INVENSYS LEARNING SERVICES

Training Manual

Revision A
May 2013
Part Number 11-HM-10011

InTouch 2012 R2 – Part 1

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, SCADAlarm, 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 and ArchestrA IDE Overview 1-17
Lab 1 – Setting Up the Galaxy 1-25
Lab 2 – Backing Up and Restoring a Managed InTouch Application 1-37
Section 3 – System Requirements and Licensing 1-47

Module 2 Using WindowMaker and WindowViewer 2-1
Section 1 – Using WindowMaker 2-3
Section 2 – Using WindowViewer 2-11
Section 3 – Data Content Portal 2-15
Section 4 – Window Properties and Behaviors 2-21
Lab 3 – Data Portal Application 2-23
Section 5 – Tagname Dictionary and Data Sources 2-43
Section 6 – Tag Viewer 2-55
Lab 4 – Creating and Monitoring Tags 2-63
Section 7 – Large-Scale Tag Manipulation 2-85
Lab 5 – Creating Tags on a Large Scale 2-91
Section 8 – Finding and Deleting Tags 2-103

Module 3 Data Presentation 3-1
Section 1 – Working with ArchestrA Symbols 3-3
Lab 6 – Working with ArchestrA Symbols 3-11
Section 2 – Using the ArchestrA Symbol Editor 3-37
Lab 7 – Modifying Your ArchestrA Symbol 3-47
Section 3 – ArchestrA Symbol Custom Properties 3-71
Section 4 – ArchestrA Symbol Animation 3-75
Lab 8 – Animating Symbols 3-95
Section 5 – ArchestrA Symbol Status and Quality 3-115
Lab 9 – Monitoring Custom Symbol Status and Quality 3-119

Module 4 Alarms and Events 4-1
Section 1 – Enabling and Configuring Alarms 4-3
Lab 10 – Configuring Alarms for Discrete and Analog Tags 4-7
Section 2 – Alarm Client 4-29
Lab 11 – Implementing an Alarm Client 4-31
Section 3 – Alarm DB Logger 4-47
Lab 12 – Viewing Alarm History with the Alarm DB Logger 4-51
Section 4 – Alarm Printing 4-67

Module 5 Data Tracking and Trending 5-1
Section 1 – Real-Time Trend 5-3
Lab 13 – Building a Real-time Trend 5-7
Section 2 – Historical Trend 5-17
Lab 14 – Examining Data Retrieved from InTouch History 5-29

Module 6 Application Security 6-1
Section 1 – InTouch Security 6-3
Lab 15 – Securing Your Application 6-9
Lab 16 – Managing Automatic Logout 6-29

Module 7 Enhancing Your Application with QuickScript 7-1
Section 1 – Introduction to QuickScript 7-3
Section 2 – Enhancing Security with QuickScript 7-11
Lab 17 – Creating a Log Off Button 7-23
Lab 18 – Synchronizing the Navigation Tabs with Current Window 7-29
Lab 19 – Restricting User Access to Windows 7-37
Lab 20 – Defining Special User Access 7-53
Section 3 – Enhancing History with QuickScript 7-59
Lab 21 – Creating a Live Mode for the Historical Trend 7-67
Lab 22 – Changing Historical Pens at Runtime 7-75
Section 4 – Enhancing Alarms with ArchestrA Symbol Scripting 7-93
Section 5 – Enhancing Symbols with ArchestrA Symbol Scripting 7-95
Section 6 – Troubleshooting QuickScript 7-99
Lab 23 – Enhancing Alarm Client with Scripting 7-103
Lab 24 – Managing Limited Screen Real Estate 7-117

Module 8 Application Distribution 8-1
Section 1 – Application Deployment 8-3
Lab 25 – Deploying the Application 8-5
Section 2 – Application Publishing 8-23

Module 9 Putting It All Together 9-1
Section 1 – Building a New Process 9-3
Lab 26 – Putting It All Together 9-5



Learning Services

Module 1 – Introduction

Section 1 – Course Introduction 1-3

Section 2 – InTouch and ArchestrA IDE Overview 1-17

Lab 1 – Setting Up the Galaxy 1-25

Lab 2 – Backing Up and Restoring a Managed InTouch Application 1-37

Section 3 – System Requirements and Licensing 1-47

Module Objectives

- Establish course agenda
- List the system requirements and licensing
- Summarize Wonderware software solutions
- Introduce key features of InTouch HMI

Section 1 – Course Introduction

This section will familiarize you with the objectives and agenda for this course. It also outlines Wonderware products and how InTouch meets the human-machine interface requirements for visualization.

Course Description

The InTouch 2012 R2 - Part 1 course is a 5-day, instructor-led class focused on Managed InTouch Application design. The course provides a fundamental understanding of the basic principles of the Wonderware visualization module and the steps necessary to develop a Human Machine Interface (HMI) system for your specific plant floor. Lectures and hands-on labs guide you through setup, layout, best practice concepts, features, and functions of the InTouch Software platform. Hands-on labs reinforce concepts and features.

Objectives

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

- Build a tag-based Managed InTouch application
- Use WindowMaker to create and configure windows
- Use Tagname Dictionary to create and Import tags
- Build, customize, embed, and animate symbols
- Create QuickScripts
- Add alarms to an application
- Implement InTouch security in an application
- View Realtime and Historical trends
- Deploy and publish your application
- Implement the key concepts learned in this course

Audience

Plant floor operators and managers, system administrators, system integrators, and other individuals who need to use the InTouch HMI software in manufacturing processes.

Prerequisites

- Familiarity with Microsoft Windows
- Manufacturing industry experience recommended

Course Outline

Module 1 – Introduction

Section 1 – Course Introduction

This section will familiarize you with the objectives and agenda for this course. It also outlines Wonderware products and how InTouch meets the human-machine interface requirements for visualization.

Section 2 – InTouch and ArchestrA IDE Overview

This section introduces the InTouch HMI, the benefits of using InTouch, and its key capabilities. It also provides an overview of its features.

Section 3 – System Requirements and Licensing

This section describes the operating system(s) and other software requirements the host computer must meet prior to installing InTouch 2012 R2.

Module 2 – Using WindowMaker and WindowViewer

Section 1 – Using WindowMaker

This section provides an overview of WindowMaker and how to configure WindowMaker components. It also describes how to design windows and set application preferences.

Section 2 – Using WindowViewer

This section describes your application visualization node and defines WindowViewer components and features.

Section 3 – Data Content Portal

This section explores the fundamentals of designing an InTouch application. You will also learn how to integrate data connectivity into your application, and view the results in WindowViewer.

Section 4 – Window Properties and Behaviors

This section discusses window properties and how they can be configured using the options available in the Window Properties dialog box.

Section 5 – Tagname Dictionary and Data Sources

This section explores tags and the Tagname Dictionary in more detail. It also describes data sources and how they are used, and how to create data Access Names.

Section 6 – Tag Viewer

This section describes the Tag Viewer and how to view, save, and import watch windows. It also discusses Dot Fields and how to interpret value, time, and quality information in a watch window.

Section 7 – Large-Scale Tag Manipulation

This section explores using the DBDump and DBLoad utilities to streamline the development of large amount of tags containing similar functionality.

Section 8 – Finding and Deleting Tags

There are several ways to maintain and edit links and tags. This section discusses the various methods such as the InTouch Cross Reference Utility, updating user counts, and deleting, substituting, and converting tagnames.

Module 3 – Data Presentation

Section 1 – Working with ArchestrA Symbols

This section describes how to embed an ArchestrA graphic in a window in your InTouch application. Then it describes how to link tags and fields to graphics and configure custom properties to streamline how they are viewed in WindowMaker.

Section 2 – Using the ArchestrA Symbol Editor

This section discusses symbol duplication and editing. It also describes the ArchestrA symbol editor features.

Section 3 – ArchestrA Symbol Custom Properties

This section describes custom ArchestrA symbol animation and the flexibility available when using custom properties.

Section 4 – ArchestrA Symbol Animation

This section describes best practices in ArchestrA symbol animation, including the use of $System tags present in all InTouch applications.

Section 5 – ArchestrA Symbol Status and Quality

This section describes methods for monitoring the graphical representation of quality and data status.

Module 4 – Alarms and Events

Section 1 – Enabling and Configuring Alarms

This section describes methods to enable and configure alarms for discrete and analog tags.

Section 2 – Alarm Client

This section describes configuration and implementation methods for the Alarm Client. It also describes Alarm Client features such as right-click and context menu capabilities.

Section 3 – Alarm DB Logger

This section describes the Alarm DB Logger utility and how to configure the Alarm Client to connect to the logger utility.

Section 4 – Alarm Printing

This section describes alarm printer configuration and best practices for printed alarm output.

Module 5 – Data Tracking and Trending

Section 1 – Real-Time Trend

This section describes the real-time trends used to view recent activities. It also discusses strategic placement and memory management of real-time trends.

Section 2 – Historical Trend

This section introduces InTouch historical logging, and explains the historical trend configuration options.

Module 6 – Application Security

Section 1 – InTouch Security

This section provides an overview of ArchestrA, InTouch, and Operating System security models. The InTouch-based security model is then described in greater detail. Additional security options through runtime environment customizations have also been explained here.

Module 7 – Enhancing Your Application with QuickScript

Section 1 – Introduction to QuickScript

This section explains and defines the various InTouch QuickScript types, and features and functions that are unique to a particular QuickScript type. It also describes the features and functions of the QuickScript Editor.

Section 2 – Enhancing Security with QuickScript

This section describes the InTouch Script Editor functionality. It also describes the Animation Links dialog box and Action Script methods you can use to secure your application.

Section 3 – Enhancing History with QuickScript

This section describes history tracking enhancements using scripts, and includes Live HistTrend functionality. This section also describes historical pen functionality used to track components of historical pens, Historical Pen Runtime selection and unassignment, and configuration of multiple scripts using Action Script animation.

Section 4 – Enhancing Alarms with ArchestrA Symbol Scripting

This section describes symbol enhancements with ArchestrA symbol scripts. It also describes the use of comments to provide script line explanations, and how to restrict access to symbol functionality based on user login.

Section 5 – Enhancing Symbols with ArchestrA Symbol Scripting

This section builds upon the Application and Action Script functions from earlier sections, and discusses elements of script to alter height and width properties at runtime.

Section 6 – Troubleshooting QuickScript

This section describes QuickScript troubleshooting using debugging scripts. It also describes various techniques used to enable or disable portions of scripts using flags or comments.

Module 8 – Application Distribution

Section 1 – Application Deployment

This section describes how to deploy your InTouch application for runtime operators.

Section 2 – Application Publishing

This section discusses how to publish your InTouch application for your production environment.

Module 9 – Putting It All Together

Section 1 – Building a New Process

In this section, you review important features of InTouch software covered in this class. You then apply what you have learned as you design, build, and animate a process using InTouch.

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:

- **Framewor**k, 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, unequaled 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 SCADAlarm

SCADAlarm 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 and ArchestrA IDE Overview

This section introduces the InTouch HMI, the benefits of using InTouch, and its key capabilities. It also provides an overview of its features.

Overview

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

Wonderware InTouch provides you with a quick and easy way to create HMI applications for the Microsoft operating systems. InTouch applications are used around the world in the manufacture of all kinds of products including food processing, semiconductors, oil and gas, automotive, chemical, pharmaceutical, pulp and paper, transportation, utilities, and more.

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 your operation brings the right information to the right people at the right time
- History of uninterrupted software version migration path, which means your HMI applications investment is protected

Key Capabilities

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

Managed InTouch Application

The **ArchestrA Integrated Development Environment (IDE)** can be used to create an ArchestrA Galaxy in the Galaxy Repository and to create a managed InTouch application. The ArchestrA IDE can also be used to configure and deploy all necessary objects to support an InTouch runtime workstation. In the IDE, a managed InTouch application is derived from an $InTouchViewApp template object and then started in WindowMaker for creating and editing the application.

When a Managed InTouch Application has been deployed to a runtime workstation where the ArchestrA IDE is not installed, the InTouch Application Manager (InTouch.exe) can be used to start a deployed InTouch application in WindowViewer. The InTouch Application manager cannot be used to edit a managed application.

Throughout this course you will develop an InTouch application, create symbols, and configure graphics and other elements using standard functionality in the ArchestrA IDE, WindowMaker, and WindowViewer.

The following list shows the approach you will use as you develop the InTouch application:

- Start and connect the ArchestrA IDE to your Galaxy
- Create Symbols in the ArchestrA symbol editor
- Configure ArchestrA objects
- Create an InTouch application
- Add tags, scripts, and links
- Embed symbols
- Configure symbols
- Test the application

Key Features and Functionality

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 the Galaxy Repository. The Galaxy represents the system configuration and incorporates a variety of components distributed across multiple computer nodes.

The ArchestrA IDE

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

InTouch includes a diverse set of features and enhancements to existing features. All of these features and enhancements are integrated to meet a defined set of operational goals, including the following:

- A consistent, intuitive development environment that allows non-programmers, as well as software engineers, to build applications quickly and easily
- A set of centralized tools to manage your InTouch applications on a single node or in a complex multi-node environment
- Integrating ArchestrA technology with InTouch to create a runtime environment that can scale from a simple single-node HMI to an enterprise-wide solution using the Wonderware System Platform

- A comprehensive library of pre-built graphical symbols with native properties to consistently handle the visualization and quality of application data
- Graphic tools to easily create symbols with integrated scripting and animation that can be used as standardized objects in multiple applications
- Comprehensive backwards compatibility

Options to Manage InTouch Applications

You can manage your InTouch applications with either the InTouch Application Manager or with the ArchestrA 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 ArchestrA 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

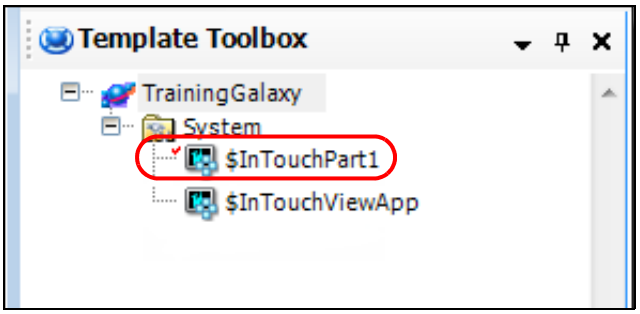
This section details how to start the ArchestrA IDE and create a Galaxy database. It explains the steps needed to create an **InTouchViewApp** template, create a new InTouch Managed application and start InTouch from the IDE.

Note: This course provides only basic concepts and procedures associated with Wonderware® System Platform's ArchestrA IDE interface. To obtain additional information and learn advanced procedures on how to use InTouch and System Platform, it is recommend you attend **Application Server** and **InTouch for System Platform**.

You can use the system platform functionality of the ArchestrA IDE 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
- Edit or add ArchestrA symbols in the InTouch applications

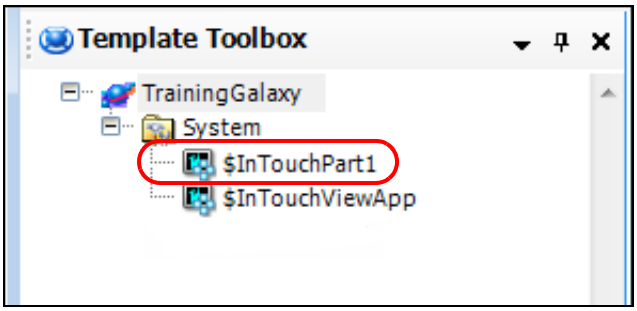
Anytime you open a Managed InTouch application, it is automatically checked out. To ensure the integrity of the application, when an application is checked out, no one else can check it out until you check it back in. Whenever a file is checked out, a check mark is shown next to the application's name, which will disappear when you check it back in.



Note: An InTouch application is automatically checked out at the time InTouch WindowMaker is started. If you close InTouch without making any changes to the application, an **Undo Check Out** is automatically performed.

Right-click the application in the **Template Toolbox** of the ArchestrA IDE, select **Check In** and confirm when prompted.

Notice that the check mark indicating that your application was checked out is no longer visible.



Windows and Graphics

You use WindowMaker to create the visual interface used by operators to view and manage your manufacturing processes. An InTouch interface shows data from and writes data back to the production environment.

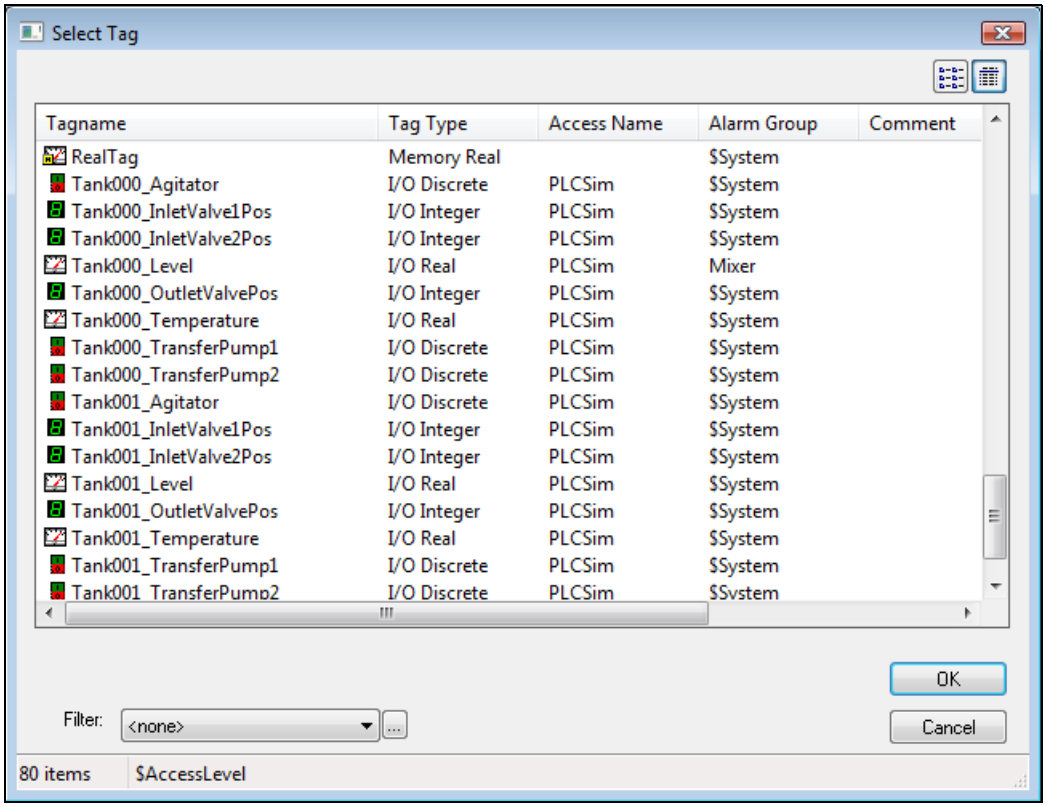
An Application window is a container for one or more graphics that model your production processes. InTouch windows are visual panels that contain the user interface of your InTouch application. You can create replacement windows that close any windows that these windows interact with when they appears on the screen, a popup window that stays on top of all other windows, and an overlay window that appears on top of the window currently open.

You can create a new window using several methods. Additional methods are as follows:

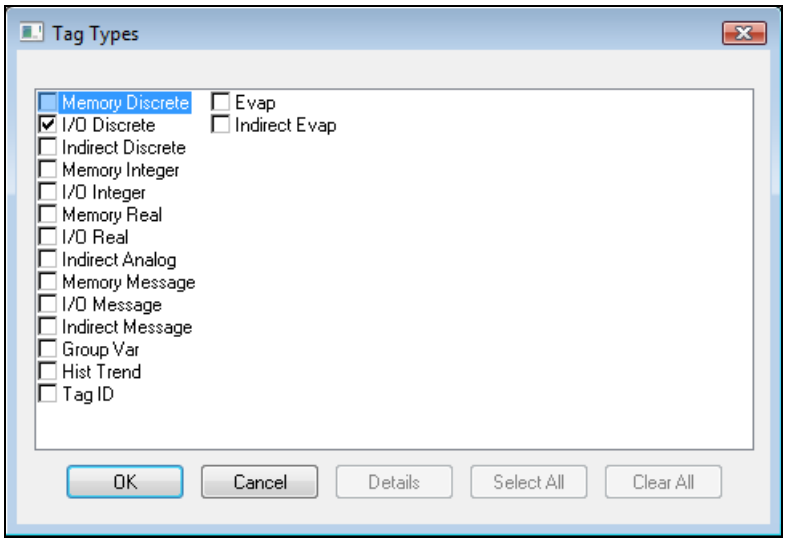
- Click **New Window** on the toolbar 
- In the **Project View**, **Windows & Scripts** pane, **Unassigned** folder, right-click the **Windows** folder and select **New Window**
- Press **Ctrl+N** on the keyboard

Data and InTouch Tags

After you create an InTouch application, you define tags for the application using the **Tagname Dictionary**, which is a WindowMaker tool. You create tags to represent a data item in your InTouch application for all process components whose properties you want to monitor or control. You can also assign additional properties to specify a connection to a remote data source.



You use different types of tags for different types of data collected from a manufacturing component. The InTouch application communicates with the tags placed into runtime memory using animation links or scripts. The InTouch application tracks the current values and other status information from the component properties assigned to tags.



Discrete tags are associated with process component properties whose values are represented by two possible Boolean states.

Indirect tags serve as pointers to other tags. When you equate an indirect tag to another source tag, the indirect tag acts as if it is the source tag. Indirect tags can be equated to integer and real tags. They can also be mapped to any other tag with the same tag type. You can also use scripts to assign input sources to an indirect tag. The script can be triggered by an application event or by operator interaction with the interface.

Integer tags can be assigned 32-bit, signed-integer numbers.

HistTrend tags reference data shown in an InTouch historical trend graph.

Memory tags define internal system constants and variables within the InTouch application. These can also act as calculated variables that are accessed by other programs.

Message tags can be assigned specific text strings.

Real tags can be assigned specific floating decimal point numbers.

Tag ID tags are used when monitoring processes or events. A Tag ID can be assigned to a specific trend pen or change the tag assigned to a trend pen.

A **SuperTag** is a template structure of other tags. The tags that belong to a **SuperTag** template are associated with common properties of a component in a manufacturing process.

You will learn about tags and the **Tagname Dictionary** in greater detail later in this course.

Alarm Subsystem

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.

The distributed alarm system provides the following features:

- Alarm Providers keep track of items that may go into an alarmed condition and provides the Distributed Alarm System with the list of these items, including information on any hierarchical grouping of the items.
- Alarm Consumers provide the Distributed Alarm System with a set of queries identifying alarm-able 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.

Alarm Client

InTouch Alarm Client to subscribe to alarms and events generated by Alarm Providers. In addition, you can configure the InTouch Alarm Client to subscribe to only selected Alarms based on its query filters.

Use the Alarm Client to create an alarm display for real-time visualization and acknowledgment of alarms coming from InTouch, or to enable or disable alarming on an automation object. You also

use the Alarm Client to create an alarm display for visualization of historical alarms and events logged in the alarm database.

Alarm Database Tools

Alarm DB Logger Manager Utility – Alarm DB Logger is an Alarm Consumer. Configure the Alarm DB Logger using an alarm query to define which alarms you want to log. You can also specify alarm queries and log the resulting alarm records. These alarm queries are sent via the Alarm Consumer interface of the Distributed Alarm System. Use the Alarm DB Logger to start and stop logging operations.

Alarm DB Logger Configuration Utility – Use this utility to configure options in the Alarm DB Logger such as database connection type, and performance tuning, and to store the registry setting. You also use the utility to configure a detailed or consolidated logging mode, to create necessary SQL tables in the database, and to specify the alarm query for the logging instance.

Historical Subsystems

The Wonderware Historian is made up of specialized subsystems, which work together to manage data as it is acquired or generated, stored, and retrieved. The following is a list of these specialized subsystems:

Configuration Subsystem – Stores and manages configuration data

Data Acquisition Subsystem – High-speed acquisition of data, allowing Historian to acquire and store process data many times faster than a traditional relational database

Data Storage Subsystem – Stores data for analog, analog summary, discrete, state summary, string, and system tags in sets of files on disk called history blocks

Data Retrieval Subsystem – Receives SQL queries from clients, locates the requested data, performs necessary processing, and then returns the results

Event Subsystem – Detects events and associates actions when they are detected

Replication Subsystem – Data from one Wonderware Historian can be replicated to one or more other Wonderware Historians, creating a tiered relationship between the historians

Trends

InTouch provides two types of trend display objects: Real-time and Historical. Both trend objects can be configured to display graphical representations of multiple tags over time. Both types of trends are created using special tools in WindowMaker. An optional 16-pen trend wizard is available.

Real-time Trend – Real-time trends available from the WindowMaker, Drawing toolset allows you to chart up to four tags. Trend wizards allow you to create real-time trends to show data from up to 16 tags or expressions.

Historical Trend – Historical trends allow you to chart up to eight trends. InTouch also provides you with complete trend configuration control. For example, you can specify the time span, value range, grid resolution, location of time stamps, location of value stamps, number of pens, and color attributes.

Security

InTouch provides three security models: ArchestrA, InTouch, and Operating System security. You can also use security to limit access to other Windows applications or the operating system.

ArchestrA Security uses the security built into the Industrial Application Server (IAS) to secure InTouch Applications, and is centralized regardless of whether NAD is used or not. IAS security activities are configured externally in the Integrated Development Environment (IDE). Please see the *ArchestrA Integrated Development (IDE) Guide* for additional information.

InTouch Security is based on a binary file containing a list of users of the application with passwords and access levels. Use InTouch security to secure the operating system, audit operators' actions, or secure the InTouch application.

Operating System Security uses the Windows Operating Security to secure InTouch Application and allows you to meet security requirements in regulated industries.

Lab 1 – Setting Up the Galaxy

Introduction

In this lab, you will create a Galaxy database within the ArchestrA IDE and derive a WinPlatform template from a base template. You will use the InTouchViewApp object to create a new Managed InTouch application. Then, you will open your Managed InTouch application in WindowMaker and create a new window.

Objectives

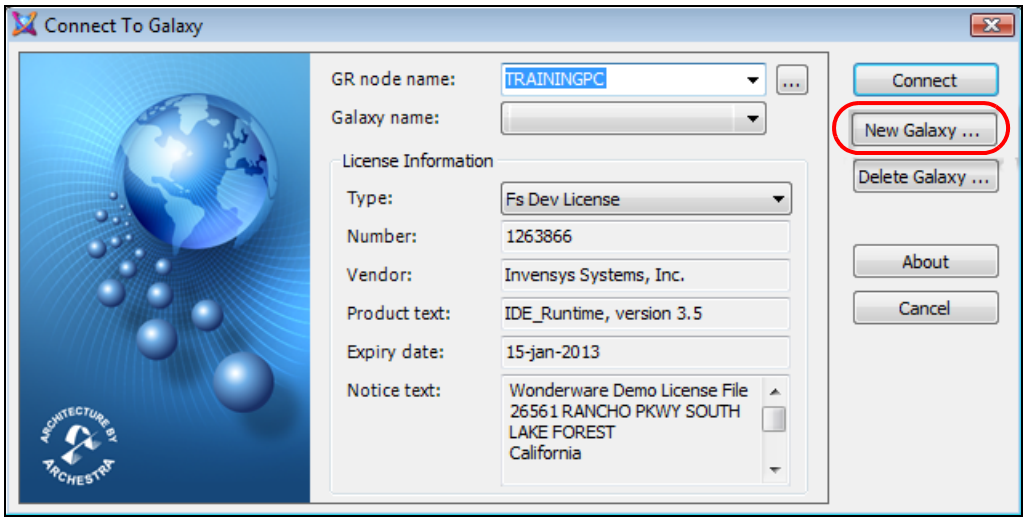
Upon completion of this lab, you will be able to:

- Create a Galaxy
- Create and deploy a WinPlatform
- Create a Managed InTouch application
- Edit a Managed InTouch application

Create a Galaxy

First, you will use the ArchestrA IDE to create a new Galaxy.

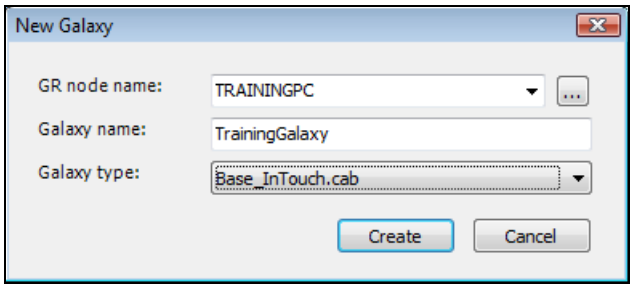
1. Open ArchestrA IDE (**Start | All Programs | Wonderware | ArchestrA IDE**).
2. Click **New Galaxy**.



The **New Galaxy** dialog box appears.

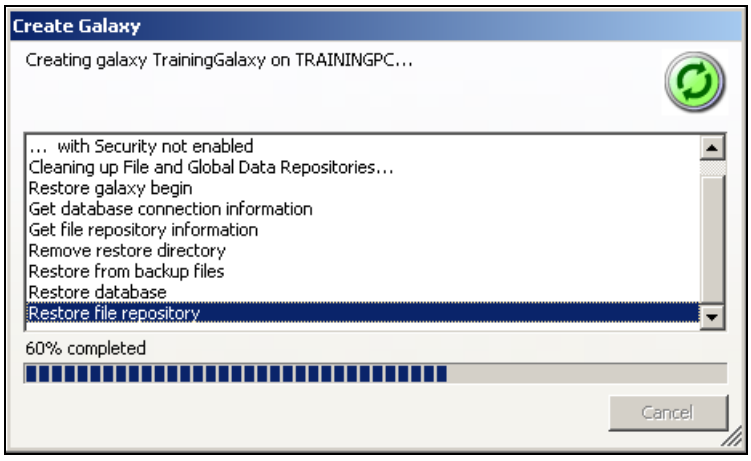
3. Configure the **New Galaxy** dialog box as follows:

GR node name: <your computer name> (provided by your instructor)
Galaxy name: TrainingGalaxy
Galaxy type: Base_InTouch.cab



4. Click **Create**.

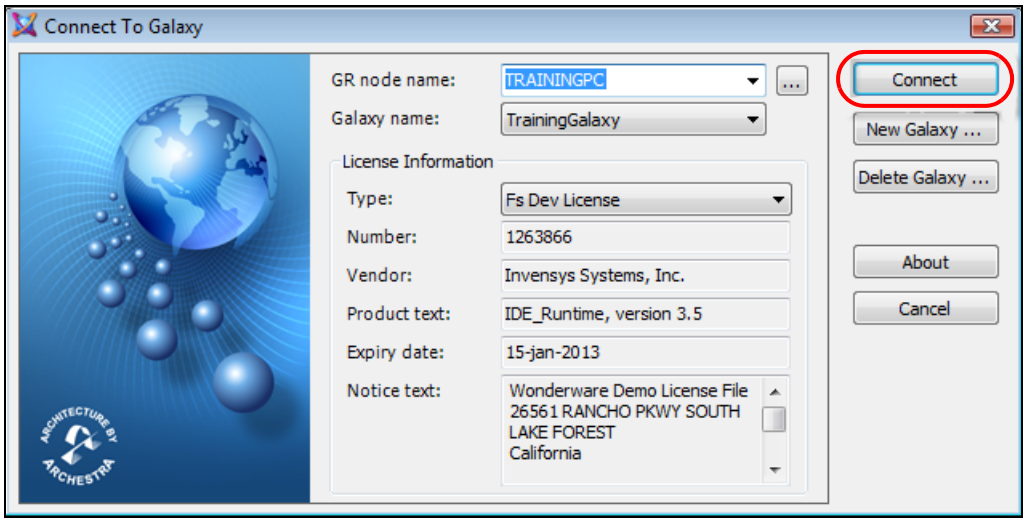
   The **Create Galaxy** dialog box appears.



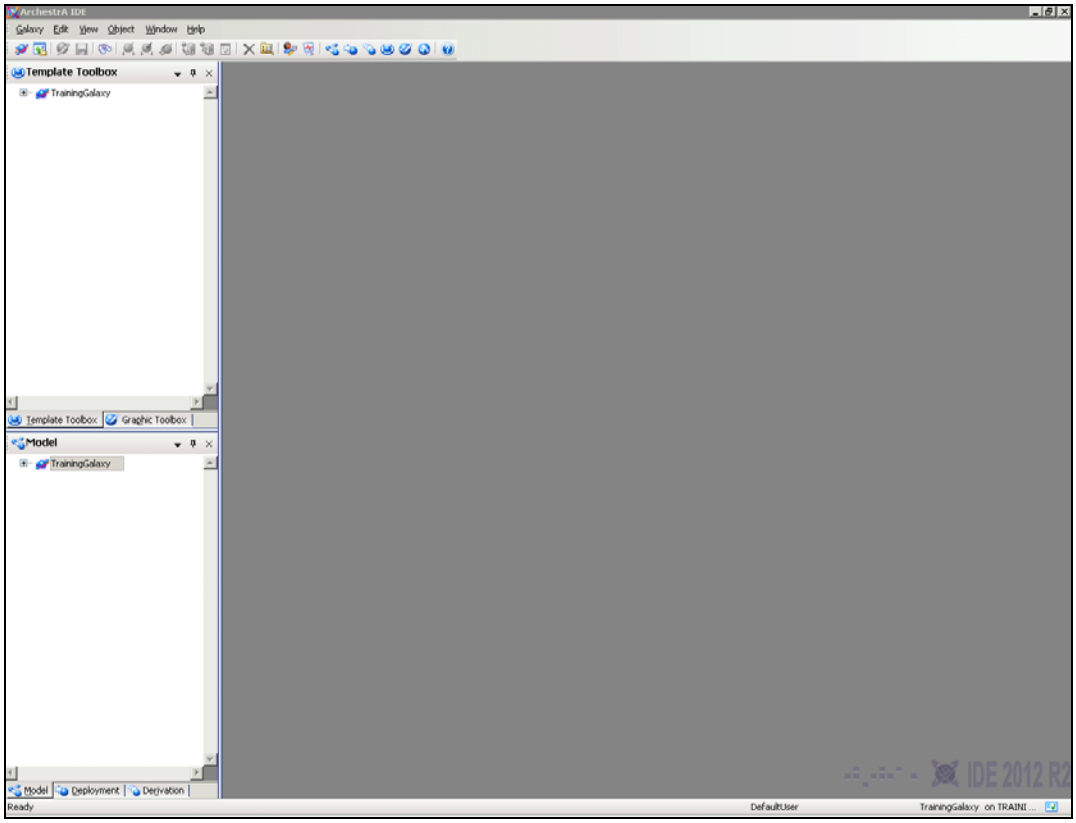
Note: This process may take several minutes to complete. Always monitor and review the displayed list of messages for possible errors. The **100% completed** message at the end of the process does not necessarily mean the process was free of errors.

5. When the bottom progress bar displays **100% completed**, click **Close** to return to the **Connect To Galaxy** dialog box.

6. In the **Connect To Galaxy** dialog box, click **Connect**.



The ArchestrA IDE appears.

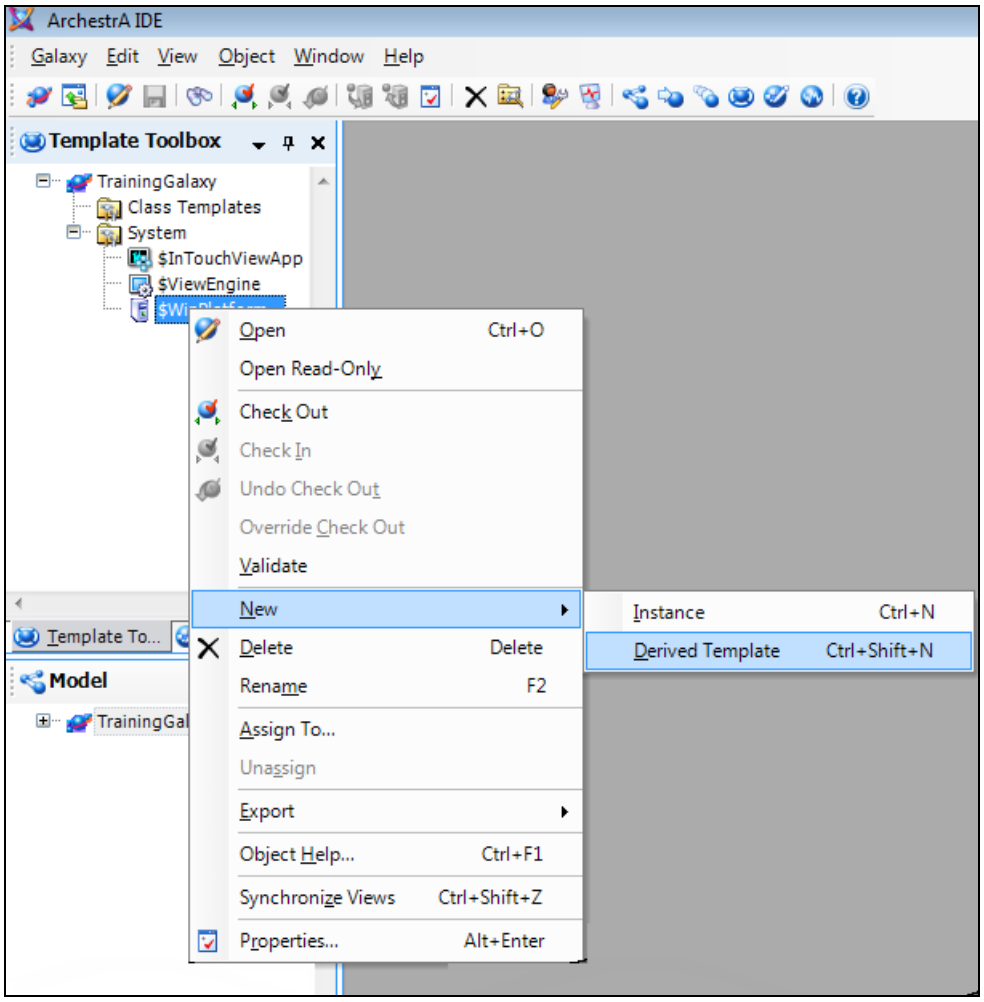


Create a New Derived Template

Next, you will create a template toolset to organize your work. You will derive a $gWinPlatform template from a base template and place it in your new template toolset.

