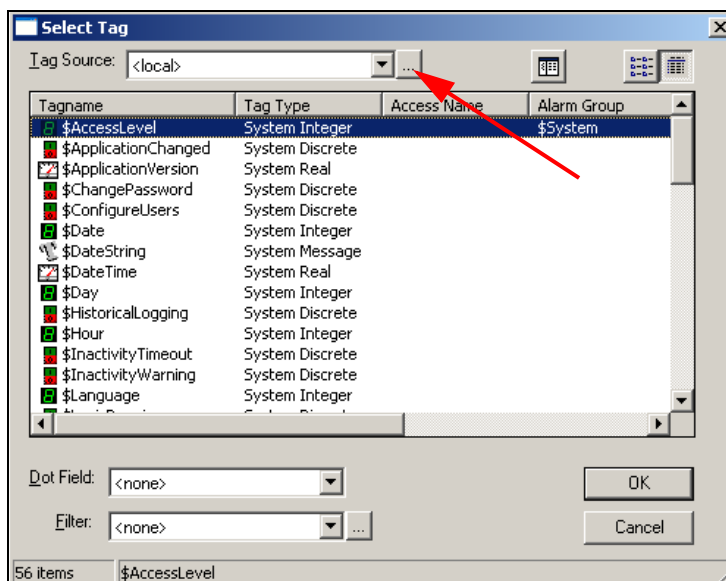
The next method you will use is to define a Tag Provider within your local InTouch application.

18. Click **Embed Archestra Graphic**.
19. Select **Archestra Symbol Library\Analog Meters**, and then double-click **AnalogMeter90Degree\_DK**.



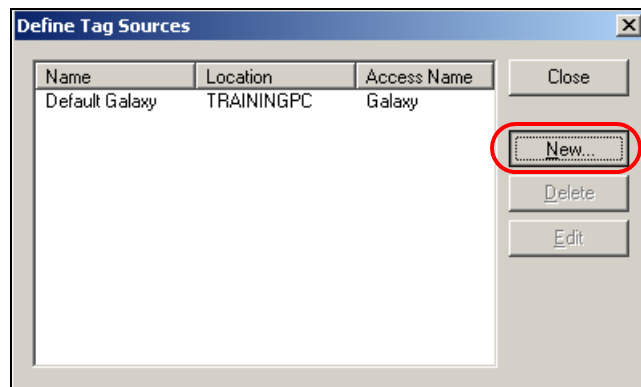
**Note:** Your symbol will have your own initials rather than the DK used in the example.

20. In the **Remote** window, place the symbol.
21. Change the label to **Temperature**, and then double-click it to open it for editing.
22. Select **Value**, then double-click in the **Default Value** field to open the **Select Tag** dialog box.
23. Click the ellipsis button to create a new **Tag Source**.



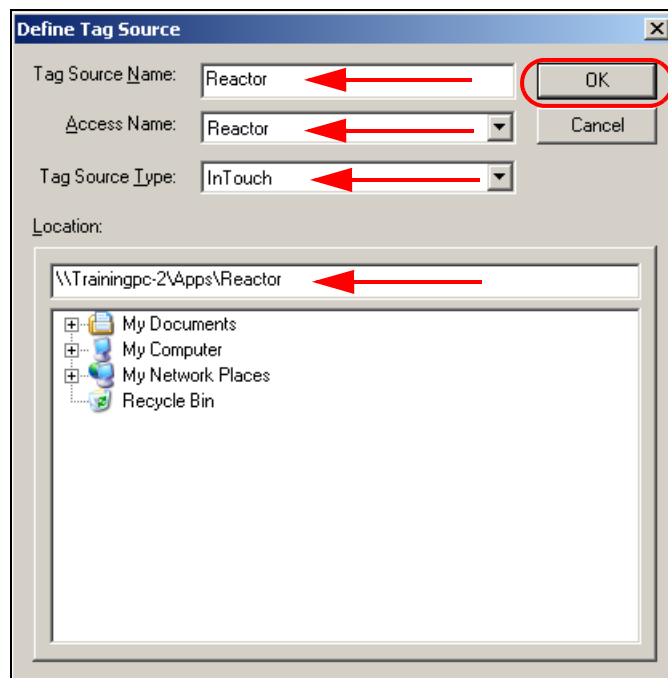


24. Click **New**.



25. Configure as follows:

**Tag Source Name:** Reactor  
**Access Name:** Reactor  
**Tag Source Type:** InTouch  
**Location:** Provided by your instructor



---

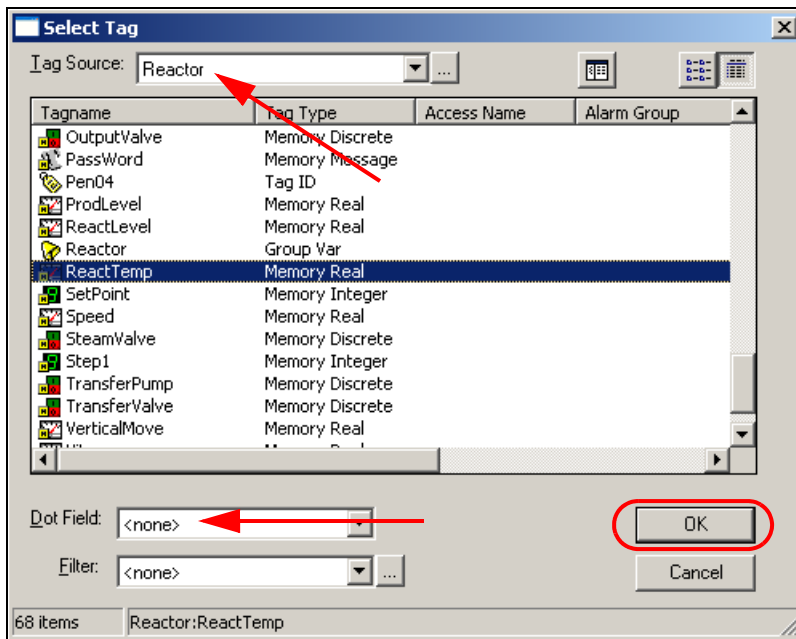
**Note:** Your node name will vary from the example, please substitute Trainingpc-2 as shown in the example with the node name provided by your instructor.

---

26. Click **OK**.

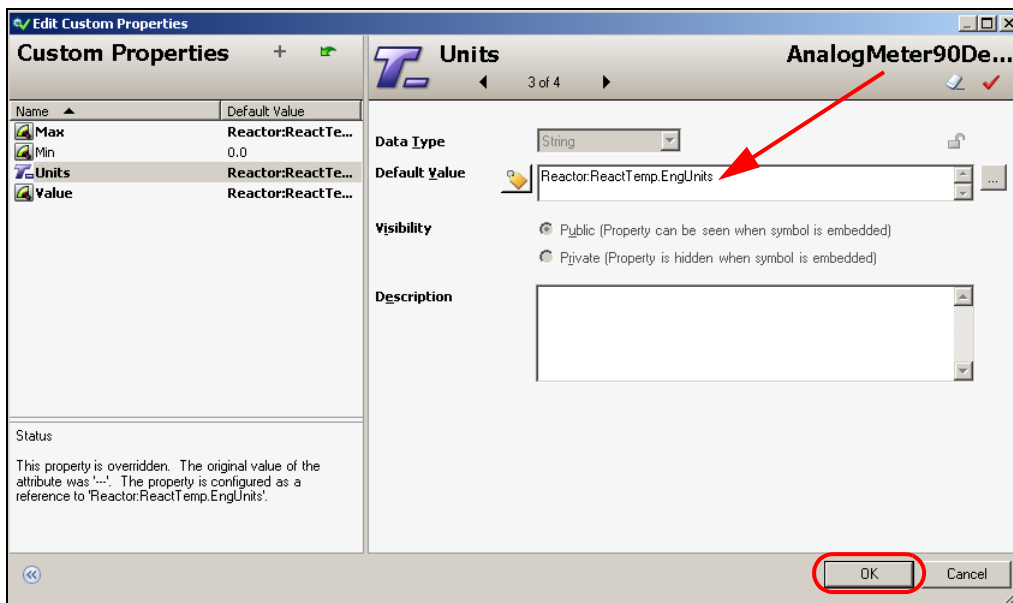
27. Click **Close**.

28. From the **Tag Source** drop-down select **Reactor**, from the **Tagname** list select **ReactTemp**, and then click **OK**.



**Note:** Make sure to clear out any **Dot Field** selections that may have carried over.

29. Select **Max**, in the **Default Value** field select **Reactor:ReactTemp.MaxEU**, and then click **OK**.
30. Select **Units**, in the **Default Value** field select **Reactor:ReactTemp.EngUnits**, and then click **OK**.



## Test What You Have Built

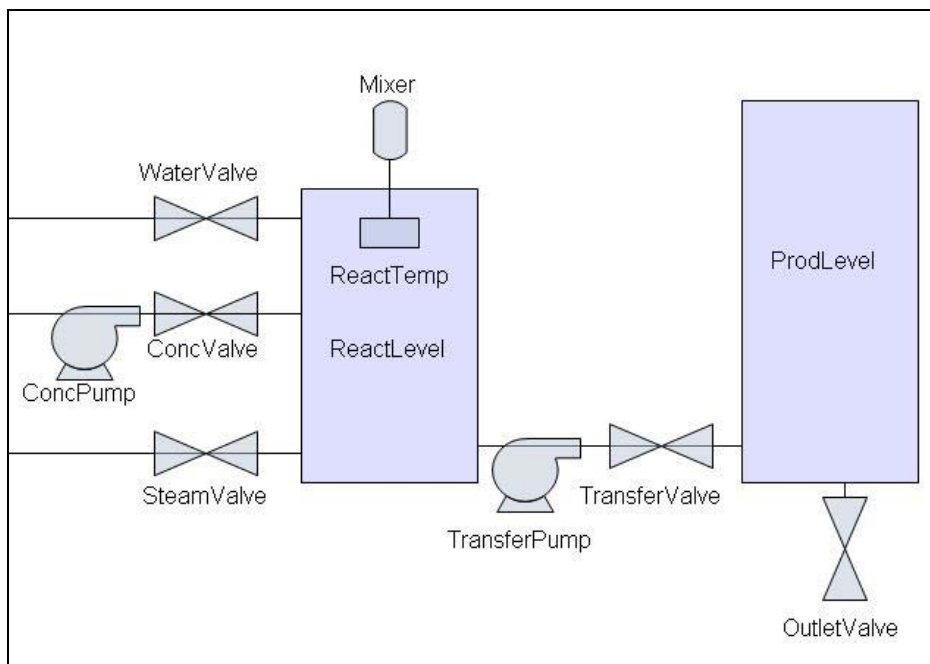
31. Click **Runtime**.
32. Ensure the animations are updating.
33. Click **Development!**.

## Create the User Interface (Optional)

For the remainder of this lab, you will create a user interface to model the process shown below. You may use graphics of your choosing to create the interface. It does not need to duplicate the exact layout shown in the graphic below.

### List of Available Remote Tags

| Function                                               | Tagname                                                                                                                       |
|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Main Tank                                              | <ul style="list-style-type: none"> <li>• ReactLevel</li> <li>• ReactTemp</li> <li>• Mixer</li> </ul>                          |
| Inputs to Main Tank                                    | <ul style="list-style-type: none"> <li>• WaterValve</li> <li>• ConcPump</li> <li>• ConcValve</li> <li>• SteamValve</li> </ul> |
| Transferring material from Main Tank to Secondary Tank | <ul style="list-style-type: none"> <li>• TransferPump</li> <li>• TransferValve</li> </ul>                                     |
| Secondary Tank                                         | <ul style="list-style-type: none"> <li>• ProdLevel</li> </ul>                                                                 |
| Discrete output control from secondary tank            | <ul style="list-style-type: none"> <li>• OutputValve</li> </ul>                                                               |





---

## Section 2 – Alarm Providers

This section discusses how to receive alarms from a remote application.

### Distributed Alarm Object Components

The InTouch Distributed Alarm System provides a notification system to inform operators of process and system conditions. This system supports the displaying, logging, and printing of process alarms and system events. Alarms represent warnings of process conditions, while events represent normal system status messages. InTouch supports a Distributed Alarm System that allows the display of alarms and events generated by the local InTouch application and by alarm systems of other networked InTouch applications.

The basic alarm communication protocol uses Wonderware's TCP/IP-based SuiteLink protocol. This uses a minimal amount of CPU and network resources. Alarm update time is very fast and includes an accurate time stamp. InTouch provides up-to-the-millisecond timestamps of the alarms when the alarm occurs, not when the consumer receives the alarm.

The Alarm Objects can be provided to an application user to enable pre-configured (by the system designer) views to be selected at runtime, either by a simple right-click command menu or via scripting.

Alarms can be acknowledged on the local InTouch node or from a remote node on the network. Alarms can also be enabled or disabled directly or indirectly under full control of the application. Alarm suppression can be applied to single alarm classes, tags, or groups to prohibit the display of alarm information on a specific view node. System-wide disablement can block alarm activity at the source.

A separate alarm comment field allows more feedback about the alarm to be provided. Expanded Dot Field support allows greater control over alarms.

The InTouch Distributed Alarm Subsystem supports logging alarms and events to a Microsoft SQL Server or MSDE database. This provides the power of a relational database, regardless of the size of the application or budget. Visualization tools allow easy and open access to these data tables via the Alarm DB View Control.

The Alarm DB Logger can be manually started on demand or can be configured to run as a Windows service. Alarm logging uses UTC (GMT) time stamping and provides compatibility with DST and across time zones.

The Distributed Alarm System allows a secondary backup alarm provider to be configured, thus establishing a fail-safe system. Alarm provider pairs are configured using a supplied utility. Should a primary alarm provider fail, the Distributed Alarm Subsystem seamlessly acquires alarm information from the backup system. On reconnection of the primary node, the Distributed Alarm Subsystem ensures that alarm acknowledgments are resynchronized prior to the returning primary system becoming live.

A summary of the distributed alarm system features include:

- Alarm Providers that keep track of items that may go into an alarmed condition and provide the Distributed Alarm System with the list of these items, including information on any hierarchical grouping of the items
- Alarm Consumers that provide the Distributed Alarm System with a set of queries, identifying alarmable items about which it wishes to receive notifications
- The ability to display and acknowledge alarms from any InTouch node on a network.
- Completely configurable Alarm Display object
- Numerous Alarm System properties
- QuickScript functions that provide dynamic control over the alarm display and alarm acknowledgment
- Name Manager, which allows multiple Alarm Groups across different applications to be called via a single name
- Capability to add comments to alarms when acknowledged

## Displaying Distributed Alarms

To display remote (network) alarm providers, the **Alarm Query** field of the alarm object will need to be modified. The default query **\\InTouch!\$System** refers to local alarms coming from InTouch.

Additional queries can be used:

- **\\NodeName\\InTouch!\$System** can be used here, where **\\NodeName** refers to a network node name where alarms are being generated by an alarm provider.
- Named queries from the Distributed Name Manager can be entered here as well, to provide access to alarm providers grouped under one name.
- **\\NodeName\\InTouch!<GroupName>!<TagName>** can be used to specify a query on a specific tag.
- **\\NodeName\\InTouch!<GroupName>!<Part of a Tag Name\*>** can be used to filter a query on similarly named tags. For example: **\\Train05\\InTouch!RemoteAlarms!Real\***.