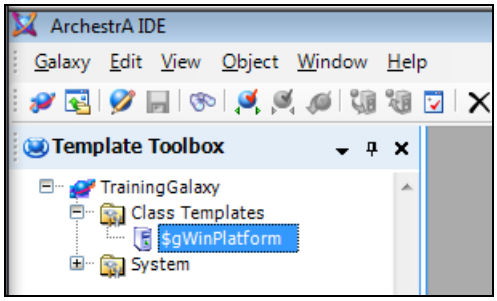
7. In **Template Toolbox**, right-click **TrainingGalaxy** and select **New Template Toolset**.
8. Name the toolset **Class Templates**.

   A new template toolset is created.
9. In the **System** toolset, right-click **$WinPlatform** and select **New | Derived Template**.

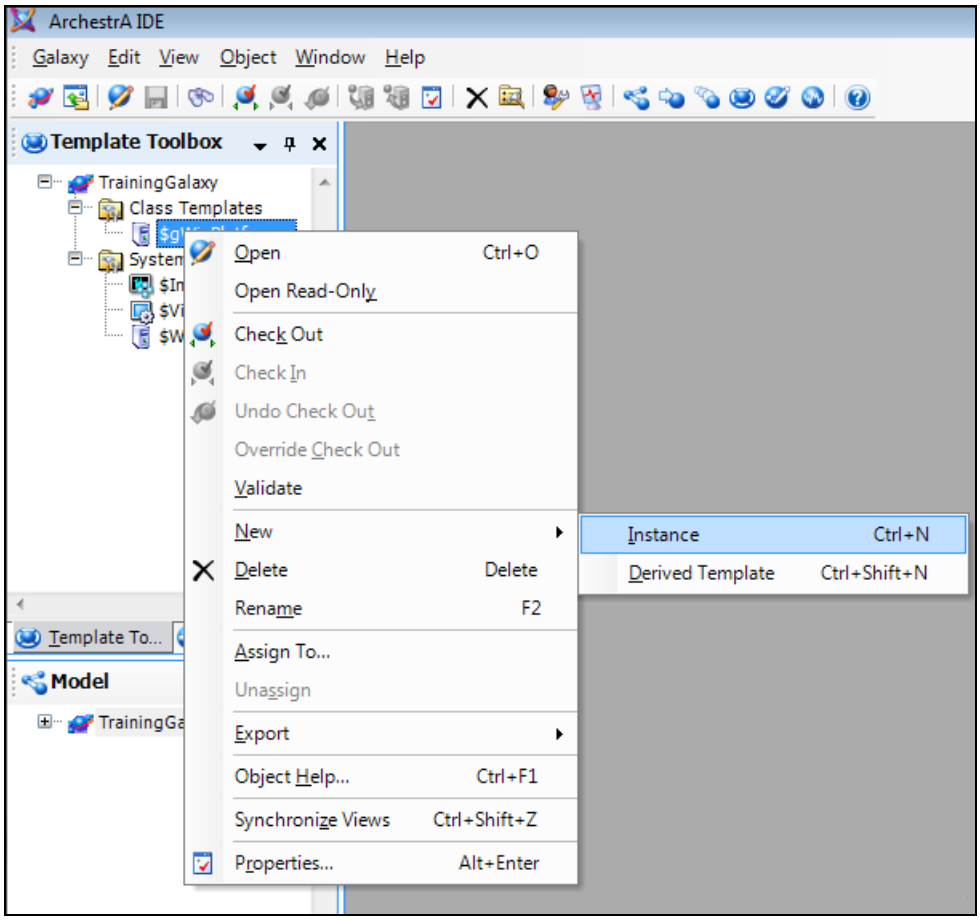


10. Change the name to **$gWinPlatform**.

11. Drag **$gWinPlatform** to the **Class Templates** toolset.

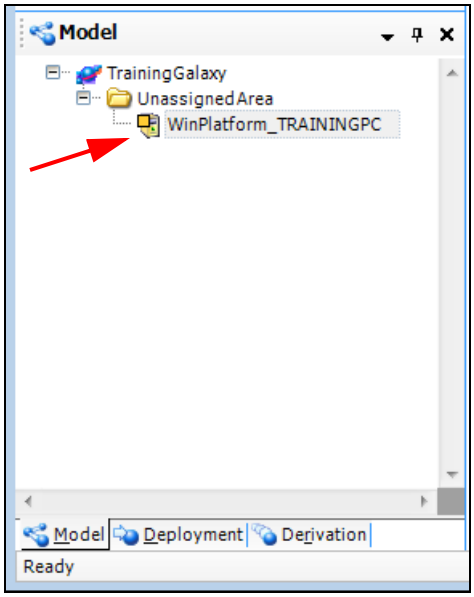


12. Right-click **$gWinPlatform** and select **New | Instance**.



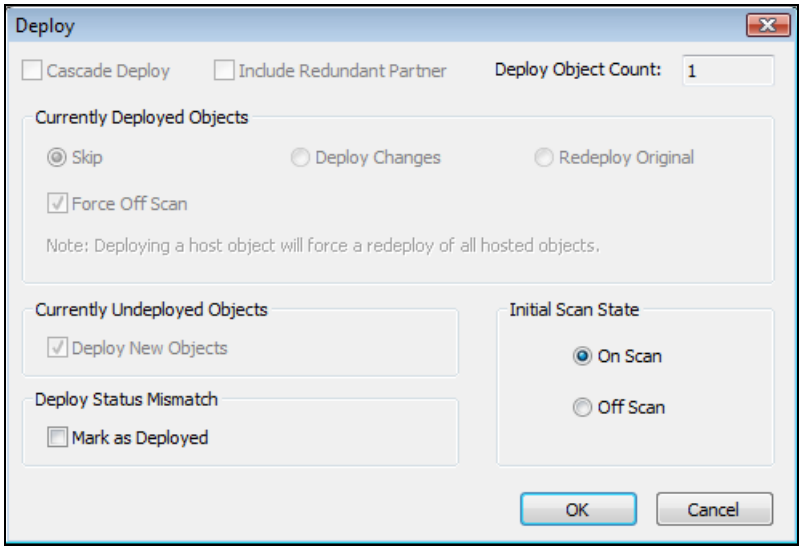
The instance is created in the **Galaxy** and can be seen in the **Model** view.

13. Change the name to **WinPlatform_<your computer name>**.



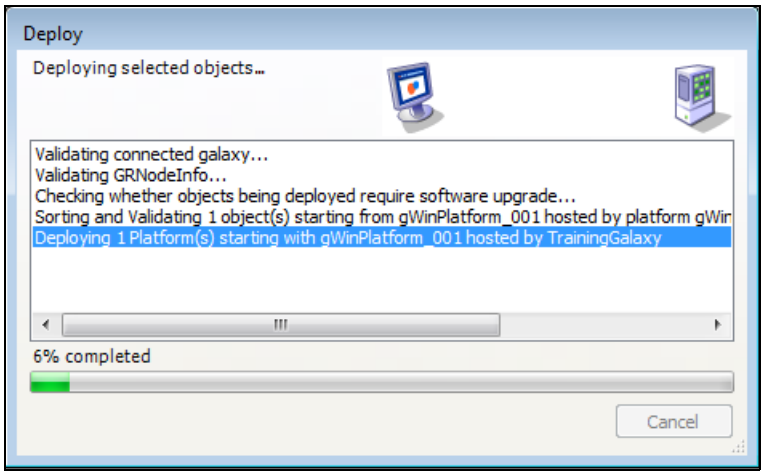
14. Right-click **WinPlatform__<your computer name>** and select **Deploy**.

    The **Deploy** dialog box appears.



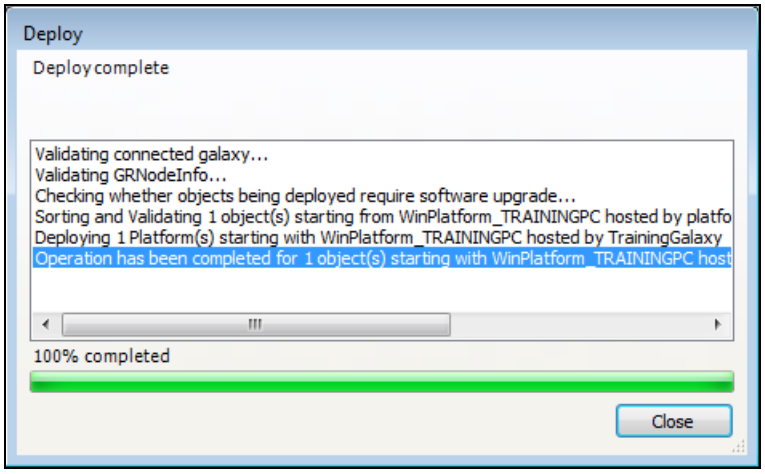
15. Keep all defaults and click **OK**.

    The system will deploy the necessary object.

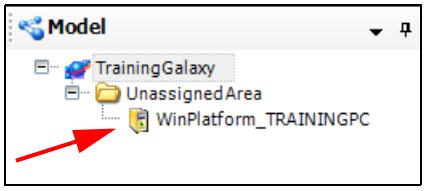


Note: This process may take several moments to complete.

16. After the deployment is complete, click **Close**.



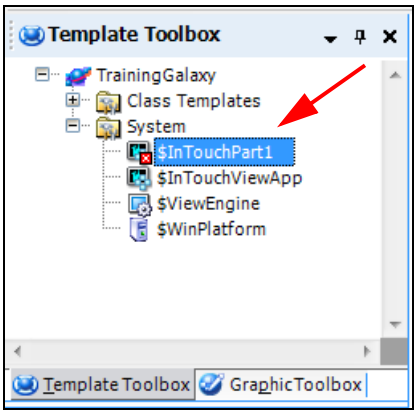
Notice that the icon now indicates the new **WinPlatform** instance is deployed.



Create the InTouch Application

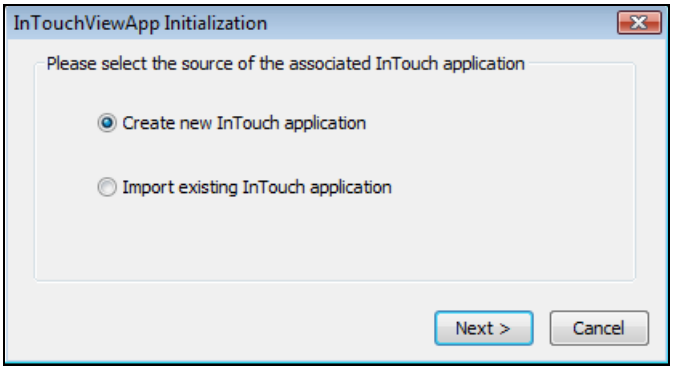
17. In **Template Toolbox**, expand **System** toolset, right-click the **$InTouchViewApp** object and select **New | Derived Template**.

    Note: You can identify a not yet configured **InTouchViewApp** object by the red **X** icon adjacent to the object name.



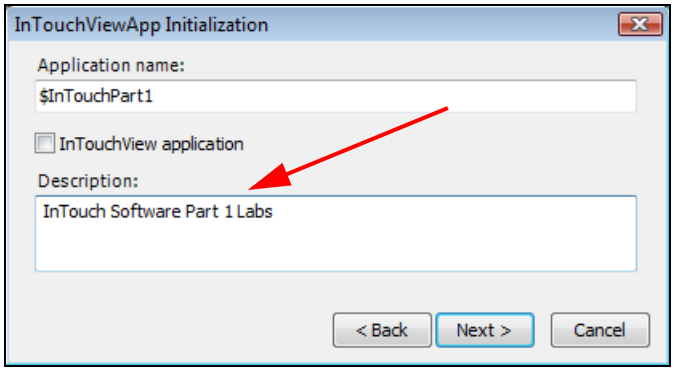
18. Rename the new template **$InTouchPart1**, then drag it to the **Class Templates** toolset.
19. Double-click **$InTouchPart1** to start the configuration wizard.

    The **InTouchViewApp Initialization** dialog box appears.



20. Accept the default setting and click **Next**.

21. In the **Description** field, enter **InTouch Software Part 1 Labs**.



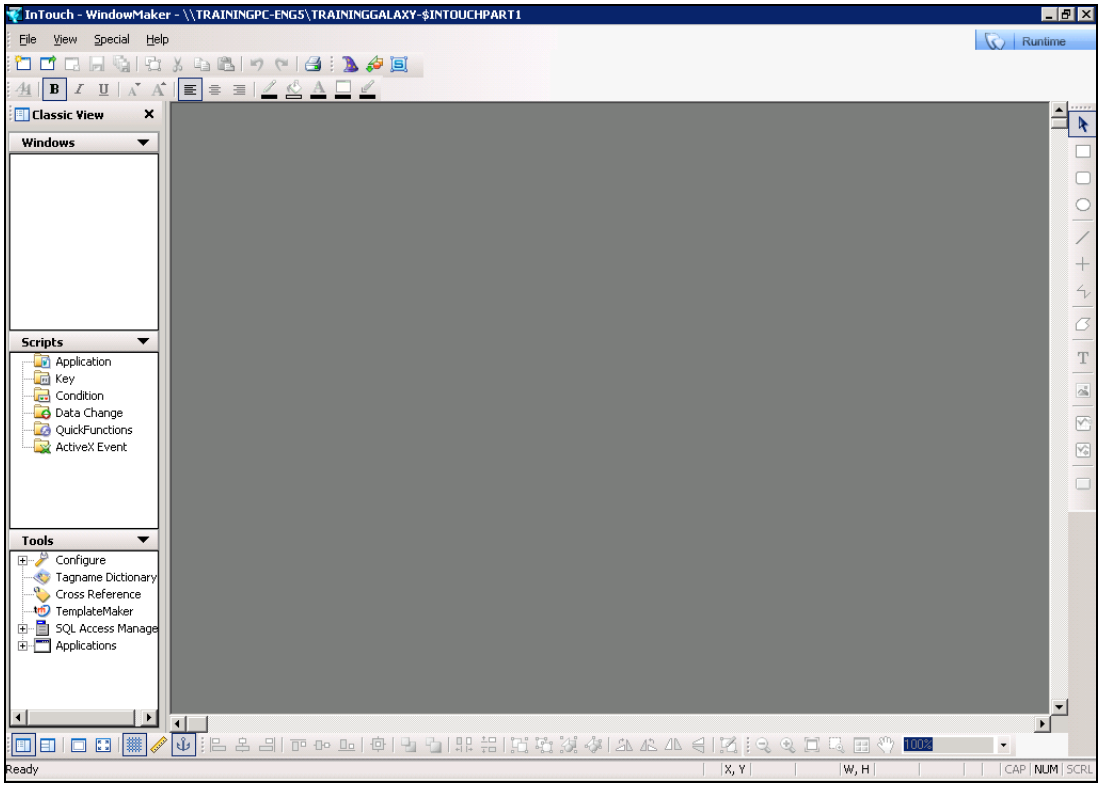
Important: Do not check the **InTouchView** application check box.

22. Click **Next**.

    The WindowMaker application opens.

23. Close **Project View**.

    Only **Classic View** remains open in WindowMaker.

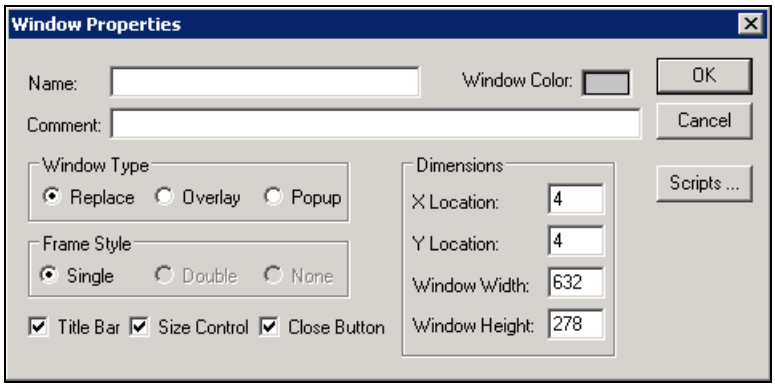


Create a Window

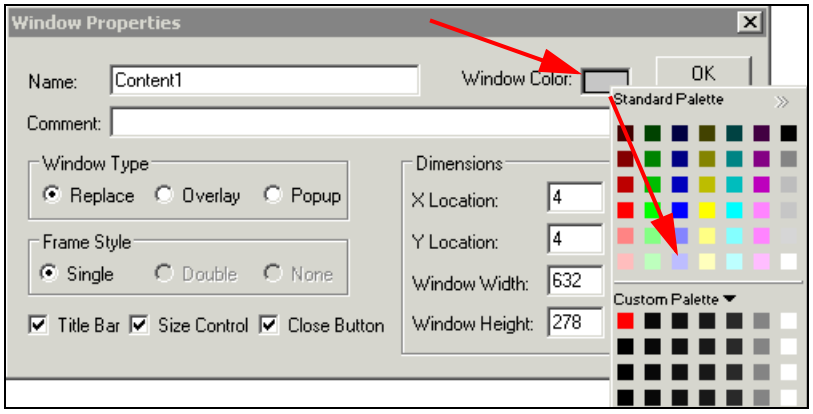
Next, you will create and configure a new window in WindowMaker.

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

    The **Window Properties** dialog box appears.



25. In the **Name** field, enter **Content1**.
26. Click **Window Color,** then on the color palette click on **light blue**.

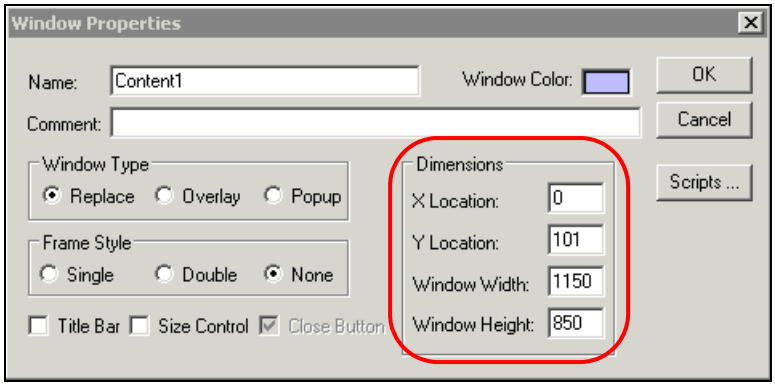


27. Ensure the **Window Type** is set to **Replace**.
28. Uncheck **Title Bar** and **Size Control**.
29. In the **Frame Style** area, click **None**.

Note: The **Frame Style** options **Double** and **None** are not enabled until the **Title Bar** is disabled.

30. In the **Dimensions** area, enter the following details:

X Location:	0
Y Location:	101
Window Width:	1150
Window Height:	850



This places the window in a specific location, with a specific height and width.

31. Click **OK**.

The **Content1** window appears.

Note: You will save your work in the next lab.

Lab 2 – Backing Up and Restoring a Managed InTouch Application

Introduction

In this lab, you will check in your application and back up your application to a location on your hard drive. You will then restore your application.

It is highly recommended that you periodically backup your InTouch application to an external location.

Objectives

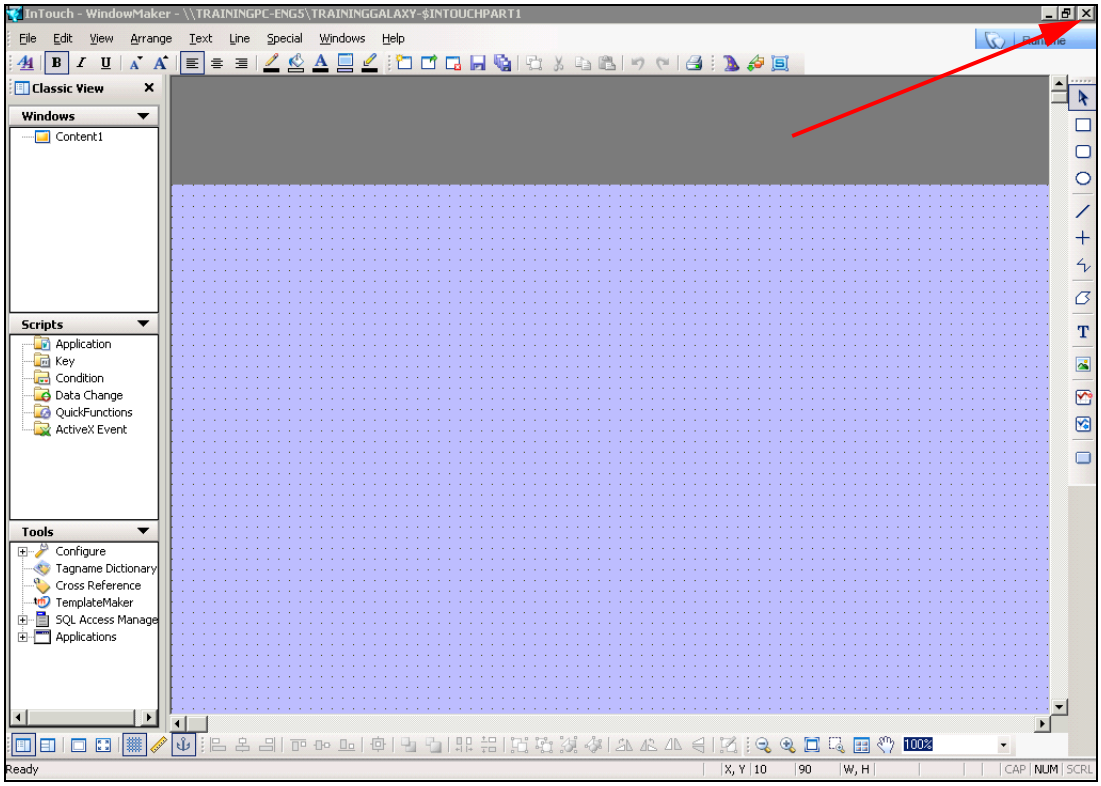
Upon completion of this lab, you will be able to:

- Back up the InTouch application
- Restore the InTouch application

Check the InTouch Application Object into the Galaxy

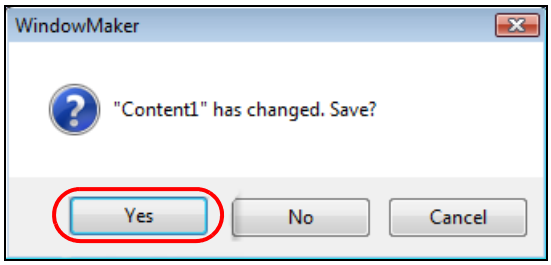
First you will save and check in your InTouch application.

1. Close WindowMaker.



A dialog box prompts you to save the window.

2. Click **Yes**.

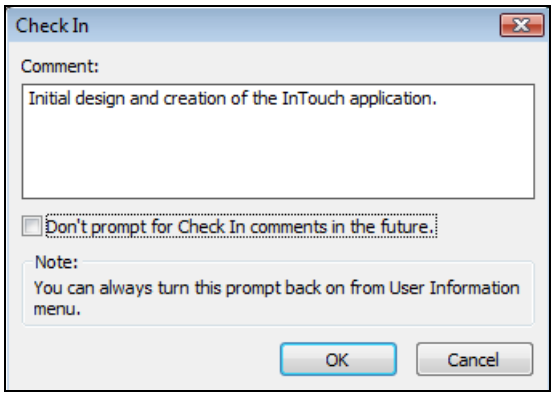


WindowMaker closes.

The **Check In** dialog box appears.

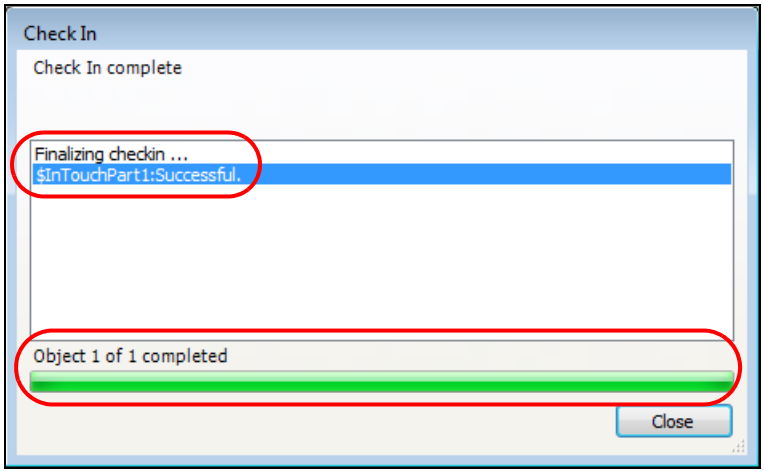
3. In the **Comment** field, enter the following:

   Initial design and creation of the InTouch application.



Note: To keep track of the changes, it is good practice to enter comments each time you check in your application.

4. Click **OK**.



   The **Check In** dialog box appears. This indicates the **Check In** is successful and the process is complete.

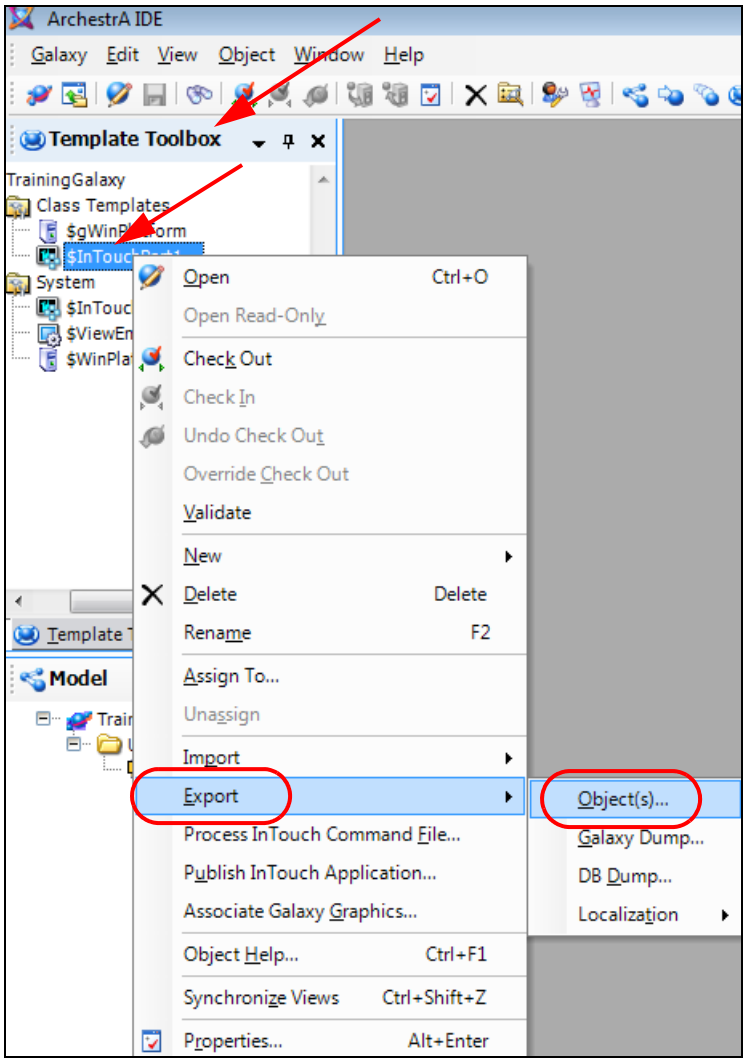
5. Click **Close**.

Back Up the InTouch Application

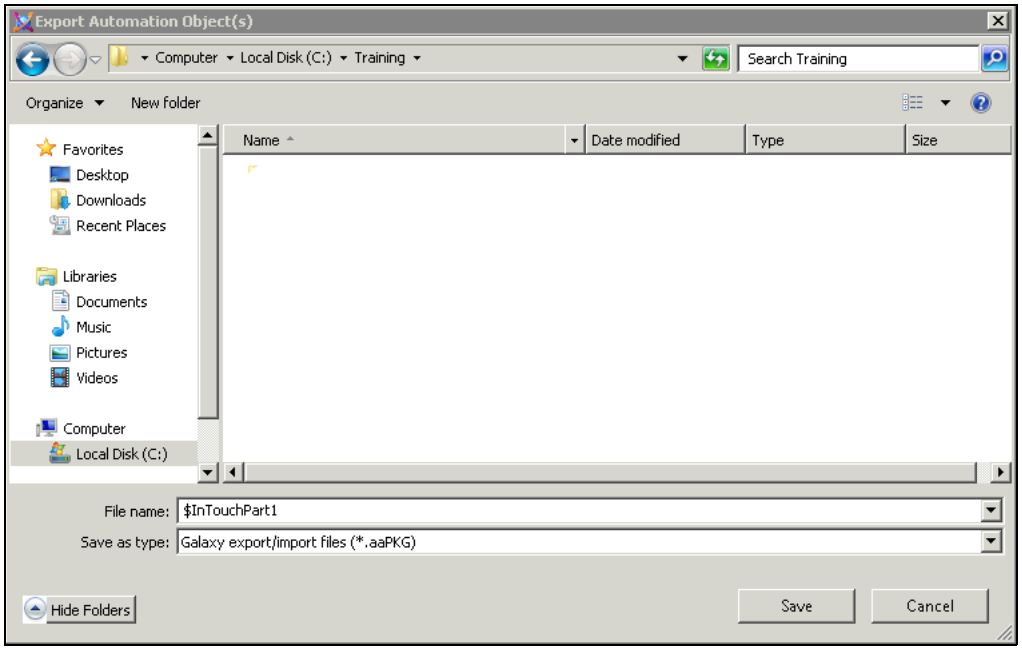
Once a Managed InTouch application is checked in, you can back it up from the ArchestrA IDE.

Follow these steps to back up your InTouch application.

6. In **Template Toolbox**, right-click **$InTouchPart1** and select **Export | Object(s)**.



The **Export Automation Object(s)** window appears.

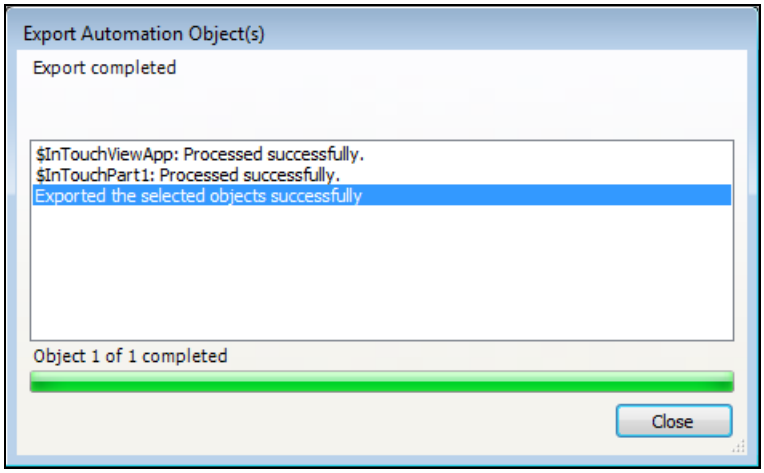


The **File name** field displays the name of your application, with the **.aaPKG** extension type selected by default.

The **Save as type** field displays **Galaxy export/import**. This compressed file is much like a CAB file, containing all parts of the application.

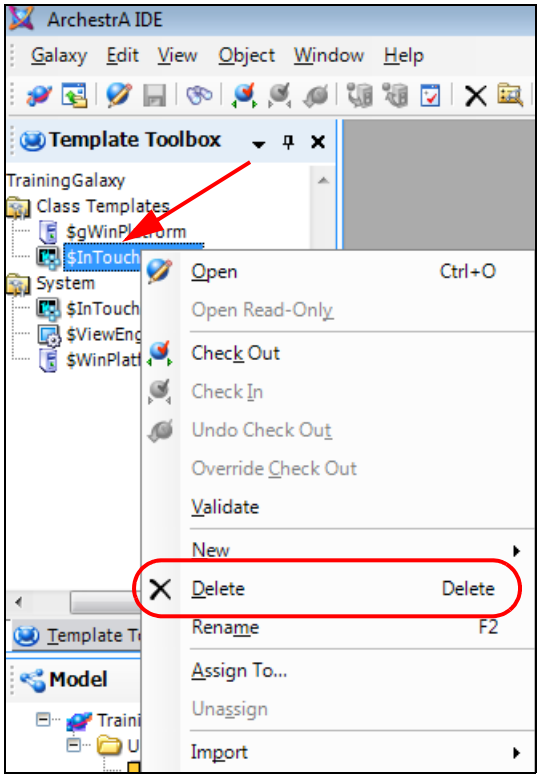
7. Navigate to **C:\Training**.
8. Click **Save**.

   The **Export Automation Object(s)** dialog box displays the status of the export.



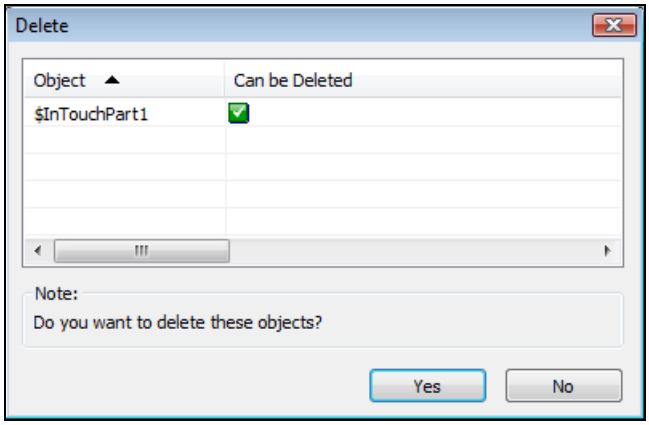
9. When the export is complete, click **Close**.

10. Right-click **$InTouchPart1** and select **Delete**.



Important: This step is performed for illustration purposes to demonstrate how to fully restore your application in the event it is accidentally deleted.

A **Delete** warning displays.

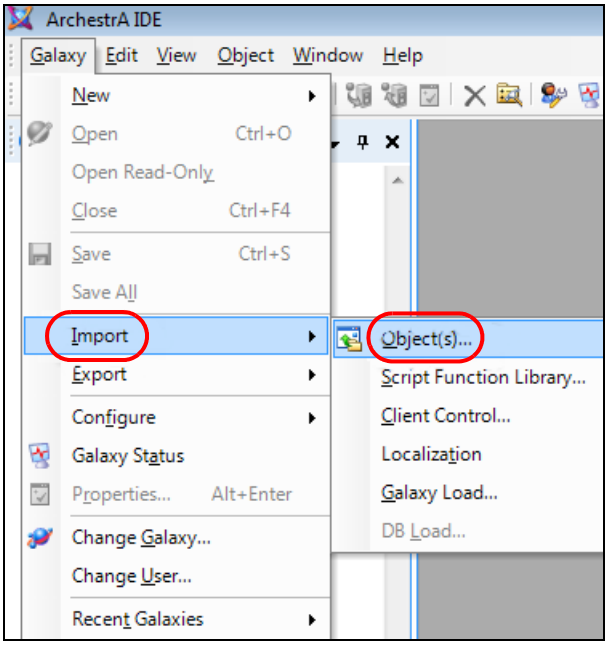


11. Click **Yes** to delete your InTouch application.

Restore the InTouch Application Object

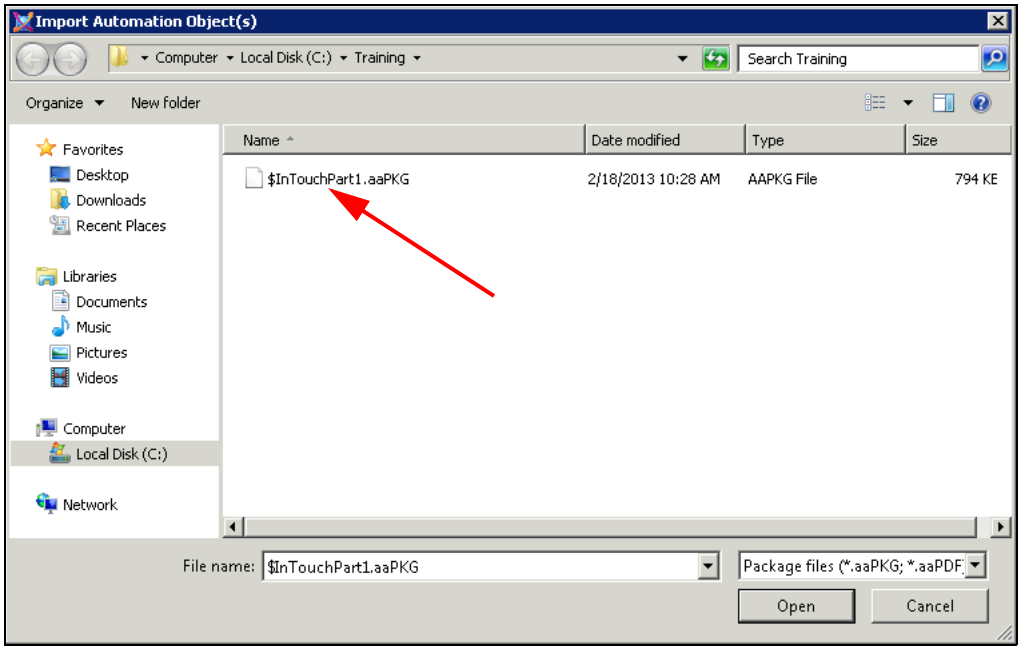
Next, you will restore your InTouch application from the .aaPKG backup you created.