Configure the other parameters of the **Default Query Properties** for the type of display and any filtering the application requires.

## Distributed Alarms

Both InTouch's Standard and Distributed Alarm Systems provide services to display, log, print and acknowledge process alarms and system events. Under the Distributed Alarm system, InTouch nodes are classified as alarm "providers" and/or alarm "consumers," allowing the display and acknowledgment of alarms generated by the local alarm systems of other InTouch applications on the network.

Since the distributed systems allows the view and acknowledgment of alarms not represented in the local Tagname Dictionary, distributed alarms combined with Remote Tagname Referencing facilitate the development of true client/server applications. InTouch's Distributed Name Manager can be used to combine both local and remote alarm groups into customized queries that can be called under a single name. The distributed alarm system also includes script functions that provide dynamic control over alarm display and acknowledgment, as well as the ability to add comments to alarms when acknowledged.

---

# Lab 21 – Distributed Alarming

## Introduction

In this lab, you will be using the Tag Server from the previous lab as an alarm source to illustrate the concept of Distributed Alarming, which provides the ability to receive alarms from a remote alarm provider.

## Objectives

- Connect to a remote alarm provider for all alarm consumers
- Display provider information in the alarm controls

---

**Note:** In this lab you may be required to provide additional information depending on your operating system. Your instructor will provide this information as necessary.

---

## Distributed Alarming

First modify the queries of the various alarm consumers, such as: Alarm Object, Alarm Tree Viewer, and Alarm Database are looking at.

## Alarm Control Modifications

1. Close all windows except for **Security/Tag Viewer** and **Alarms**.
2. Double-click the **AlarmViewer ActiveX** control to open it for editing.

| Time                   | Name   | State | Class  | Type  | Priority | Group     | Provider | Value | Limit |
|------------------------|--------|-------|--------|-------|----------|-----------|----------|-------|-------|
| 04/24/2012 02:28:40 PM | Alarm1 | UNACK | Value  | HIHI  | 1        | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm2 | UNACK | Value  | HI    | 250      | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm3 | UNACK | Value  | LO    | 500      | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm4 | UNACK | Value  | LOLO  | 750      | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm5 | ACK   | Dev    | Minor | 1        | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm6 | ACK   | Dev    | Major | 250      | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm7 | ACK   | ROC    | 1     | 500      | GroupName | Provider | 15    | 14    |
| 04/24/2012 02:28:40 PM | Alarm8 | ACK   | Custom | 1     | 750      | GroupName | Provider | 15    | 14    |

3. Click the **Query** tab, and then change the **Alarm Query** to:  
**\\localhost\Intouch!\$System \\trainingpc-2\Intouch!\$System**

**AlarmViewerCtrl1 Properties**

Control Name | General | Color | Time Format | **Query** | Properties | Events

From Priority: 1 To Priority: 999

Alarm State: All Query Type: Summary

Alarm Query: \\localhost\Intouch!\$system \\trainingpc-2\Intouch!\$system

Query Favorites File: \\TRAININGPC\TrainingGa ... Edit Query Favourites...

Sort Column: Time ☐ Auto Scroll to New Alarms

Secondary Sort Column:

Sort Direction: ☒ Ascending ☐ Descending

OK Cancel Apply Help

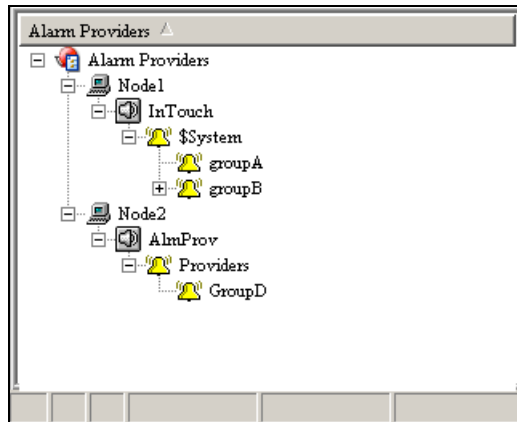
**Note:** Make sure you put a space between the word **System** and the double-backslash **\\**. You will also need to substitute **trainingpc-2** with the node name provided by your instructor.

4. Click **OK**.

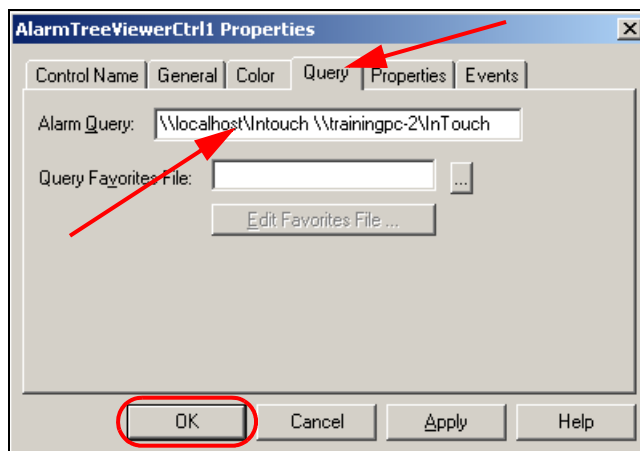


## Alarm Tree Viewer Modifications

5. Double-click the **AlarmTreeView** control to open it for editing.



6. Click the **Query** tab, and then change the **Alarm Query** to:  
**\\localhost\Intouch \trainingpc-2\Intouch**

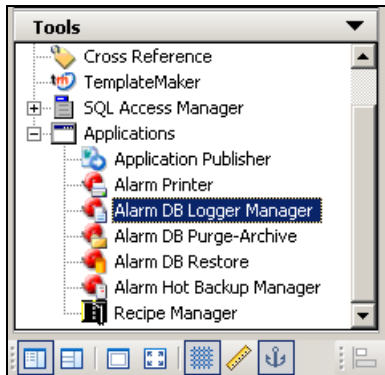


**Note:** Make sure you put a space between the word **System** and the double-backslash \\ . You will also need to substitute **trainingpc-2** with the node name provided by your instructor.

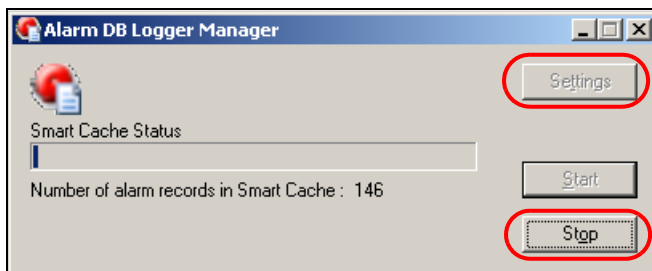
7. Click **OK**.

## Alarm Database Modifications

8. In **Tools** pane, expand **Applications**, and then double-click **Alarm DB Logger Manager**.

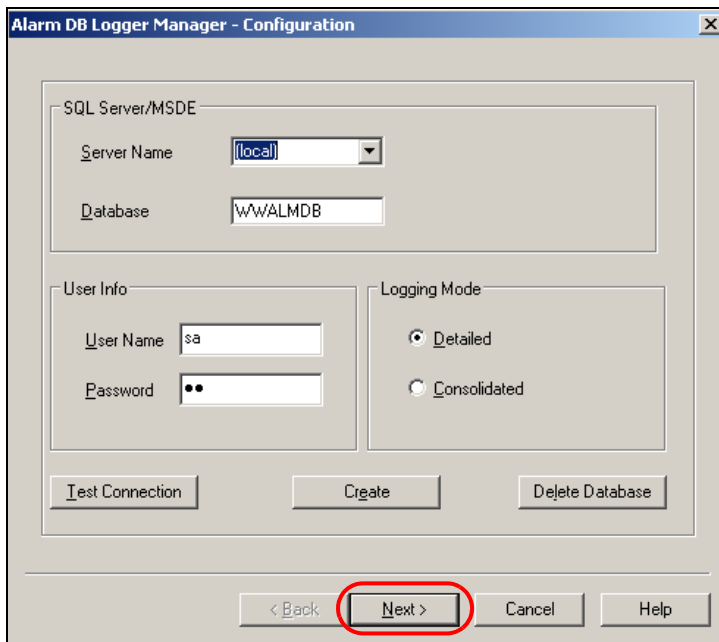


9. Click **Stop**, and then click **Settings**.



This may take a few moments.

10. Verify your server name is set to **(local)**, and then click **Next**.



11. In the **Alarm Query** field, enter the following:

**\\localhost\Intouch\System**

**\\trainingpc-2\Intouch!\$System**

Alarm DB Logger Manager - Query Selection

Query Details

Alarm State: All From Priority: 1

Query Type: Historical To Priority: 999

Alarm Query

\\localhost\Intouch!\$system  
\\trainingpc-2\Intouch!\$system

< Back Next > Cancel Help

**Note:** Substitute **trainingpc-2** with the node name provided by your instructor.

12. Click **Next**.
13. Click **Finish**.

Alarm DB Logger Manager - Advanced Setting

Running Logger As

☐ Windows Service

☒ Normal application

Miscellaneous

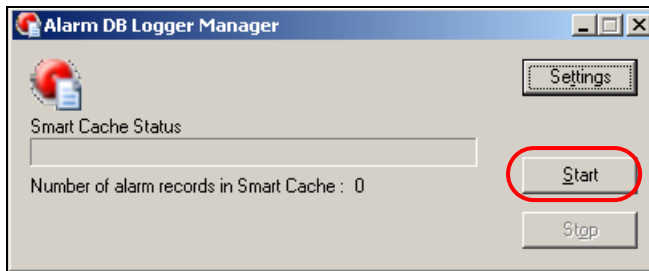
☒ Log Events

Performance Tuning

Log Alarms Every: 100 m sec

< Back Finish Cancel Help

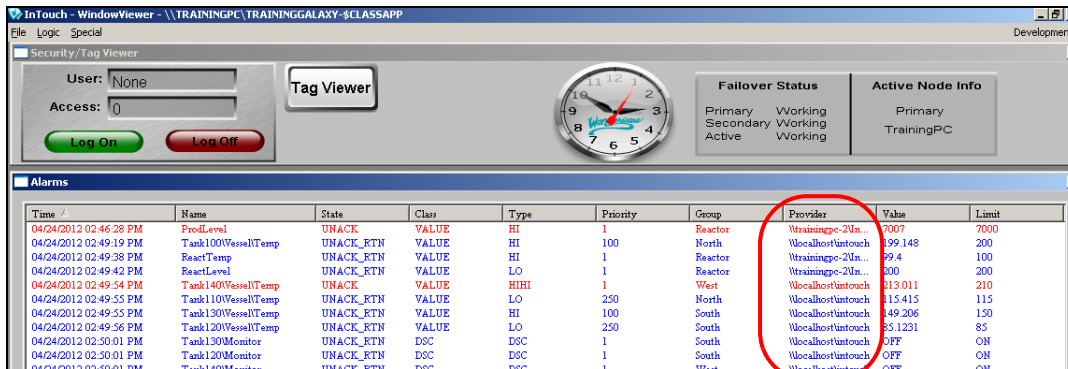
14. Click **Start**.



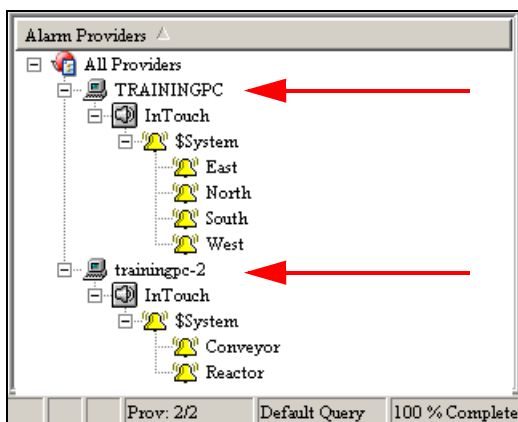
15. Click X to close

## Test What You Have Built

16. Click **Runtime**.
17. Under the **Provider** column you will see two providers, one is the local host the other is specific to your node name.

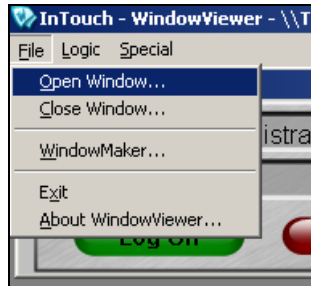


18. Log in as **Administrator** (password: **Wonderware**)  
You are now able to acknowledge alarms.
19. In the **Alarm Providers** pane, you can see alarms from both providers.

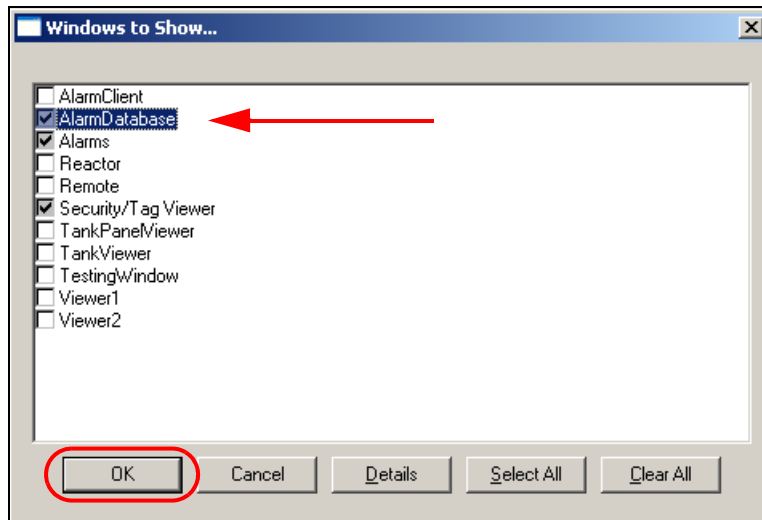


**Note:** Clicking **All Providers** will not display both providers. Instead, right-click in the **Alarm Viewer** and select **Query Favorites**, and then double-click **Default Query**.