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

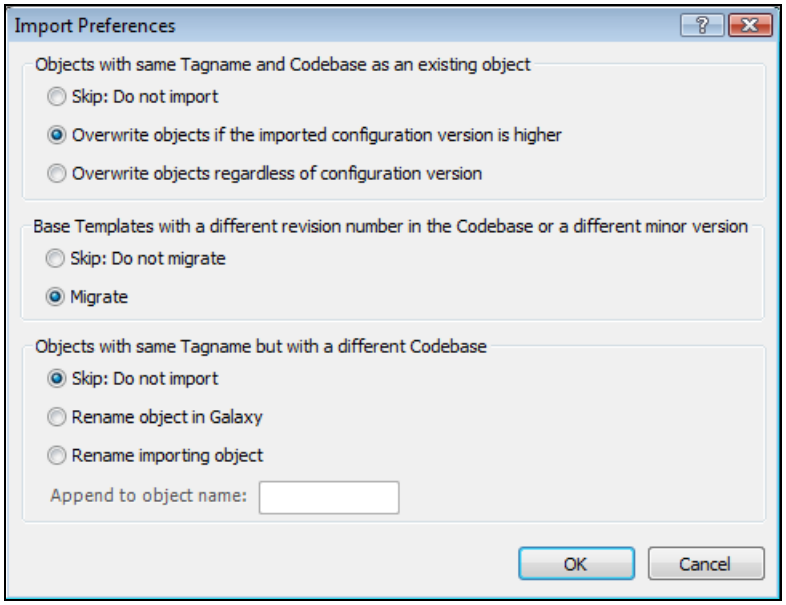


The **Import Automation Object(s)** dialog box appears.

13. Click **$InTouchPart1.aaPKG**, then click **Open** to import the object into your Galaxy.

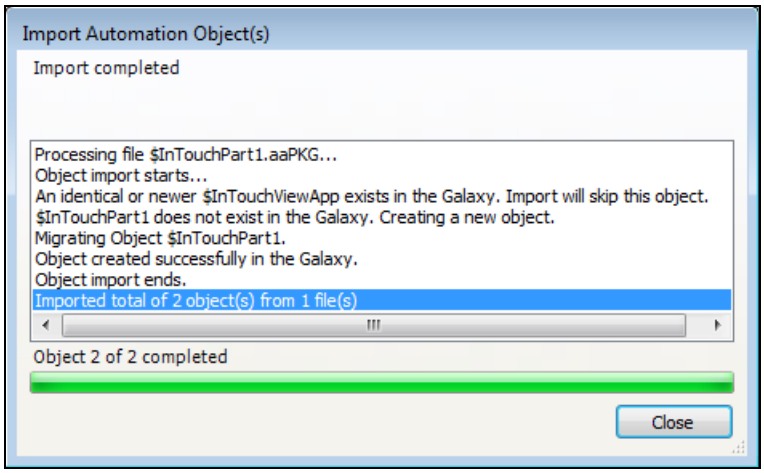


The **Import Preferences** dialog box appears.



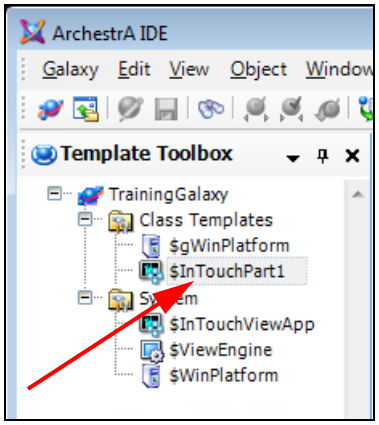
14. Accept the defaults and click **OK**.

    The **Import Automation Object(s)** dialog box displays the import status.



15. When the process is complete, click **Close**.

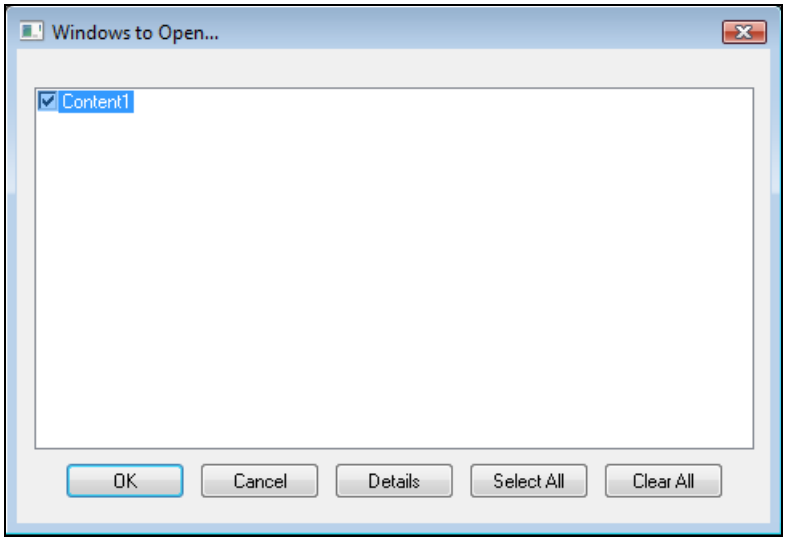
    You can now see your restored InTouch—in the **Class Templates** toolset.



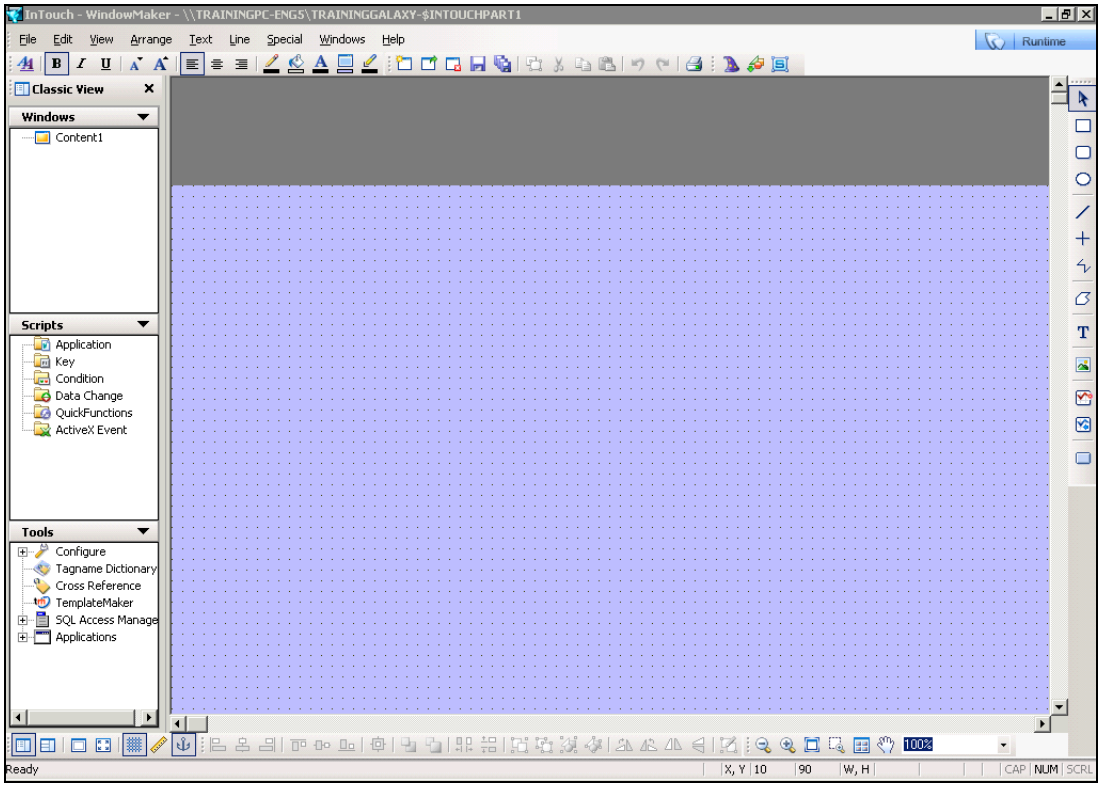
16. Double-click **$InTouchPart1** to start WindowMaker.

    The **Windows to Open** dialog box appears.

17. Leave the default and click **OK**.



The **Content1** window displays.



You were successful in backing up and restoring your InTouch application.

Section 3 – System Requirements and Licensing

This section describes the operating system(s) and other software requirements the host computer must meet prior to installing InTouch 2012 R2.

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 ArchestrA symbols from Application Server 3.1 SP3 or earlier, run the ArchestrA 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 ArchestrA symbols or InTouchViewApp objects containing ArchestrA 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 ArchestrA 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 (GHx)	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.mspx

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.

	Product name Components				
Operating Systems	**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.



Learning Services

Module 2 – Using WindowMaker and WindowViewer

Section 1 – Using WindowMaker	2-3
Section 2 – Using WindowViewer	2-11
Section 3 – Data Content Portal	2-15
Section 4 – Window Properties and Behaviors	2-21
Lab 3 – Data Portal Application	2-23
Section 5 – Tagname Dictionary and Data Sources	2-43
Section 6 – Tag Viewer	2-55
Lab 4 – Creating and Monitoring Tags	2-63
Section 7 – Large-Scale Tag Manipulation	2-85
Lab 5 – Creating Tags on a Large Scale	2-91
Section 8 – Finding and Deleting Tags	2-103

Module Objectives

- Use WindowMaker
- Use WindowViewer
- Build an InTouch application

Section 1 – Using WindowMaker

This section provides an overview of WindowMaker and how to configure WindowMaker components. It also describes how to design windows and set application preferences.

Overview

WindowMaker is the InTouch application editor you use to create and define windows, graphics, animations, the layout, and the look and feel of your HMI application. You use the WindowMaker graphical tool set to design Wonderware InTouch user interfaces with a familiar behavior and style. This allows InTouch application developers to rapidly develop and configure applications with ready-to-use features and functionality.

InTouch WindowMaker allows for rapid application design and deployment, animation and display of data, and supervisory control. InTouch Applications can be enhanced with built-in features of Scripting, Alarming, Historical Logging, and Security.

You use WindowMaker to create the visual interface used by operators to view and manage your manufacturing processes. You can use scripts, graphic tools, ActiveX controls, .NET clients, and miscellaneous tools to enhance the visual interface. An InTouch interface shows data from and writes data back to the production environment.

The WindowMaker Interface

The WindowMaker development environment allows an InTouch Application developer to configure most of the available user interface elements including toolbars, view panes, status bar, classic and project view panes, even the title and application directory path displayed in the WindowMaker title bar. You can turn off user interface elements to provide additional work space or rearrange the toolbars and the view panes to suit your style and efficiency. You can also display the optional ruler and turn the visible grid on and off which is used to control the accurate placement of graphical elements on the open window. You can adjust the levels of undo supported by WindowMaker or change the spacing of the visible grid.

Classic and Project Views

The WindowMaker view panes provide hierarchical graphical views of application elements. These views display the windows, scripts, menus, wizards and other items you have configured in your application and provide easy access to them. They also provide quick access to many of WindowMaker's most commonly used commands and functions.

You can create and configure Windows and other InTouch elements from the view panes.

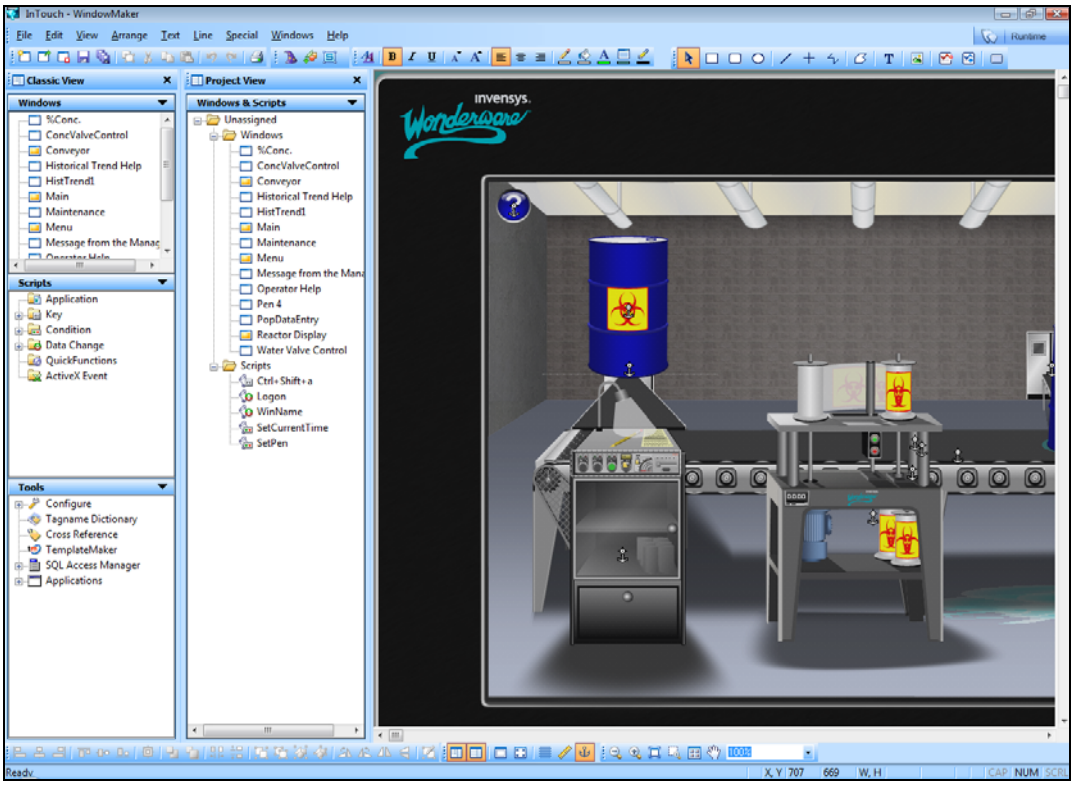
WindowMaker has two views: Classic View and Project View. These views show both your InTouch application windows and the tools available for use and differ only in their presentation of these elements. Like other toolbars, they can be hidden or shown and be docked with the WindowMaker application frame or undocked to float over the workspace.

The Project View allows you to create project folder(s) to help you better organize and manage your application elements. For example, creating a folder for Production Area which might contain all of the windows that represent processes found in the Production Area. This makes finding what you are looking for much easier in an application with hundreds of windows.

Both Classic and Project Views are two ways of looking at the same basic information; to save on space select one and hide the other. This course will uses the Classic View. To hide/show Project View, click the **Project View hide/show** toggle button in the **View** toolbar.

Navigating WindowMaker

The following figure shows the elements of the WindowMaker development environment. When you create a new application and run WindowMaker for the first time, the toolbars, views, panes, and layout are in the default configuration.



WindowMaker helps you build an InTouch application that is comprised of windows that can contain graphics, text, animations, navigation elements, window scripts and action scripts. When you create a window, you define the window properties of the window including background color, title, and screen position.

The **General** toolbar contains buttons for most of the commands also found in the **File** menu, including create new window, delete window, and save window.

The **Special** menu contains the configuration options that are also available from the **Tools** pane, which appears in both the **Classic View** or in the **Project View** (if the Classic View is hidden). In any view where you see the **+** or **–** indicators, you may expand or collapse the groups or if you double-click on a group, the button expands and displays the group's members. When you double-click on a member, it opens that member. All groups that contain members are preceded with a plus sign. You can expand each group and view its members, or collapse the group and hide its members.

Adding Applications to WindowMaker

A powerful feature of WindowMaker Explorer view is its ability to start other ArchestrA and third-party Windows applications from within WindowMaker. For example, you can run or configure an I/O Server program and develop your application simultaneously. You can start third-party Windows programs that you frequently use, such as Windows Notepad, Microsoft Excel, Microsoft Word, Microsoft Paint, among others. You can also configure the Explorer view to open a specific document in a program.

Once the application is added:

- It appears in the **Tools** pane under the **Applications** option
- You can run the application at any time without exiting WindowMaker
- To make changes to the properties of an application added previously, right-click the application name in the **Tools** pane, and select the **Properties** option

Customizing the Development Environment

Using the **WindowMaker Properties** dialog box, you can configure preferences and options that will affect the behavior of WindowMaker, such as:

- Set default text font, size and style for text displays
- Set default fill style, color, texture, transparency, and so forth
- Set default line style, color, width and ends
- Customize the application title bar to display a specific string of text in runtime
- Toggle the grid on and off, and set the pixel spacing for it
- Toggle the option to close WindowMaker when switching to WindowViewer on and off
- Select to pick through hollow objects
- Set the number of undo levels

Setting WindowMaker Properties

To display the **WindowMaker Properties** dialog box, expand the **Configure** list in the **Tools** panel of Explorer view and double-click **WindowMaker**, or select **WindowMaker** from the **Special | Configure** menu.

The WindowMaker configurable properties are as listed in the table below:

Option	Purpose
Title Bar Text	Enter the title that you want to display in your InTouch application title bar while in WindowMaker.
Show Application Directory	Check to include the path to the application's directory in the title bar.
Spacing	Enter the number of pixels that you want spaced between the snap to grid's coordinates.
Show Grid	Display a grid in your windows when you turn on WindowMaker's Snap to Grid functionality. To enable and disable the snap to grid functionality, click the Snap to Grid button on the View toolbar. You can also select Snap to Grid from the Arrange menu to toggle the snap to grid functionality on and off.

Option	Purpose
Close on Transfer to WindowViewer	Check to automatically close WindowMaker when WindowViewer starts. If memory is not an issue and you are moving often between WindowViewer and WindowMaker, this option should not be selected. Upon selecting this option, the CloseWindowViewer option that can be found in the WindowViewer Properties General Properties sheet is also automatically selected. This option does not apply to Managed InTouch applications.
Pick Through Hollow Objects	Check to enable selecting objects that are behind other hollow objects.
Enable Fast Switch	Check to enable fast switch between display modes (WindowMaker and WindowViewer) by displaying a command in the menu bar for both applications. If this option is selected in WindowMaker, the fast switch is the word Runtime, displayed in the upper right hand corner of your screen. In WindowViewer, the word that displays in the same location is Development! Click either to toggle between view modes. When you use the fast switch, WindowMaker automatically saves all changes made to all open windows before WindowViewer is started.
Line Selection Precision	Enables you to enter the number of pixels that your cursor can be away from a line and still be able to select it. In most cases, the default setting of 4 should be sufficient.
Levels of Undo	Enter the number of undo / redo levels to save (up to 25 levels). Enter zero (0) to turn the undo / redo functionality off. One level represents one action. The under and redo stacks are empty when a new window is created or an existing window is opened. Both stacks are emptied when the window is saved.
Configure Fonts area	Click Text to open the Font dialog box, where you can select the default font style, size, color, and effects for text. Click Button to open the Font dialog box, where you can select the default font, style, size, color, and effects for buttons.

WindowMaker Toolbars

The buttons on the WindowMaker toolbars are grouped by common functionality. For example, the **Arrange** toolbar contains tools that you can use to quickly apply most of the commands found on the **Arrange** menu.

The toolbars within WindowMaker can be "docked" to any edge of the WindowMaker window, or "floated" anywhere within the WindowMaker window. When the toolbars are floating, the name of the toolbar displays in the title bar.

When you hover your cursor over a button, a **Tool Tip** appears displaying the tool name.

General Toolbar

The General toolbar consists of buttons that execute most of the window commands found on the **File** menu. The buttons also include the Microsoft Windows clipboard tools found within the **Edit** menu.



Wizards Toolbar

The Wizards toolbar contains the buttons used to access the **Wizard Selection** dialog box, the **SmartSymbol Selection** dialog box, and the **Embed ArchestrA Graphic Selection** dialog box. You can add any installed wizard or ActiveX control to the toolbar.



Wizards are pre-drawn, individual graphical components that are embedded in the InTouch window during development for use in a runtime application. They save a considerable amount of time during application development because they are ready made, easy to use, and easy to configure. Once the required configuration information is entered, the Wizard is ready to use in runtime.

A managed InTouch application has access to the ArchestrA library of graphics, which includes hundreds of prebuilt graphic symbols. These may be embedded in your windows and linked to tagnames for visually displaying data in runtime.

Using the Wizard tool, ActiveX programs can be installed and made available to use in your application. ActiveX controls are complex wizards that can perform many operations. An example is an ActiveX control that brings in a streaming video feed from a camera in the field. ActiveX controls are provided by 3rd party vendors.

Format Toolbar

The Format toolbar is grouped with tools that execute most of the text formatting commands found on the **Text** menu. It also contains the tools used to access the color palette to select line, fill, text, window background, and transparent color.



View Toolbar

The View toolbar is grouped with tools that execute most of the window commands found in the View menu. These commands are used to control the state of the WindowMaker window. In Stand Alone InTouch applications, the View toolbar does not display the Show/Hide ArchestrA Graphic Anchor Points button. This is a feature that is only available for Managed InTouch Applications.



Selecting and Sizing Graphics

The term “selected” is a key concept of WindowMaker graphics editing. Placing the cursor over a graphic item and click directly on the item selects it. Clicking on a blank area of the window deselects any currently selected item(s) in that window.

When an item is selected, small boxes appear around its borders. The boxes are handles and are used to resize and/or reshape the graphic.

In general, any command that you execute is applied to all selected items if it is valid for each item. All objects can be selected using the **F2** key, or selecting **Select All** from the **Edit** menu, or by right-clicking an unoccupied spot on the currently opened window and choosing **Select All** from the context menu. To add to a multi graphic selection hold down the **Shift** key and click the graphic object. Hold down the **Shift** key and clicking again deselects a graphic object.

Snap to Grid

When you are arranging objects in your windows, the Snap to Grid forces objects to align with the grid. The grid must be enabled first. Then when an object is moved or positioned, it will align to the grid. However, enabling the grid does not realign objects that have already been placed on the window.

By default, the grid is set to 10 pixels and visible when you initially start WindowMaker. You can configure the pixel interval for the grid through the WindowMaker **Properties** dialog box. Click the **Snap to Grid** button on the **View** toolbar to turn snap to grid on and off.

While the snap to grid is enabled, using the arrow keys move an object with single grid spacing in that direction. With **Shift** held down, the arrow keys move a selected graphic two grid spaces, and with **Ctrl** held down, the arrow keys move a selected object four grid spaces.

If Snap to Grid is off, the arrow keys move an object 1 pixel at a time. If **Shift** is held down with the arrow keys, an object is moved 10 pixels, and with **Ctrl** held down, 50 pixels.

Ruler

You can use the WindowMaker ruler to do precision alignment of the objects in your windows, and to determine the movement of an object during an animation sequence.

The small tick marks are spaced 5 pixels apart. The medium tick marks are spaced 10 pixels apart. The numbered large tick marks are spaced 50 pixels apart.

Pan and Zoom Toolbar

The Pan and Zoom toolbar allows you to quickly move around a window and enlarge or reduce the magnification level.



If your mouse has a scroll wheel, you can hold down the **Ctrl** key and scroll forward to zoom in and scroll backward to zoom out.

Arrange Toolbar

The Arrange toolbar is grouped with tools that execute most of the object arranging commands found on the Arrange menu. Arrange toolbar element buttons are activated if one or more objects are selected.



Arranging Objects

WindowMaker provides tools to arrange objects in your windows. The Arrange toolbar contains tools that you can use to quickly apply most of the commands found on the Arrange menu.

You can align objects by their left or right edges, center, center points, top, middle, or bottom edges.

With the **Space Horizontal** or **Space Vertical** tools, you can distribute a group of selected graphics horizontally or vertically at equal intervals.

To specify the stacked order of multiple objects that obscure each other, use the **Send to Back** or **Bring to Front** tools.

Rotating Objects

Most objects can be rotated in WindowMaker utilizing the Arrange toolbar. Objects can rotate clockwise or counter-clockwise in 90 degree increments. Any affected animation links attached to the object rotate with the object.

Mirroring and Flipping Objects

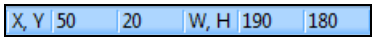
When an object is flipped, it is transformed into its horizontal or vertical mirror image. Slider, Location, Width, and Height animations flip with the object.

WindowMaker Status Bar

When an object is selected in a window, the WindowMaker status bar displays the object's top-left X and Y pixel coordinates and the object's pixel height and width. The X and Y pixel coordinates displayed are relative to the window the object is in, rather than the overall WindowMaker workspace. When you select multiple objects, the status bar displays the width and height for the entire group.

In the WindowMaker status bar, you can modify the X, Y, W, and H fields with specific values to position and resize a graphic. Select an item on a window, then click the numerical fields in the status bar to enter exact X,Y coordinates or W,H values. You can also select multiple objects and move or resize them as a group.

You must press **Enter** after each entry to apply manual positioning and sizing entries.



When you click a blank area of a window, the status bar displays the X and Y coordinates for the current location of the cursor in the window.

Drawing Toolbar

The Drawing toolbar is grouped with all the tools used to draw both simple graphic objects (rectangles, ellipses, lines, or text objects) and complex objects (real-time trends, historical trends, bitmaps, and 3-dimensional buttons with labels). This toolbar displays along the right side of the window.



Hold the **Shift** key down while drawing rectangles, rounded rectangles, and ellipses to constrain the height and width and to create exact squares and circles.

Right-Click Window Menus

To access the various commands that can be applied to a window, right-click a blank area of the open window, then click the appropriate command on the right-click menu. When developing your application, you can open as many windows as the workstation's memory will allow.

The **General** toolbar contains most of the basic window manipulation tools, such as for opening, closing, and saving windows.

Use the **Open Window** button to display the **Windows to Open** dialog box listing all windows in your InTouch application, and select the window(s) you want to open.

Use the **Close Window** button to close the window. At the **Windows to Close** dialog box, select the names of the windows you want to close. If there are unsaved windows open, InTouch will prompt you to save them or cancel. To quickly close a single window, right-click the window name and click **Close**.

Use the **File / Delete** menu command to delete a window. WindowMaker will prompt you to confirm the deletion. To quickly delete a single window, right-click the window name and select **Delete**.

Editing Text Objects and Labels

You can edit the characters within InTouch text objects using the **Substitute Strings** dialog box. There are three methods to access this dialog box:

- Select the text object, then from the menu select **Special | Substitute Strings**
- Use the keyboard shortcut **Ctrl+L**
- Right-click the text object and select **Substitute |Substitute Strings**

The **Substitute Strings** dialog box displays the current string and a field to enter the changes to the text string. In addition, you can use the Substitute Strings command on text strings contained within a WindowMaker symbol or cell. You can also use Substitute Strings with ArchestrA symbols which contain text or labels.

Search and Replace Text in a Graphic Item

Once a text string is substituted in one graphic, InTouch allows you to globally change all or a portion of a text string throughout all selected graphics containing the same text string. To perform a global search and replace for partial or whole text strings within all selected graphics, you use Substitute Strings and the **Replace** option. In the **Old Text** field, enter the text string characters to be replaced, then in the **New Text** field enter the characters which will replace the Old Text entries.

Section 2 – Using WindowViewer

This section describes your application visualization node and defines WindowViewer components and features.

Overview

InTouch WindowViewer is the runtime visualization application. It executes scripts, interfaces with the alarm manager, the historical data retrieval and storage system, drives animations, links and displays tagname data, displays graphics and text, and provides interactive tools for supervisory control and data acquisition.

WindowViewer is a Human Machine Interface for users to view and control processes in a visual display format with animations and user interfaces. WindowViewer is a flexible environment that can be customized to your requirements, for example the menus in WindowViewer can be hidden, the title bar can also be hidden, it can be configured as impossible to close so it cannot be shutdown accidentally, it can be localized in different languages, and dynamically change based on which workstation it is running. Additionally, WindowViewer can cater the user experience to the available memory, video resolution, processor power, and other local environmental requirements.

Functionality

Typically, menu items are left enabled during the testing phase of application development, to allow runtime options and features to be used easily by application designers. For example, you can select commands to restart I/O conversations, halt scripting, configure security, start Tag Viewer, and halt historical logging. You can configure properties to hide menus, commands, and standard window components to ensure operators can only interact with your InTouch application from their computers.

File Menu

The File menu contains general options to open, close, and exit the application. You can also change the active window and view information about your WindowViewer version.

Option	Purpose
Open Window	Displays the **Windows to Show** dialog box with a list of the windows in the application. Check the window you want to view and click **OK**. You can also view Type and Last Modified details for the available windows, select all windows, or clear all window selections.
Close Window	Displays the **Windows to Hide** dialog box. You can select an option from a list of all open windows.
WindowMaker	Displays the **Windows to Edit** dialog box. Check the window you want to edit and click **OK**. You can also view Type and Last Modified details for the available windows, select all windows, or clear all window selections.
Exit	This option closes WindowMaker.
About WindowViewer	Locates information about your WindowViewer version, serial number, version, and expiration date. You can also access your WindowViewer license and license agreement from the dialog box.

Logic Menu

You can use a wide variety of runtime triggers to start scripts while an application is running. You use options on the Logic menu to start or stop these scripts.

Option	Purpose
Start Logic	Starts running all QuickScripts.
Halt Logic	Stops QuickScripts. This also prevents operators from starting new asynchronous QuickFunctions. Currently executing asynchronous QuickFunctions are not stopped.

Special Menu

Use the items in the Special menu to perform additional tasks in WindowMaker. Some items may be restricted based on Security settings.

Option	Purpose
Start Uninitiated Conversations	Starts all I/O connections defined in the InTouch application.
Reinitialize I/O	Reinitializes Access Names using the InTouch default settings. You can reinitialize all Access names, or select specific Access Names.
Restart Historical Logging	Restarts logging after it is manually stopped with the Stop Historical Logging option.
Stop Historical Logging	Stops logging for the duration of the current application session. Logging remains stopped for the current session until it is manually started again.
Security	Allows Individuals to log on to the system, change their password, and log off. Individuals with sufficient security can also add or update users, and change the security level for a user from this option.
Language	Changes the currently displayed language to another language configured in the application. If no additional language is configured, only the current language appears.
Tag Viewer	Allows you to watch and monitor tags and modify tag values at run time. It provides you with a list of all available tags in the application arranged hierarchically, based on their alarm groups. Tag Viewer displays only the tags available in the local InTouch application and does not support remote references.

Security

InTouch supports three types of security:

Option	Purpose
InTouch Security	Users that are allowed to access WindowMaker and WindowViewer, are defined and managed in InTouch.
OS Security	Users that are allowed to access InTouch, are defined and managed in the User Manager of the Windows operating system.
ArchestrA-based Security	Users that are allowed to access InTouch, are defined and managed in the ArchestrA IDE.

All types of InTouch security authenticate users during the logon process with a user name and password combination. Each type of security provides a different mechanism to verify the user name and password during the authentication process.

Applying security to your application is optional. By default, an InTouch application is not secured. However, you can restrict functionality in your application through animations, such as disable

visibility through linking to InTouch system tagnames that contain security information, such as which user is logged into the system or what level of access the currently logged in user has.

To secure your InTouch application at runtime, you can restrict access to specific windows or options, turn off certain keystrokes, and other settings to insure that users do not change the application environment, close the application, or escape from the application to other programs. You can also set a visibility or disable link to an object to limit access by user log in, and turn off menu bar options.

Section 3 – Data Content Portal

This section explores the fundamentals of designing an InTouch application. You will also learn how to integrate data connectivity into your application, and view the results in WindowViewer.

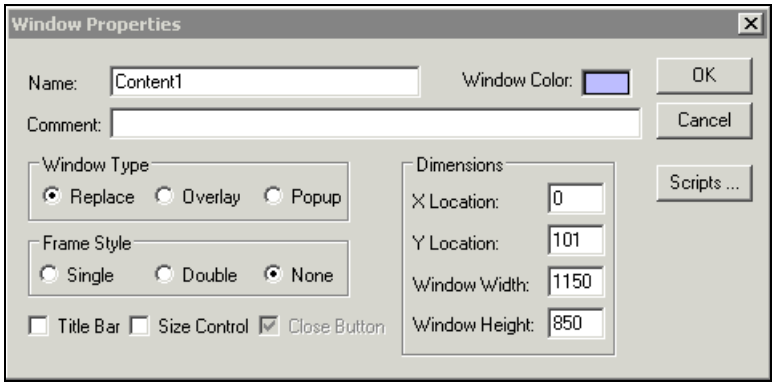
Overview

InTouch allows you to create graphical representations of your plants, equipment, and machinery. Once the graphics are created, you can link them to tag data using custom properties to simulate what is going in the real world. In the next lab, you will quickly and easily develop a custom graphical view using the comprehensive set of graphic tools. Then you will use tags and scripting to animate the result in WindowViewer.

Creating a New Window

Consider the overall layout of your WindowMaker application. To create a new window, click the **New Window** button. In the **Window Properties** dialog box, type the name for your new window, add a comment to be stored with the window, define the **Window Type** and **Frame Style**, select a **Window Color**, enable/disable a runtime **Title Bar** and **Size Controls**, and determine the window dimensions and location.

By default, the settings in the Window Properties will reflect those of any previously created windows. If you select this command while a window is open in WindowMaker, the settings will reflect those of the active window.



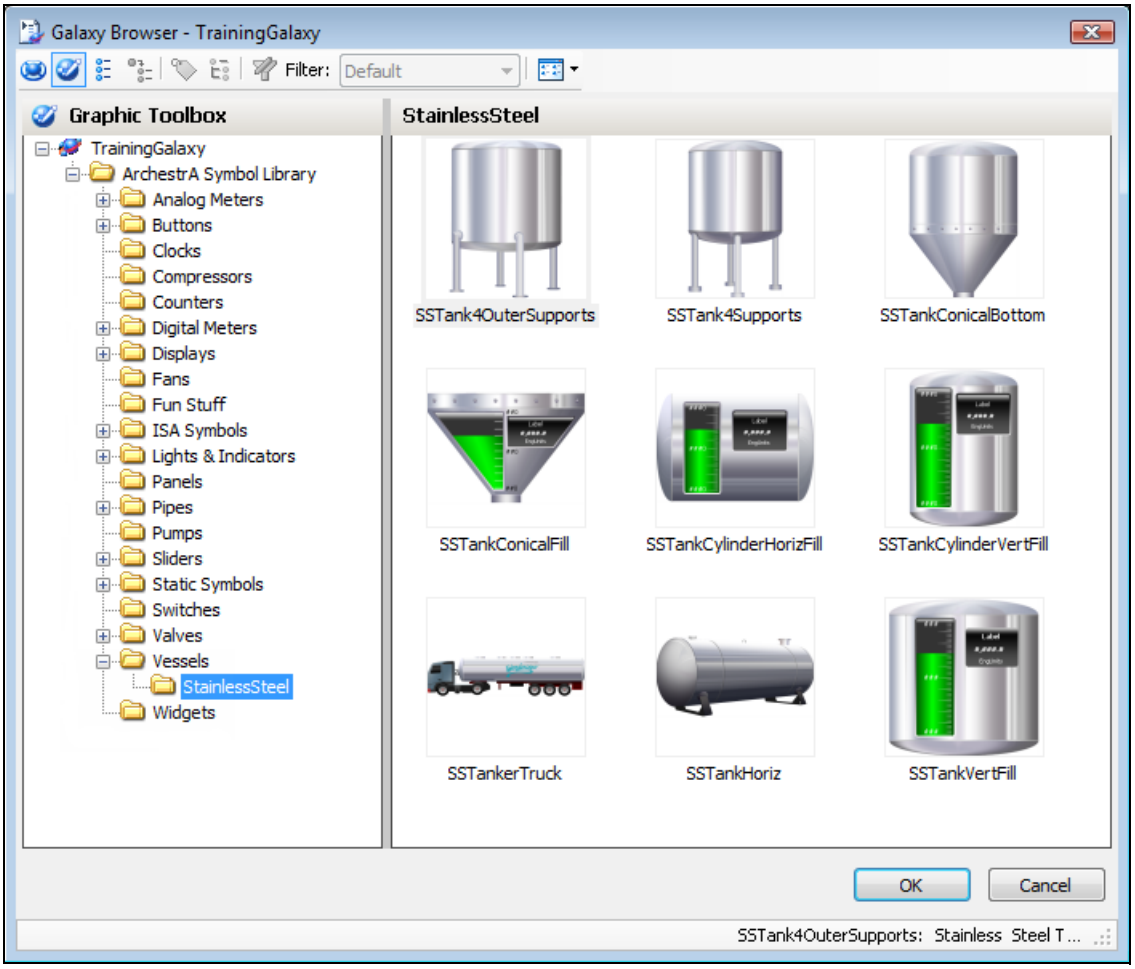
WindowMaker Color Palette

The WindowMaker color palette allows you to apply color to static and dynamic properties of lines, rectangles, round rectangles, ellipses, polylines, polygons, and text. You can also use it to select a window background color and the transparent color for graphics, allowing objects to be viewed behind graphics. The color palette appears when a colored square in a dialog box is clicked or when one of the color tools is used to apply line, fill, or text color to a selected object.

The WindowMaker color palette allows you to define custom colors and add them to your palette. You can import palettes created in other Windows applications and add them to the standard palette. You can also export custom palettes to other Windows applications.

Embedding an ArchestrA Graphic

This feature provides you with the ability to embed pre-built symbols into your applications from the ArchestrA **Graphic Toolbox**. Use the **Embed ArchestrA Graphic** button to open the **ArchestrA Symbol Library** and view the library of available ArchestrA graphic symbols.



Select the graphic you want to embed and click **OK**, then click the window canvas to place it on the window. When symbols are embedded, their custom properties are included. You can then double-click the graphic to edit available custom properties.