20. On the **File** menu, click **Open Window**.



21. In the **Windows to Show** dialog box, check **AlarmDatabase**, and then click **OK**.



The database now shows alarms from both providers.

| AlarmDatabase      |           |       |      |          |                 |         |                    |        |  |
|--------------------|-----------|-------|------|----------|-----------------|---------|--------------------|--------|--|
| Time               | State     | Class | Type | Priority | Name            | Group   | Provider           | Value  |  |
| 04/24/2012 02:5... | UNACK_ALM | DSC   | DSC  | 1        | Tank130Monitor  | South   | localhost\Intouch  | ON     |  |
| 04/24/2012 02:5... | UNACK_ALM | DSC   | DSC  | 1        | Tank120Monitor  | South   | localhost\Intouch  | ON     |  |
| 04/24/2012 02:5... | UNACK_ALM | DSC   | DSC  | 1        | Tank110Monitor  | North   | localhost\Intouch  | ON     |  |
| 04/24/2012 02:5... | UNACK_ALM | DSC   | DSC  | 1        | Tank100Monitor  | North   | localhost\Intouch  | ON     |  |
| 04/24/2012 02:5... | UNACK_RTH | VALUE | LO   | 1        | ReactLevel      | Reactor | trainingpc-2UnT... | 200    |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | HI   | 1        | ReactTemp       | Reactor | trainingpc-2UnT... | 101    |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | HI   | 1        | ReactLevel      | Reactor | trainingpc-2UnT... | 1850   |  |
| 04/24/2012 02:5... | UNACK_RTH | VALUE | HI   | 1        | ReactTemp       | Reactor | trainingpc-2UnT... | 99.4   |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | LO   | 1        | ReactLevel      | Reactor | trainingpc-2UnT... | 155    |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | HI   | 1        | ProdLevel       | Reactor | trainingpc-2UnT... | 7007   |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | HI   | 1        | ReactTemp       | Reactor | trainingpc-2UnT... | 179.9  |  |
| 04/24/2012 02:5... | UNACK_RTH | VALUE | HI   | 1        | ReactLevel      | Reactor | trainingpc-2UnT... | 1775   |  |
| 04/24/2012 02:5... | UNACK_ALM | VALUE | HI   | 1        | Tank120(Vessel) | South   | localhost\Intouch  | 190.37 |  |

localhost - WWAlmDb      Displaying 1 to 100 of 1698 records.      Connected

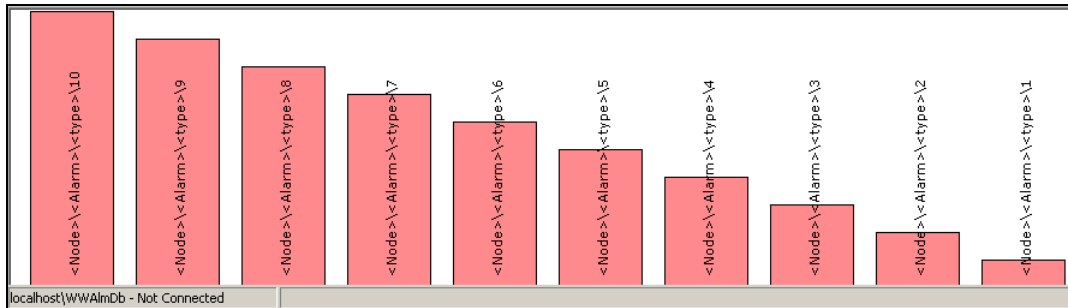
☐ Last Minute  
☐ Last 5 Minutes  
☐ Last 15 Minutes

**Note:** The pareto does not show the provider information yet.

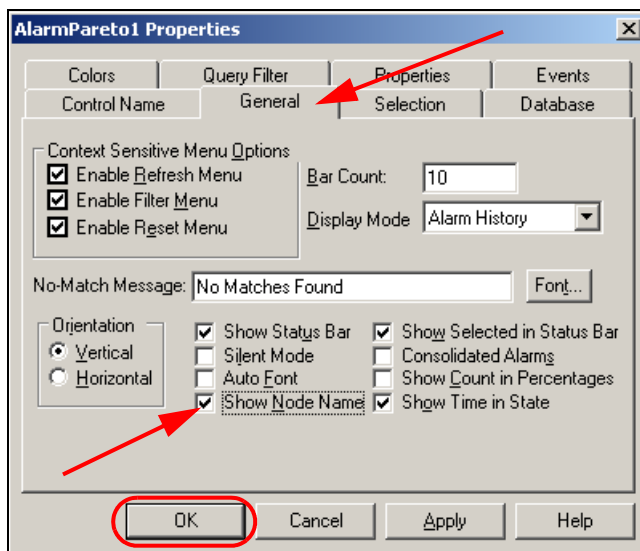
22. Click **Development!**.

## Pareto Modifications

23. In the **AlarmDatabase** window, double-click **Pareto** to open it for editing.



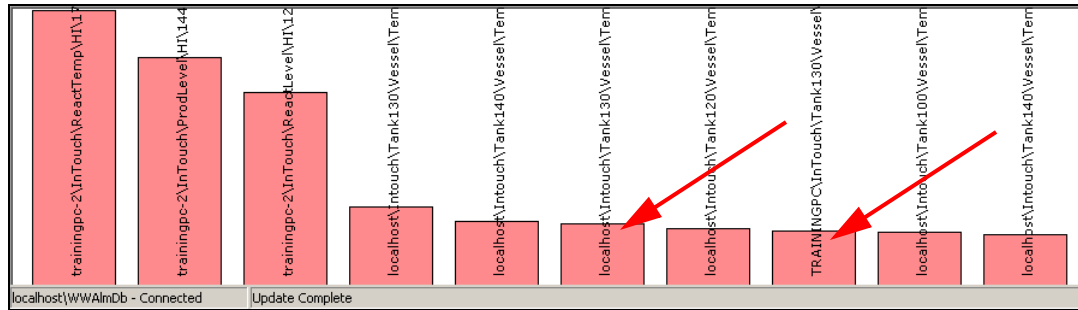
24. Click the **General** tab, and then check **Show Node Name**.



25. Click **OK**.

## Test What You Have Built

26. Click **Runtime**.
27. The **Pareto** will now show alarm provider information.



28. Click **Development!**.





## Section 3 – Access Tag Server History

This section discusses using InTouch history from a remote application.

### Distributed History System

InTouch provides a distributed history system that allows retrieval of historical data from any InTouch application, even those across a network. This system extends the capabilities of the standard InTouch history by allowing remote retrieval of data from multiple historical databases simultaneously. These databases are referred to as history providers. Up to eight history providers can be displayed simultaneously, one for each historical trend chart pen.

### Configure Distributed or Local Historical Logging

Use Historical Logging Properties to enable global tagname logging and enter of the location where the log files will be stored. Enter a valid Universal Naming Convention (UNC) path.

For example:

\\Node\Share\Path.

The screenshot shows the 'Historical Logging Properties' dialog box. It has a blue title bar and a light beige background. At the top left, there is a checked checkbox labeled 'Enable Historical Logging'. To its right are 'OK' and 'Cancel' buttons. Below this is a section titled 'Historical Log File' in blue. It contains a 'Keep Log Files for:' field with a value of '0' and the unit 'days'. There are two radio buttons: 'Store Log Files in Application Directory' (which is selected) and 'Store Log Files in Specific Directory:' (which is unselected). Below the radio buttons is a text field labeled 'Name of Logging Node:'. The next section is titled 'Printing Control' in blue. It contains three fields: 'Default % of page to print on:' with a value of '50' and a '%' symbol; 'Max consecutive time to spend printing:' with a value of '500' and the unit 'msec'; and 'Time to wait between printing:' with a value of '2000' and the unit 'msec'. At the bottom of this section are two buttons: 'Select Printer Font ...' and a checkbox labeled 'Always use color when printing'.

Enter the name of the node that will be logging to the history log file, but note that this setting will only allow the node named here to log to the file.

---

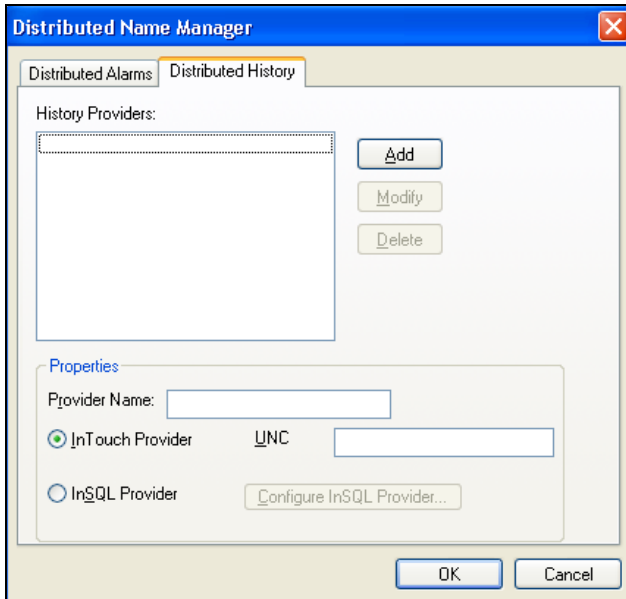
**Note:** When an application with the Enable Historical Logging option selected is distributed to a WindowViewer node, that node checks this option to determine if it should log or not. If Enable Historical Logging is selected, the possible settings are Field equals name of Node - Logging enabled and Field does not equal name of Node - Logging disabled.

---

## Configure the Distributed History Provider List

Each remote history provider, from which historical data will be received, must be registered in the InTouch history provider list. This list specifies a name and network location for each history provider. These names will be used when referring to a history provider in InTouch.

In the Distributed Name Manager, Distributed History tab, configure the Provider Name with the new historical provider that should be used.



To access an InTouch application's log file, select **InTouch Provider** and enter the **UNC** (Universal Naming Convention) path for the InTouch application directory. The valid format is:

\\Node\Share\ApplicationPath

---

**Note:** This path is limited to 51 characters. C\$ is only available to administrators. This location cannot be on the root and must be at least one layer down. The directory that is connected to must be shared on that node. The History Provider Name should be the same as the associated Access Name. If not, enter the History Provider Name in place of the Access Name when selecting a tag. If the UNC location is password-protected, first establish a connection using the Windows Explorer.

---

## Access Tags From the History Provider

In runtime, dynamically configure a historical trend's remote history provider by creating a QuickScript that specifies the remote history provider tagname reference in the HTSetPenName function.

For example:

```
HTSetPenName("HistTrendTag", 1, "HistPrv1.Boiler1");
```

where, 1 specifies the pen that will plot the specified remote history provider tagname.

Another method is to use a dot field for a HistTrendTag.

For example:

```
HistTrendTag.Pen1 = Tagname.TagID
```

---

# Lab 22 – Distributed History

## Introduction

In this lab, you retrieve InTouch historical data from a remote InTouch application. For the purposes of this lab, you access the remote historical data from the same node you accessed the alarms and tag data in the previous labs.

## Learning Objectives

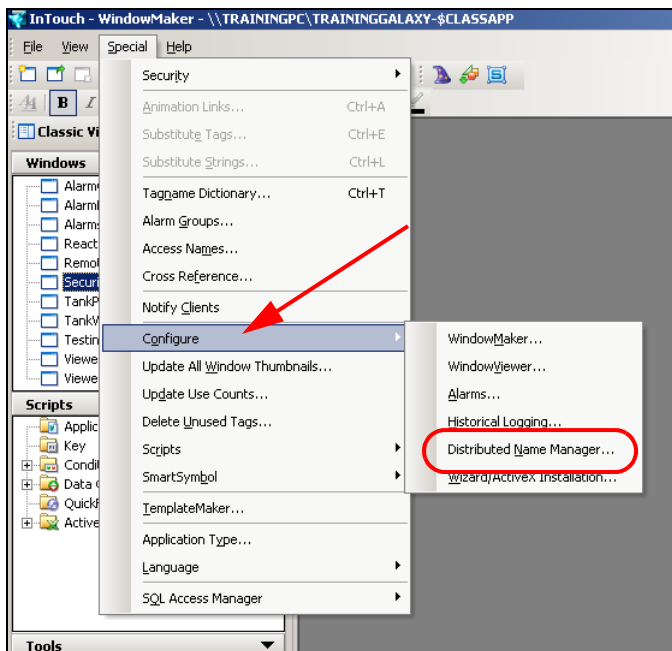
Upon completion of this lab, you will be able to:

- Configure the distributed name manager
- Access history from a history provider

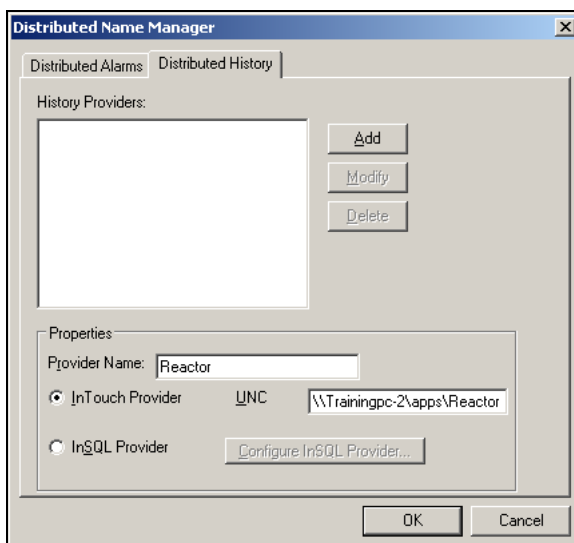
## Configure InTouch

The first thing you do is to configure InTouch to receive historical data from the remote application.

1. Close WindowViewer.
2. In WindowMaker, on the **Special** menu, click **Configure | Distributed Name Manager**.



3. Click the **Distributed History** tab.
4. In the **Provider Name** field enter **Reactor**, ensure the **InTouch Provider** radio button is selected, and then enter the **UNC** information provided by your instructor.



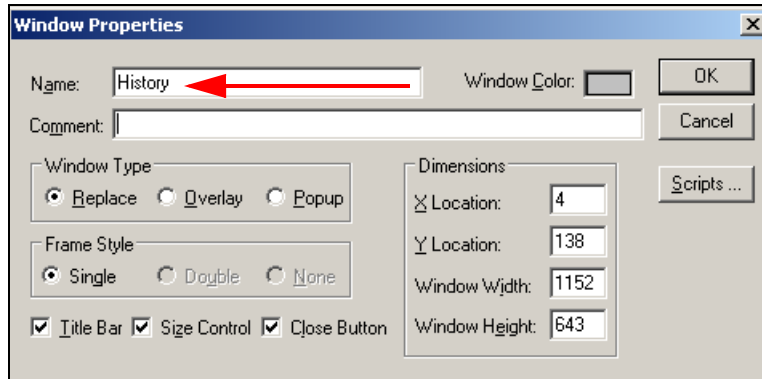
**Note:** This example uses **\\Trainingpc-2\\Apps\\Reactor**, but this will vary. Your instructor will provide you with the specific information for your location.

- Click **Add**, and then click **OK**.

Next, you will create a new InTouch window and add the required graphics.

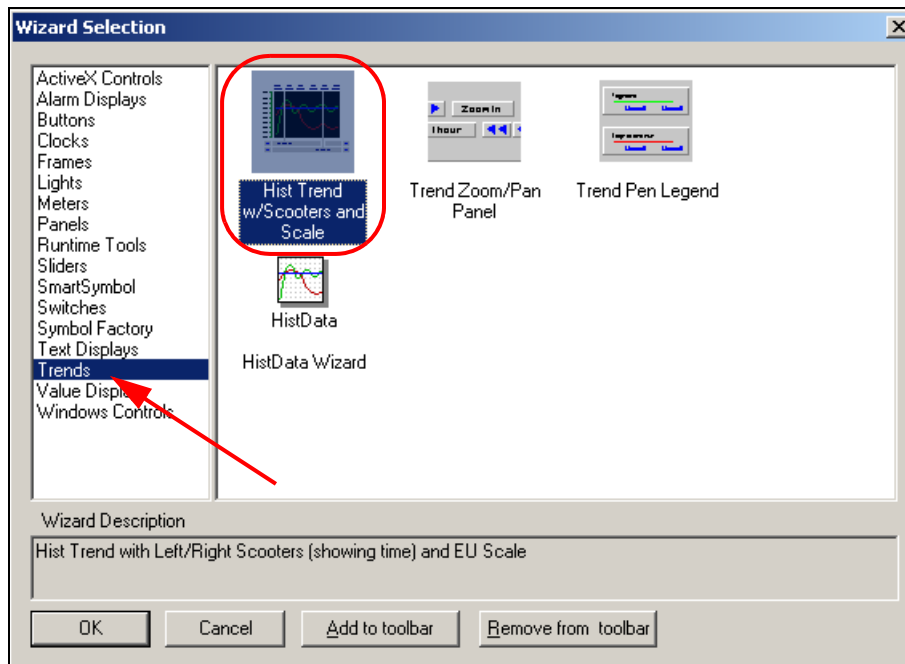
- On the **File** menu, click **New Window**.
- Name the window **History**.

It should have the same dimensions as the **Reactor** window.



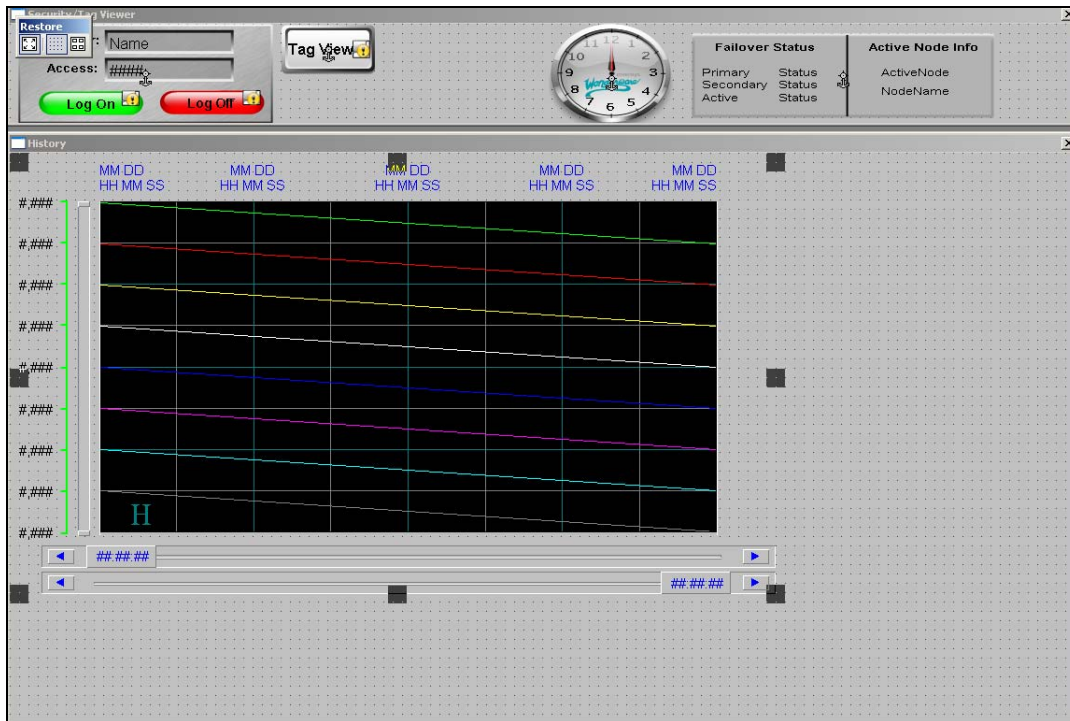
**Note:** The window dimensions will vary based on your resolution.

- Close all windows except for **History** and **Security/Tag Viewer**.
- Click **Wizards**, select **Trends**, and then double-click **Hist Trend w/Scooters and Scale**.

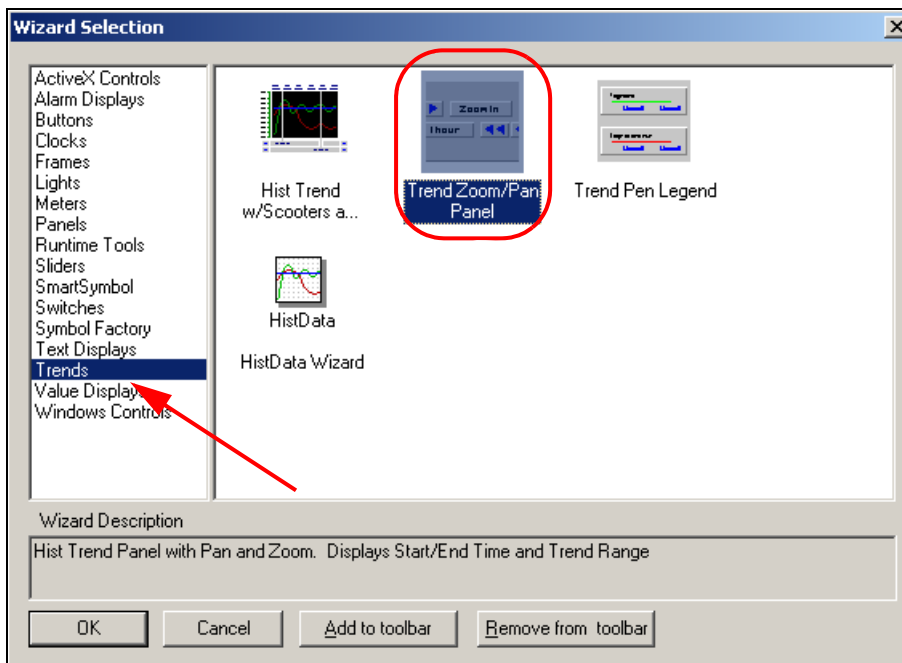


10. Place **Hist Trend w/Scooters and Scale** in the top-left of the **History** window.

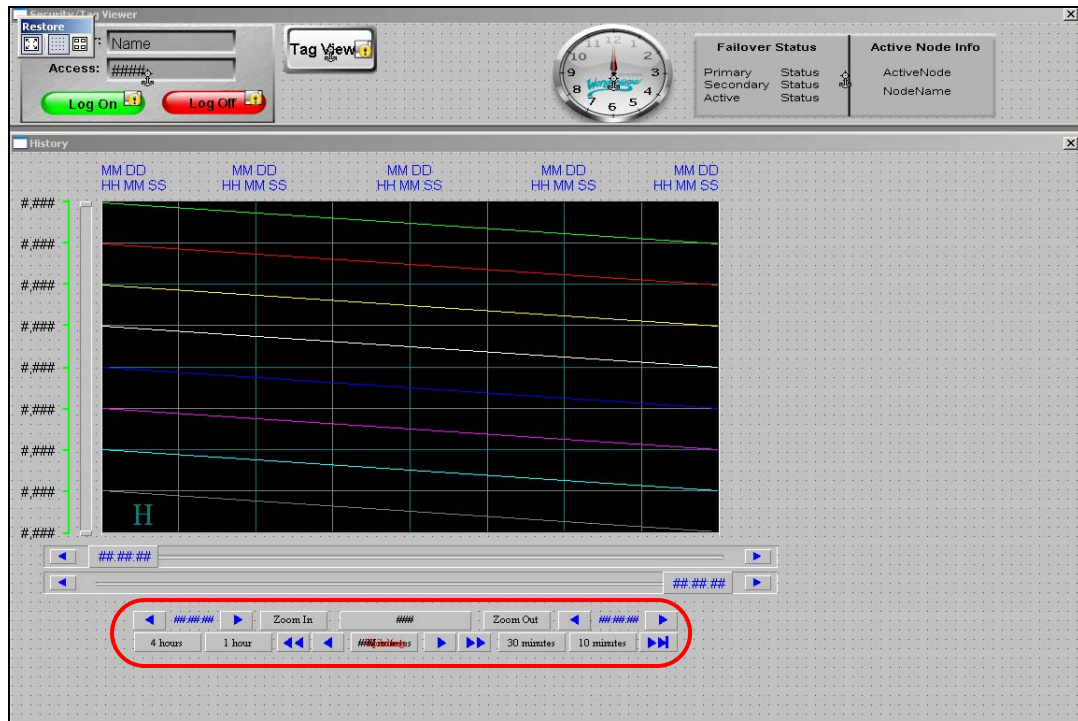
You may wish to go to full screen mode to increase its size.



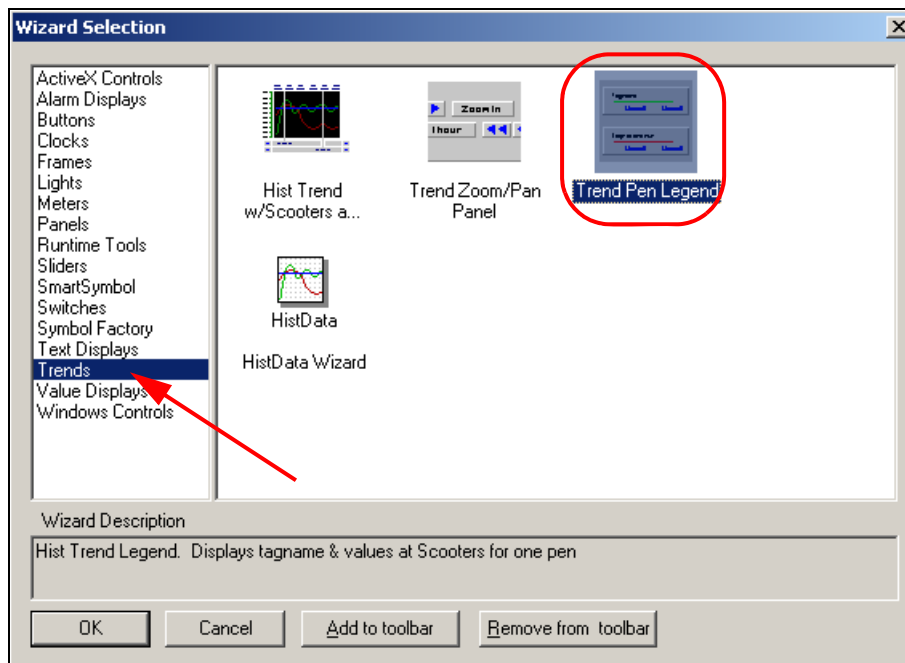
11. Click **Wizards**, select **Trends**, and then double-click **Trend Zoom/Pan Panel**.



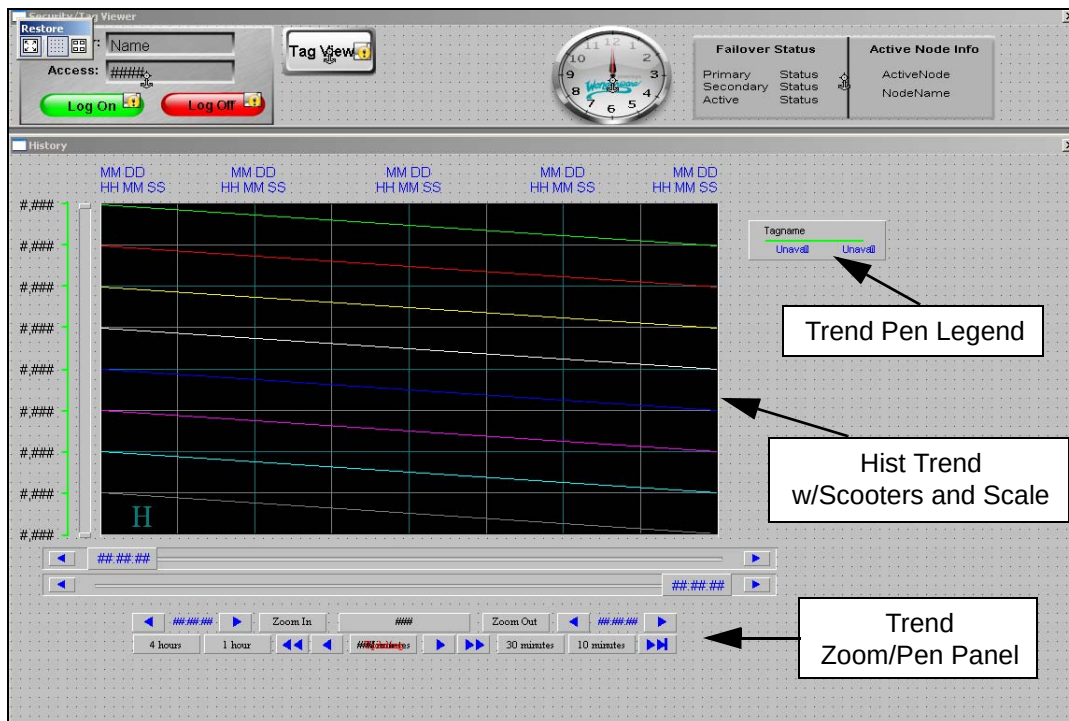
12. Place **Trend Zoom/Pan Panel** directly below the **Hist Trend w/Scooters and Scale** graphic.  
You may wish to go to full screen mode to increase its size.



13. Click **Wizards**, select **Trends**, and then double-click **Trend Pen Legend**.



14. Place **Trend Pen Legend** to the right of the **Hist Trend w/Scooters and Scale** graphic.



## Configure Hist Trend w/Scooters and Scale

15. Double-click on the **Hist Trend w/Scooters and Scale** graphic to open it for editing.
16. Click **Suggest** to populate the **HistTrend** and **PenScale** fields.

17. Click **Pens**.

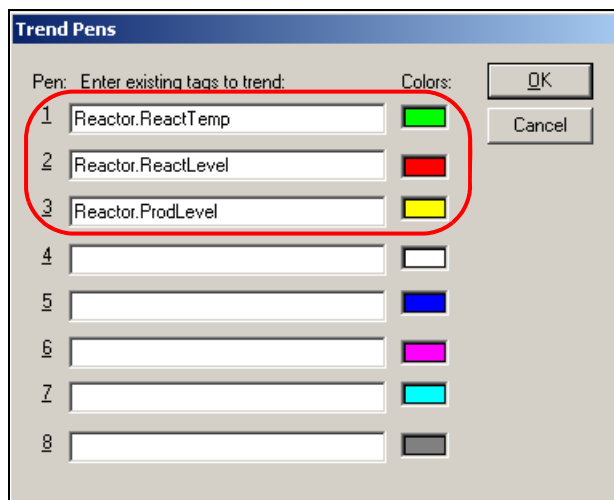
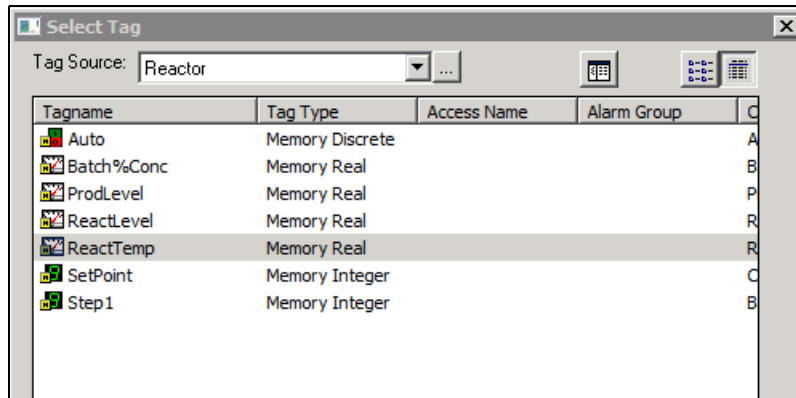
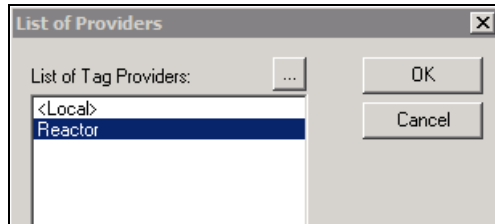
**Note:** If you get a message stating that the **viewer is running, cannot configure pens**, simply click **Cancel** and close WindowViewer. Then click on **Pens** again.