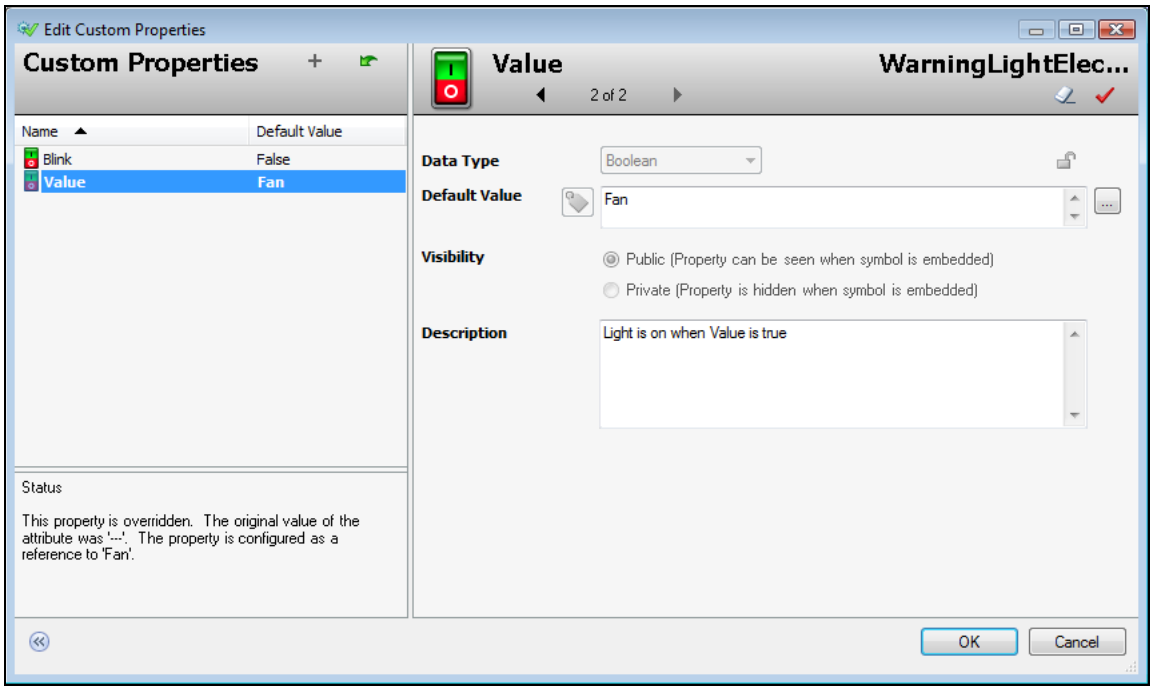
Managing Memory for WindowViewer

You can configure how WindowViewer uses memory for windows and popup symbols to improve performance at run time. Windows and popup 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*.

Configuring Custom Properties

Custom properties allow you to make the most use of your symbols by making them customizable when embedded. You can use custom properties to extend the functionality of a symbol.



A custom property can contain:

- A value that can be read and written to
- An expression that can be read
- An ArchestrA attribute that can be read and written to if the attributes allow
- A property of an element or symbol
- A custom property of a symbol

Custom properties can be set to either be Private or Public.

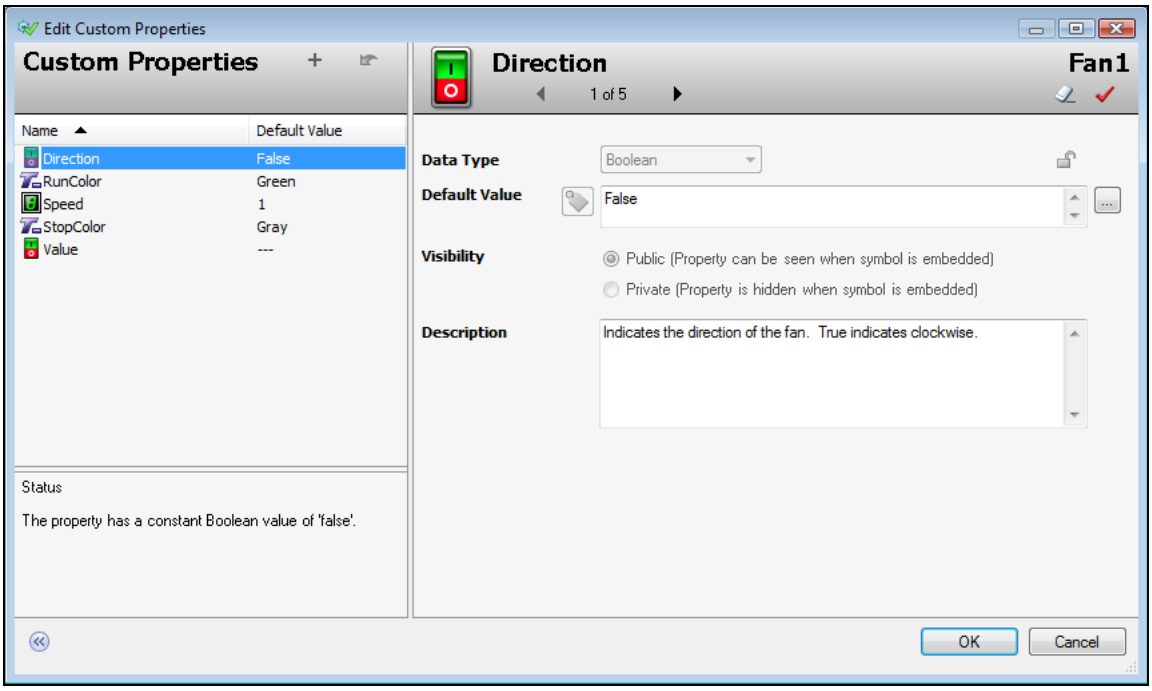
- **Private**: Property is not exposed when the symbol is embedded
- **Public**: Property can be customized when the symbol is embedded

Custom properties can also have Absolute or Relative references.

- **Absolute**: A reference to an attribute that is fully defined (Tank1.InletValve.PV)
- **Relative**: A reference that goes up the hierarchy to parent objects (Me.InletValve.PV)

Managing Custom Properties

You can manage all custom properties of a symbol in the **Edit Custom Properties** dialog box. When you edit the custom properties of a symbol embedded from the **ArchestrA Symbol Library**, each property listed has a default value data type, default value, default visibility, and default description. The description is useful because it indicates what the symbol can do and to a degree, how to configure it.



You can see the name of the symbol and the custom property above the configuration panel.



To add a custom property, display the **Edit Custom Properties** dialog box and click **+**. In the **Data Type** list, then select the data type of the custom property. You can select one of the following:

Data Type	Symbol
Boolean	
Double	
Elapsed Time	
Float	
Integer	
String	
Time	

In the **Default Value** field, type a literal value, reference, or expression. You can also browse for a reference using the **Browse** button ....

If the selected data type is **String**, **Time** or **Elapsed Time**, you can:

- Select the **T** icon to indicate that the default value is a static value.
- Select the **Label** icon to indicate that the default value is a reference to a value.

For **Visibility**, select one of the following:

- **Public**, so that the custom property is visible and can be used in a parent symbol if the symbol is embedded or inserted.
- **Private**, so that the custom property is not visible outside of the defining symbol and cannot be used if the symbol is embedded or inserted.

In the **Description** field, type a meaningful description for the custom property.

Linking Custom Properties to External Sources

You can link custom properties of a symbol directly to external sources as follows:

- **Configure Automation** objects that point to external sources and then point the custom property to the corresponding attribute reference.

OR

- Use **InTouch:tagname** to link to an InTouch tag.

Using Tags

A Tag is simply defined as a value. The value represents a tank level, temperature, communication status, switch position, valve position, or other real-time values in your 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 tags, InTouch requires more information in order 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 the tags in the database. To create the runtime database, InTouch requires information about all of the variables being created. Each tag must be assigned a tagname and tag 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 (for example: created, value change, or configuration change), the changes are stored in the Tagname Dictionary.

Manual creation and configuration of tags is performed using two interfaces of the Tagname Dictionary: the **Tag Editor** and the **Tag Browser**.

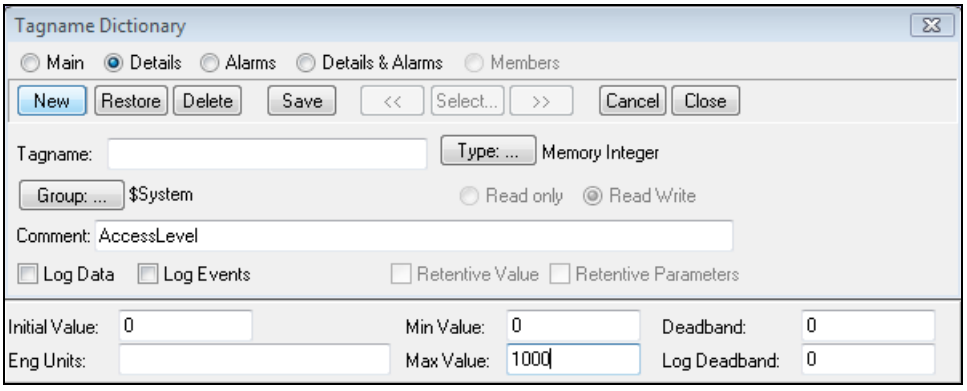
Tag Types

InTouch tags are either **Memory** tags or **I/O** tags. For example, if the tag reads or writes values to or from an external application, such as an I/O Server, it is an I/O tag. When you are defining tags in the InTouch database, you must assign a specific type to each tag according to its usage.

Creating Tags

Tags are created using the Tagname Dictionary. You access the Tagname Dictionary from the main WindowMaker menu **Special | Tagname Dictionary**. You can also access the Tagname Dictionary from the **Tools** panel of **Application Explorer**. Anytime you use a tagname in a script, animation, or a link that has not been defined, you are prompted to define it as soon you save changes or validate. This allows you to create tags on the fly as they are needed.

Click **New** to create a new tag, and click **Type** to define the tag type you want to create. You then configure tag values and save the tag to the dictionary.



Options available in the Tagname Dictionary will vary based on the tag type selected.

Section 4 – Window Properties and Behaviors

This section discusses window properties and how they can be configured using the options available in the Window Properties dialog box.

Overview

Your InTouch application is comprised of windows that can contain graphics, text objects, animations, and action scripts. When you first create the window, you define properties of the window itself such as background color, title, and screen position.

When you create a new window, you define behaviors and aesthetic properties in the **Window Properties** dialog box. Specify if you want the window to automatically close other windows it intersects with or position the window on top or behind open windows. You also define border style, position and size, and background color.

Available options include **On Show**, **While Showing**, or **On Hide**. You can further refine window behaviors using the **Window Script** dialog box.

Creating a New Window

You can create a new window using several different methods.

- From the **File** menu, select **New Window**.
- Click the **New Window** button on the toolbar.
- In the **Project View**, right-click **Windows** and select **New Window**.
- In the **Classic View**, right-click an existing window and select **New**.

When you create a window, you are only required to provide a window name. All other items are optional.

When you create a window, the default settings reflect those of the previously created or of the currently active window. Window names can have up to 32 characters and can include any character on the keyboard except quotation marks.

You can include a comment for a window, but it is for design information purposes only. By default, the window dimension values are set to the dimensions of the previously created window. These values are also automatically modified if you manually change the window size by dragging the window border.

Window Properties

At the **Window Properties** dialog box, you can type the name for your new window, add a comment to be stored with the window, define the **Window Type** and **Frame Style**, select a **Window Color**, enable/disable a runtime **Title Bar** and **Size Controls**, as well as determine the window's dimensions and location.

By default, the settings in this dialog box reflect those of any previously created windows. If you select this command while a window is open in WindowMaker, the settings reflect those of the active window. If a window script(s) is attached to the active window, a message box will appears asking if you want the window script(s) copied to the new window.

To make changes to the properties, right-click the window name in one of the views and select the **Properties** option.

When you create a new window, you provide a unique name that identifies the window. You can also enter an optional comment in the **Comment** field of 50 characters or less.

Click the **Window Color** box to select the background color for the window.

Window Type options include the following:

- **Replace** – Automatically closes any window(s) it intersects when it appears on the screen, including popup and replace type windows.
- **Overlay** – Appears on top of currently displayed window(s) and can be larger than the window(s) it is overlaying. Clicking on any visible portion of a window behind an overlay window makes that window active.
- **Popup** – Is similar to an overlay window, yet always stays on top of all other open windows (even if another window is clicked). Popup windows usually require a response from the user in order to be removed.

The **Title Bar** can be enabled or disabled. Use the check box to toggle between options. You can also use **Size Controls** to resize a window in WindowMaker.

Frame Style options include the following:

- **Single** – By default, displays a single line around the window. When the **Title Bar** is unchecked, it enables the two additional options.
- **Double** – Displays a 3-D effect border around the window. This option is enabled when the **Title Bar** option is unchecked.
- **None** - Displays a window with no border. This option is enabled when the **Title Bar** option is unchecked.

Use the **Dimensions** options to define the location and size of a window. Dimensions are measured in pixels.

Click the **Scripts** button to define scripts for your window. Available window script types include **On Show**, **While Showing**, or **On Hide**.

- **On Show** executes one time when the window is initially shown (opened).
- **While Showing** executes continuously at the specified frequency while the window is open.
- **On Hide** executes one time when the window is hidden (closed).

Lab 3 – Data Portal Application

Introduction

In this lab, you will create two new InTouch windows and write a window script. You will then create a new tag and animate some of your graphics. This will provide a broad view of InTouch application design and introduce some shortcuts to demonstrate its ease of use and broad capabilities.

Objectives

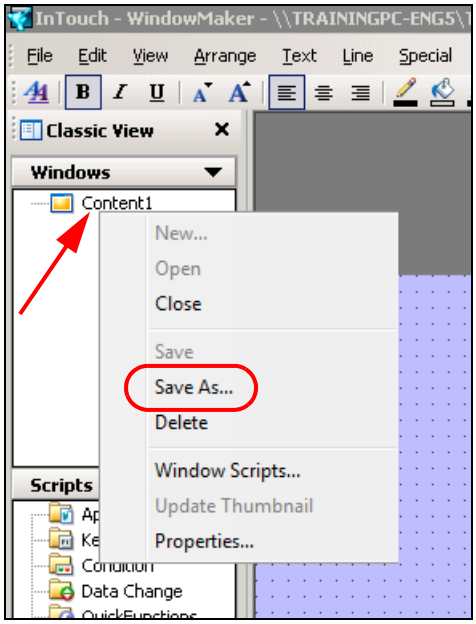
Upon completion of this lab, you will be able to:

- Examine the big picture of InTouch application design including the use of ArchestrA graphics, tags, and scripting

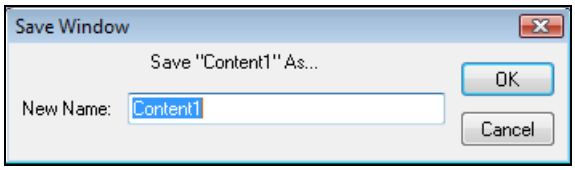
Create Two InTouch Windows

Follow these steps to create two new windows.

1. In WindowMaker, in **Windows** pane, right-click **Content1** and select **Save As**.



The **Save Window** dialog box appears.



2. Change the window name to **Content2**.
3. Click **OK**.

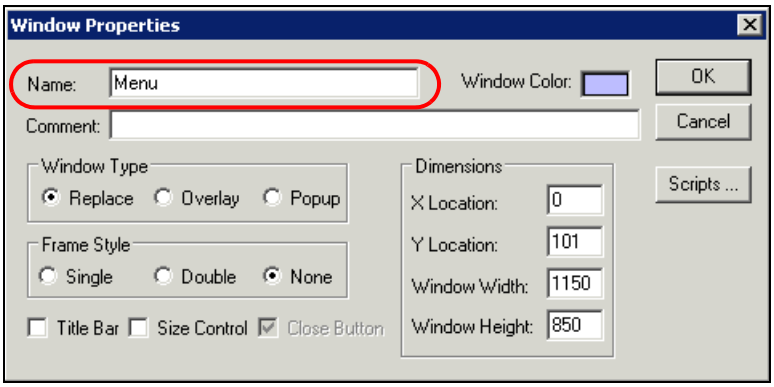
Next, you will create a menu bar.

4. Inside **Windows** pane, right-click and select **New**.



The **Window Properties** dialog box appears.

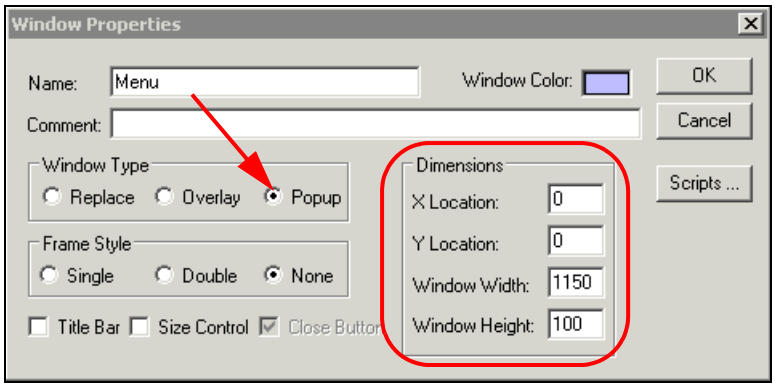
5. In the **Name** field, enter **Menu**.



Note: Window properties are retained from the last window created.

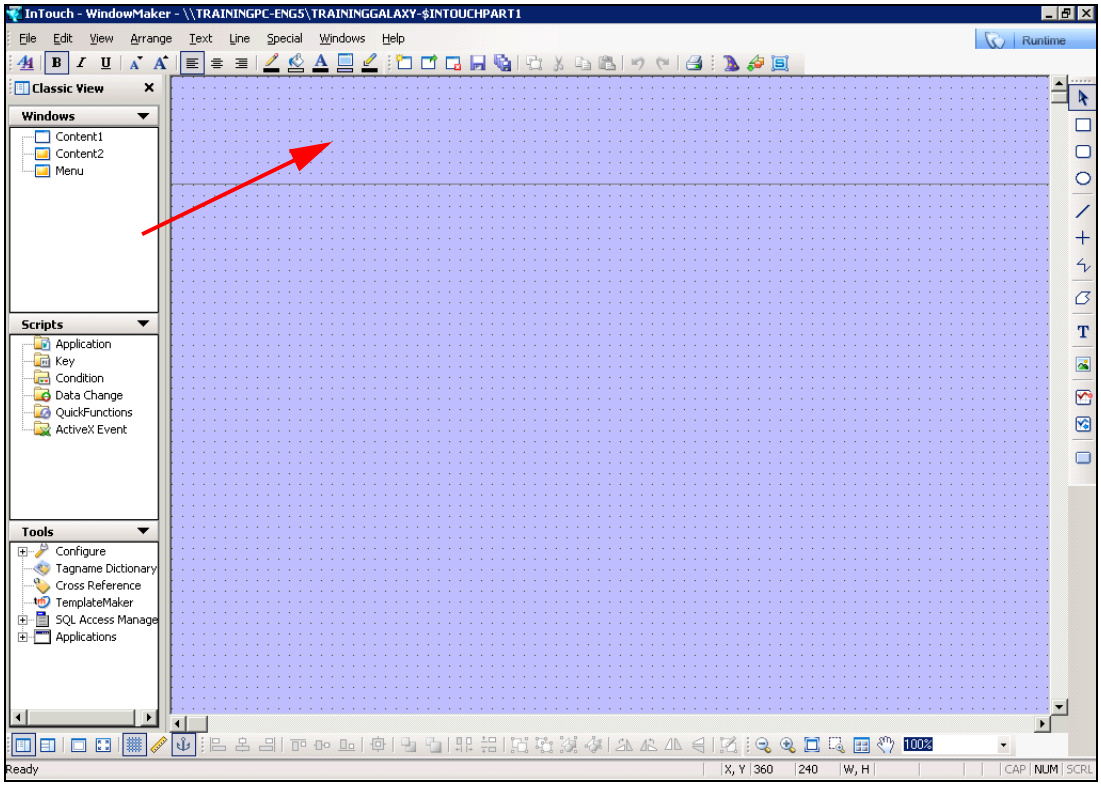
6. In the **Window Type** area, select **Popup**.
7. Change the **Dimensions** as follows:

X Location:	0
Y Location:	0
Window Width:	1150
Window Height:	100



8. Click **OK**.

   Notice the new **Menu** popup window fills the space above the **Content 2** window.



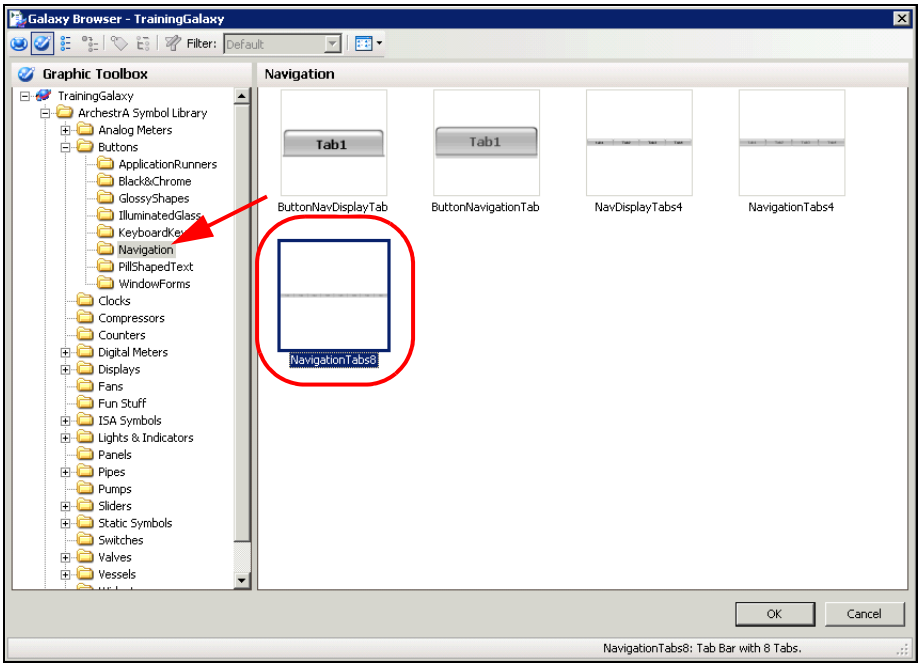
Next, you will add a pre-built graphic from the ArchestrA Symbol Library.

9. On the toolbar, click **Embed ArchestrA Graphic** .



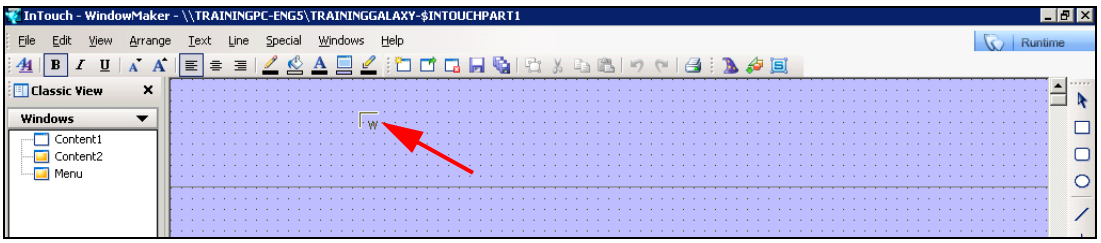
The **Galaxy Browser** appears.

10. In **Graphic Toolbox**, expand **TrainingGalaxy\ArchestrA Symbol Library\Buttons**, click **Navigation**, then select **NavigationTabs8**.



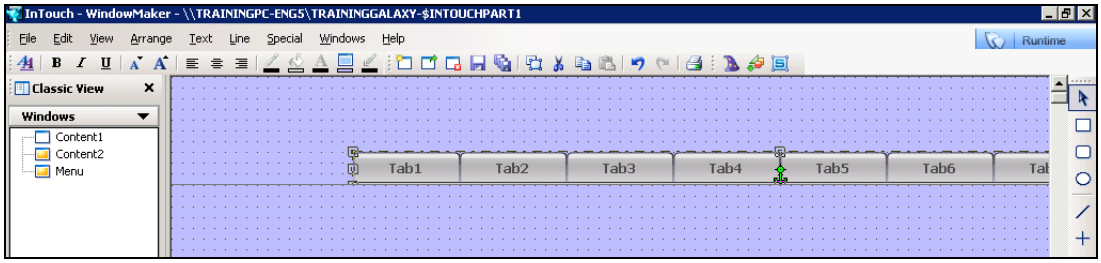
11. Click **OK**.

The cursor changes to a placement handle.



12. On the **Menu** window, click to place the graphic.

13. Position the **NavigationTabs8** graphic at the right corner of the **Menu** window, leaving the left area open for additional items you will add later.



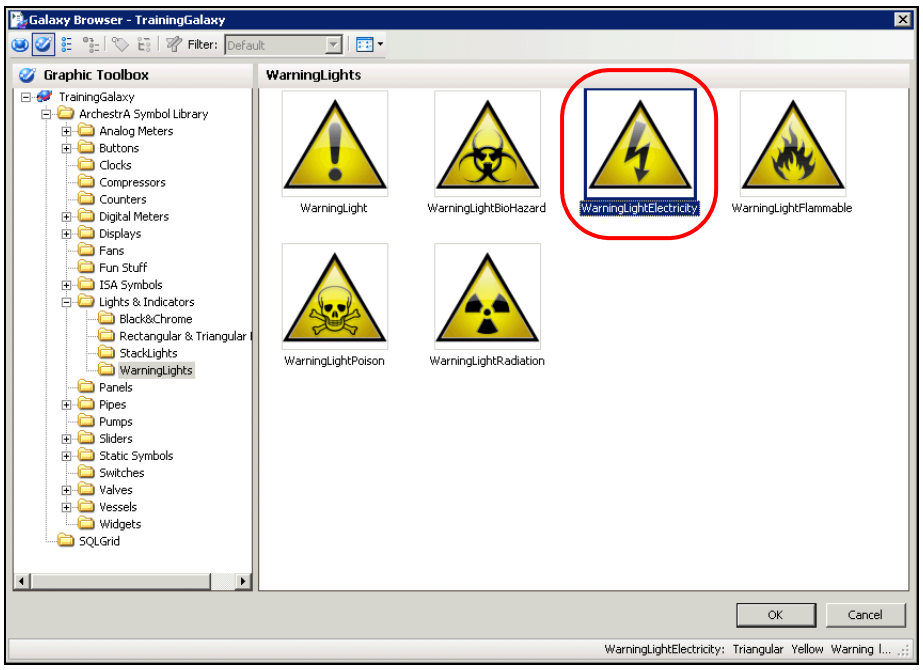
Add ArchestrA Graphics

Next, you will add a graphic to the Content2 window.

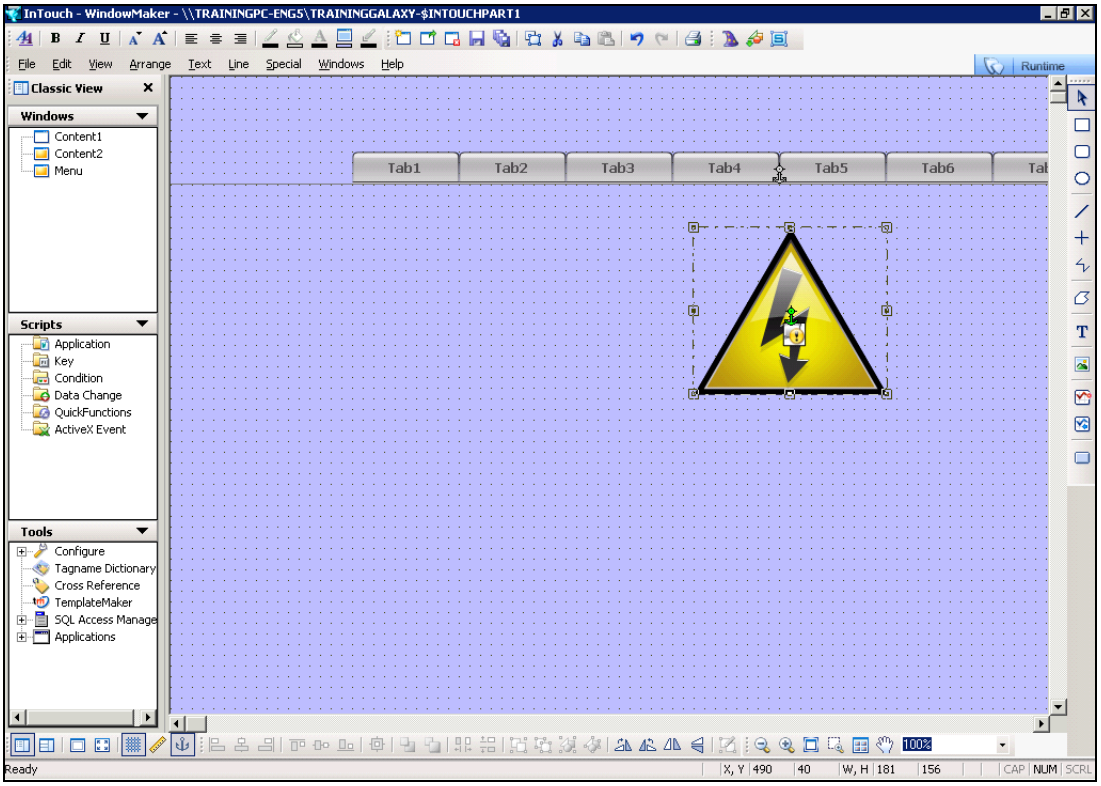
14. On the toolbar, click the **Embed ArchestrA Graphic** .

15. In **Graphic Toolbox**, expand **TrainingGalaxy\ArchestrA Symbol Library\Lights & Indicators**, then click **WarningLights**.

16. Double-click **WarningLightElectricity**.



17. In the **Content2** window, place the graphic.



Next, you will add a graphic to the Content1 window.

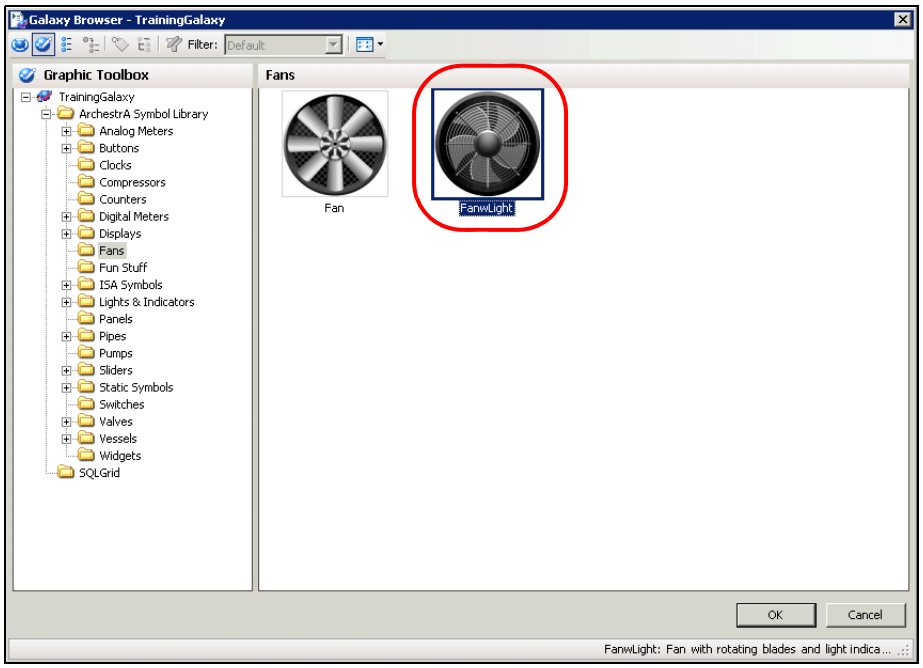
18. In **Windows** pane, double-click the **Content1** window to open it.

    Notice that **WarningLightElectricity** disappears because **Content1** is now on top of **Content2**.

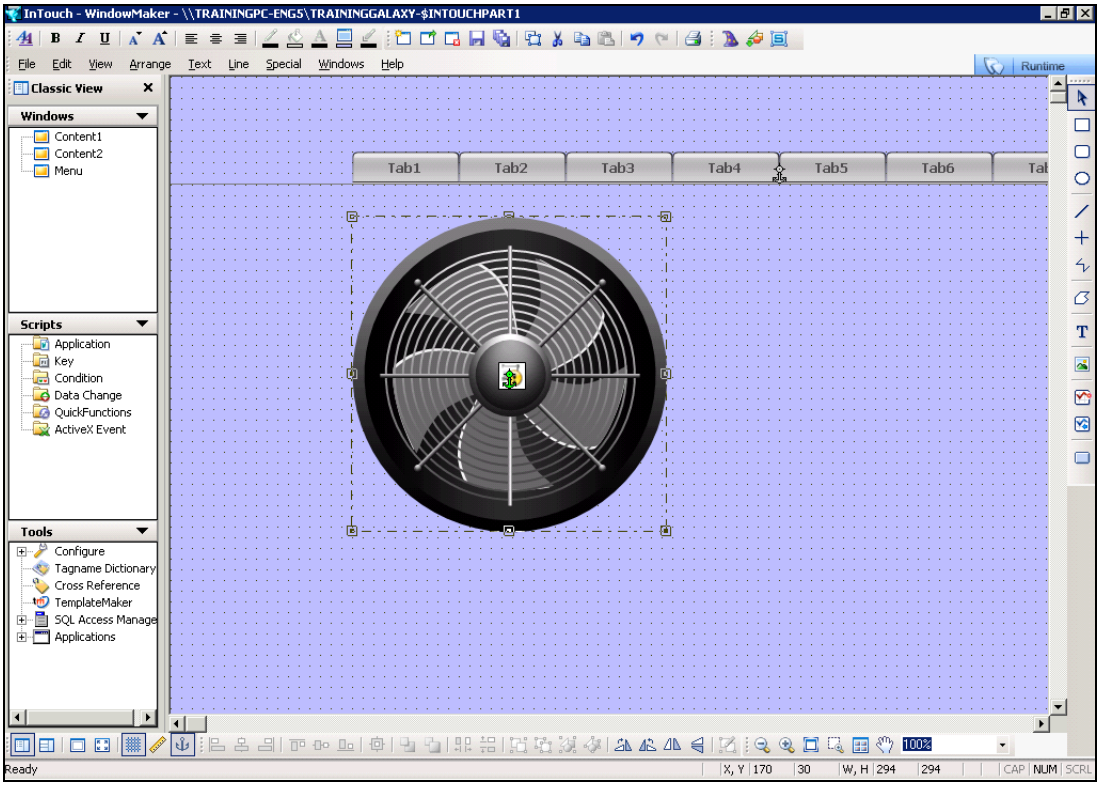
19. Click **Embed ArchestrA Graphic** .

    The **Galaxy Browser** appears.

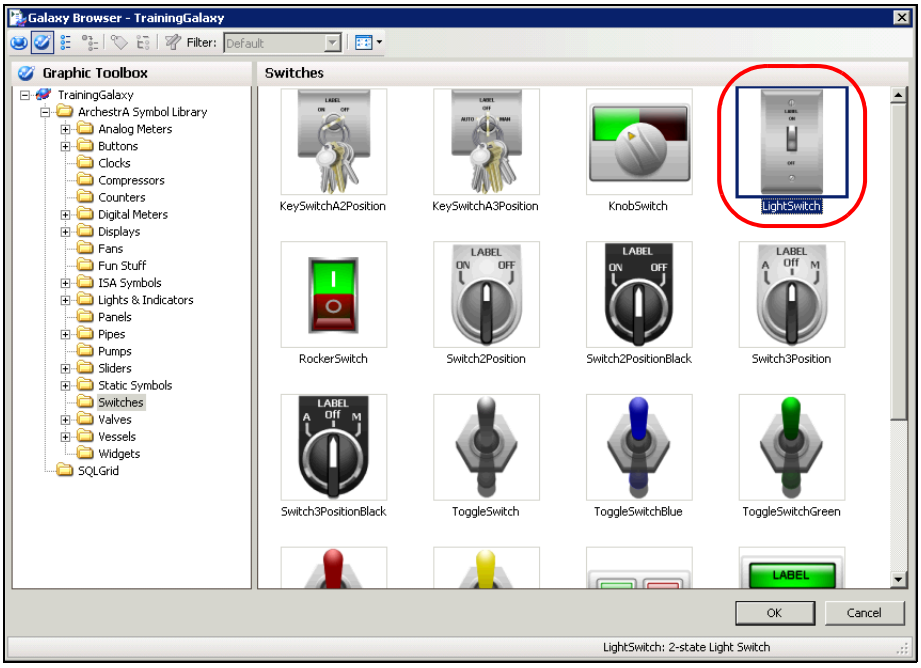
20. In **Graphic Toolbox**, click **Fans**, then double-click **FanwLight**.



21. In the **Content1** window, place the graphic.



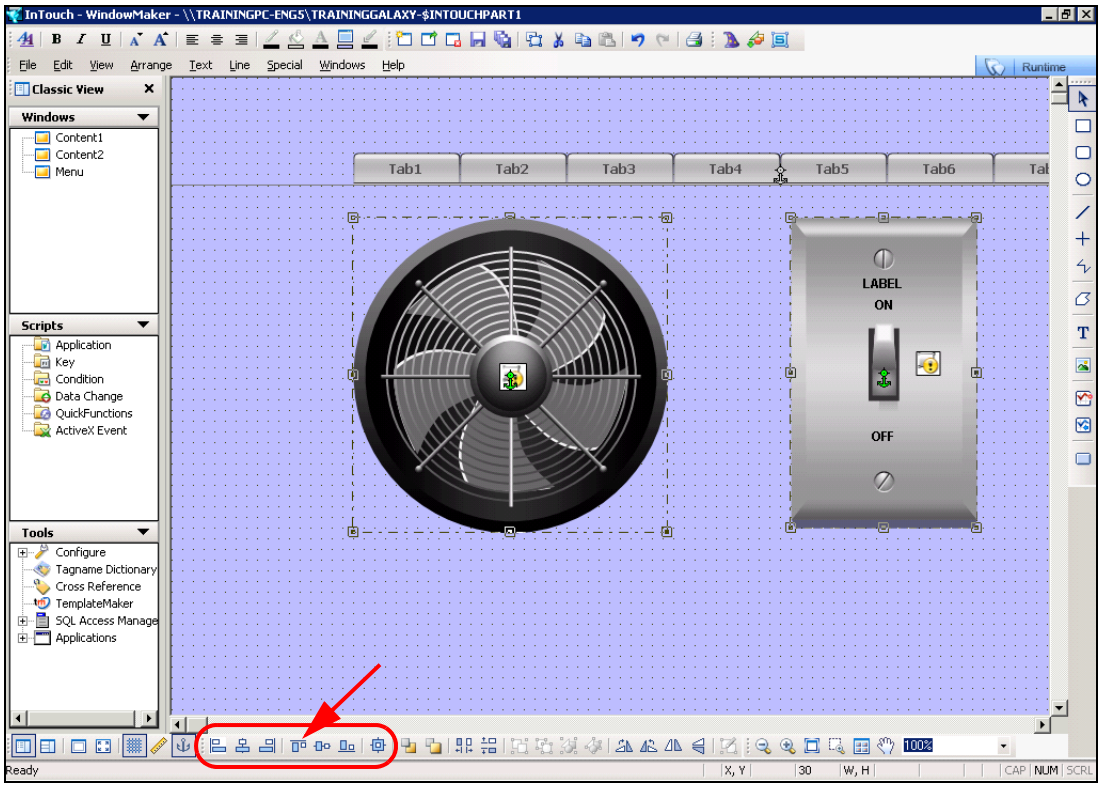
22. In **Graphic Toolbox**, expand **TrainingGalaxy\ArchestrA Symbol Library\Switches**, then double-click **LightSwitch**.