18. To associate a historically logged tag with a pen, double-click in the tag field, double-click **Reactor** from the list of **Providers**, and then browse to the desired tag using the **Reactor Tag Source**.

Configure the pens as follows:

| Pen # | Browse to Tag:     |
|-------|--------------------|
| 1     | Reactor.ReactTemp  |
| 2     | Reactor.ReactLevel |
| 3     | Reactor.ProdLevel  |



**Note:** Do not forget to set the **Dot Field** to **<none>**, if necessary.

19. Click **OK**.

20. Click **OK** to close the **Historical Trend Chart Wizard**.

## Configure Trend Zoom/Pan Panel

21. Double-click the **Trend Zoom/Pan Panel** object to open it for editing.
22. Click **Suggest** to populate the **HistTrend** and **Pan Minutes** fields, and then remove the **1** or any other number that populates from both suggested tagnames.

**Trend Panel Wizard**

The Trend Panel requires 2 Tags. Enter these below.  
If the tags that you enter below do not exist, the Wizard  
will create them. Click Suggest for suggestions on names.

Hist Trend:  (Hist Trend)

Pan Minutes:  (Memory Integer)

The Pan Minutes tag is used to adjust the pan range on the panel.  
It is not used elsewhere nor is it needed on any other Trend Wizard.

23. Click **OK**.

## Configure Trend Pen Legend

24. Double-click the **Trend Pen Legend** object to open it for editing.
25. Click **Suggest** to populate the **HistTrend** and **Pen Scale** fields, and then remove any extra numbers that populate from either of the suggested tagnames.

**Trend Legend Wizard**

The Trend Legend requires 2 Tags. Enter these below.  
If the tags that you enter below do not exist, the Wizard will create them. Click Suggest for suggestions on names.

Hist Trend:  (Hist Trend)

Pen Scale:  (Memory Integer)

The Pen Scale tag is used to display Engineering Units. If you also use the Trend Chart Wizard, specify this same Pen Scale tagname there as well.

Colors:

Tagname Label:

Values:

Trend Pen:

Pen Number:

Value Format:

For Invalid Values Display:

26. Ensure the legend is pointing to **Pen Number 1** and the **Trend Pen** color is green, and then click **OK**.
27. Duplicate the **Trend Pen Legend** object, place it directly below the original, and then double-click to open it for editing.
28. Change the **Pen Number** to **2**, the **Trend Pen** color to red, and then click **OK**.

**Trend Legend Wizard**

The Trend Legend requires 2 Tags. Enter these below.  
If the tags that you enter below do not exist, the Wizard will create them. Click Suggest for suggestions on names.

Hist Trend:  (Hist Trend)

Pen Scale:  (Memory Integer)

The Pen Scale tag is used to display Engineering Units. If you also use the Trend Chart Wizard, specify this same Pen Scale tagname there as well.

Colors:

Tagname Label:

Values:

Trend Pen:

Pen Number:

Value Format:

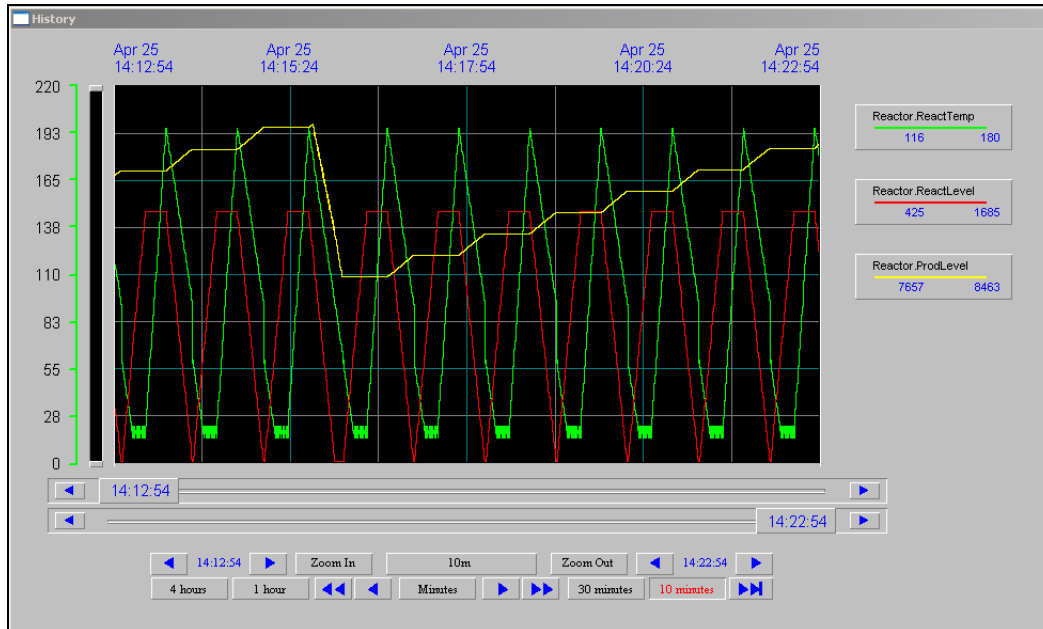
For Invalid Values Display:

29. Duplicate the **Trend Pen Legend** object again, place it directly below the other two, and then double-click to open it for editing.

30. Change the **Pen Number** to **3**, the **Trend Pen** color to yellow, and then click **OK**.

## Test What You Have Built

31. Click **Runtime**.
32. You should see historical data for all three tags in green, red, and yellow. The legend on the right should show the correct tag names and values.



33. Click **Development!**.



## Module 8 – Deployment

---

|                                           |             |
|-------------------------------------------|-------------|
| <b>Section 1 – Deploying Applications</b> | <b>8-3</b>  |
| <b>Lab 23 – Application Deployment</b>    | <b>8-5</b>  |
| <b>Section 2 – Terminal Services</b>      | <b>8-21</b> |

### **Module Objectives**

- Configure the Galaxy for InTouch application deployment

## Section 1 – Deploying Applications

This section discusses how InTouch is used to create a Platform, ViewEngine, and instance of the InTouch application for deployment. Graphical issues regarding deployed applications using show symbol scripting will be discussed.

### Development and Runtime

The Development environment is where the configuration of the Galaxy takes place. It encompasses the creation and configuration of the objects, as well as any Galaxy-wide setting such as the definition of the security model. The development environment is comprised of the Galaxy repository, the Galaxy configuration database, and the Archestra IDE

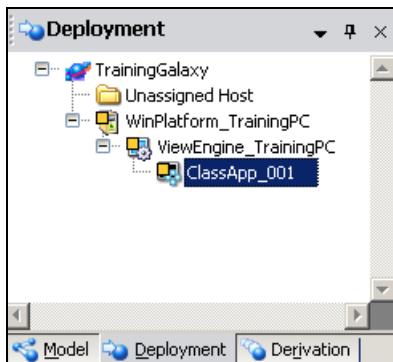
The Runtime environment is where the execution of the Galaxy takes place. Access to field devices (I/O), alarm generation, historization of data, script execution, and visualization of data, occurs only in the runtime environment. The runtime environment is comprised of execution components (Platforms and Engines) and client components (InTouch, ObjectViewer, and Information Server).

### Deployment View

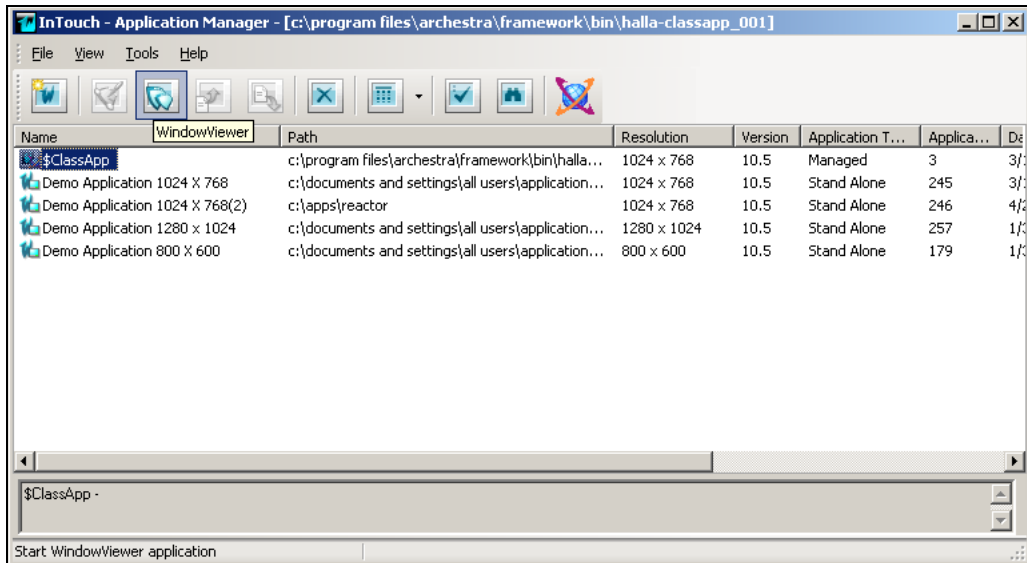
The Deployment view is used to display the assignment of the automation scheme to physical machines and process engines, showing which object instances are hosted on which computers. This view displays where the objects are running.

### Deploy InTouch Application

To deploy the instance of the InTouch application, create a new derived template of the Platform, the ViewEngine, and then create a new instances of each of the new objects for each node of the network. Assign the new ViewEngine object to the new Platform to ensure that the ViewEngine running the InTouch application, is hosted on the appropriate WinPlatform.



When deployed, the application will be available to the user from within the InTouch Application Manager.





---

# Lab 23 – Application Deployment

## Introduction

In this lab, you deploy an InTouch application to an operator's station using Archestra IDE.

## Objectives

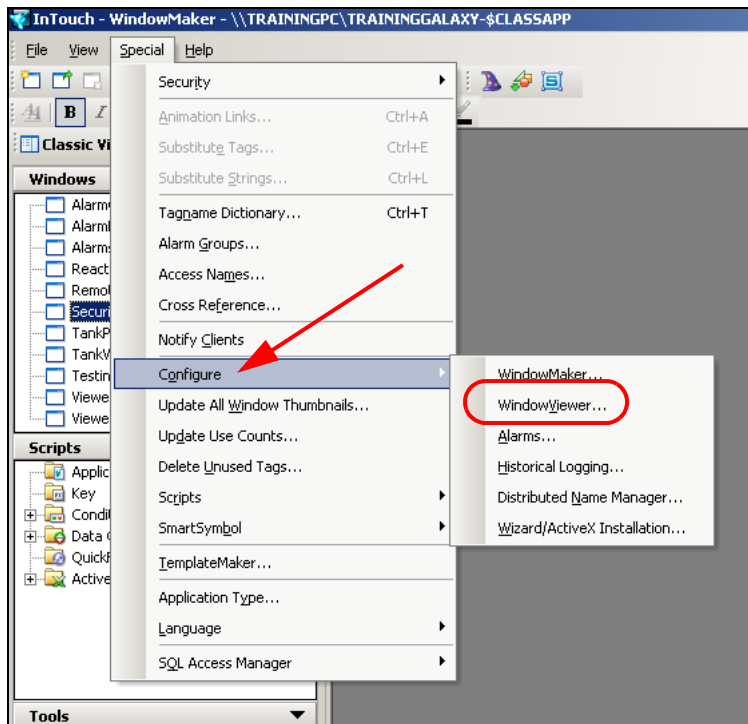
Upon completion of this lab, you will be able to:

- Create a Platform
- Create a ViewEngine
- Configure the application for deployment
- Create instances of objects
- Deploy objects
- Propagate changes to the application

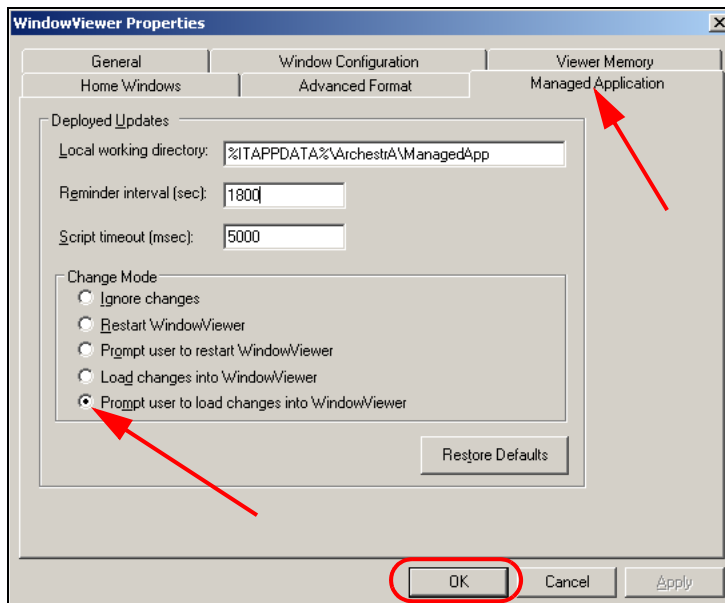
## Configure for Deployment

The first thing you do is to verify the application's configuration.

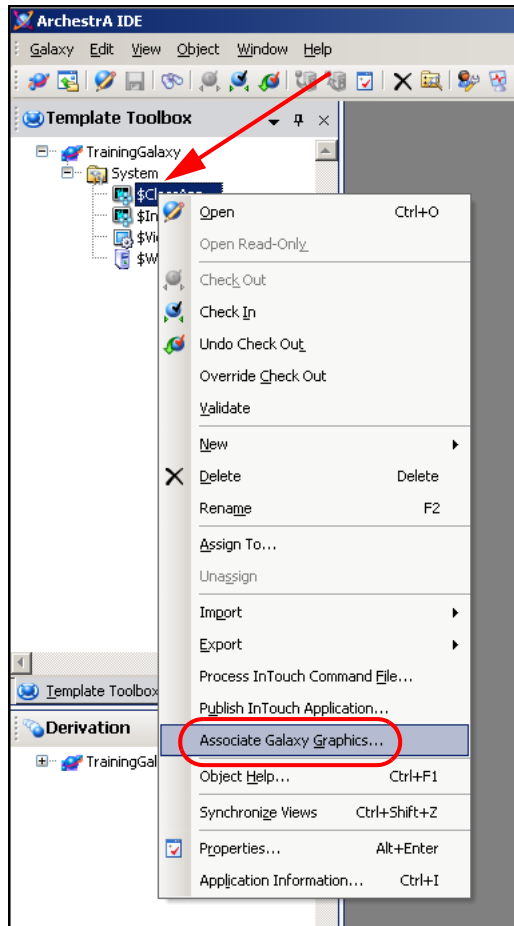
1. In WindowMaker, on the **Special** menu, click **Configure | WindowViewer**.



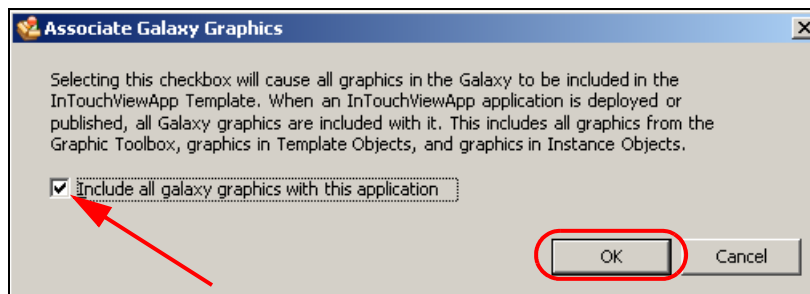
2. Click the **Managed Application** tab, ensure the **Prompt user to load changes into WindowViewer** radio button is selected, and then click **OK**.



3. Close WindowMaker, and then click **OK** when prompted to check in.
4. Click **Close**.
5. **Save, Exit**, and then **Check In** any other graphics that may be open.
6. In Arcestra IDE, in **Template Toolbox**, right-click **\$ClassApp** and select **Associate Galaxy Graphics**.



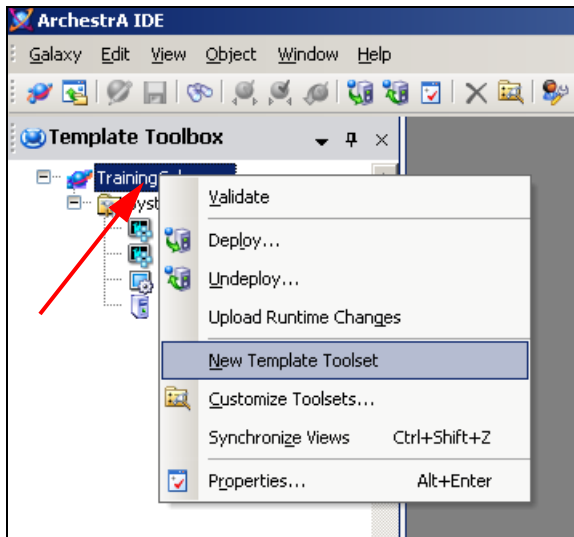
7. Check the box **Include all galaxy graphics with this application**, and then click **OK**.



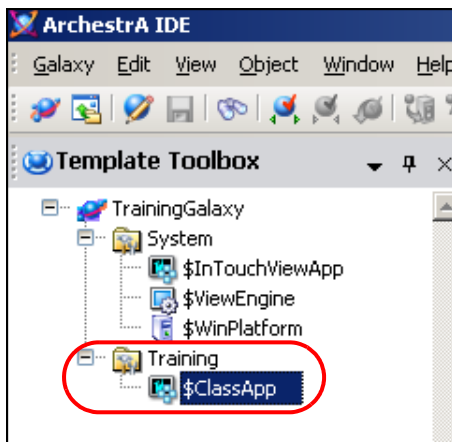
8. Click **Close**.

Next you will need to create and organize a template toolset.

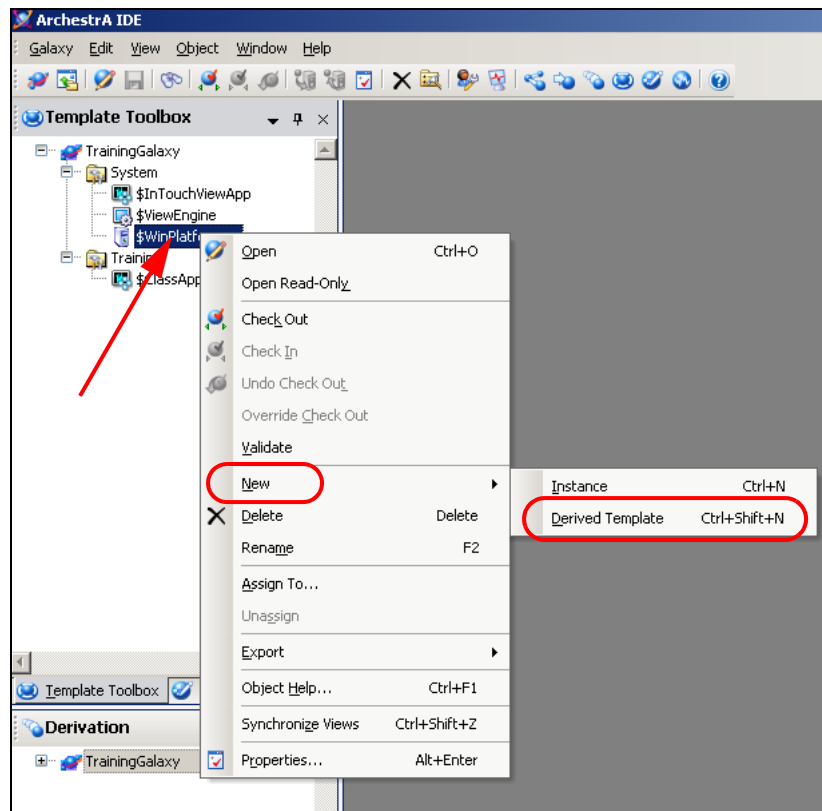
9. From **Template Toolbox**, right-click **TrainingGalaxy** and select **New Template Toolset**.



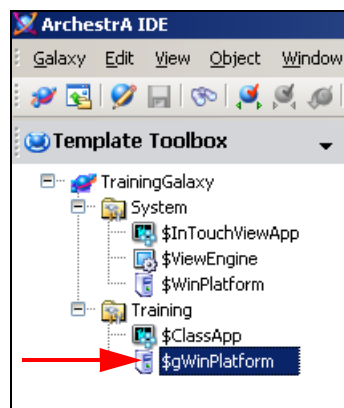
10. Name the new toolset **Training**, and then move **\$ClassApp** into it.



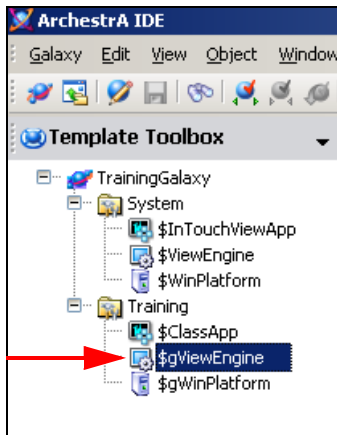
11. Right-click **\$WinPlatform** and select **New | Derived Template**.



12. Name it **\$gWinPlatform**, and then move it to the **Training** toolset.



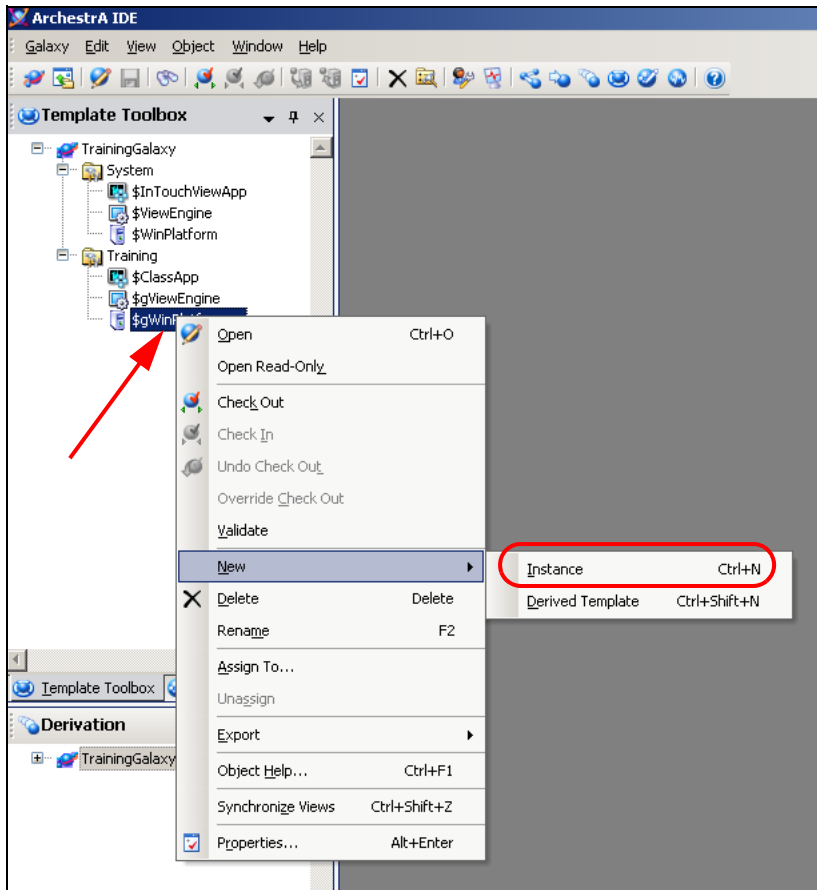
13. Right-click **\$ViewEngine** and select **New | Derived Template**.
14. Name it **\$gViewEngine**, and then move it to the **Training** toolset.



Next, create instances of the new objects for each node on your network.

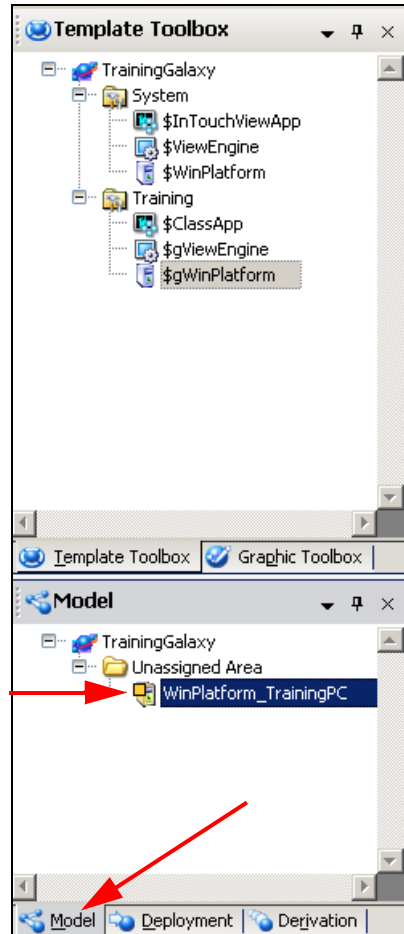
### Local Node

15. Right-click **\$gWinPlatform** and select **New | Instance**.

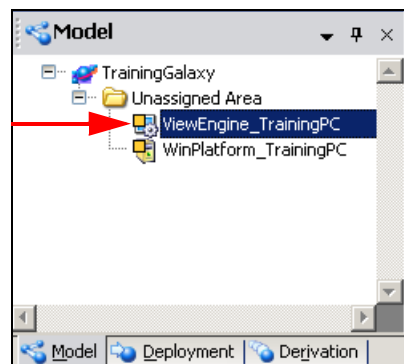


16. In the pane directly below the **Template Toolbox**, click the **Model** tab, and then rename the new instance **WinPlatform\_TrainingPC**.

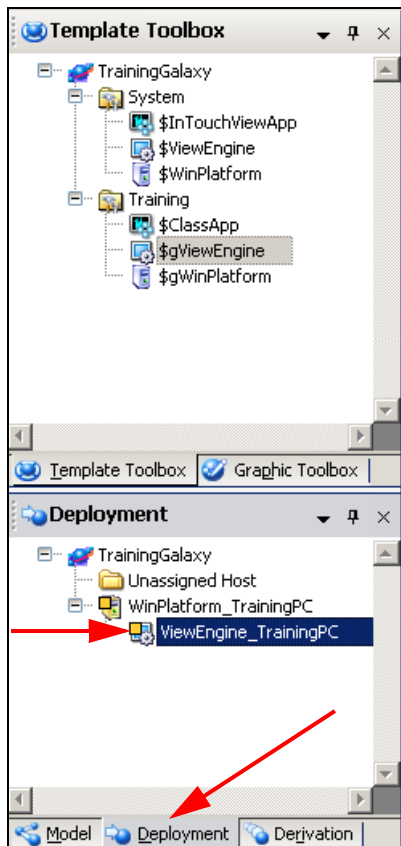
**Note:** **TrainingPC** is used in these steps; you will enter the local node provided by your instructor.



17. In **Template Toolbox**, right-click **\$gViewEngine** and select **New | Instance**.
18. In the **Model** view, rename the new instance **ViewEngine\_TrainingPC**.



19. Click the **Deployment** tab, and then drag **ViewEngine\_TrainingPC** to **WinPlatform\_TrainingPC**.

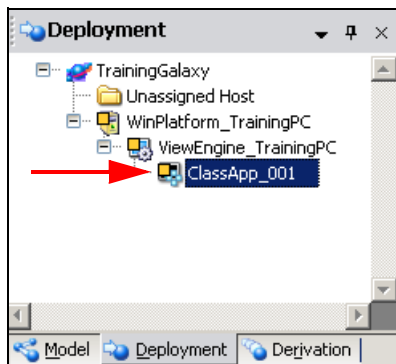


---

**Note:** This will assign **ViewEngine\_TrainingPC** to the **WinPlatform\_TrainingPC** and ensure that the **ViewEngine** that will be running the InTouch application is hosted on the appropriate **WinPlatform**.

---

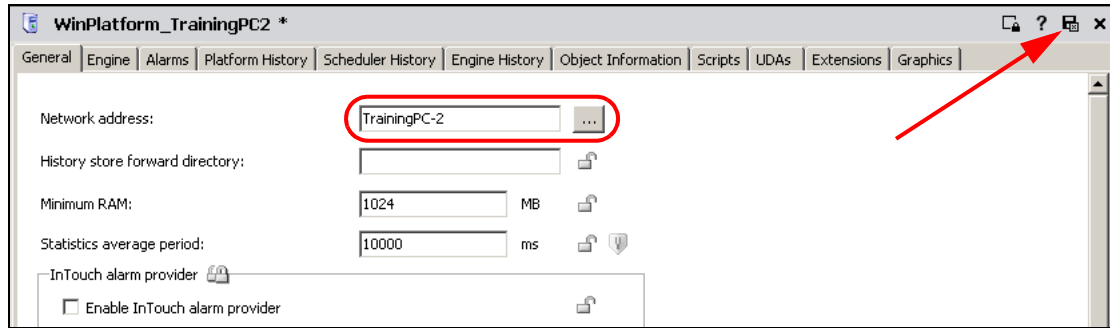
20. In **Template Toolbox**, right-click **\$ClassApp** and select **New | Instance**.
21. In the **Deployment** view, accept the default name of **ClassApp\_001**, and then drag it to **ViewEngine\_TrainingPC** to assign it.