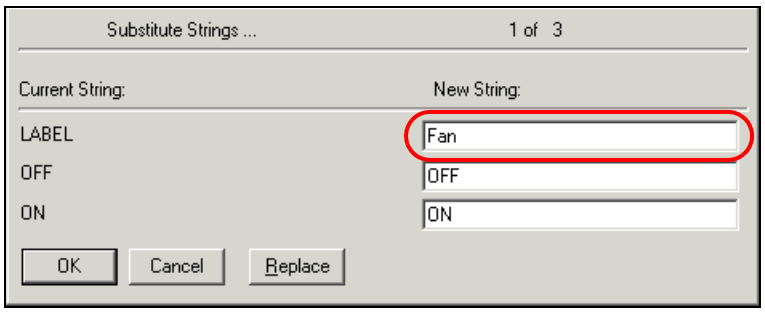
23. Place the **LightSwitch** to the right of the fan symbol.

You can easily align objects using the alignment options on the bottom toolbar.

24. Click the **FanwLight** graphic, then hold the **Shift** key and click the **LightSwitch** graphic to select both, then click **Align Top** .



25. Right-click the **LightSwitch** graphic and select **Substitute | Substitute Strings**.
26. Change the **LABEL** to **Fan**.



27. Click **OK**.
28. On the top toolbar, click **Save All**.



Link Graphics with an InTouch Tag

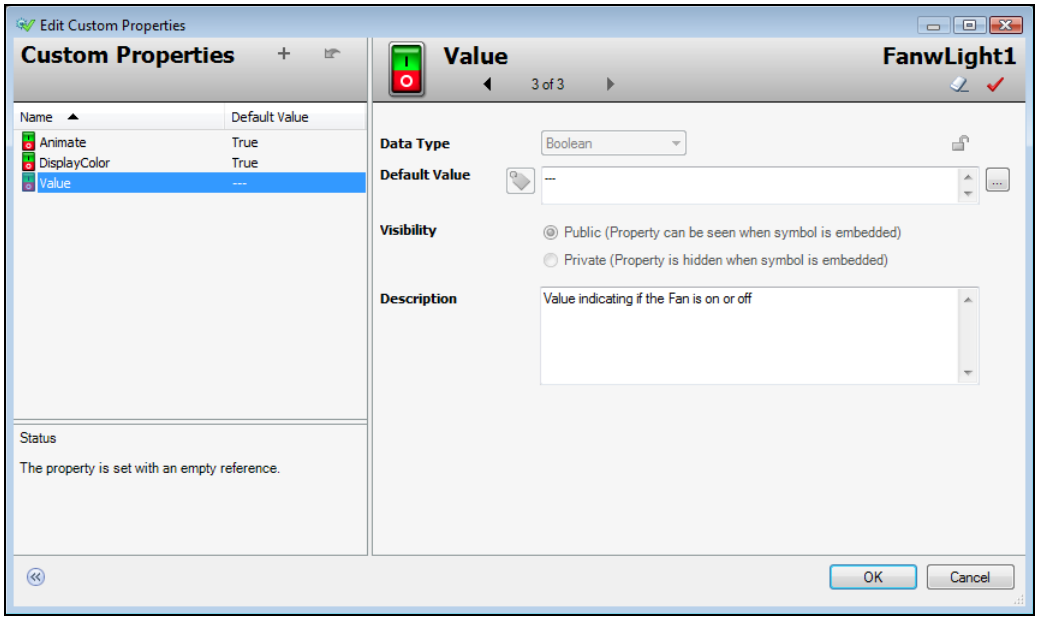
Next, add an InTouch tag to link the switch, the fan, and the light together.

29. In the **Content1** window, double-click the **FanwLight** graphic.

    The **Edit Custom Properties** dialog box appears.

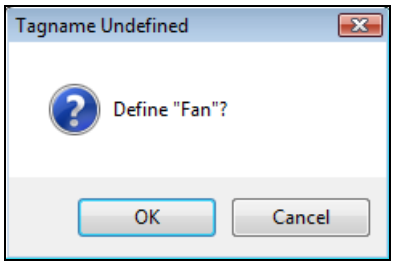
30. In **Custom Properties** pane, click **Value**.

    The **Value** properties display in the right pane.



31. In the **Default Value** field, replace **---** with **Fan**.
32. Click **OK**.

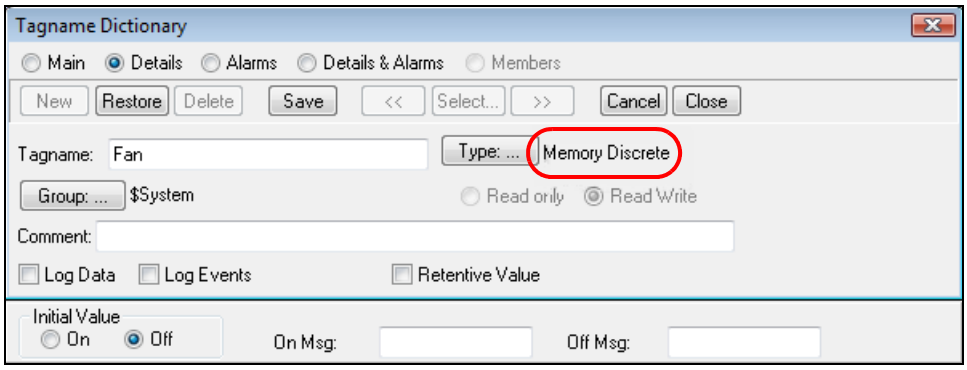
    The **Tagname Undefined** dialog box appears.



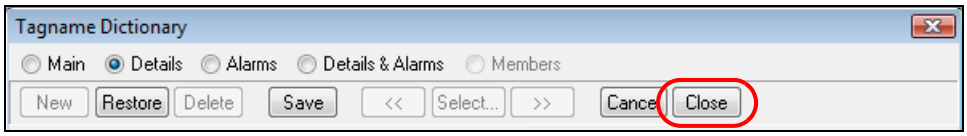
33. Click **OK** to define a tag named **Fan**.

    The InTouch **Tagname Dictionary** appears.

    Notice the tagname data **Type** is **Memory Discrete**.



34. Click **Close** to save and commit this tag.

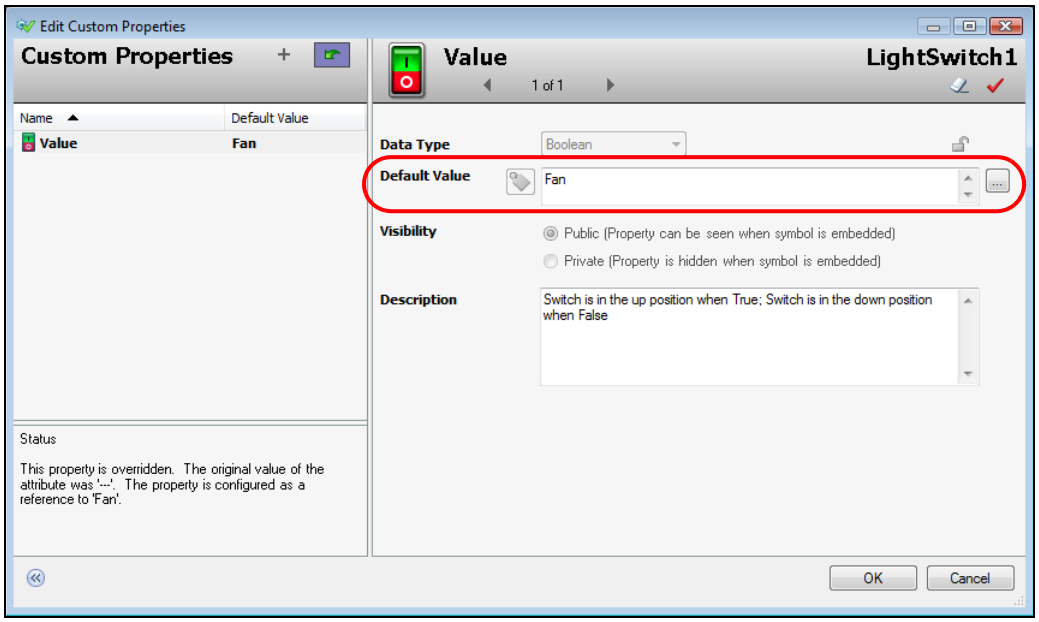


    The tag is saved in the **Tagname Dictionary**. You will reuse this tag for subsequent graphics.

35. In the **Content1** window, double-click the **LightSwitch** graphic to open the **Edit Custom Properties** dialog box.

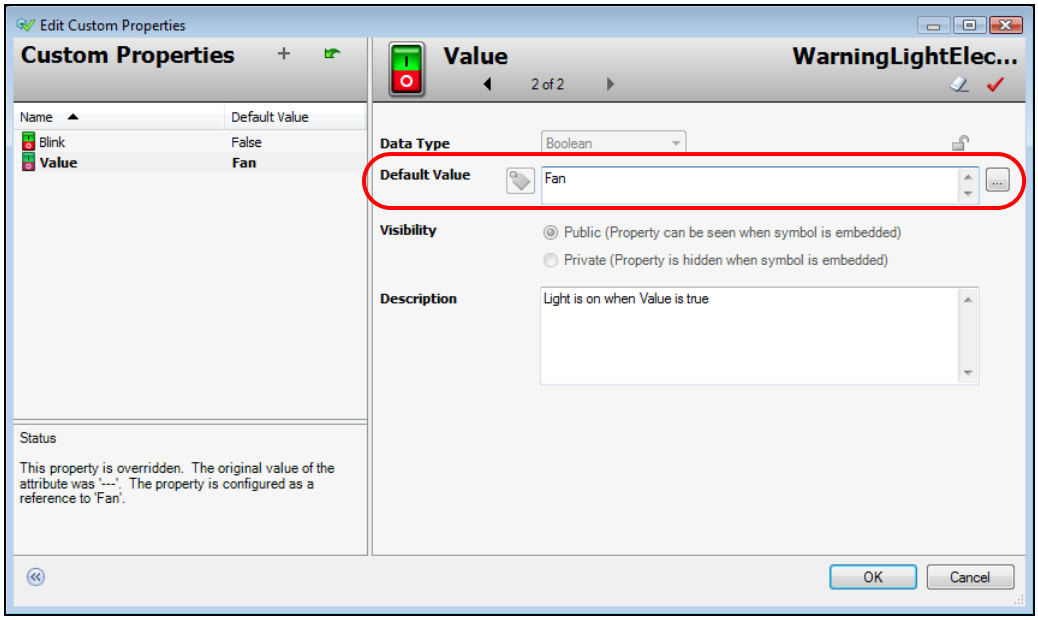
36. In **Custom Properties** pane, click **Value**.

37. In the **Default Value** field, enter **Fan**.

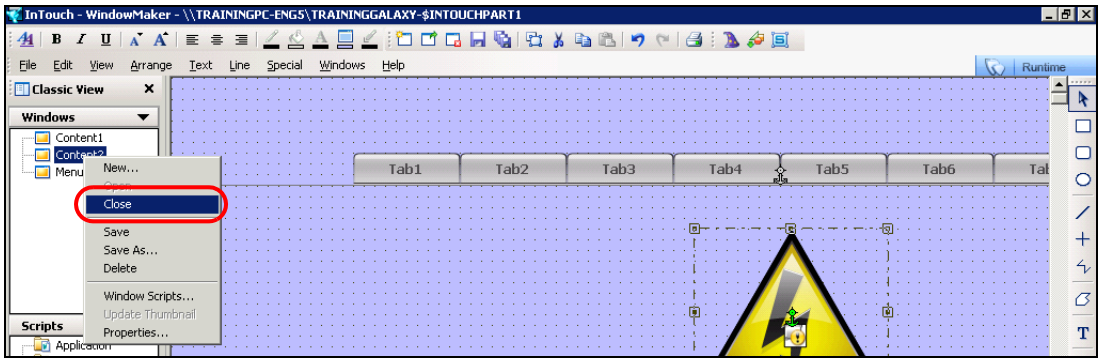


38. Click **OK**.

39. Open to the **Content2** window, then double-click the **WarningLightElectricity** graphic to open the **Edit Custom Properties** dialog box.
40. Click **Value**, then in the **Default Value** field enter **Fan**.



41. Click **OK**.
42. In **Windows** pane, right-click **Content2** and select **Close**.



A warning prompts you to save your changes.

43. Click **Yes**.

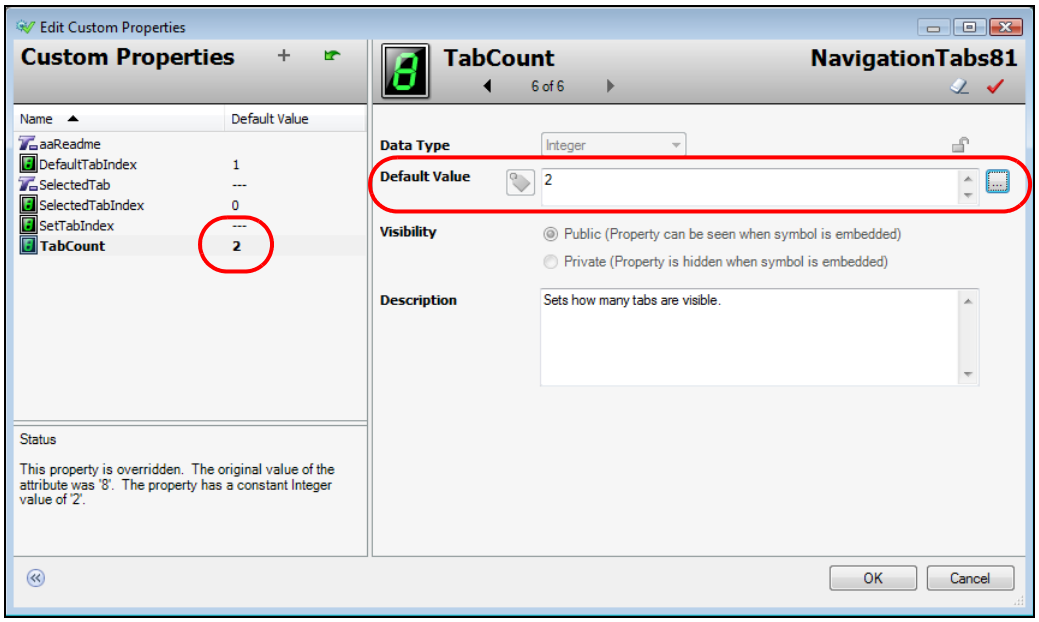
Build Application Navigation

Now you will change the labels on the tabs and modify the number of visible tabs in the graphic.

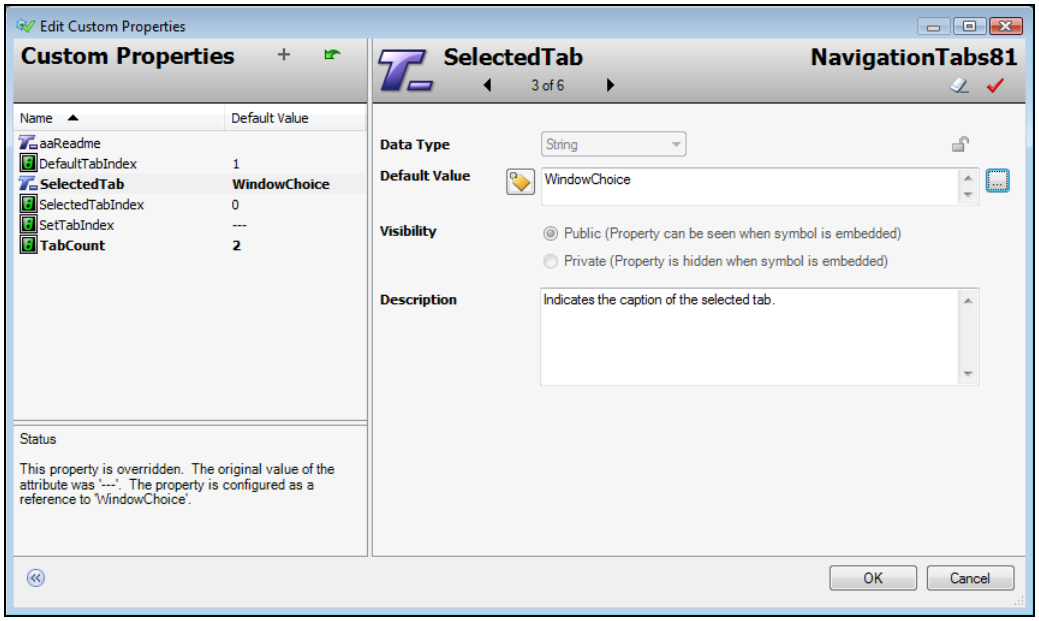
44. In the **Menu** window, double-click **NavigationTabs8** to open the **Edit Custom Properties** dialog box.

45. In **Custom Properties**, click **TabCount** and change the **Default Value** to **2**.

Note: This will only affect the appearance of the tabs in runtime.

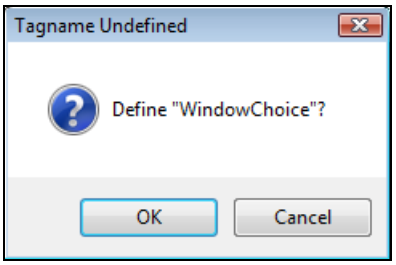


46. In **Custom Properties**, click **SelectedTab** and change the **Default Value** to **WindowChoice**.



47. Click **OK**.

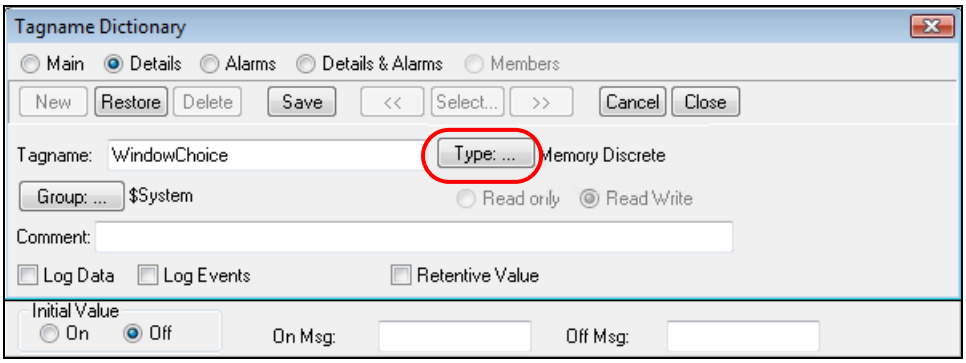
    The following message displays:



48. Click **OK** to define a new **WindowChoice** tag.

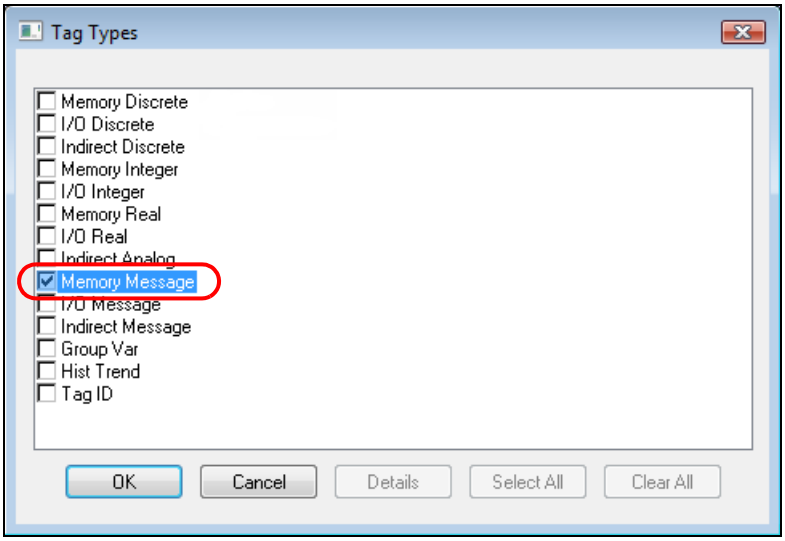
    You will use this tag to monitor the tab a user selects on the navigation bar and determine which window to open at runtime.

49. Click **Type** to define a data type for this tag.



    The **Tag Types** dialog box appears.

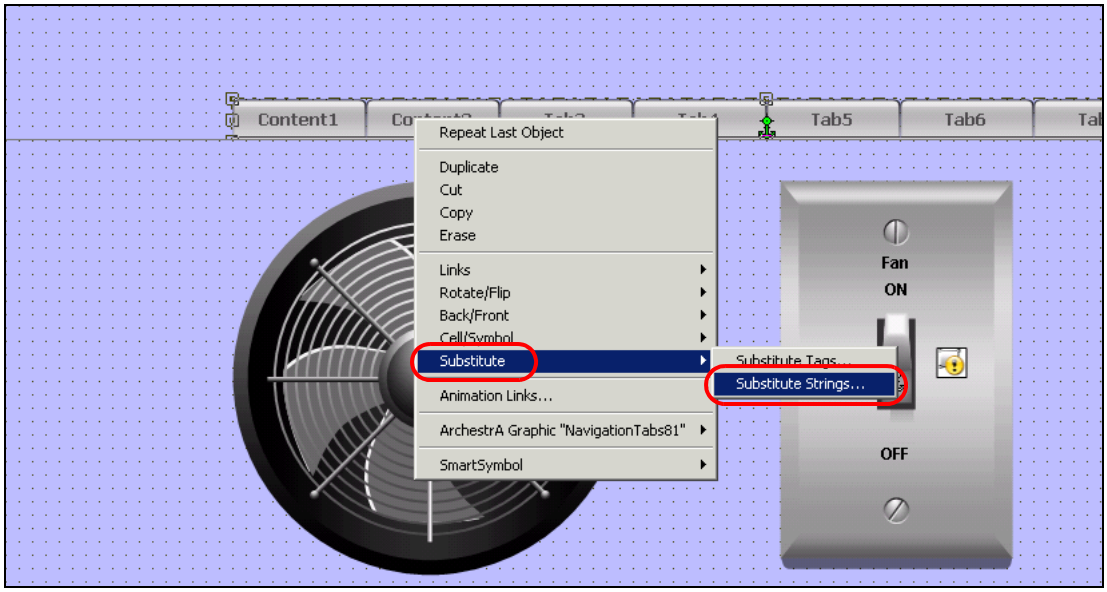
50. Check **Memory Message**.



51. Click **OK**.

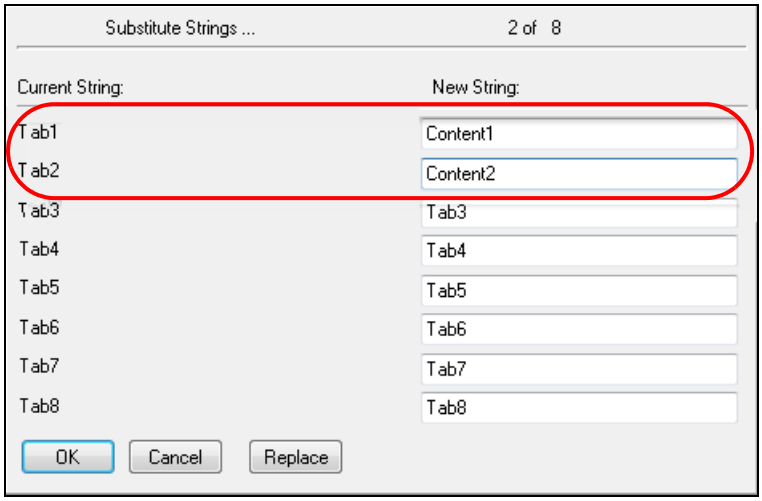
52. Click **Close** to save and commit this change.

53. Right-click the **NavigationTabs8** and select **Substitute | Substitute Strings**.



This opens the **Substitute Strings** dialog box.

54. In the **New String** field for **Tab1**, change the label to **Content1**.
55. In the **New String** field for **Tab2**, change the label to **Content2**.

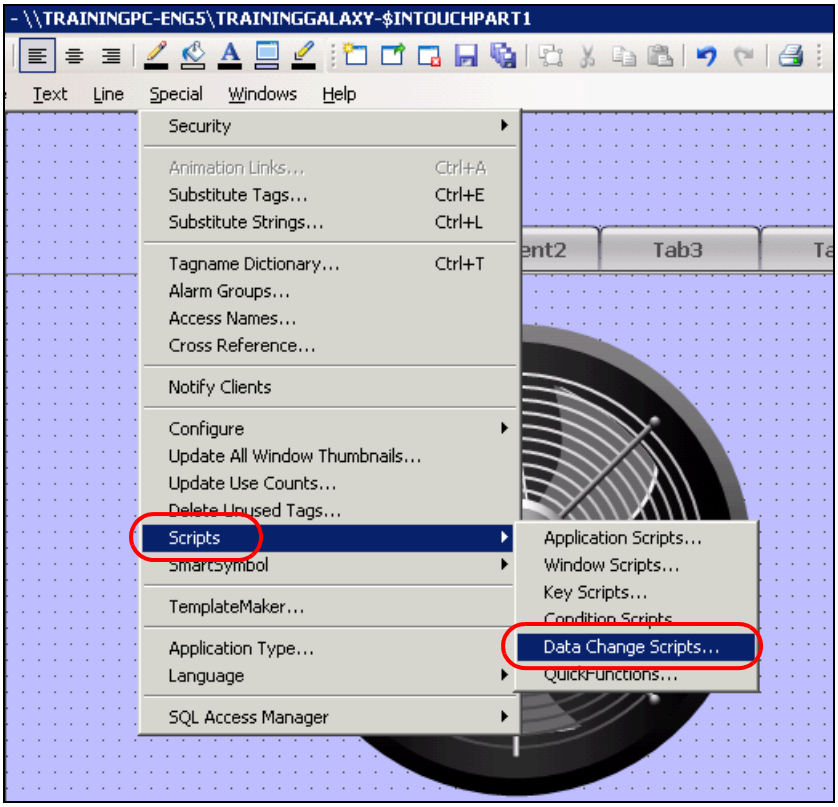


56. Click **OK** to return to the main window.

Create a Script

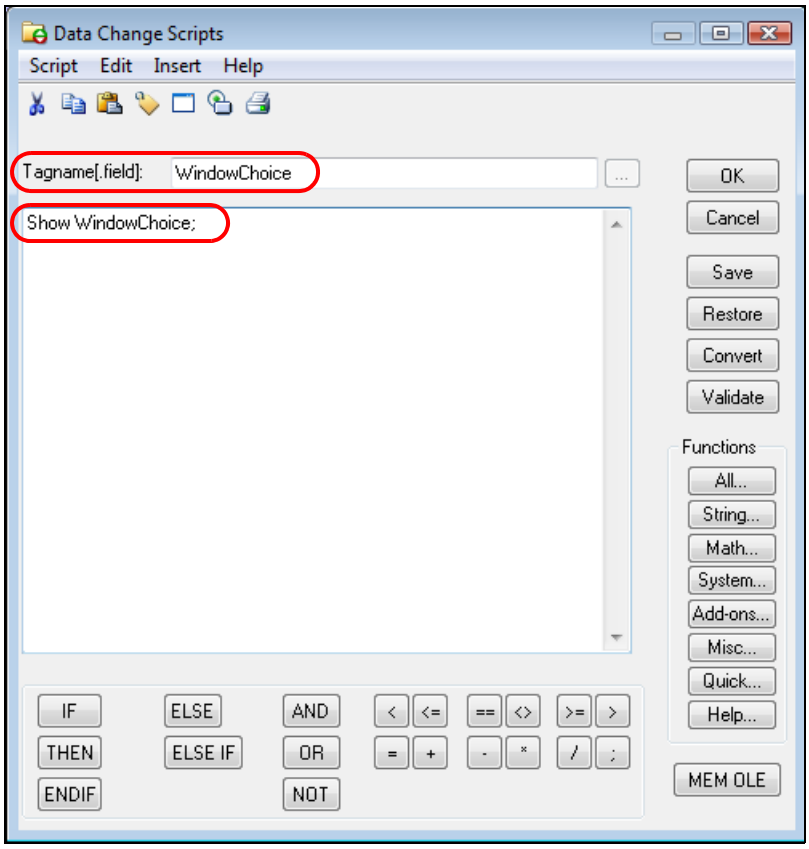
Next, you will create a script to monitor the selection a user makes on the navigation tabs to open the appropriate window.

57. On the **Special** menu, select **Scripts | Data Change Scripts**.



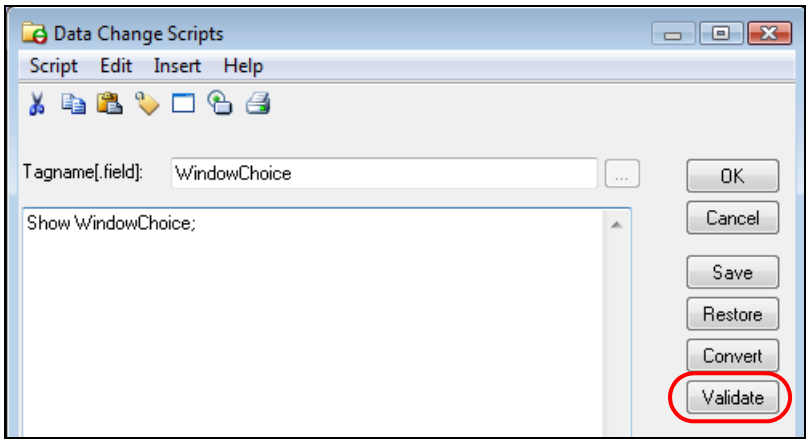
The **Data Change Scripts** dialog box appears.

58. In **Tagname[.field]**, enter **WindowChoice**.
59. In the body of the script, enter **Show WindowChoice**;



The **Show** command is the built-in function to display a window.

60. Click **Validate** to ensure the script does not contain errors.



61. Click **OK**.

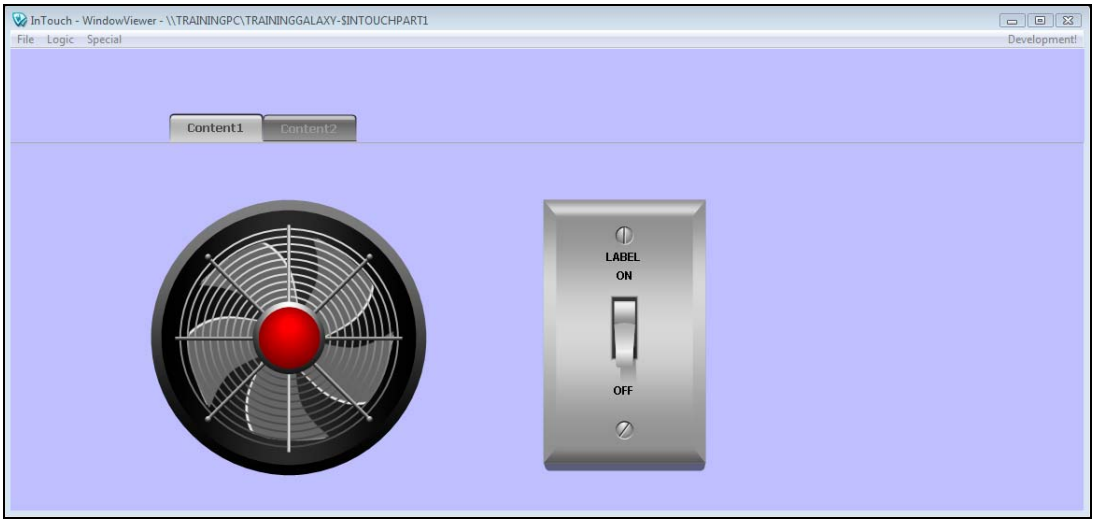
Test in Runtime

You are now ready to test your application in runtime.

62. On the WindowMaker menu bar, click **Runtime**.



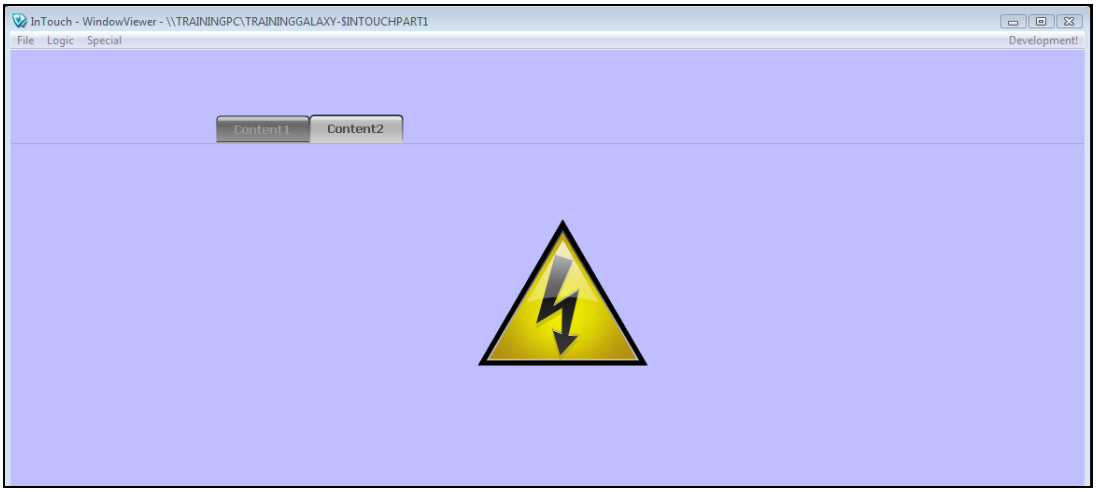
The application opens in WindowViewer.



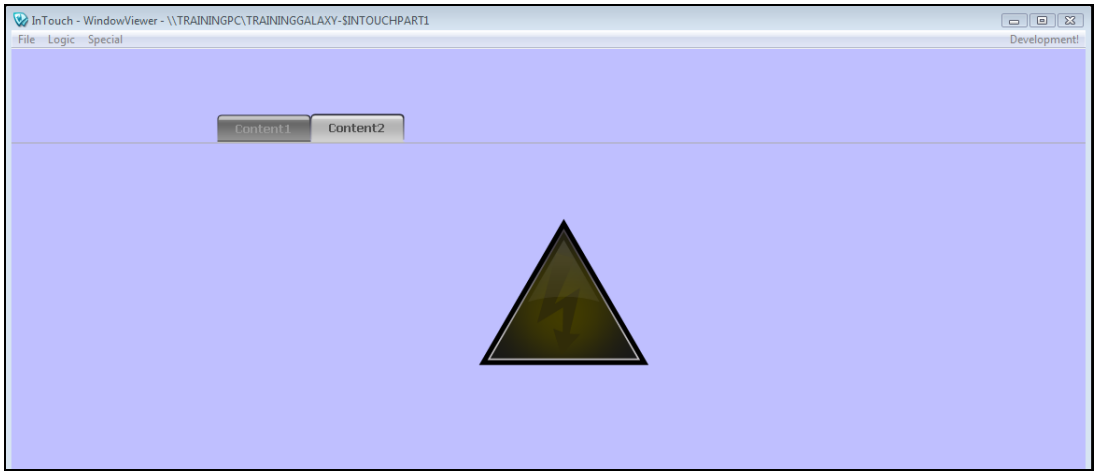
63. Click the light switch to turn on the fan.

    Notice that the blade rotates and the center of the fan changes to green.

64. Click the **Content2** tab to confirm the warning light is on.



65. Return to the **Content1** tab and click the light switch to turn off the fan.
66. Click **Content2** and confirm the warning light is off.



Section 5 – Tagname Dictionary and Data Sources

This section explores tags and the Tagname Dictionary in more detail. It also describes data sources and how they are used, and how to create data Access Names.

Overview

At the most basic level, a Tag is a named value. Tags are used to store and manipulate values and their properties, such as minimum and maximum ranges or alarm information. Defining tagnames and their properties is done in the InTouch Tagname Dictionary**.** For communication bandwidth, storage size and efficiency, and proper handling, each tag is defined with a specific data type, which confines that tag to a specific usage. In runtime, WindowViewer loads and maintains a copy of the dictionary, which is updated with all current values and properties for these tagnames.

During runtime, an InTouch application accesses the Tagname Dictionary and provides specific tag information via animation links, and InTouch subsystems such as Alarms, History, Security, and scripting.

You can set up an InTouch application to identify an element of data stored on another node by using a three-part addressing convention. This convention includes the node, application, and topic names. To obtain data from a remote node, you must configure an **Access Name** for your InTouch application that specifies these three items.

You must associate InTouch I/O tags or remote tag references with an Access Name. An Access Name defines a communication link with another I/O data source. Each Access Name specifies an I/O address consisting of a node name, an application name, and a topic.

You can develop distributed applications in which the functional components of an InTouch system are located on different nodes, and then specify the type of data located on the remote node. This information determines the I/O type for the tag when it is defined in the Tagname Dictionary.

Tags, including I/O configuration, and Access Names are described in more detail in the following sections.

Memory Tags, I/O Tags, and $System Tags

InTouch data tags, which include discrete, integer, real, and message, are categorized as either **Memory** tags or **I/O** tags. Additionally the dictionary has predefined tags called **$System** tags that have very useful built-in functionality, such as Date and Time information.

When WindowViewer acts like an I/O client and requests or writes data with an external I/O aware application, this can only be done using an I/O tag; memory tags cannot be defined to read or write data with an external source.

WindowViewer is also an I/O server and it does not distinguish between $System Tags, Memory Tags, or I/O Tags when handling a request to provide data to an external I/O client requesting that tag. Data from $System Tags, Memory Tags, and I/O Tags can be served to an external I/O client.

Memory Tags

Memory Tags are tags that are used internally within your InTouch Application and do not acquired their data from a remote source. For example, Memory Tags can store the results from a mathematical calculation performed in an InTouch QuickScript. WindowViewer can act as an I/O server and provide this result to an external client application requesting this Memory Tag. Memory Tags are also used in animations such as a counter used to control a Location or Fill animation.

Memory Tags can be used as a logic state flag in a QuickScript or used to track the toggle state of an action pushbutton. Memory Tags also serve as simulation data during the design phase of an InTouch application where linking to I/O data is not yet possible or presents a risk until the design is tested and proven.

There are four **Memory** types:

- **Memory Discrete**: Internal discrete tag with a value of either 0 (False, Off or Low) or 1 (True, On or High)
- **Memory Integer** (Analog): A 32-bit signed integer value in whole numbers
- **Memory Real** (Analog): Floating (decimal) point value
- **Memory Message**: Text string tag that can be up to 131 characters

I/O Tags

All tags that read or write their values to or from an external source are I/O tags. This includes all inputs and outputs from programmable controllers, process computers, and data from network nodes.

I/O tags are accessed either through the Microsoft Dynamic Data Exchange (DDE) or Wonderware SuiteLink communication protocols. SuiteLink is recommended for network communication and diagnostics.

When the value of a read/write I/O tag changes, it is immediately written to the remote application. The tag can also be updated from the remote application when the item to which the tag is linked changes in the remote application.

By default, all I/O tags are set to **Read/Write**. However, you can restrict them to read only by selecting the **Read Only** option in the **Tagname Dictionary** dialog box.

There are four **I/O Types**:

- **I/O Discrete**: Discrete input/output tag with a value of either 0 (False, Off) or 1 (True, On)
- **I/O Integer** (Analog): A 32-bit signed integer value
- **I/O Real** (Analog): Floating (decimal) point tag
- **I/O Message**: Text string input/output tag that can be up to 131 characters long

$System Tags

There are a number of pre-built $System Tags in the Tagname Dictionary. These $System tags provide data for security, date information, time information, and miscellaneous functions, such as determining the vertical and horizontal position of the current selection. $System tags cannot be deleted or renamed.

Some $System tags allow data to be written to them such as **$OperatorEntered** and **$PasswordEntered** allowing these tags to be used to create a login feature for InTouch Security. One tag, **$ConfigureUsers**, opens the configure users security dialog box when a value of **1** is written to the tag, but only if the user has the proper level of access. Several of the $System tags are also predefined to log their events to the Alarms and Events subsystem when it is enabled. For example, each time a new alarm occurs, the **$NewAlarm $System** tag is set to **1**, and this incident is recorded in the **Event** history.

Special Purpose Tag Types

There are several special tag types that you can assign to tags that perform complex functions, such as creating dynamic alarm displays, historical trends, monitoring or controlling the tag that each historical trend pen is plotting, and so on.

Group Var

Use the Group Var type for a tag with an assigned **Alarm Group** to create dynamic alarm displays, disk logs and print logs. Group Var tags create alarm windows or alarm logs that display all alarms associated with a specific group variable. You can also control the alarms that are displayed or logged by assigning a different Alarm Group to the Group Var tag.

You can also use a Group Var tag to create application buttons the operator can click to selectively display alarms for different areas of a plant within the same alarm window. All of the .fields associated with Alarm Groups can be applied to Group Var tagnames.

Hist Trend

InTouch requires a Hist Trend tag type when you create a historical trend. All of the .fields associated with historical trends can be applied to Hist Trend tags.

Tag ID Tags

Use Tag ID tags with historical trend objects. You use Tag ID tags to retrieve information about tags being plotted in a historical trend. In most cases, you would use Tag ID tags to display the name of the tag assigned to a specific pen or to change the tag assigned to the pen.

Indirect Tags

Indirect tags allow you to create one window, and reassign the tags in that window to different sources. When you assign a source tag to an indirect tag through scripting, the indirect tag becomes linked to the source tag and allows you to access the value and .fields of the source tag. The indirect tag can then be used to represent the source tag in animations, graphics, and scripts.

Using indirect tags in an application design reduces the number of windows that must be created to represent similar processes, allowing the reuse of windows, animations, and scripts. A window designed with indirect tags should provide a method of navigation for an operator. The operator makes a navigation selection and a script runs, which changes the source tags linked to the indirect tags on the window. This allows operators to view any one of a group of similar processes, rooms, devices, or equipment without closing the current window.

For example, a window designed using indirect tags is showing the cooling equipment in a cold storage room of an industrial refrigeration facility with dozens of similar cold rooms. On the cold storage window, the operator selects Whole Bird Chicken Room 27 from a Listbox and instantly the graphics, animations, and scripts on the current window are showing the selected cold room.

By choosing which room the operator wants to view, a script runs that changes source tags linked to the indirect tags used on the window. Thus the window, graphics, animations, and scripts in a single window are reused over and over to view dozens of cold storage rooms.

If an indirect tag has its retentive value property enabled in the dictionary, its link to the source tag will persist between restarts of WindowViewer.

SuperTags

A SuperTag is a structure of tags that allow you 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 segments of plants with identical or similar equipment in multiple quantities.

Tag Naming Conventions

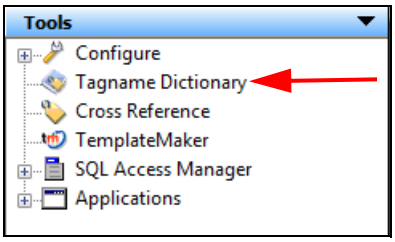
When creating tagnames, consider the following naming conventions:

- The maximum length is 32 characters
- Spaces are not permitted
- If you start with a number, do not follow with another number or a function character
- The backlash character "\" is reserved for supertags
- You must use at least one alpha character

Creating Tags

Tags and their tagnames (and other attributes) are created using the Tagname Dictionary's **Tag Editor**.

Use **Special | Tagname Dictionary** from the main WindowMaker menu or double-click on the **Tagname Dictionary** in the **Tools** panel of Application Explorer.

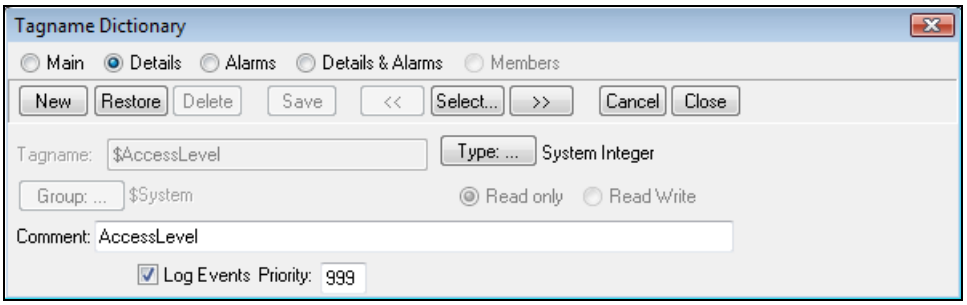


There are several ways to open the **Tagname Dictionary**:

- **Tools** pane shortcut: **Tagname Dictionary**
- **Menu** bar: **Special | Tagname Dictionary**
- **Keyboard** shortcut: **Ctrl+T**

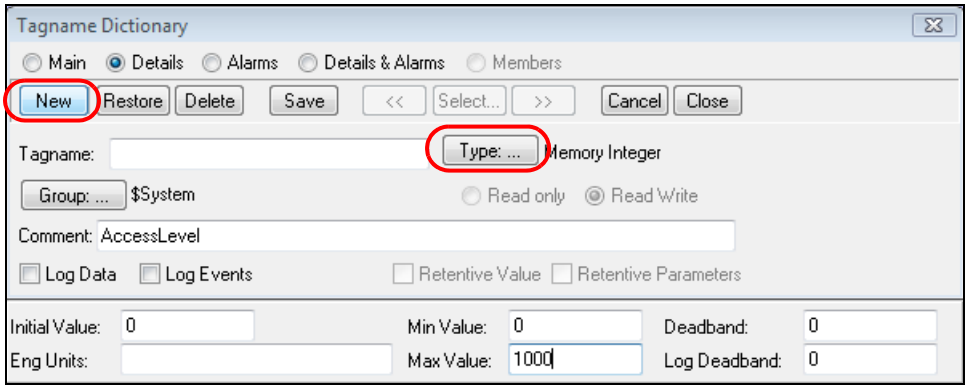
The first time you access the Tagname Dictionary, the definition for the internal system tagname **$AccessLevel** appears. This is the first tag in the **Tag Browser** list.

After saving the tag definition in the **Tag Editor,** the definition for the tag you edited last is displayed.

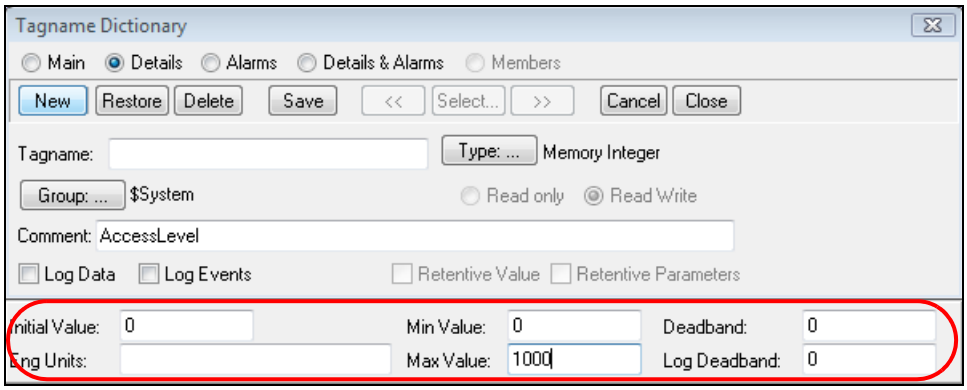


While in the **Tag Editor**, use the left [<<] and right [>>] browse buttons to browse one tag at a time through the tagname definitions currently stored in your Tagname Dictionary.

To create a tag, use the **New** button and enter a unique tagname. Use the **Type** button to select the tag type from a list of available options.



Many tags require greater level of detail to be properly handled. A specific **Details** area is displayed for each type of tag to define the details and alarm types for the tag type. This section is seen only if the **Details** or **Details & Alarms** radio buttons are selected at the top of the **Tagname Dictionary**.



The fields which are shown in the details area vary based on the tags type selection.

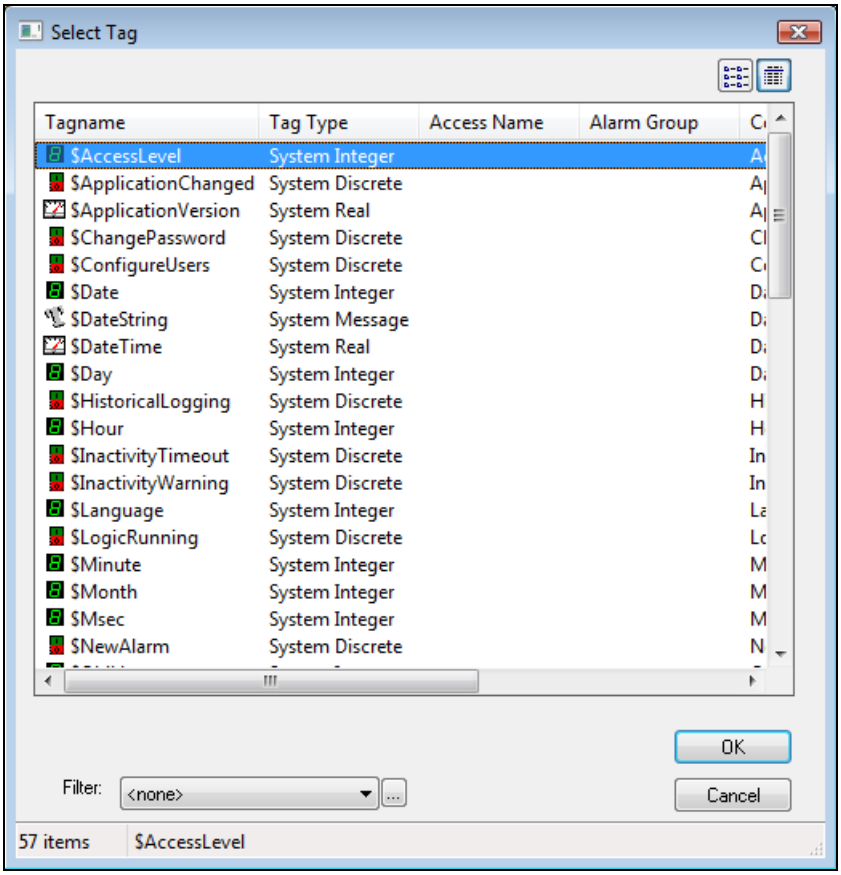
Tag Browser

The Tag Browser displays all tags contained within the Tagname Dictionary. This is your primary tool for viewing and selecting local and remote tagnames and tagname Dot Fields from applications or any other tag source that supports the InTouch **T**agname Dictionary interface. It enables selection of existing tags, addition of new tags and viewing of basic Tagname Dictionary information.

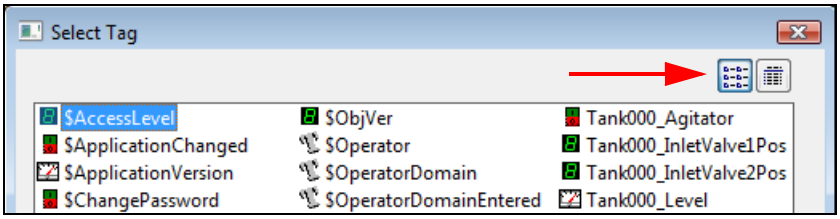
The Tag Browser is also used to access the Tag Editor to perform editing, replication and configuration of remote references to remote tag sources. The first time the Tag Browser is accessed, by default, **<local>** is selected for the tag source. This means that the tagnames in the local application Tagname Dictionary are displayed. Thereafter, the last accessed tag source's tagnames are displayed.

The Tag Browser operates in either **Dictionary** or **Selection** mode. The mode is determined by the method used to access it.

From the **Tagname Dictionary**, the **Select** button opens the **Select Tag** dialog box in Dictionary mode.



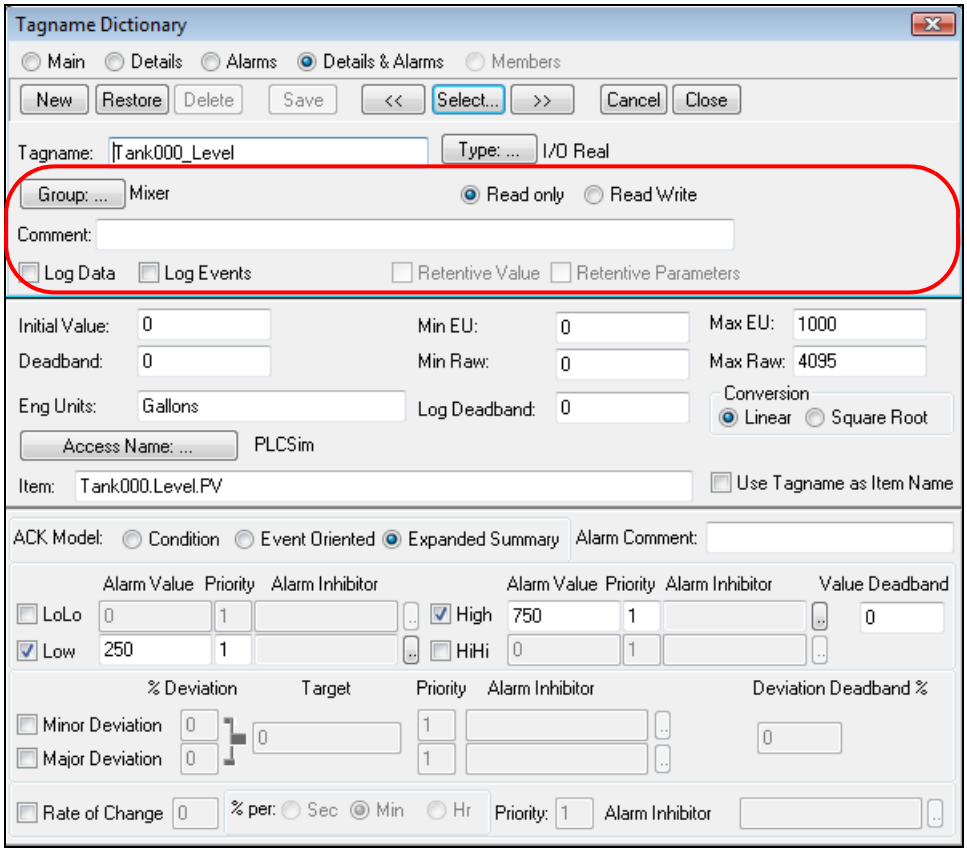
Click the **List View** button to remove the detail columns from the **Select Tag** dialog box.



The **Select Tag** dialog box displays without tag details.

Configuring Tags

The following configuration options are available in the **Tagname Editor**.



Group

The group button assigns the tag to a specific Alarm Group. Click this button to display the **Alarm Group** dialog box and a list of available alarm groups or create new alarm groups.

- If you do not assign the tag to a specific Alarm Group, InTouch assigns it to the root group $System by default
- Once you create a tag and assign it to an Alarm Group

Read Only and Read Write

Read Only and Read Write are used to set restrictions for I/O tags. These options are disabled for Memory tags that cannot be restricted to Read Only, all Memory tags are writable.

Read Only is selected if you want to restrict an I/O tag to prevent writing values where prohibited during runtime.

Read Write is selected to grant the I/O tag both read and write capabilities in runtime.

Comment

Contains any miscellaneous comment you want the system to store regarding your tag. The comment cannot exceed 50 characters.

Log Data

Logs the tag data to the historical log file during runtime when its engineering unit value changes more than the specified **Log Deadband** value or, by default, once an hour, regardless of change.

- For tag values to be logged, you must enable historical logging by selecting **Special | Configure Historical Logging** from the main WindowMaker menu.
- If this option is disabled later (tag value NOT logged), the data previously logged for the tag will be inaccessible to InTouch in historical trend displays.
- If the option is re-enabled, all logged data is accessible, but a historical trend would display a gap in the trend trace.
- Also, if you make logging changes in WindowMaker while WindowViewer is running, the changes do not take effect until WindowViewer is restarted.

Log Events

Logs all data changes to the tag that are initiated by the operator, I/O, a QuickScript, or by the system to the alarm system.

- When you define a tagname to do event monitoring, an event message is logged to the Alarm System each time the tagname value changes. The event message logs how the value changed and whether the change was initiated by the operator, I/O, scripts, or the system.
- When the **Log Events** option is selected, the **Priority** field becomes active. Valid entries in this field are **1** to **999**, where 1 is the highest and 999 is the lowest priority. Default is 999.

Retentive Value

Retains the current value of the tag when WindowViewer is closed. This value is used as the initial value for the tag when WindowViewer is restarted. Retentive values cannot be selected or cleared for new or existing tags when WindowViewer is running.

Retentive Parameters

Retains any changes the operator makes to the value of any alarm limit fields for the tag. This value is used as the initial value for the alarms when WindowViewer is restarted.

Alarm Comment

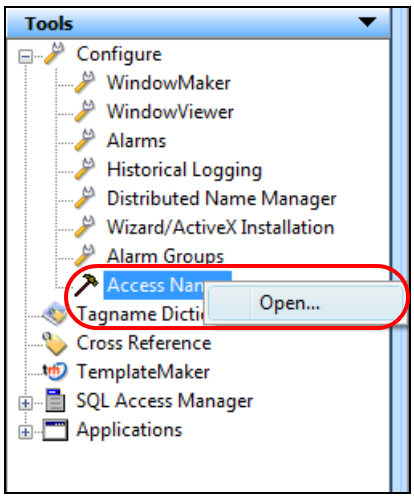
You can configure your alarm windows to display these comments when the tag is in alarm. You can configure the distributed alarm system to use the tag's **Alarm Comment** field to store an operator's comments regarding an acknowledged alarm.

Examining InTouch Access Names

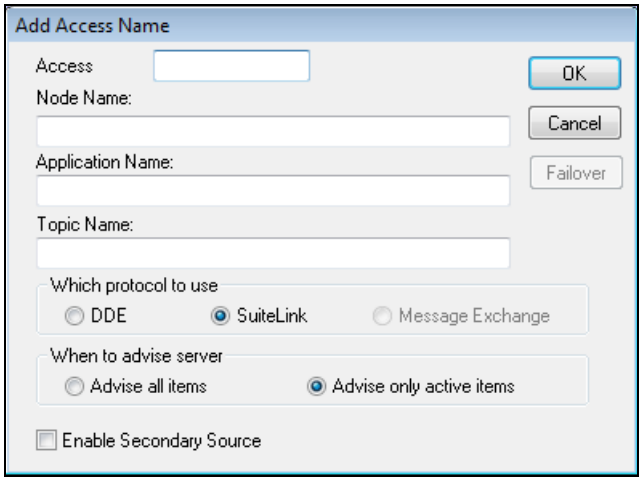
When you create I/O tags or remote tagname references, you must associate them with an Access Name. Access Names contain the information that is used to communicate with other I/O data sources including the node name, application name and topic name.

Creating an Access Name

To configure an Access Name, expand the **Configure** option from the **Tools** panel of Application Explorer (**Classic View**), right-click **Access Names**, and select **Open**.

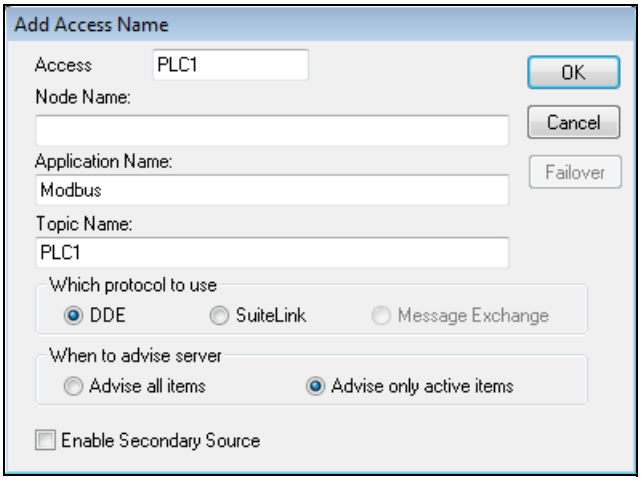


From the **Access Names** dialog box, click **Add**. At the **Add Access Name** dialog box, enter the requisite information.



Option	Purpose
Access Name	Enter the new Access Name (for simplicity, use the same name that you will use for the Topic Name. InTouch uses Access Names to reference real-time I/O data. Each Access Name is associated with an I/O address, which can contain a Node, Application, and Topic. In a distributed application, I/O references can be set up as global addresses to a network I/O Server or local addresses to a local I/O Server.
Node Name	Used when the data is from a remote I/O Server over the network.
Application Name	Enter the actual program name for the I/O Server program from which the data value will be acquired. **Note:** Do not enter the **.exe** extension portion of the program name.
Topic Name	Enter the <topic name you wish to access>. The **Topic Name** is an application-specific subgroup of data elements. In the case of data coming from a Wonderware I/O Server program, the topic name must be the same name defined in the I/O server.
Which Protocol to use	Select DDE (local node only) or SuiteLink (Wonderware proprietary network protocol).
When to advise server	**Advise all items**: Select if the server program is to poll for all data whether or not it is in visible windows, alarmed, logged, trended or used in a script. Selecting this option impacts performance and is not recommended. **Advise only active items**: Select if the server program is to poll only points in visible windows and points that are alarmed, logged, trended or used in any script. An InTouch Pushbuttons action script is not polled unless it opens in a visible window.

The following image shows an example of a completed Access Name (if the I/O server is on the local node, the **Node Name** field can be left blank, as shown in the following figure).



Modifying or Deleting an Access Name

Display the **Access Names** dialog box (from the **Tools** panel of Application Explorer, right-click **Configure** and select **Open**).

At the **Access Names** dialog box, select the **Access Name** to modify, and click **Modify**.

Repeat this procedure if you need to modify other access names.

To delete an Access Name, in the **Access Names** dialog box, select the **Access Name** to be deleted, and click **Delete**. Access names used by tags cannot be deleted.

Examining InTouch I/O Addressing

InTouch identifies an element of data in an I/O Server program using a three-part naming convention that includes the **application name**, **topic name**, and **item name**. To obtain data from another application, InTouch (the client) opens a channel to the server program by specifying these three items. In addition, InTouch needs to know the data type: discrete, integer, real (floating point), or message (string).

This information determines the I/O type for the tagname when it is defined in the InTouch database. When WindowViewer is running, it automatically performs all of the actions required to acquire and maintain the value of this item.

When another DDE Aware Windows application requests a data value from InTouch, it also must know the three I/O address items. For example, if the application name is Excel, the topic name is the name of the specific spreadsheet that contains the data, and the item name is the identification of the cell on the spreadsheet where the date is written to or read from.

- **VIEW**
  The application name that identifies the InTouch runtime application WindowViewer (view.exe) that contains the data element.
- **TAGNAME**
  The topic name recognized by WindowViewer for reading/writing to any tagname in the InTouch database.
- **TAG**
  The item, any tagname memory or I/O defined in the Tagname Dictionary.
  For example, to access a data value in InTouch from Excel running on the same node, a DDE Remote Reference formula would be entered in the cell into which the data value is to be written. The application (VIEW) is separated from the topic (TAGNAME) by a pipe “|” symbol, the topic and the item ($Second) are separated by an explanation mark, and finally the item or tag must be surrounded by the apostrophe (sometimes called single quote) marks:
  =VIEW|TAGNAME!'$Second’
  When this formula is placed in a cell in Excel, this example retrieves the current seconds of the internal clock as seen by WindowViewer.

Defining an I/O Tagname

Tagname is the name used within InTouch to refer to a data value (tag). The Item is the name used by a remote Windows application to refer to the same value.

All I/O tags receive their values from other Windows application programs, such as Excel and I/O Servers. This value is referred to as the **Raw** value. When you define a tag in the Tagname Dictionary, you must enter values for the **Min Raw** and **Max Raw**. These values are used by the database as clamps on the actual raw value received from the I/O device. For example, if you set the Min Raw value to 50 and the actual value received from an I/O Server is 0, database will force the Raw value to 50.

InTouch does not display raw values. Instead, it displays **Engineering Units (EU)**. When you define an I/O tag in the Tagname Dictionary, you must specify values for the **Min EU** and **Max EU**. These values are used to scale the raw value to the displayed value. If you do not want to do scaling or your I/O device does the scaling for you, set the **Min/Max EU** values equal to the **Min/Max Raw** values.

For example, assume that a flow transmitter wired to a PLC register generates a value of zero at no flow and a value of 9999 at 100% flow. The following values would be entered:

Min EU = 0 Max EU = 100

Min Raw = 0 Max Raw = 9999

A raw value of 5000 would be displayed as 50.

Assume that a flow transmitter wired to a PLC register generates a value of 6400 at no flow and a value of 32000 at 300 GPM.

Min EU = 0 Max EU = 300

Min Raw = 6400 Max Raw = 32000

In this case, a Raw value of 12800 would be displayed as 150. A raw value of 6400 would be displayed as 0 and a Raw value of 0 would be displayed as 0. All values outside the boundaries set by the Min Raw and Max Raw values are clamped.

The above scaling works in reverse when the I/O tag data is written from the InTouch Tagname Dictionary to other Windows applications.

To add an I/O tag, open the **Tagname Dictionary**. Ensure that the **Details** radio button is selected, and add a new tag by completing the following:

- **Tagname**: enter a name for the new tag
- **Tag Type**: I/O Integer

The I/O Integer tag **Details** is displayed.

Upon returning to the **Tagname Dictionary**, specify all the required details for defining the **Item**, as follows:

- **Access Name** – Select the Access Name to be used or define a new Access Name.
- **Item** – Enter the item name or number for the data value in the I/O Server application or associated application. If the Item name is the same as the tagname, select the **Use Tagname as Item Name** check box.

Section 6 – Tag Viewer

This section describes the Tag Viewer and how to view, save, and import watch windows. It also discusses Dot Fields and how to interpret value, time, and quality information in a watch window.

Overview

Tag Viewer is an external application that allows you to watch and monitor tags and modify tag values at runtime. It provides you with a list of all available tags in the application arranged hierarchically, based on their alarm groups. It displays tags available in the local InTouch application and does not support remote references.

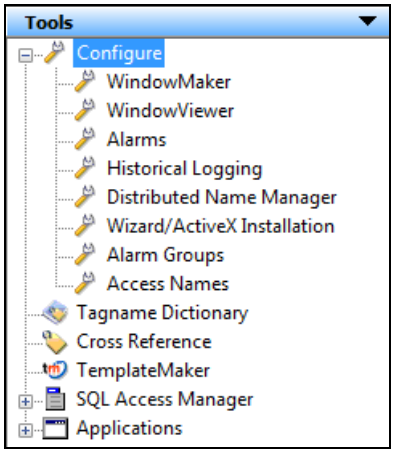
Tag Viewer is useful during engineering trouble shooting and diagnostic testing. It also reduces application development and diagnostic time, and enhances application stability and reliability by providing real-time direct runtime access to the Tagname Dictionary.

To use Tag Viewer, you must first enable it in WindowMaker. You can then start the application at runtime. It can be started through a script or the special function menu.

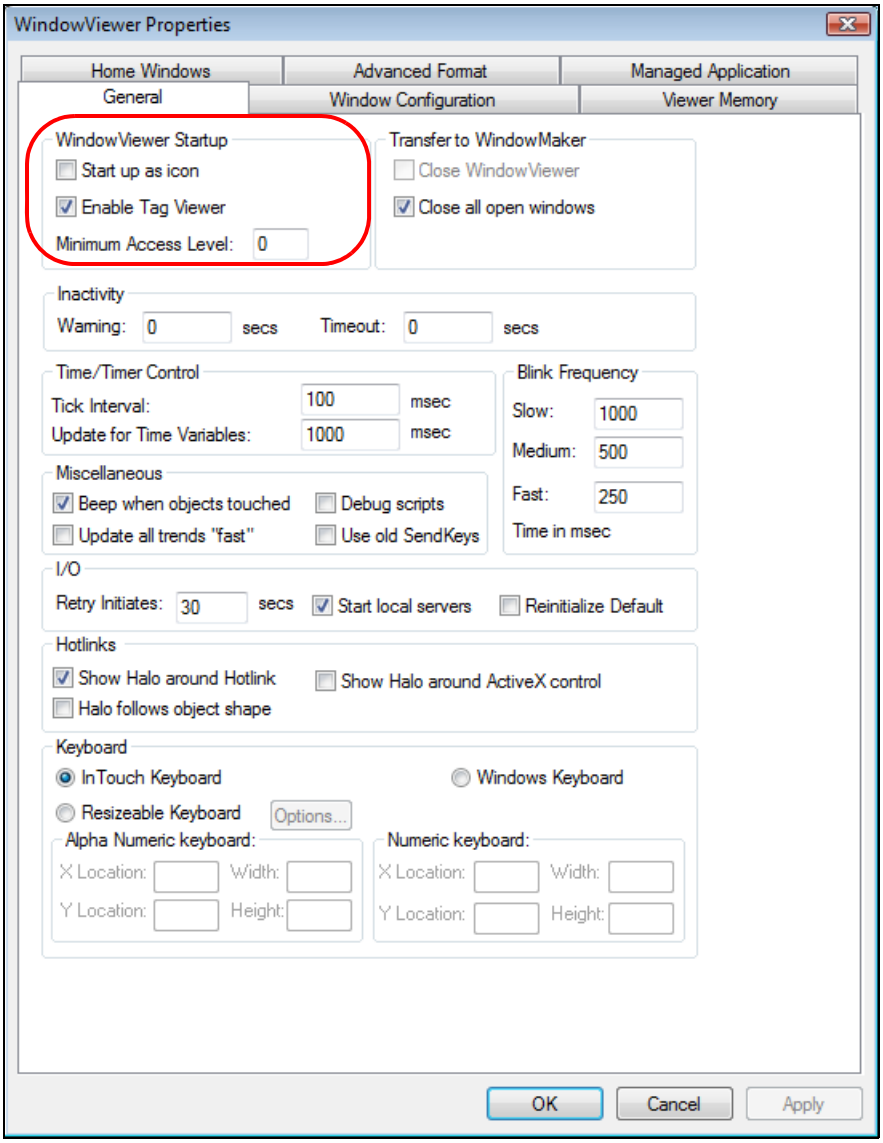
Enabling Tag Viewer

You can configure the WindowViewer properties at design time to enable Tag Viewer to run. You can also configure the WindowViewer menu so that the **Tag Viewer** option is displayed in the **Special** menu.

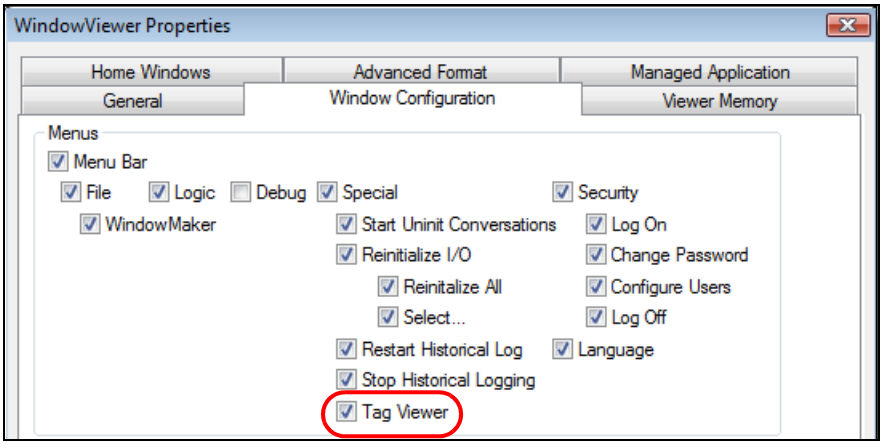
Access WindowViewer configuration options in the **Tools** pane to enable **Tag Viewer**.



On the **General** tab, check **Enable Tag Viewer** and set a minimum security access level.



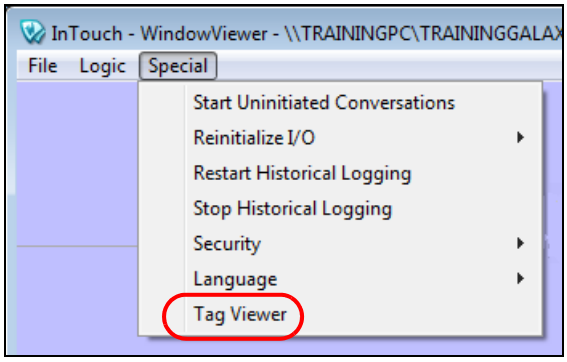
On the **Window Configuration** tab, add **Tag Viewer** to the WindowViewer **Special** menu.



If WindowViewer is already open, you must restart WindowViewer for your changes to become effective.

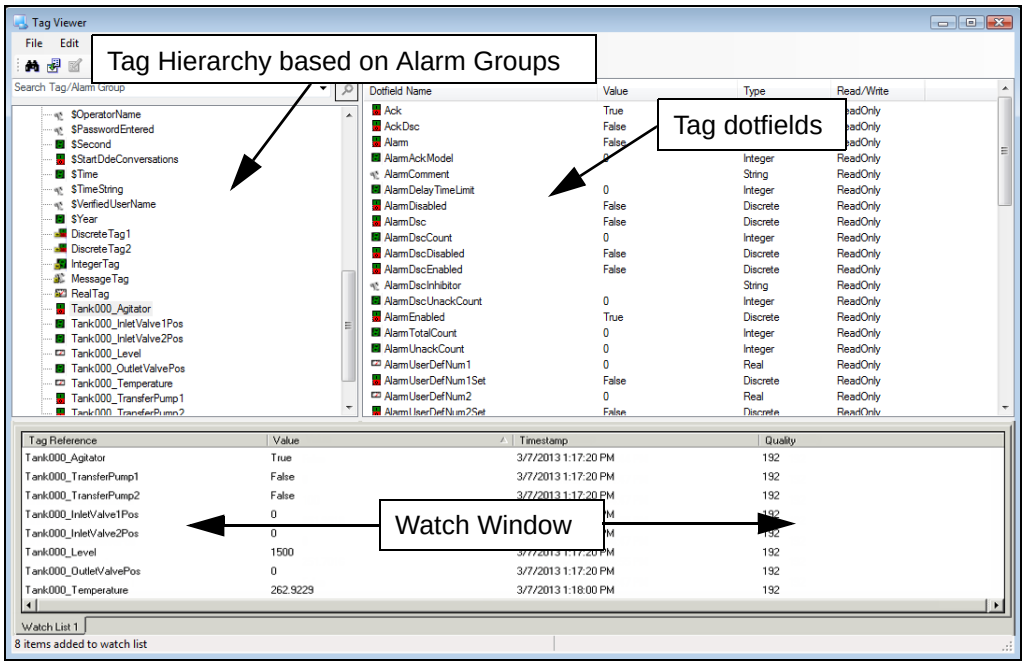
Starting Tag Viewer

After configuring WindowViewer to enable **Tag Viewer**, you click the **Special** menu, and select **Tag Viewer**.