## Remote Node

22. Right-click **\$gWinPlatform** and select **New | Instance**.
23. In the **Deployment** view, rename the new instance **WinPlatform\_TrainingPC2**.
24. Double-click to open it for configuration.
25. In the **General** tab, enter the remote network address provided by your instructor, and then in the top-right of the window, click the **Save and Close** button.

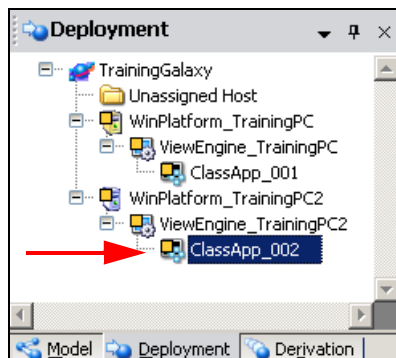


**Note:** TrainingPC-2 is used in this example but this will vary by location.

26. Check in the object.
27. In **Template Toolbox**, right-click **\$gViewEngine** and select **New | Instance**.
28. In the **Deployment** view, rename the new instance **ViewEngine\_TrainingPC2**.

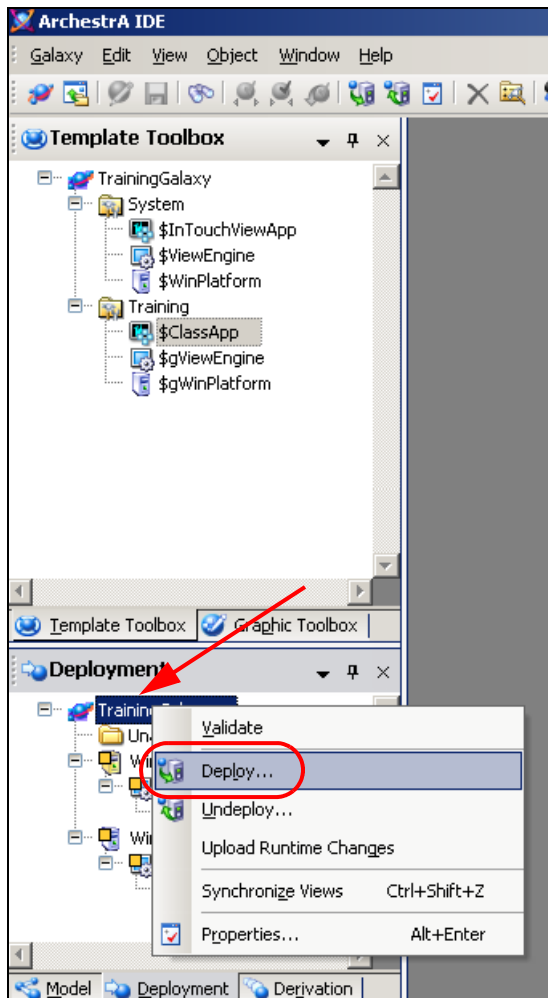
**Note:** TrainingPC2 is used for this example; you will enter the remote node provided by your instructor.

29. Drag **ViewEngine\_TrainingPC2** to **WinPlatform\_TrainingPC2** to assign it to the appropriate **WinPlatform**.
30. In **Template Toolbox**, right-click **\$ClassApp** and select **New | Instance**.
31. Accept the default name of **ClassApp\_002**, and then drag it to **ViewEngine\_TrainingPC2** to assign it.

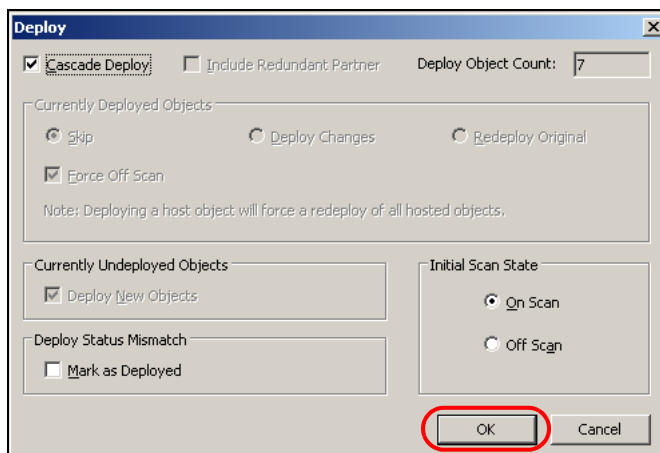


## Deploy the Galaxy

32. In the **Deployment** view, right-click on **TrainingGalaxy** and select **Deploy**.



33. Click **OK**.



34. Click **Close**.

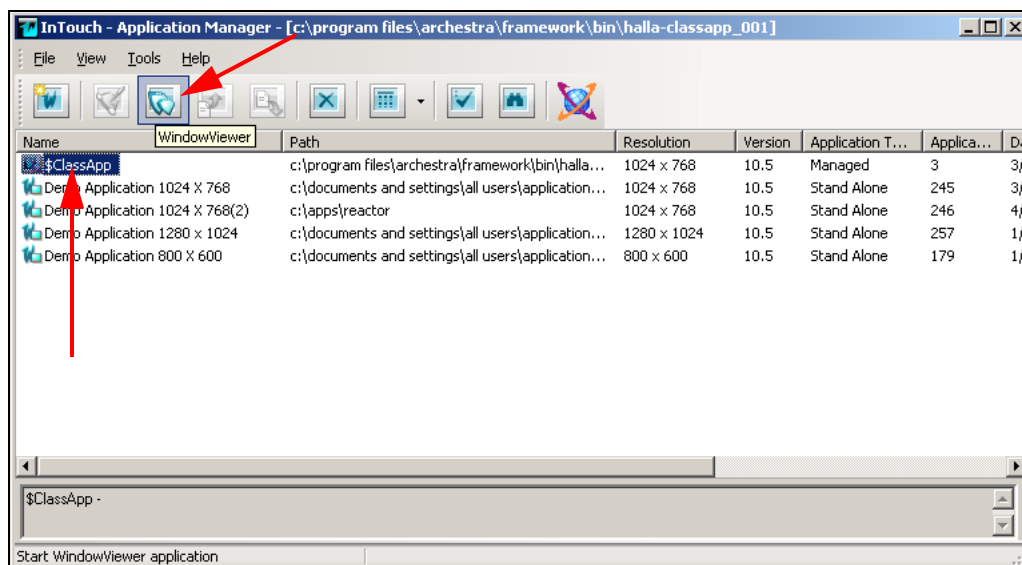
## Connect to the Remote Node

In this section of the lab, you will connect to the remote node where you have deployed your application.

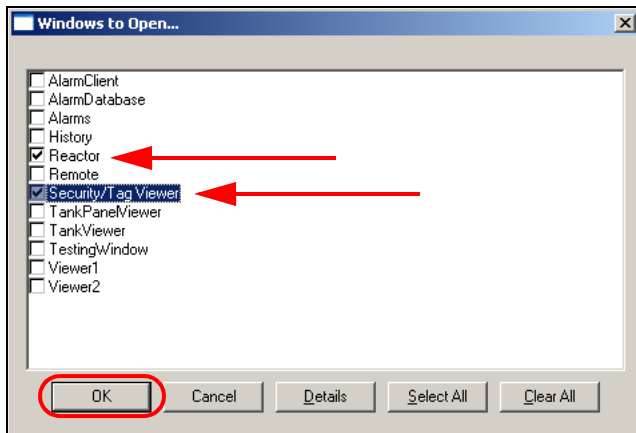
35. Connect to the remote node using the steps provided by your instructor.
36. Click **Start | All Programs | Wonderware | InTouch**.



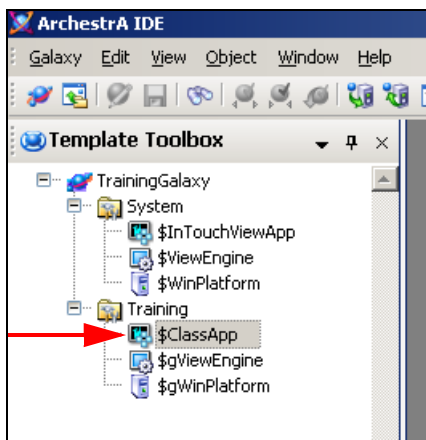
37. Select **\$ClassApp**, and then on the toolbar click **WindowViewer**.



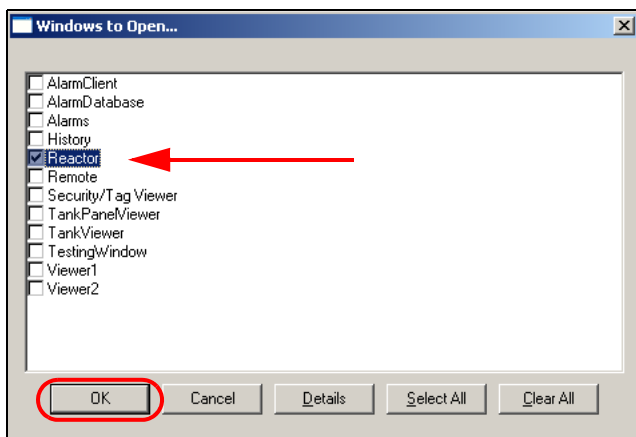
38. Select the **Reactor** and **Security/Tag Viewer** windows, and then click **OK**.



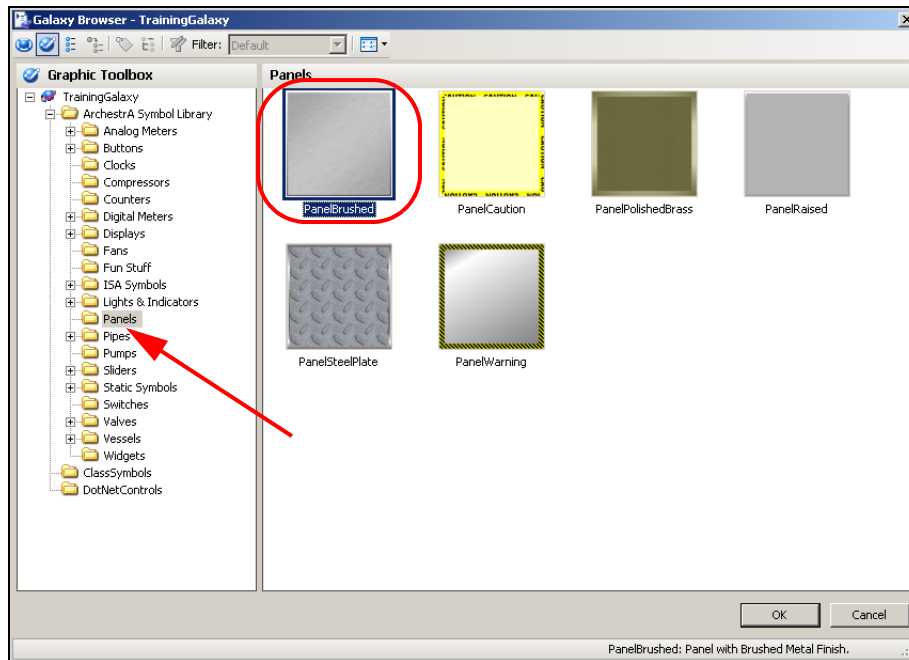
39. Select any of the tanks to see the data display.  
40. Switch back to your primary development node.  
41. In Archestra IDE, in **Template Toolbox**, double-click **\$ClassApp** to launch WindowMaker.



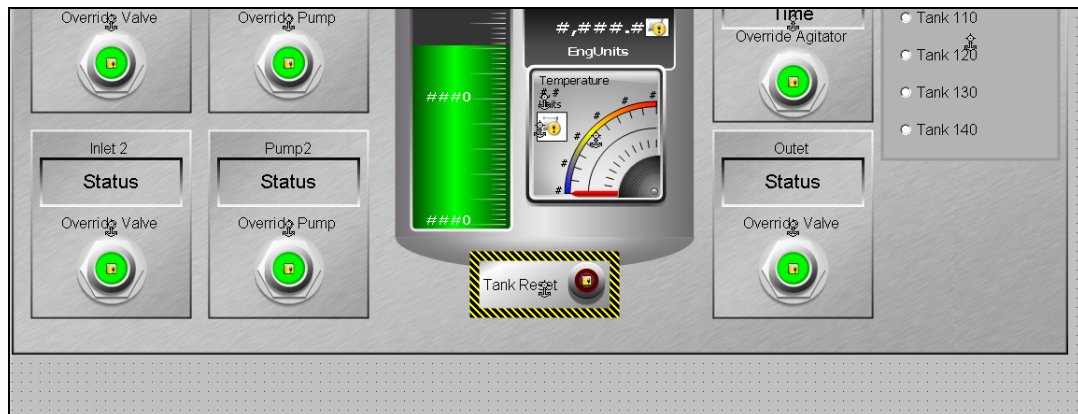
42. Select the **Reactor** window, and then click **OK**.



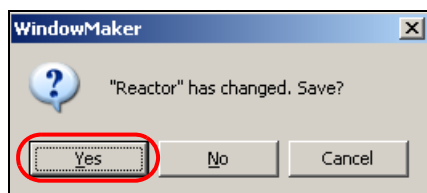
43. Click **Embed ArchestrA Graphic**.
44. Select **ArchestrA Symbol Library\Panels**, and then double-click **PanelBrushed**.



45. Place it in the **Reactor** window so that it provides a background for all of the objects in the window. You may need to switch to full screen mode to accomplish this.

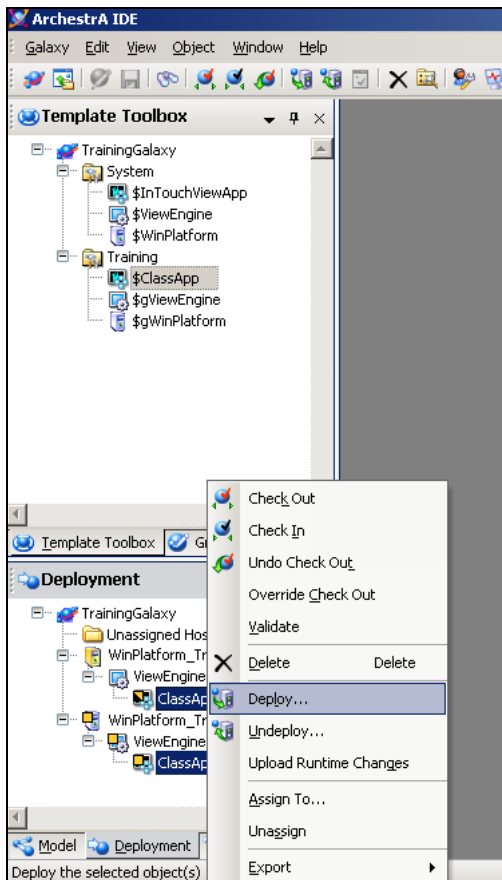


46. Exit WindowMaker, and then click **Yes** when promoted to **Save**.

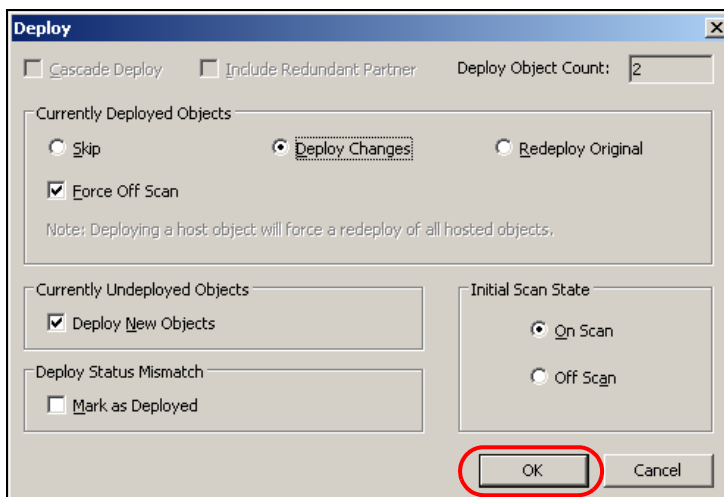


47. Click **OK** to check in.
48. Click **Close**.

49. In the **Deployment** view, hold down the **Ctrl** key to select both **ClassApp\_001** and **ClassApp\_02**, and then right-click and select **Deploy**.