The Tag Viewer window contains three parts:

- The left pane, which displays the tags in a hierarchical structure, based on their alarm groups
- The right pane, which displays the list of all available Dot Fields for the tag or alarm group selected on the left pane
- The watch window at the bottom, which displays the run-time values of tags that you want to monitor.
- The **Value**, **Timestamp**, **Quality** columns that represent Value Time Quality (VTQ) I/O data monitoring, display if the data you are receiving is good and how to identify data related problems.



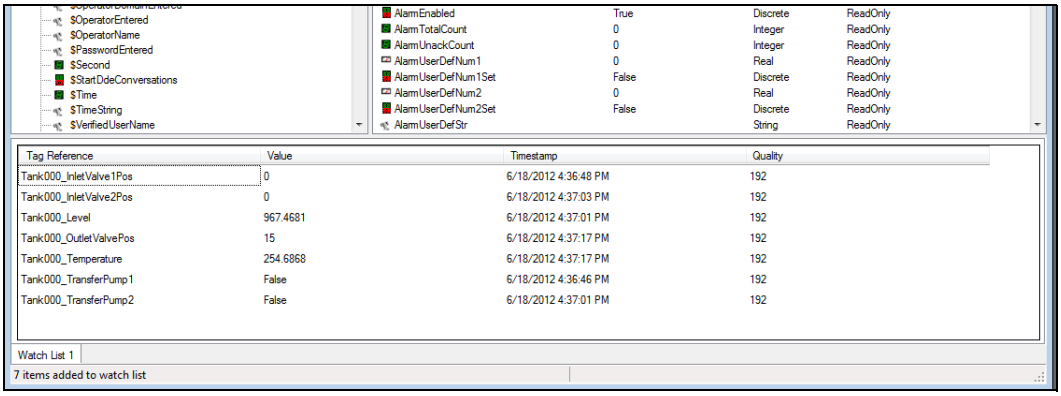
Adding Tagnames to Watch Window

The Watch Window is displayed at the bottom of the Tag Viewer by default. You can use this window to view the value of tags at runtime. You can create additional watch windows to group relevant tags together. You can also remove a watch window, if required.

Using Tag Viewer

You can assign tags to a Watch Window to view current tag values. You can add tags using several methods.

- Right-click a tag and select **Add to Watch**
- Click a tag and drag it to the **Watch Window**
- Double-click a tag



Note: You can open only one instance of **Tag Viewer** at a time.

Value Time Quality (VTQ) places a timestamp and quality indicator on all data values delivered to VTQ-aware clients.

You can monitor the value, time, and quality of a tag in the Watch Window to determine if the data you receive is good, and identify data related issues.

VTQ Dot Fields (.fields)

Value Time Quality (VTQ) places a time stamp and quality indicator on all data values delivered to VTQ-aware clients. VTQ provides indicators of data quality useful for diagnostics and troubleshooting.

Value: .Value field is the tag value.

Time: .Time fields represent various time stamps indicating the last update time for the tag.

Quality: .Quality fields represent I/O diagnostics of the tag information.

.Value

Contains the value of the specified tagname. This is also the default .field for every InTouch tagname within the system. If no other .field is specified, this .field is assumed.

.Time

Time Dot Fields represent update time stamps and can be integer, string, real tag, or message tags based on various time values. For example, the .TimeDate indicator is an integer tagname Dot Field used to display the whole number of days since 01/01/1970 which have passed since an I/O value provided by an I/O Server when the I/O connection is valid.

. Quality

The Wonderware Data Quality standard is based on the OLE for Process Control (OPC) proposed quality, which in turn is based on Fieldbus Data Quality Specifications.

Quality flags represent the quality state for an item's data value. This design makes it fairly easy for both Servers and Client applications to determine how much functionality they want to implement.

The low 8 bits (Least Significant Byte) of the Quality flags are currently defined in the form of three bit fields; Quality, Substatus and Limit status arranged as follows: QQSSSSLL

The Quality field allows the user to access the quality of an I/O tagname as provided by an I/O Server.

Note: If the I/O connection becomes invalid, the quality .fields are automatically reset to the initial value of zero. The **.ReferenceComplete** .field is also set to zero to indicate incomplete.

When the client application is unable to communicate with the Server **.QualityStatus** is **0**.

The conditions under which each of these quality states will be reported are as follows:

Good (.Quality of 192)

- The Communications link has been verified.
- The PLC understood our Poll request and returned a valid response packet.
- If a write occurred, there were no errors during the write process.
- There were no conversion problems with the data contained in the response packet.

Example: The value 0x0000A is returned due to a poll of a register containing 10 (decimal).

Clamped High (.Quality of 86)

- The Communications link has been verified.
- The PLC understood our Poll request and returned a valid response packet.
- The register was read or written without error.
- It was necessary to clamp its intended value to a limit because the value was larger than the maximum allowed.
- In the case of a string, the string is truncated.

Example: An unsigned 16 bit integer is clamped to 65535.

Clamped Low (.Quality of 85)

- The Communications link has been verified.
- The PLC understood our Poll request and returned a valid response packet.
- The register was read or written without error.
- It was necessary to clamp its intended value to a limit because the value was smaller than the minimum allowed.

Example: An unsigned 16 bit integer is clamped to 0.

Cannot Convert (.Quality of 64)

- The Communications link has been verified.
- The PLC understood our Poll request and returned a valid response packet.
- The data from the PLC could not be converted into the desired format.

Possibilities for cannot convert include, but are not limited to:

- The Server may return a constant in place of the data or return quality information alone.
- The data is not usable.
- It is not known whether the value is too large or too small.
- The data returned from the PLC is of the incorrect data type.
- A Floating Point number is returned, but is not value (for example: Not A Number).

Example: The value of 0x000A is returned from a BCD register in a PLC.

Communications Failed (.Quality of 24)

Any combination of the following:

- Data communications are down.
- The Topic is in slow poll (or equivalent) mode.
- There have been no link validating messages.
- Lack of resources in the Server. For example, a TSR (or driver) cannot allocate memory.
- Lack of resources in the communications link.
- The communications link is off-line.
- All communications channels are in use.
- The network is unable to route the message to the PLC.

Example: Attempting to read data from a PLC that has been powered off.

Cannot Access Point (.Quality of 4)

- The Communications link has been verified.
- The PLC understood our Poll request and returned a valid response packet.
- The PLC reported that it could not access the requested point.

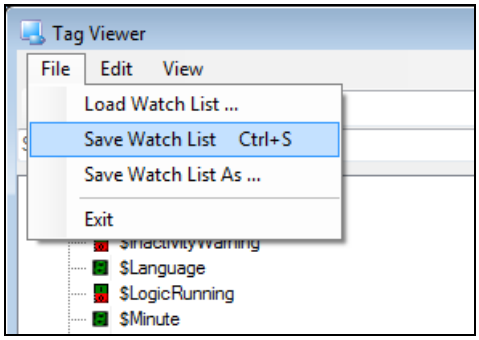
Possibilities for lack of accessibility include, but are not limited to:

- Item does not exist in PLC memory.
- Item is not currently available (locked in some way due to resource contention).
- Item is not of the correct format/data type.
- A write attempt was made, but item is read-only.
- In most cases, a group of items will be affected when one item is invalid. This is due to the block-polling scheme used by the Servers. For example, if one item in a block of 10 is invalid, then the entire block is marked invalid by the PLC. The Server will report invalid quality for all items in the block.
- The data is unusable.

Example: Attempting to read R40001 but R40001 is not defined in the PLC's memory map.

Saving Watch Windows

You can create multiple Watch Lists and save them to a central location for later viewing. Use standard windows functionality to rename the Watch List. You can then save it using the **Save Watch List** option on the **File** menu. A **Windows Explorer** window appears, allowing you to navigate to a central location and provide a name for your Watch List.



Your Watch List can contain multiple tabs to help you organize your watched tag lists.

Lists can be saved and later loaded back into an instance of **Tag Viewer**. When you load a watch list, any existing Watch List is removed from the view.

Lab 4 – Creating and Monitoring Tags

Introduction

In this lab, you will create a series of Real, Integer, Discrete, and Message type tags. You will use these tags in subsequent labs to monitor processes using graphics and animations.

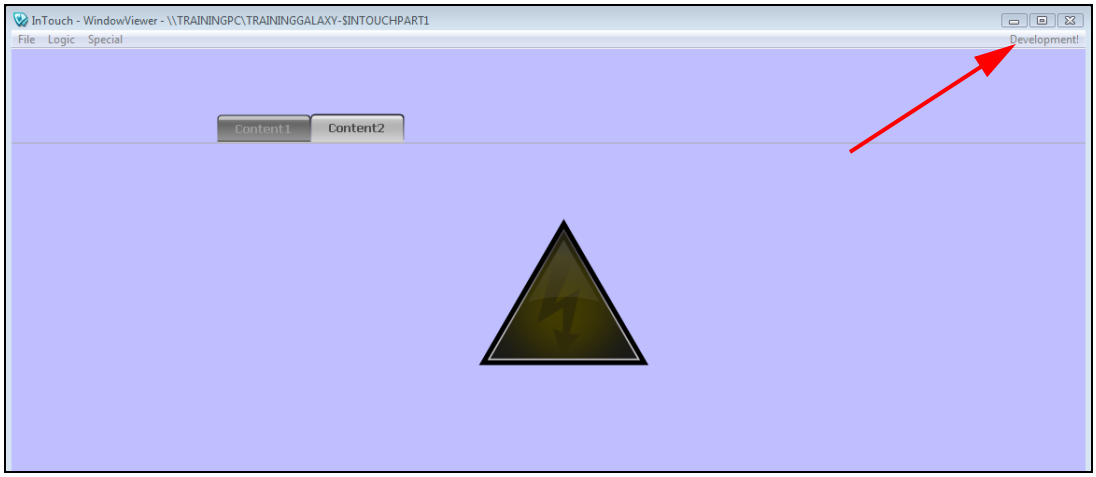
Objectives

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

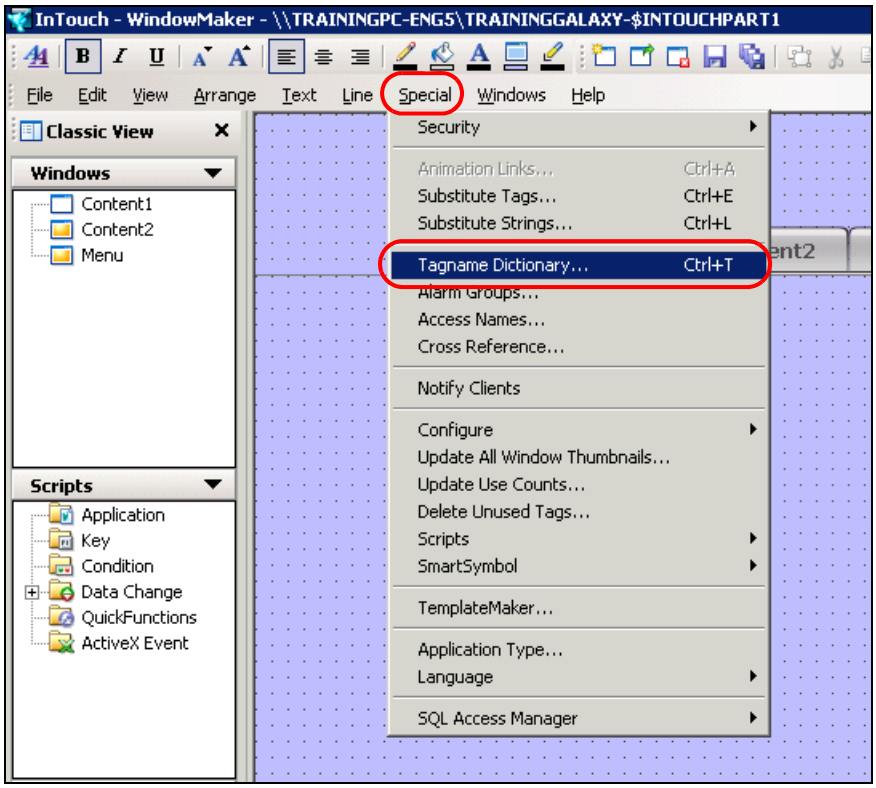
- Create tags
- Access I/O data
- Use Tag Viewer to monitor tags
- Validate quality with VTQ

Create Tags in the Tagname Dictionary

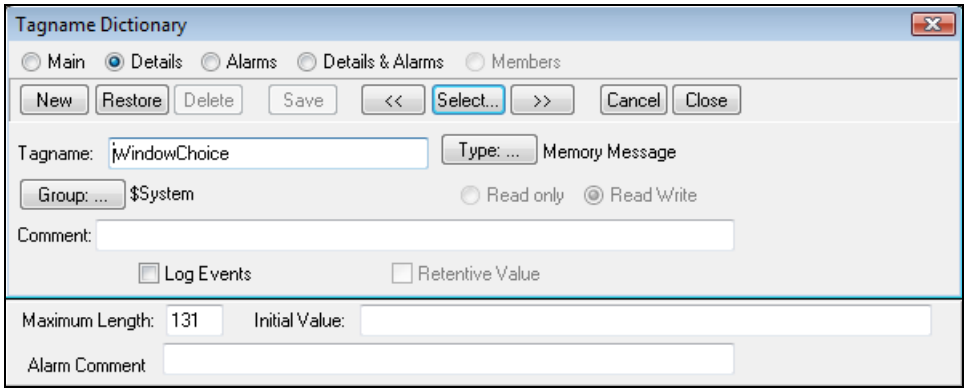
1. At the top-right corner of WindowViewer, click **Development**!.



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



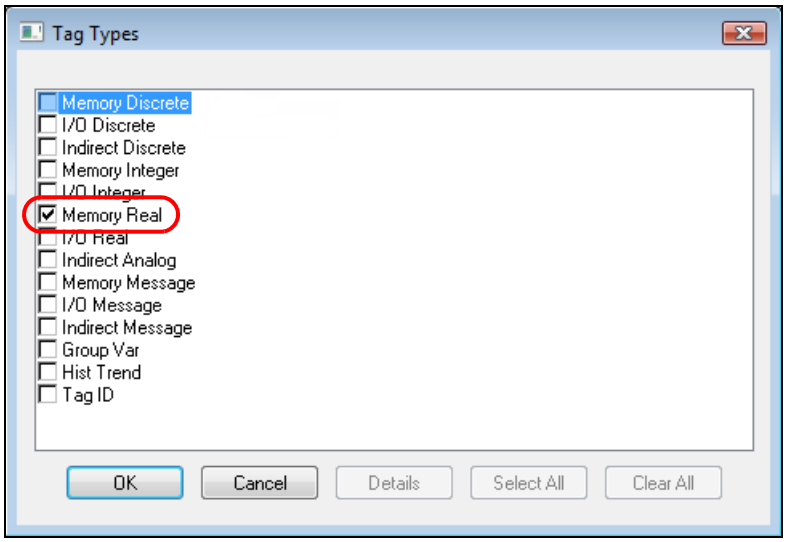
The **Tagname Dictionary** appears.



3. Click **New** to create a new tag.

Note: Changing attributes, such as the name of the tag without first clicking **New**, modifies the currently selected tag definition.

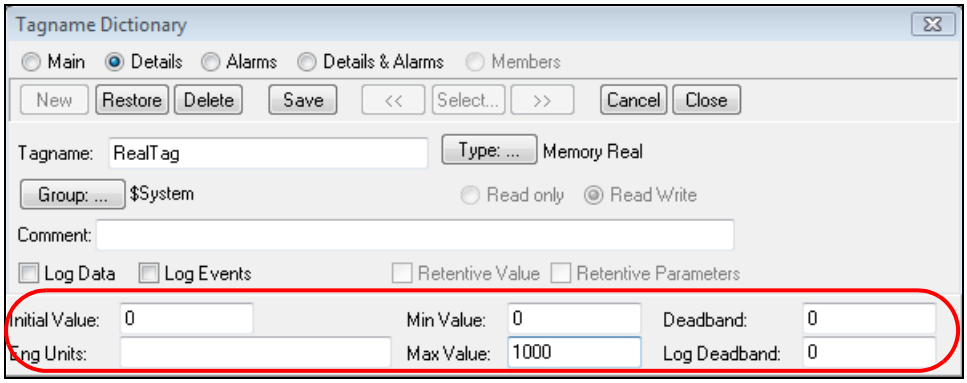
4. In the **Tagname** field, enter **RealTag**.
5. Click **Type**, then in the **Tag Types** dialog box check **Memory Real**.



6. Click **OK**.

7. In the details area of the dialog box, enter the following values:

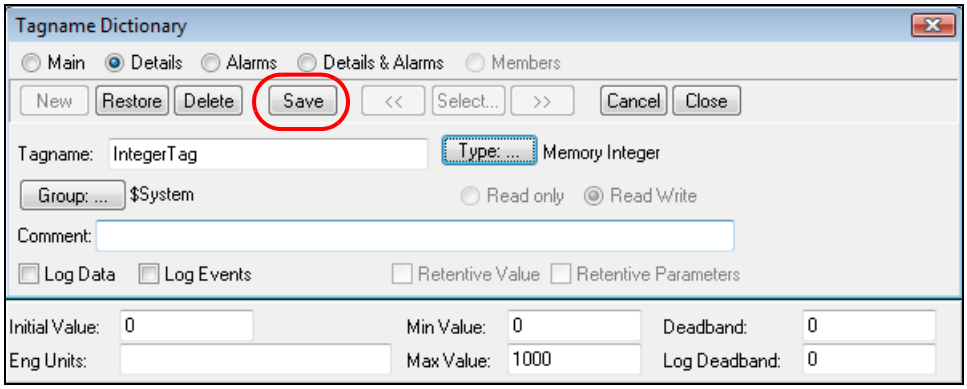
   Initial Value: 0
   Eng Units: (leave blank)
   Min Value: 0
   Max Value: 1000
   Deadband: 0
   Log Deadband: 0



8. Click **Save**.
9. Click **New** to define another tag.
10. In the **Tagname** field, enter **IntegerTag**.
11. Click **Type**, then in the **Tag Types** dialog box check **Memory Integer**.
12. Click **OK**.
13. Change **Min Value** to **0**.
14. Change **Max Value** to **1000**.

    Keep the remaining default values.

15. Click **Save**.

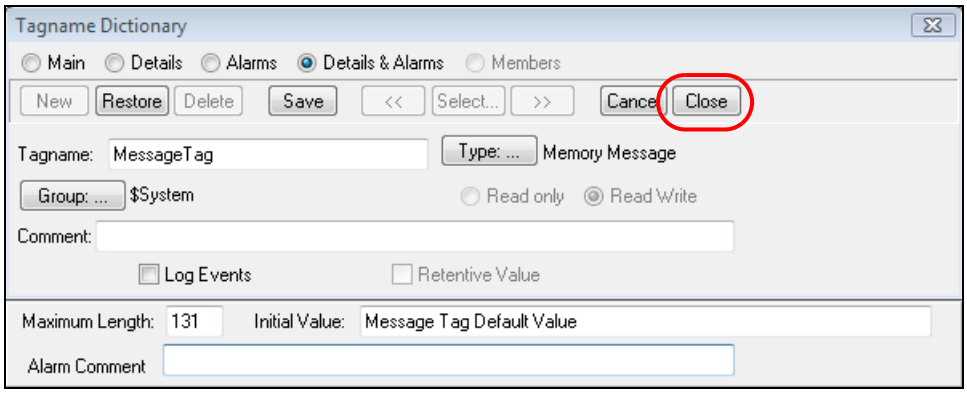


16. Create the following additional tags.

    Remember to click **New** before starting each definition:

Tagname	Type	Initial Value
DiscreteTag1	Memory Discrete	(leave defaults)
DiscreteTag2	Memory Discrete	(leave defaults)
MessageTag	Memory Message	Message Tag Default Value

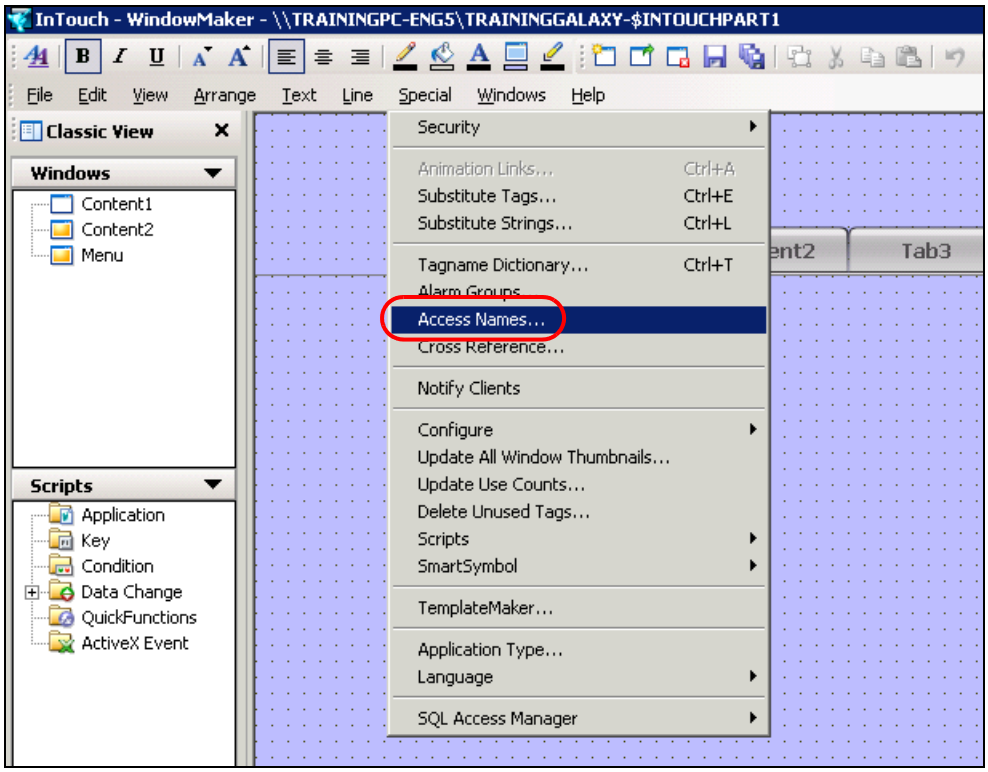
17. Click **Close** to save and exit the **Tagname Dictionary**.



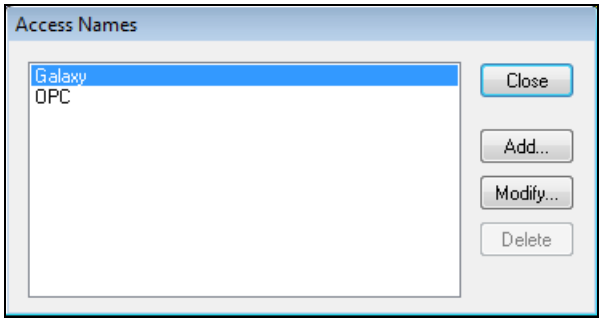
Set Up Access Names

One of the most powerful features of InTouch is its ability to acquire data from I/O sources. First you use InTouch WindowMaker to create an Access Name and then configure tags to access I/O data. The I/O server you connect to is used to simulate a vessel, where ingredients are added, mixed, and then drained. You will track various process values associated with the vessel.

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

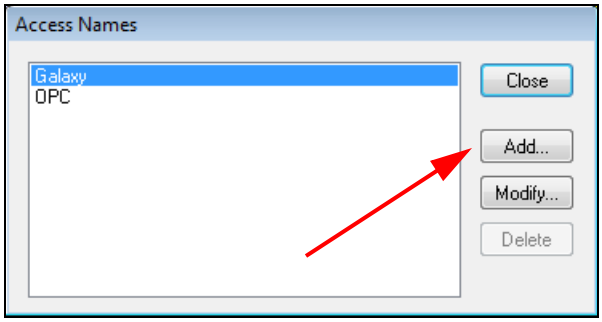


The **Access Names** dialog box appears.

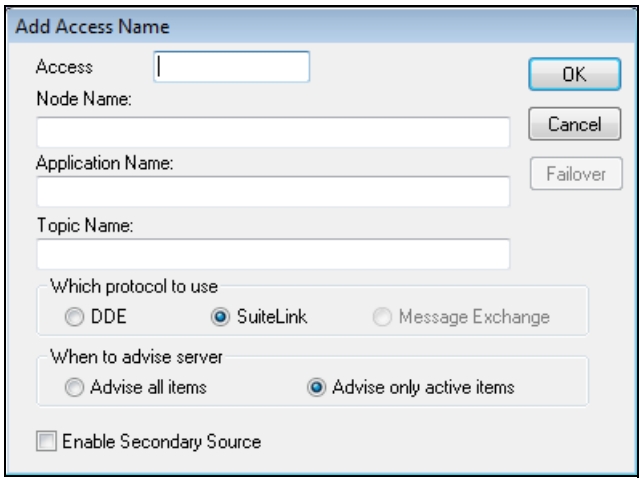


InTouch provides two default **Access Names** (**Galaxy** and **OPC**).

19. In the **Access Names** dialog box, click **Add**.

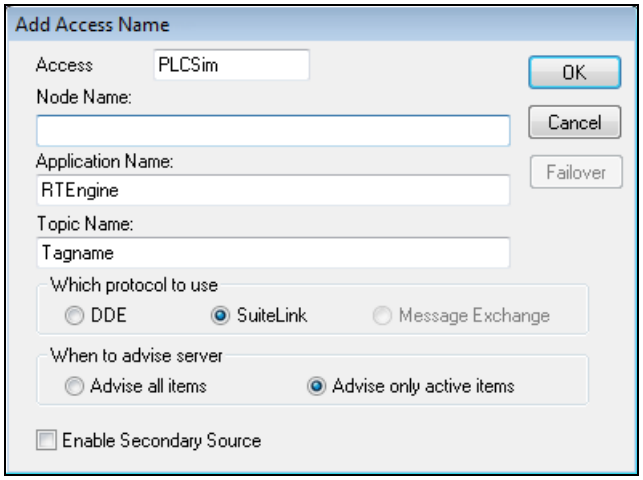


The **Add Access Name** dialog box appears.

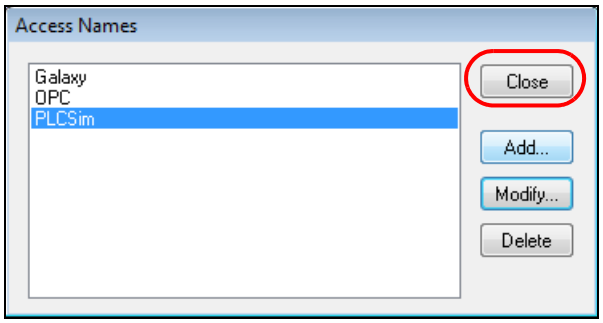


20. Enter the following information.

Access:	PLCSim
Node Name:	<instructor will provide>
Application Name:	RTEngine
Topic Name:	Tagname

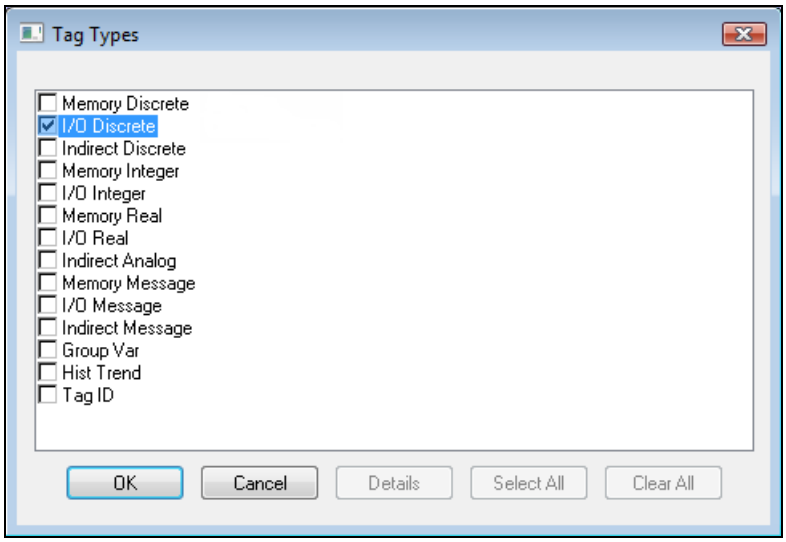


21. Keep the remaining default settings, and click **OK**.
22. Click **Close** to commit the changes.



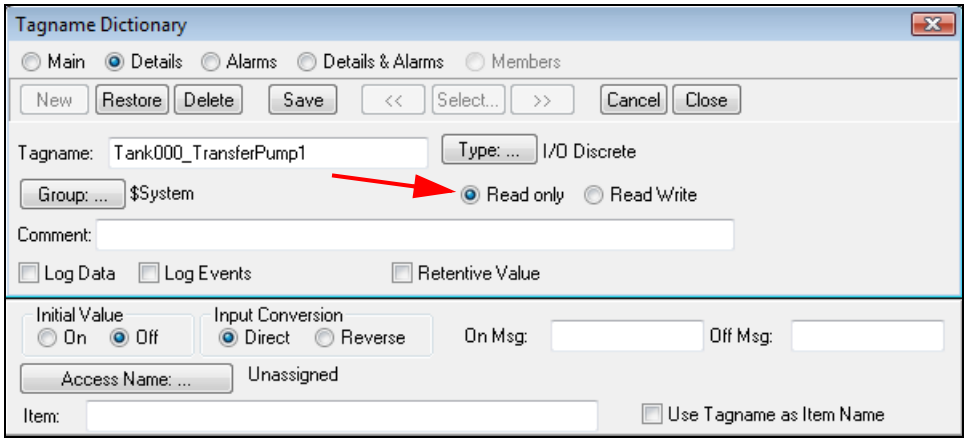
Create I/O Discrete Tags

23. Open **Tagname Dictionary**.
24. Click **New**.
25. In the **Tagname** field, enter **Tank000_TransferPump1**.
26. Click **Type** and check **I/O Discrete**.

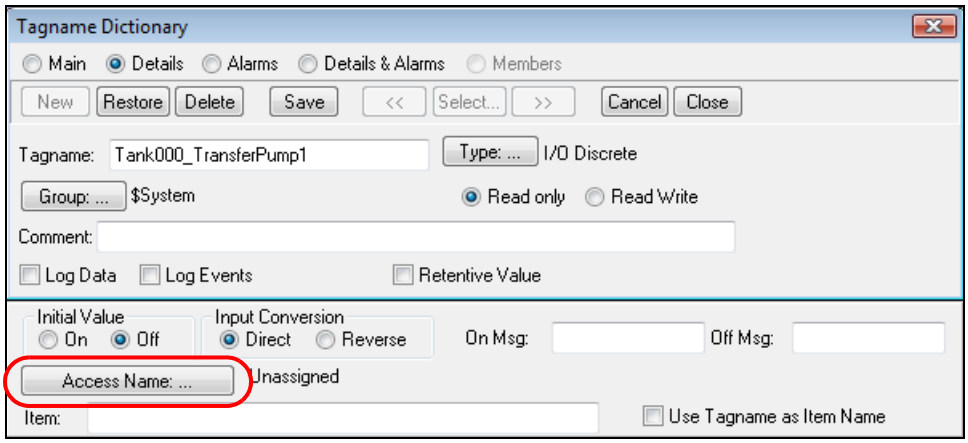


27. Click **OK**.

28. Click **Read only**.

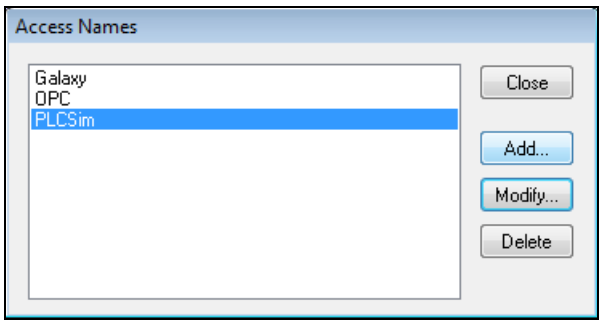


29. Click **Access Name**.



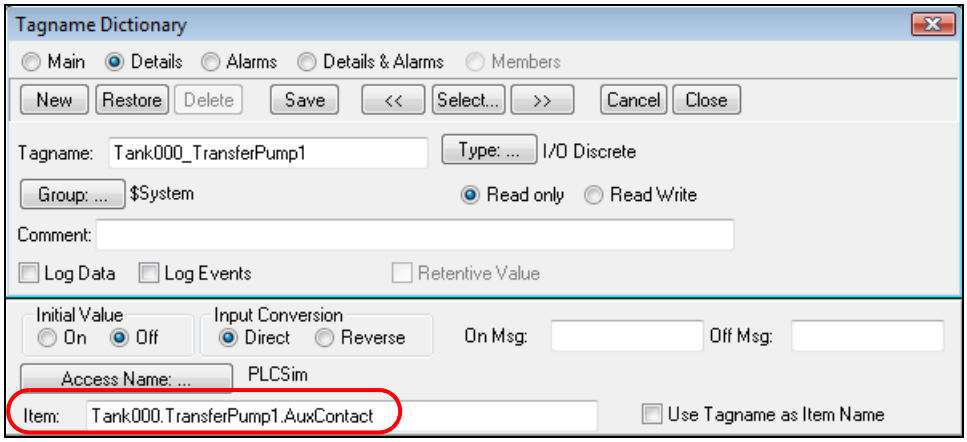
The **Access Names** dialog box appears.

30. Click **PLCSim**.



31. Click **Close**.

32. In the **Item** field, enter **Tank000.TransferPump1.AuxContact**.



33. Keep the remaining defaults and click **Save**.
34. Create the following **I/O Discrete** tags.

    After each entry, click **Save**, then click **New**.

Tagname	Type	Access Name	Item
Tank000_TransferPump2	I/O Discrete	PLCSim	Tank000.TransferPump2.AuxContact
Tank000_Agitator	I/O Discrete	PLCSim	Tank000.Agitator.AuxContact

Create I/O Integer Tags

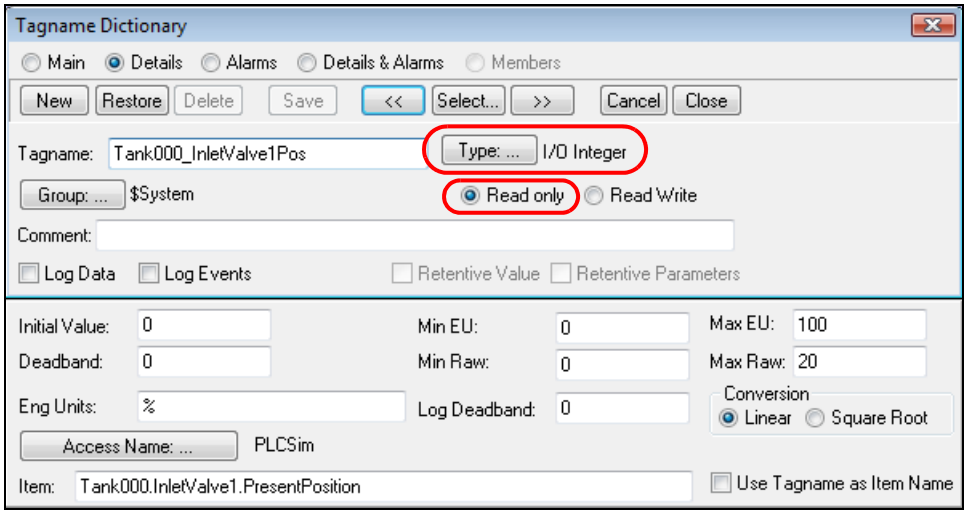
Next, you will create three I/O Integer tags.

35. In **Tagname Dictionary**, click **New**.
36. In the **Tagname** field, enter **Tank000_InletValve1Pos**.
37. Click **Type**, then check **I/O Integer.**
38. Click **OK**.
39. Ensure **Read only** is selected.

Next, you will configure the details of the tag, including scaling, engineering units, access name, and item.

40. In **Details** pane, enter the following values:

Initial Value:	0
Deadband:	0
Eng Units:	%
Min EU:	0
Max EU:	100
Min Raw:	0
Max Raw:	20
Access Name:	PLCSim
Item:	Tank000.InletValve1.PresentPosition



41. Click **Save**.

42. Create the following **I/O Integer** tags, keeping all values as above.

    After each entry, click **Save**, then click **New**.

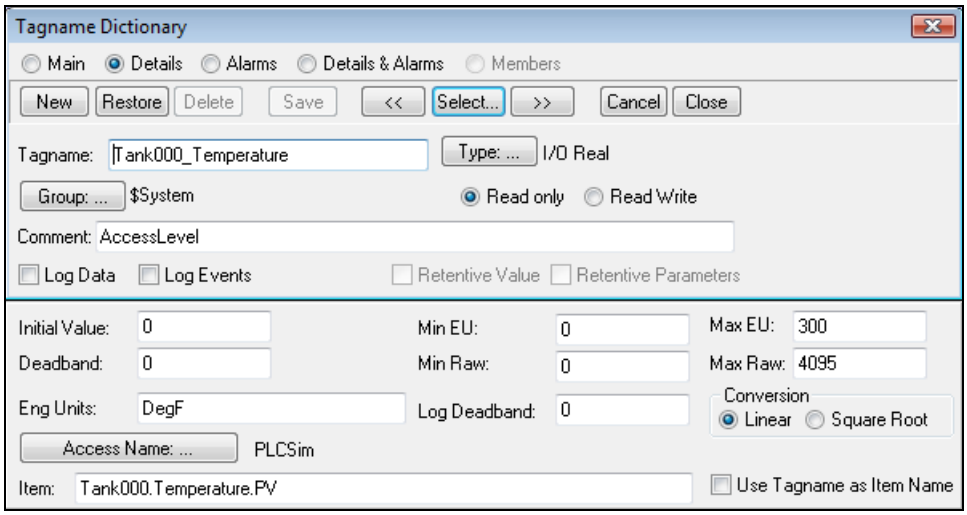
Tagname	Type	Access Name	Item
Tank000_InletValve2Pos	I/O Integer	PLCSim	Tank000.InletValve2.PresentPosition
Tank000_OutletValvePos	I/O Integer	PLCSim	Tank000.OutletValve.PresentPosition

Create I/O Real Tags

43. Create the following **I/O Real** tag:

    Tagname Tank000_Temperature
    Type I/O Real
    Initial Value: 0
    Deadband: 0
    Eng Units: DegF
    Min EU: 0
    Max EU: 300
    Min Raw: 0
    Max Raw: 4095
    Log Deadband: 0
    Access Name: PLCSim
    Item: Tank000.Temperature.PV

    Keep all remaining default values.