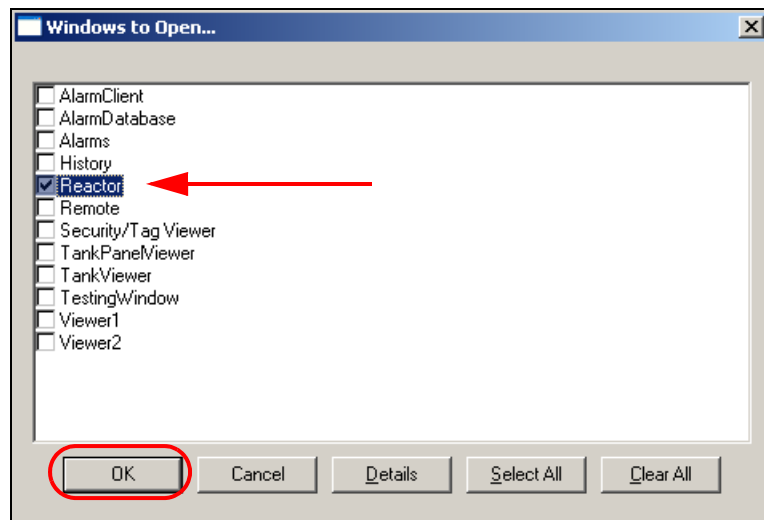


50. Click **OK**.

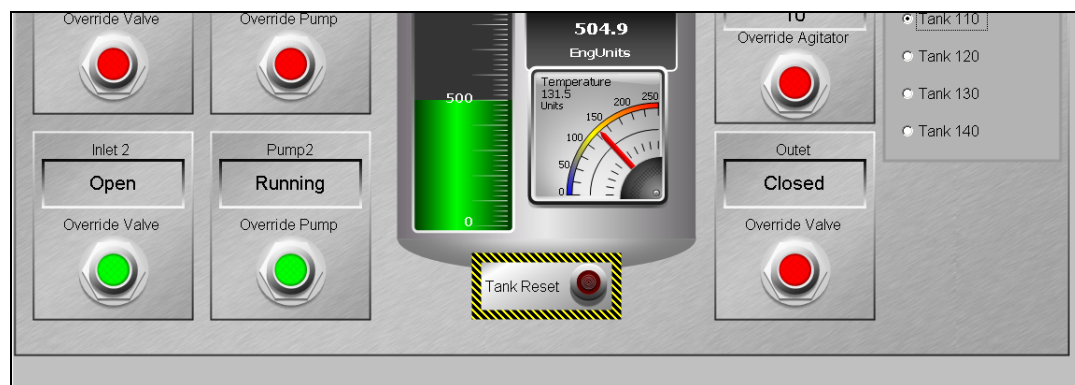


51. Click **Close** when the deployment finishes.
52. Switch back to the remote node, and then click **Yes** when prompted to restart WindowViewer.

53. Select the **Reactor** window, and then click **OK**.



54. The **PanelBrushed** element will now show in the **Reactor** window, select any of the tanks to see the data display.







## Section 2 – Terminal Services

This section discusses how InTouch is used in a Terminal Services environment and in a Virtualization environment.

### Windows Server 2008 R2 Hyper-V Feature

A virtualized environment can run multiple virtual machines (VMs) on a single server, thereby reducing the number of physical servers required on the network. Hyper-V provides a virtualized computing environment on Windows Server 2008 R2. Hyper-V is a hardware-assisted virtualization platform that uses partitions to host VMs. One of the benefits that Hyper-V provides is isolation, which ensures that the child VMs execute in their individual partitions and exist on the host as separate machines. This allows multiple operating systems and conflicting applications to run on the same server. Windows Server 2008 R2 Hyper-V provides support for using Virtual LANs (VLANs) on both parent and child partitions. By configuring VLAN, VMs can communicate over the specified VLAN using Virtual Network switch.

Microsoft introduced RemoteApp with the release of Windows 2008 Terminal Services. In the past, Windows 2008 TS Microsoft Terminal Services solutions only supported the publication of a full desktop using the RDP protocol. In Windows 2008, it was possible to start an application seamlessly from a Terminal Server, making it appear as it were running locally on the client machine. RemoteApp is an application that runs from a Terminal Server running seamlessly to the client.

### Understanding Virtualization

Virtualization is the creation of an abstracted or simulated - virtual, rather than actual version of something, such as an operating system, server, network resource, or storage device. Virtualization technology abstracts the hardware from the software, extending the life cycle of a software platform.

In virtualization, a single piece of hardware, such as a server, hosts and coordinates multiple guest operating systems. No guest operating system is aware that it is sharing resources and running on a layer of virtualization software rather than directly on the host hardware. Each guest operating system appears as a complete, hardware-based OS to the applications running on it.

### Terminal Services Benefits

**Bringing Windows Based Operating Systems to desktops faster:** For example, Terminal Services helps bridge the gap while older desktops are migrated to Windows XP Professional, providing the Windows XP desktop experience "virtually" to non-computer desktops and computers that require hardware upgrades to run a full Windows XP operating system locally. Terminal Services clients are available for many different desktop platforms including MS-DOS, Windows-based Terminals, Macintosh, and UNIX. (Connectivity to MS-DOS, Macintosh, and UNIX-based computers requires additional software).

**Takes full advantage of existing hardware:** Terminal Services extends the model of distributed computing by allowing computers to operate as both thin clients and full-featured personal computers simultaneously. Computers can continue to be used as they have been within existing networks, while also functioning as thin clients capable of emulating the Windows XP Professional desktop.

**Remote administration and support:** Terminal Services provides remote administration for the Windows 2003 Server, giving system administrators a method of remotely managing their server from any client over a WAN or dial-up connection. Administrators can remotely manage Windows 2003 servers from a single desktop. Administrators have access to system management tools and can perform all administrative tasks, including software installation, as if they were performing them locally at the server.

**Replace text-based terminals:** Because many Windows-based terminals also natively support terminal emulation on the same device, replace text-based terminals with Windows-based terminals. Windows-based terminals enable working with data from mainframe systems to have access to newer Windows-based applications, such as email.

**Increased Security and Reliability:** Multiple encryption levels enable administrators to encrypt all or some of the data transmitted between the Windows 2003 Server and Terminal Services clients at three different levels (low, medium, or high), depending upon security needs. In addition, the Terminal Services logon process includes change password, unlock desktop, and unlock screen saver features. The logon process is encrypted, ensuring secure transfer of user name and password. Terminal Services supports both 40-bit and 128-bit encryption (available only in the U.S. and Canada) between server and client.

Because no application or user data ever resides on the client, Terminal Services provides more control for security. Terminal Services also provides multi-level encryption support, which can be enabled whenever there is a risk of unauthorized transmission interception on the link between the server and the client. There are three levels of encryption available: low, medium and high. All levels of encryption use the standard RSA RC4 encryption.

The use of thin clients can help prevent the loss of data from computer failure. Since the data is processed and stored on the Terminal server, damage to the client does not lead to destruction of data. This decreases the number of nodes that need to be hardened for data protection.

For more information, refer to the Microsoft Windows 2003 Server Administrator's Companion.

**Remote Control:** The Remote Control feature allows an Administrator to temporarily control another user's session and see the user's actions. The administrator can also interact with the user and execute commands on behalf of the user. To take advantage of Remote Control using RDP, both clients must be connect to a Windows 2003 Terminal Server.

---

**Note:** Before enabling remote control of a session, inform the user by displaying a message on the client machine and requiring client permission to view or take part in the session. The settings of the remote control can be set so that control can take place without user acceptance. The mouse and keyboard inputs can be enabled or turned off.

---

**Centralized deployment of programs:** With Terminal Services running on a Windows 2003 Server, all program execution, data processing, and data storage occur on the server, centralizing the deployment of programs. Terminal Services ensures that all clients can access current versions of a program. Software is installed only once on the server, rather than every desktop throughout the organization, reducing the costs associated with updating individual computers.

**Centralized Management:** Windows Terminal Services provides the ability to manage centrally, while still allowing the individual user the flexibility of using the Windows desktop environment. Since all applications in a Windows Terminal Services session reside only in one place on the server, all upgrades of processing power and systems software versions are conducted at the server level rather than at the desktop level, thus eliminating time-consuming field upgrades. There is no client application software to develop, install and update. Therefore, installation and related soft costs are significantly reduced. User profiles can be stored on the Terminal Server so client desktops are administered centrally, thus improving applications management efficiency (centralized backup and restore, revision level upgrades, and so on).

In addition, Windows Terminal Services allows an administrator to view what is happening in a user's session, or even to directly control it. Support personnel can actually see exactly what the user is seeing without leaving their desks. If the user is configured accordingly, the support person can share control of the session, walking the user through a difficult problem. Resulting in lowered maintenance costs because of less time spent trouble-shooting; decreased threat of local viruses, software upgrades are easier and failed platforms can be easily swapped with no loss of data.

When configured in remote administration mode, Windows Terminal Services can also be used as a management tool. Administrators can log on directly to the server from their desktop and perform normal system maintenance without having to sit at the server console.

By implementing Windows Terminal Services, three components that traditionally reside at the user's desktop are centralized:

- Application support and maintenance
- Maintenance of the operating system
- Storage management

## Why Terminal Services for InTouch?

Terminal Services for InTouch allows InTouch to run in a multi-user environment. For organizations wanting to increase flexibility in process visualization and to control operator workstation management costs, a Terminal Services for InTouch architecture offers an important enhancement to the traditional two or three tier client-server architecture.

## Benefits of Terminal Services for InTouch

Beyond cost and scalability improvements, Terminal Services for InTouch also provides many technological advantages. For example, one can remotely control an InTouch application for quick troubleshooting and operator training. Using Microsoft's new Terminal Services Advanced Client (TSAC), view the process over the web for a super-thin client, full InTouch experience. Also roaming operators with real-time information and control by using wireless Ethernet is provided.

---

**Note:** Wonderware does not currently support Microsoft's Terminal Services Advanced Client (TSAC). It is strongly recommended to limit the ability to remotely control industrial applications through the Internet.

---

Using Terminal Services for InTouch provides a full desktop experience on hardware that would otherwise be unable to support such operating systems. Embedded clients are generally dedicated purpose devices. Due to InTouch licensing and hardware requirements, full-featured HMI functionality has not been available for embedded-type applications – until now. Terminal Services for InTouch fully supports very thin hardware – hardware with less components than a desktop computer. Not only are these clients less likely to fail, but they can be replaced in less than 60 seconds, reducing the overall MTTR (mean time to repair).

---

**Note:** Terminal Services scalability does not consider the impact on the control network. Data fan out can occur when InTouch sessions exceed the number of topics/update rates that SuiteLink or the I/O devices can support. For a more scalable solution, consider **Wonderware Information Server**.

---

## Managed InTouch Application With Network Load Balancing

The features provided by Remote Desktop are made available through the Remote Desktop Protocol (RDP). RDP is a presentation protocol that allows a Windows-based terminal (WBT), or other Windows-based clients, to communicate with a Windows-based Terminal Server. RDP is designed to provide remote display and input capabilities over network connections for Windows-based applications running on a Windows XP Professional desktop.

In this topology, clients can access the InTouch System Platform node via Remote Desktop. Whenever a new connection is requested to the InTouch System Platform Node, a new session is created. So all the traffic goes to the system platform node and degrades the performance of the InTouch node.

| Virtual Machine 3: InTouch TS node |                                          |
|------------------------------------|------------------------------------------|
| Processor                          | Host Compatible Processor with 2-4 Cores |
| Operating System                   | Windows Server 2008 R2 Standard          |
| Memory                             | 4 GB                                     |
| Storage                            | 80 GB                                    |
| System Platform Products Installed | InTouch with TS enabled                  |

## Data Access Server

In case of Live and Quick migration of an I/O Server node (for example, DASSIDirect), InTouch I/O Tags acquiring data from that I/O server need to be reinitialized after the I/O server node is migrated. To automatically acquire the data for these tags from the I/O server after migration, it is recommended to have an InTouch script which monitors the quality status of any of those tags and triggers reinitialize I/O once the quality goes to bad. Execute this script every 3 to 5 seconds until the tag quality becomes good.

## Start Up Programs

Ensure that InTouch Window Viewer is added to the Start Up programs so that the view is started automatically when the virtual machine reboots.

## Network Adapters

Each System Platform node can have various combinations of the following network adapters, depending on the configuration:

---

**Note:** It is assumed that the host virtualization server is configured with one external virtual network switch to communicate with the domain network, one external virtual network switch to communicate with the plant network, and one internal virtual network switch for the internal VM to VM communication.

---

---

## InTouch Network Adapters

- An external network adapter to communicate with the external domain network
- An external network adapter to communicate with the external plant network (This is to acquire the data from the IO Server which is connected to the plant network)
- An internal network adapter to communicate between the other VM nodes configured on a Hyper-V host server (For example, to a Historian VM node)

## Network Load Balancing Observations

The Network Load Balancing (NLB) feature is qualified for InTouch managed application. InTouch TSE license is required on each of the NLB cluster nodes.

Local InTouch Tag Alarms are local to the session. Local InTouch Tag alarms updated in a session remain local to that session only.

ArchestrA Alarms are common across all sessions. ArchestrA Alarms updated in one of the sessions get reflected across all the sessions.

For I/O tags pinging in one session, the data reflects across all the sessions. However, while pinging local InTouch tags, data does not get updated across all sessions, since it is local to the session.

When the NLB cluster node is lost with the active sessions, all the active sessions on the NLB cluster node closes. To retain all the active sessions, configure the NLB Cluster in a Microsoft Failover Cluster in a Hyper-V environment. The NLB cluster nodes are VM nodes hosted by Hyper-V host servers and Hyper-V host.