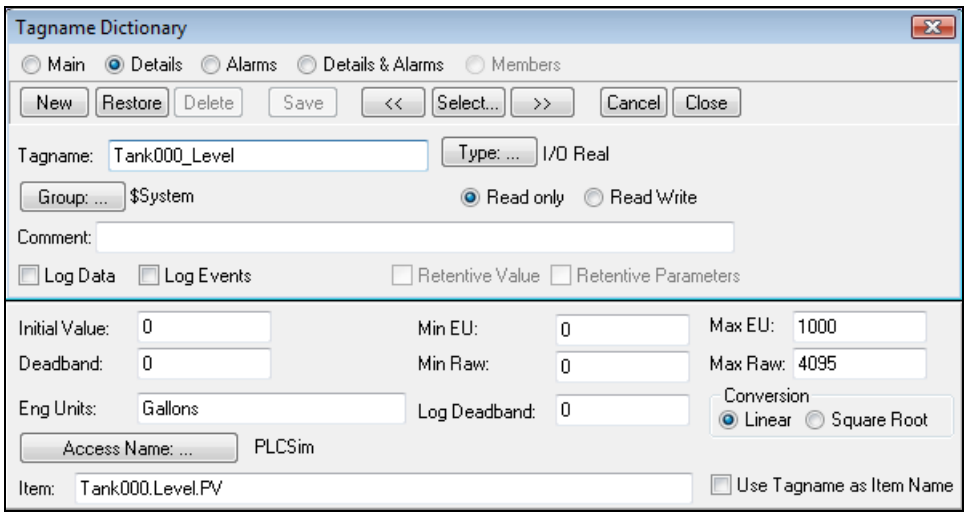


44. Click **Save**.

45. Create the following **I/O Real** tag.:

Tagname:	Tank000_Level
Type:	I/O Real
Initial Value:	0
Deadband:	0
Eng Units:	Gallons
Min EU:	0
Max EU:	1000
Min Raw:	0
Max Raw:	4095
Log Deadband:	0
Access Name:	PLCSim
Item:	Tank000.Level.PV

Keep remaining defaults.

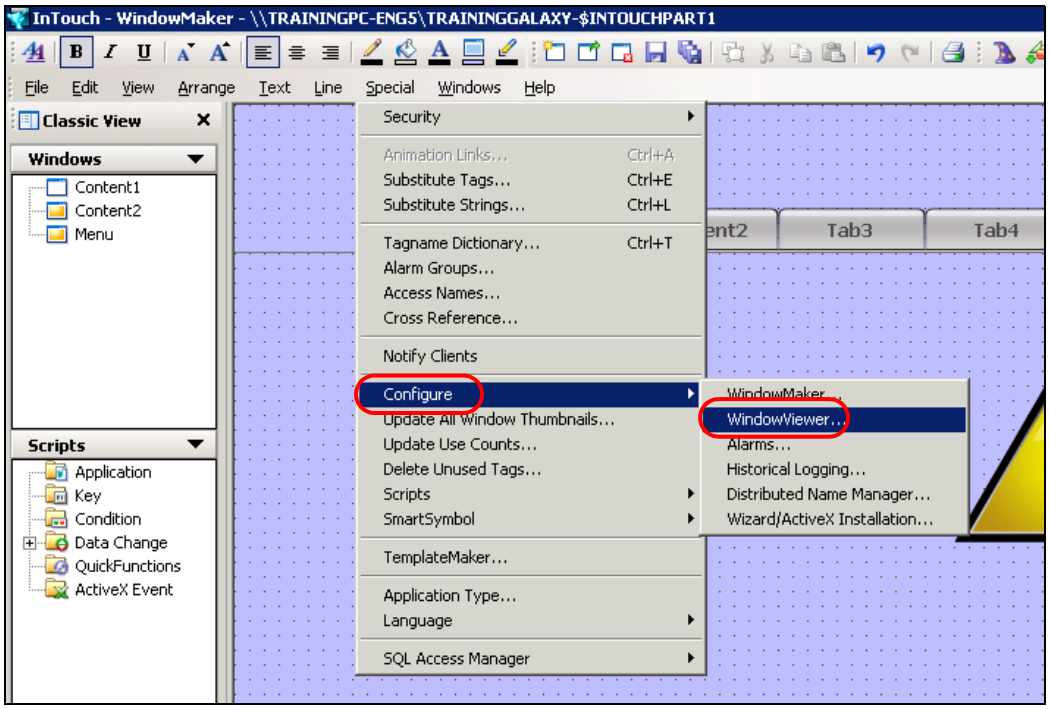


46. Click **Save**.
47. Click **Close**.

Validate Tags

Next, you will use Tag Viewer functionality to monitor I/O data and check the health of I/O communication for your new tags. To set up the Tag Viewer, you must first enable it in the WindowViewer configuration.

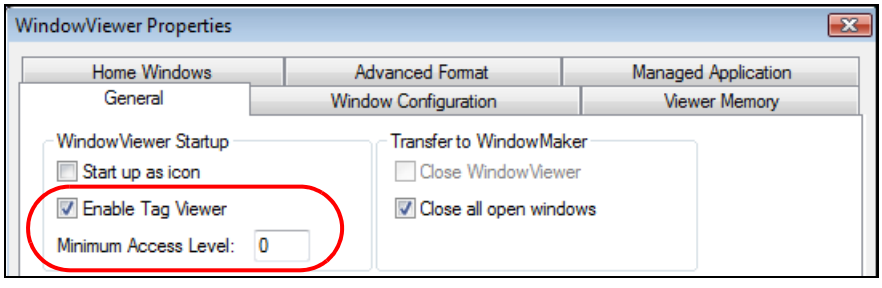
48. On the **Special** menu, click **Configure | WindowViewer** to configure WindowViewer.



The **WindowViewer Properties** dialog box appears.

49. On the **General** tab, **WindowViewer Startup** area, check **Enable Tag Viewer**.

50. In the **Minimum Access Level** filed, enter **0**.

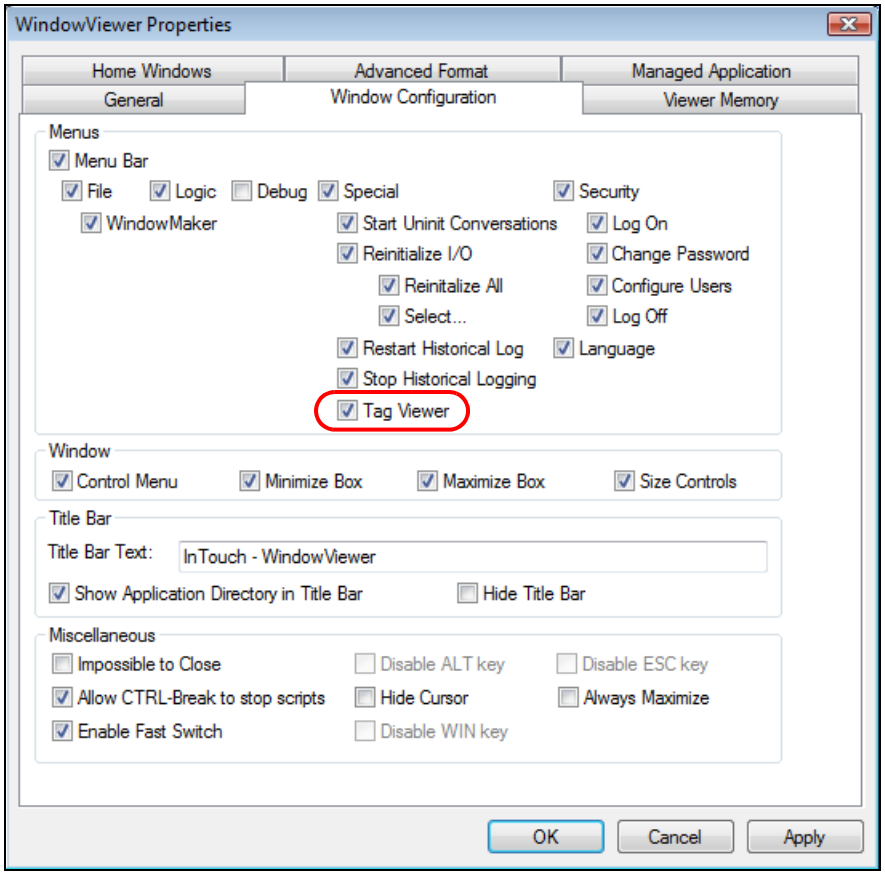


This will allow access to the **Tag Viewer** for all users.

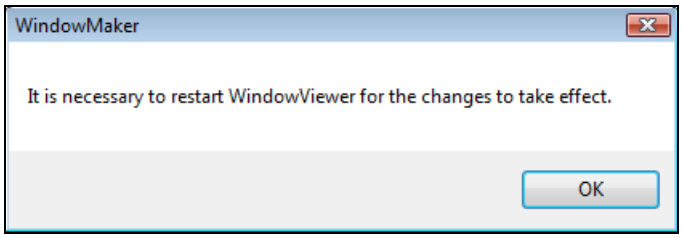
Note: Later you will change the **Minimum Access Level** to a higher value to secure this from access by guests or non-authenticated personnel.

---

51. Click the **Window Configuration** tab, then in the **Menus** area, check **Tag Viewer**.



52. Click **OK** to close.
53. Click **OK**.



54. Close WindowViewer and proceed to the next step.

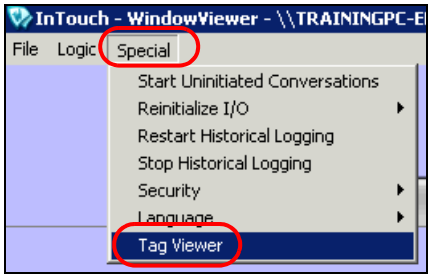
Test the Tag Viewer

Next, you will switch to Runtime to view the Tag Viewer.

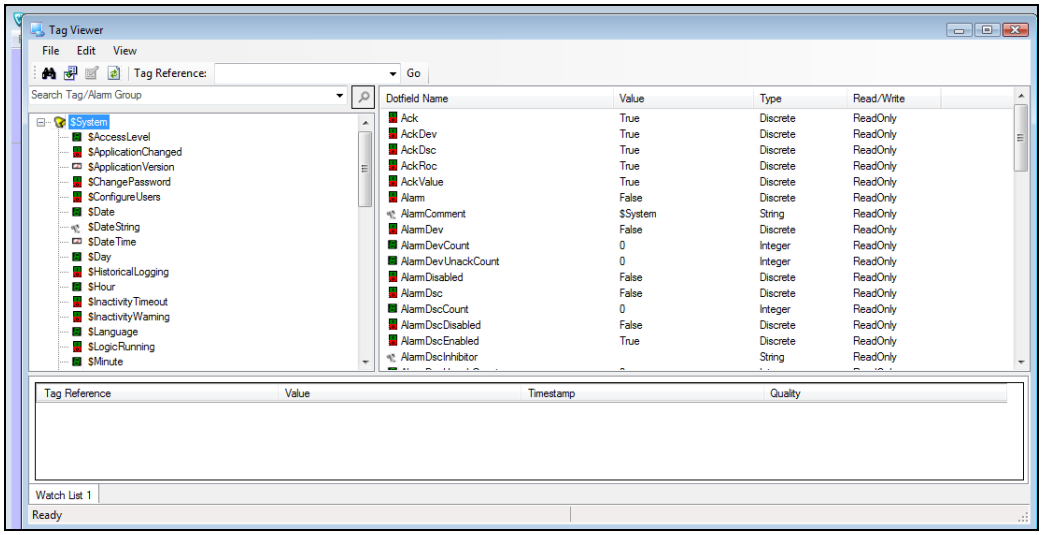
55. Click **Runtime** to open WindowViewer.



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



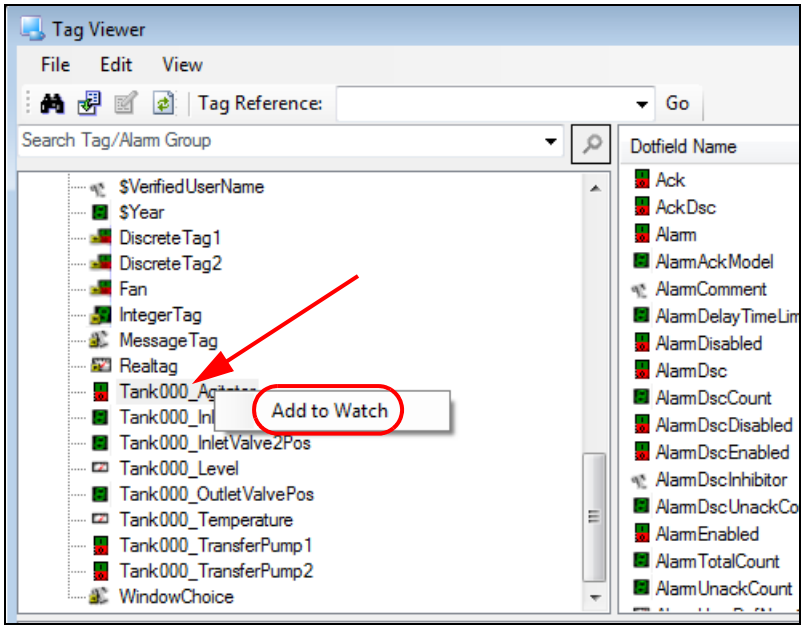
The **Tag Viewer** window appears.



Add a Watch Window

You can assign tags to a Watch Window to view current tag values.

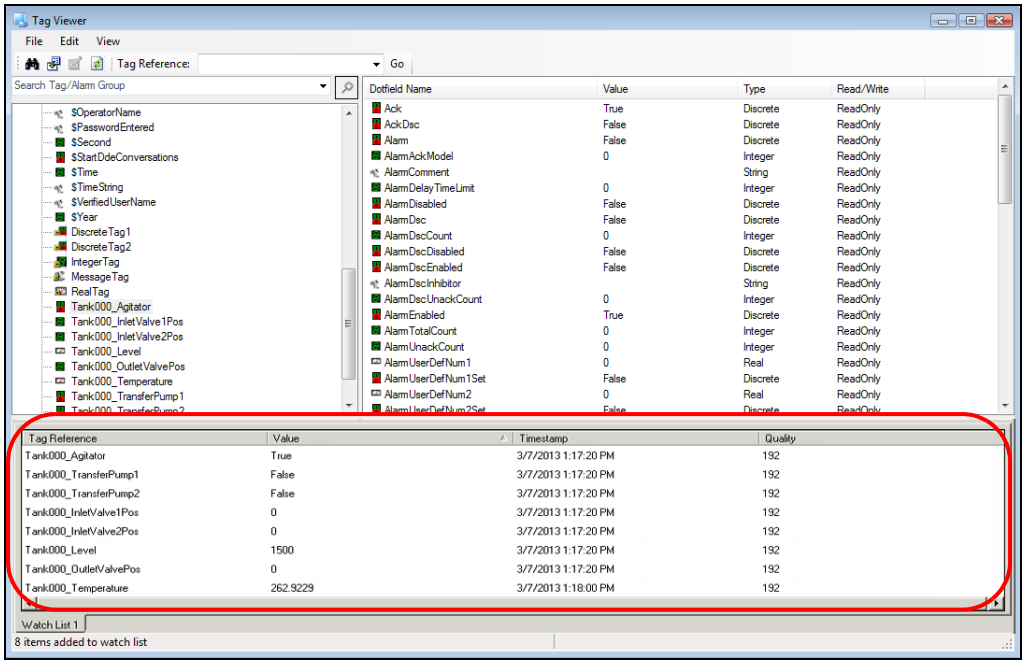
57. Right-click **Tank000_Agitator**, then click **Add to Watch**.



Note: The new tags you created in this lab appear below the system tags near the bottom of the list.

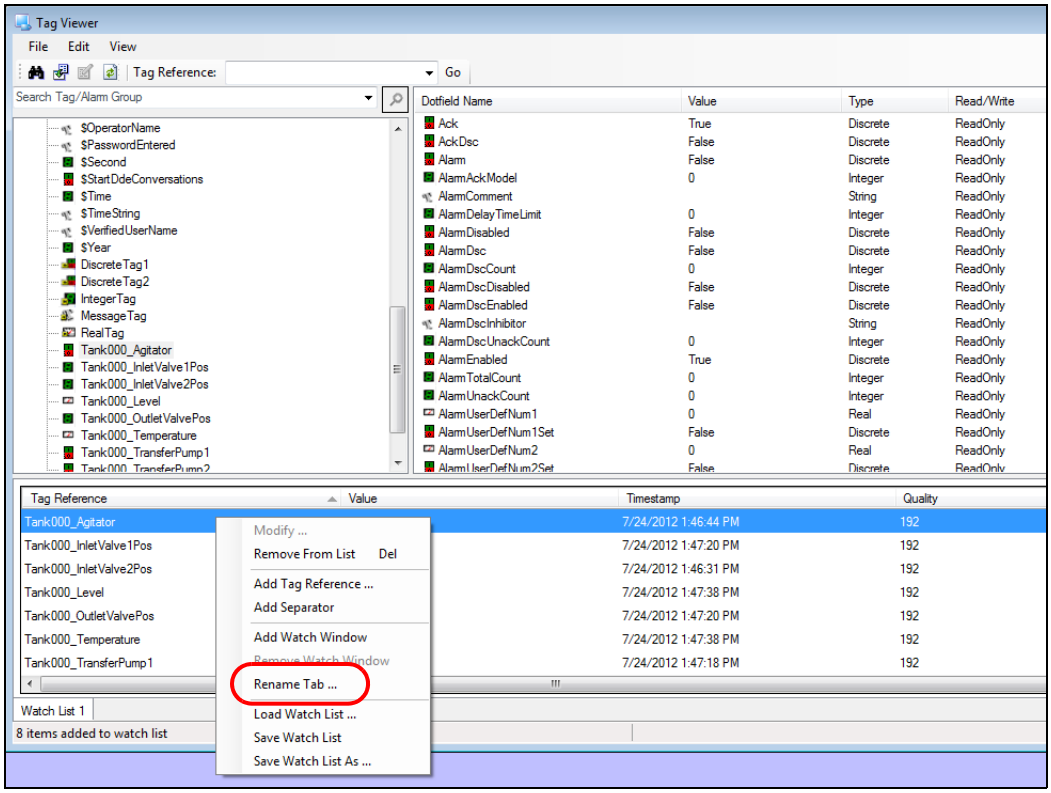
58. Double-click **Tank000_TransferPump1** to add it to the **Watch List**.
59. Drag and drop **Tank000_TransferPump2** to the **Watch List**.
60. Maximize the **Tag Viewer** window to see your entire list.

61. Add the following tags to the **Watch List**.
    - **Tank000_InletValve1Pos**
    - **Tank000_InletValve2Pos**
    - **Tank000_Level**
    - **Tank000_OutletValvePos**
    - **Tank000_Temperature**

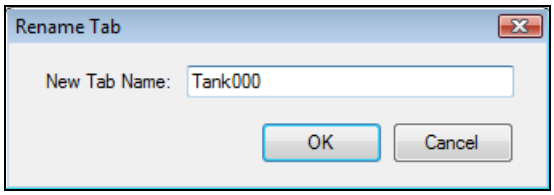


Notice that in the **Watch List**, your tag values change and the **Quality** of all tags is **192**. This indicates a healthy communication with the I/O source.

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



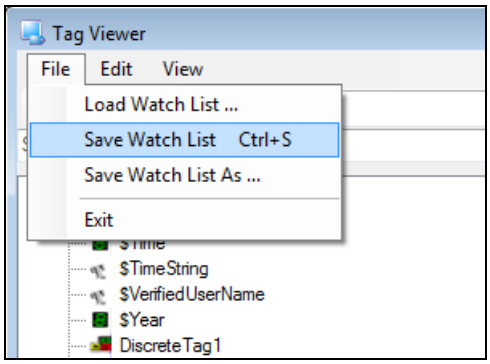
63. In the **New Tab Name** field, enter **Tank000**.



64. Click **OK**.

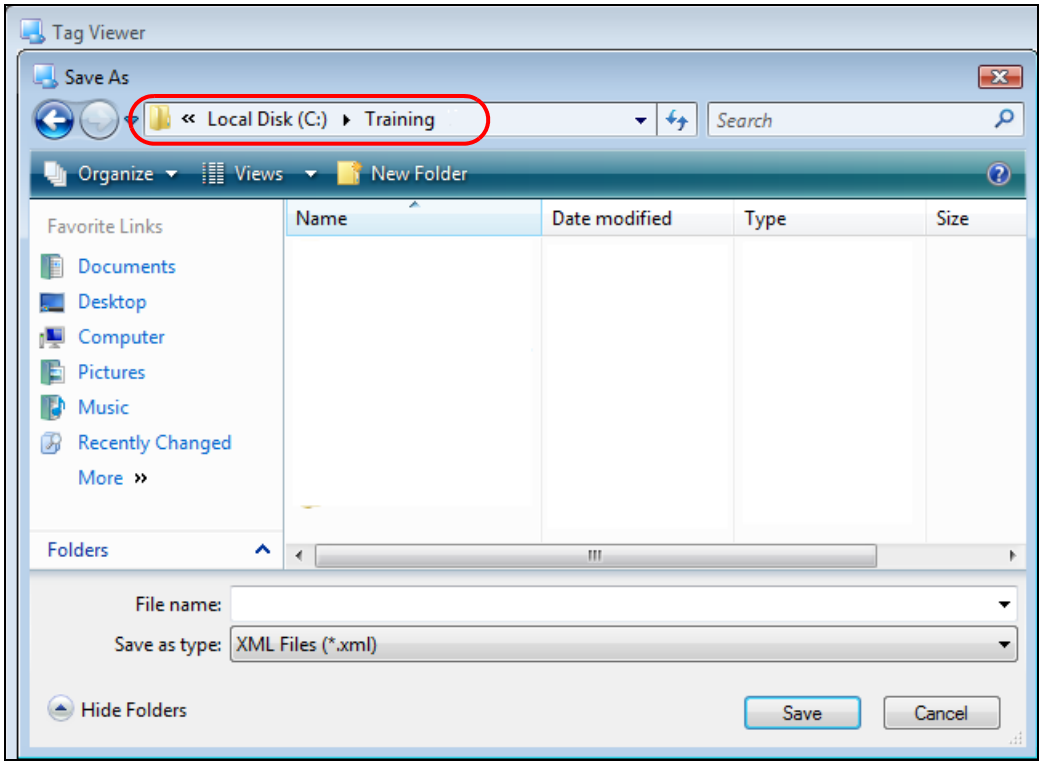
Now you will save your Watch List to a central location for later viewing.

65. On the **File** menu, click **Save Watch List**.

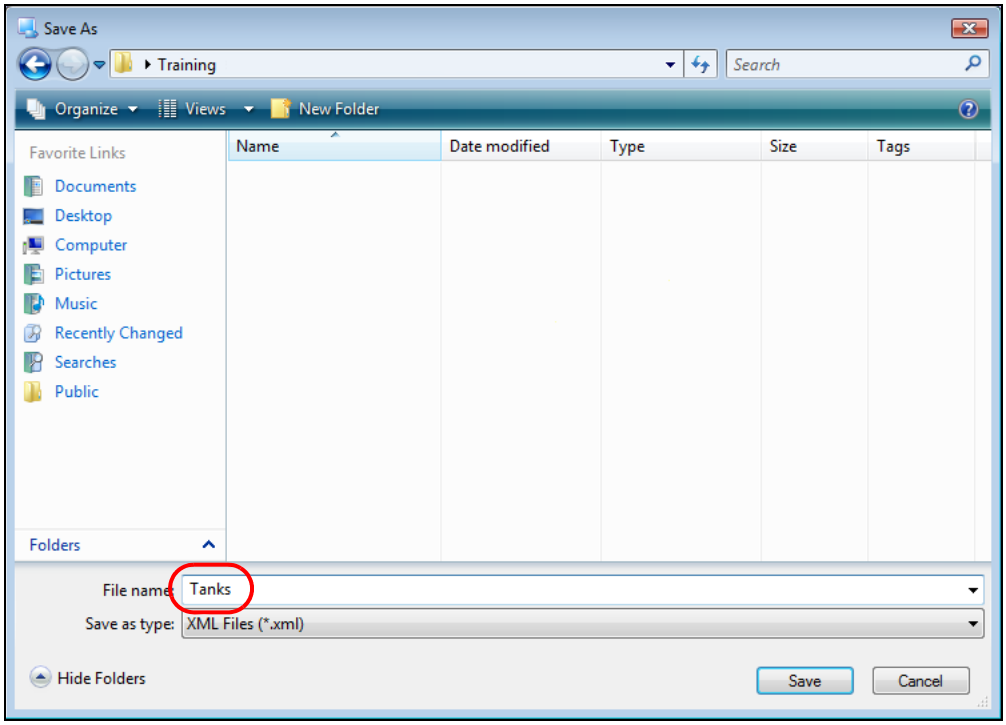


The **Save As** dialog box appears.

66. Navigate to **C:\Training**.



67. In the **File name** field, enter **Tanks**.



68. Click **Save**.

69. Close the **Tag Viewer** window.



70. Click **Development!** to return to WindowMaker.

Section 7 – Large-Scale Tag Manipulation

This section explores using the DBDump and DBLoad utilities to streamline the development of large amount of tags containing similar functionality.

Overview

When you back up the Tagname Dictionary using the DBDump utility, a copy of the Tagname Dictionary is exported to a file that can be opened by external applications. These files can be manipulated externally and loaded back into the Tagname Dictionary, replacing existing entries or adding new tags using the DBLoad utility.

The Tagname Dictionary is a convenient tool for creating tags. It's a simple interface and it makes this job quite easy; however, when creating thousands of tags for a large application, this task could be time consuming. You can streamline tag development of large amounts of similar tags using DBDump and DBLoad to make massive additions or changes on a wide scale.

Backing Up and Restoring the Tagname Dictionary

InTouch has two utility programs to back up and restore your **Tagname Dictionary**:

DBDump Exports an InTouch application Tagname Dictionary as a .CSV (Comma-Separated Value) file that can be viewed or edited in any program supporting .CSV files. Microsoft Excel is recommended for this task.

DBLoad Imports an InTouch application Tagname Dictionary as a .CSV (Comma-Separated Value) file that can be viewed or edited in any program supporting .CSV files. Microsoft Excel is recommended for this task.

These two utilities allow you to copy, modify, or develop tags in separate portions and then merge into one application. The DBLoad utility can also be used as an alternative to the InTouch TemplateMaker to create SuperTag instances.

Both the DBDump and DBLoad utilities can be started from the InTouch App Manager for Stand Alone InTouch, or the ArchestrA IDE for a Managed application.

DBDump

To use the DBDump utility to export the Tagname Dictionary of your InTouch application, in the ArchestrA IDE, on the **Galaxy** menu, select **Export | DBDump** to display the .**CSV File to Dump To** dialog box.

The backup file is named after the application. It ends with the .CSV (Comma-Separated Value) extension. In this course, the file name is **$InTouchPart1.csv**. Upon saving the file, you are prompted to confirm the DBDump option. It also allows you to group the extracted tagnames either **by tag type** or **by tagname** (tag type is the default).

DBLoad

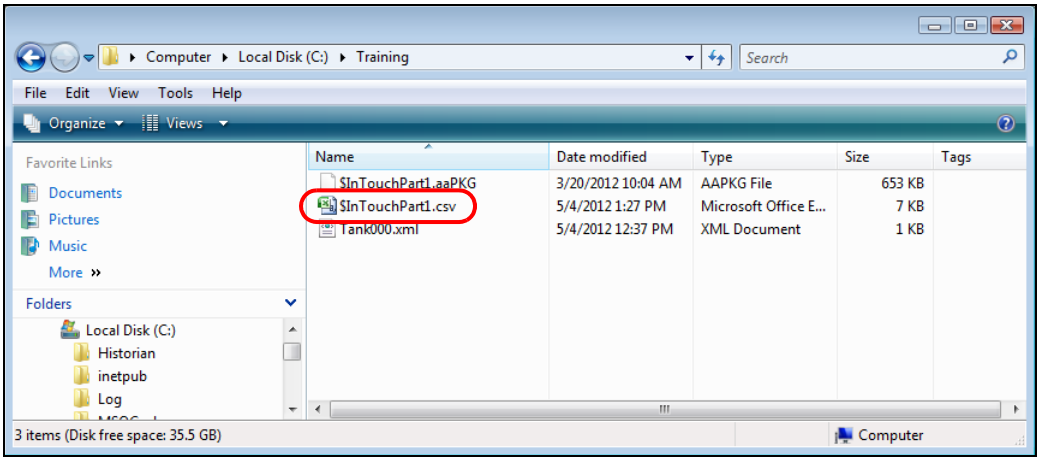
You use the DBLoad utility to load or merge a Tagname database file into an existing Managed InTouch application Tagname Dictionary.

You can manually create Tagname Dictionary import files with any application that supports the .CSV file format. But, creating an entire import file can be time consuming and prone to errors. Using an existing .CSV file as a template is faster and more reliable.

Prior to using the DBLoad utility to backup an InTouch application database, you must close both WindowViewer and WindowMaker and check in the application.

Next, determine if you want to load the database input file into a new InTouch application, or merge it into an existing InTouch application database.

To merge the database input file into a new InTouch application, open the **.CSV File to Load From** dialog box and from the **Galaxy** menu select **Import | DBLoad**, and choose the **.CSV** file to use.



The database information contained in the selected file will begin uploading to the selected application's Tagname Dictionary. After a brief transition; the **Database Load** message box notifies you of the successful completion of the **Load** operation.

Upon completing the restore operation, the ArchestrA IDE prompts to **Check In** your InTouch application.

Creating a Database Input File

The DBDump and DBLoad database utilities are the tools used to perform batch operations on a Tagname Dictionary. Database input files can be created in any program that supports the .CSV file format.

The database input file must be saved as a .CSV format. Once an input file is created, the DBLoad program is used to load/merge the data contained in the file into an existing InTouch application database.

You can create a database input file template by creating a new InTouch application and then running the DBDump program to dump its database to a correctly formatted .CSV file. This makes entering your modifications easier than creating the input file from scratch.

Database Input File Format

The first line of a database input file specifies the operating **:mode** for the file when it is loaded or merged into an application using the DBLoad utility.

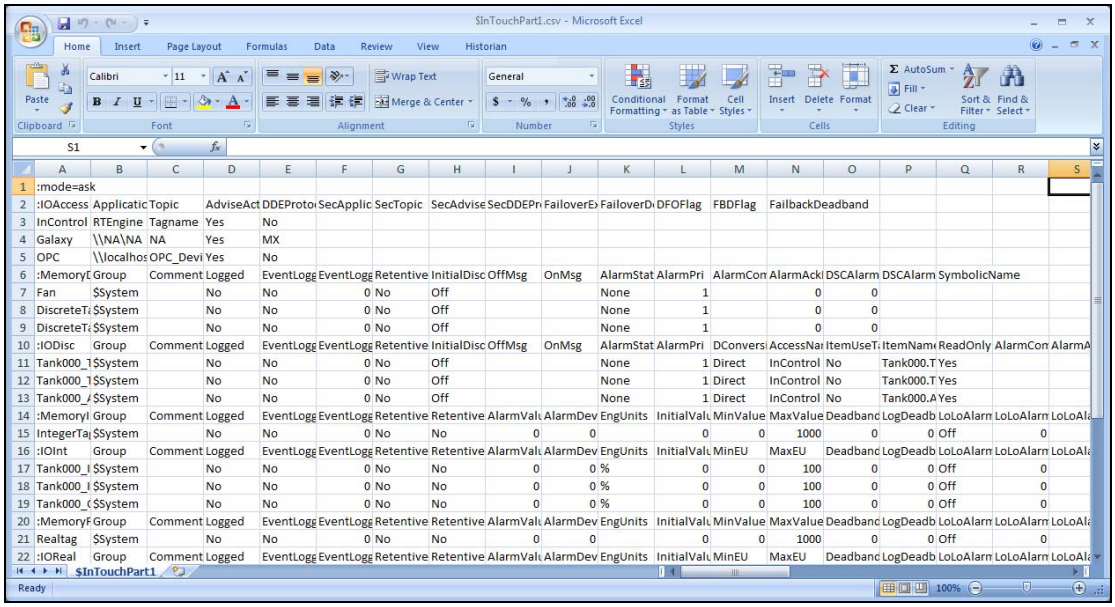
All data records must begin with the valid keyword for the tagname **:type**, followed by the valid keyword for each data record (separated by commas):

:mode=test

:IOMsg,Group,Comment,Logged,EventLogged,Event Logging Priority, …

There is a valid **keyword** for each tagname type and data record.

When the .CSV file is opened in Microsoft Excel, it interprets the comma as a delimiter and automatically separates the data records into columns.



When the .CSV file is opened in Notepad, each data record is separated by a comma.

Using DBLoad, it is possible to:

- Create tags
- Modify existing tag properties, such as Access Names and ItemNames
- Change logging properties
- Make mass changes to the Tagname Dictionary

If the tag is renamed, DBLoad creates a new tag rather than delete the original tag instance. DBLoad cannot delete a tag from the Tagname Dictionary.

Database Input File Operating Modes

The following lists the valid operating mode keywords and the actions that occur in each mode when a duplicate tagname is encountered during the load:

:MODE=ASK Change the tagname of the input or the existing entry to your specified tagname and then insert the new definition into the Tagname Dictionary.

:MODE=REPLACE Delete the existing entry and replace it with the new entry.

:MODE=UPDATE Overwrite the existing definition with only the fields that are explicitly defined in the input file; i.e., include DELTA changes only.

:MODE=TEST In this mode, DBLoad acts as if it were in the replace mode, but does not modify the database.

Test mode scans the file before loading it, then reports errors and gives the location/line number.

It is recommended to use this mode to locate possible Tagname Dictionary format errors. After the errors are resolved, change the mode to Update or Replace and load the DB.

Note: Additional **:Mode=** options are in the *InTouch User Guide*.

Database File Key Words

The DBLoad import file contains a set of keywords that organizes Access Names, alarm groups, and tag data within the file.

- A colon (:) precedes all keywords.
- To continue a line, enter a backslash (\) at the end of the line.
- To enter comments, precede them with a semi-colon (;).

The following table lists the keywords within a DBLoad import file. The table lists the keywords in the order they are specified when you create the file with DBDump. But you can specify keywords in any order within the file.

Keyword	Description
:Mode	Specifies how duplicate tag records are handled when importing the contents of the DBLoad file to an application's **Tagname Dictionary**.
:IOAccess	Access names defined for the InTouch application.
:AlarmGroup	Alarm groups defined for the InTouch application.
:MemoryDisc	Memory discrete tags.
:IODisc	I/O discrete tags.
:MemoryInt	Memory integer tags.
:IOInt	I/O integer tags.
:MemoryReal	Memory real tags.
:IOReal	I/O real tags.
:MemoryMsg	Memory message tags.
:IOMsg	I/O message tags.
:GroupVar	Group Var tags.
:HistoryTrend	Hist Trend tags.
:TagID	Tag ID tags
:IndirectDisc	Indirect discrete tags.

Keyword	Description
:IndirectAnalog	Indirect analog tags.
:IndirectMsg	Indirect message tags.

Each keyword includes a set of associated attributes that specify the properties of Access Names, alarm groups, and tags. For example, the **:IOAccess** keyword includes attributes to specify the application, topic, and communication protocol, which are properties of every InTouch Access Name.

Lab 5 – Creating Tags on a Large Scale

Introduction

In this lab, you will export the Tagname Dictionary into a comma delimited file you can open in Microsoft Excel. You will then make a modification to some of the items to duplicate existing tags. Then you will load the files back into the application using DB Load where you can review and modify the new tags and create a watch list specific to the new tags.

Objectives

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

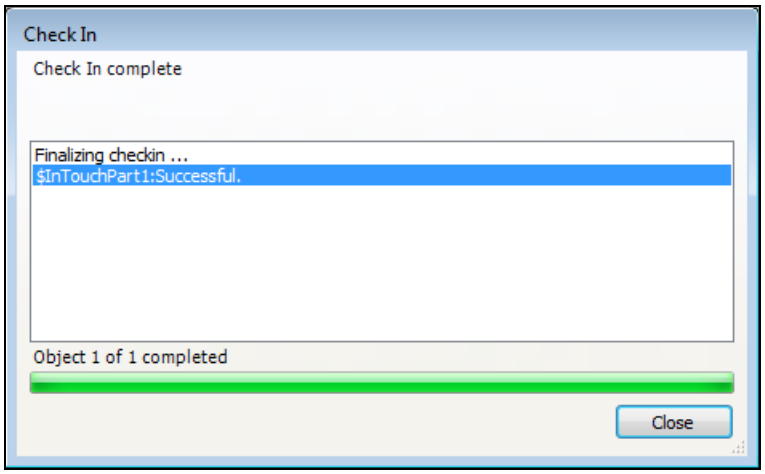
- Create a .CSV dump file using DBDump
- Load the .CSV file with DBLoad
- Add a new watch window to Tag Viewer

Use DB Dump to Export Tags

In this lab, you will use DB Dump functionality to create large numbers of tags outside the Tagname Dictionary, using Excel and CSV files.

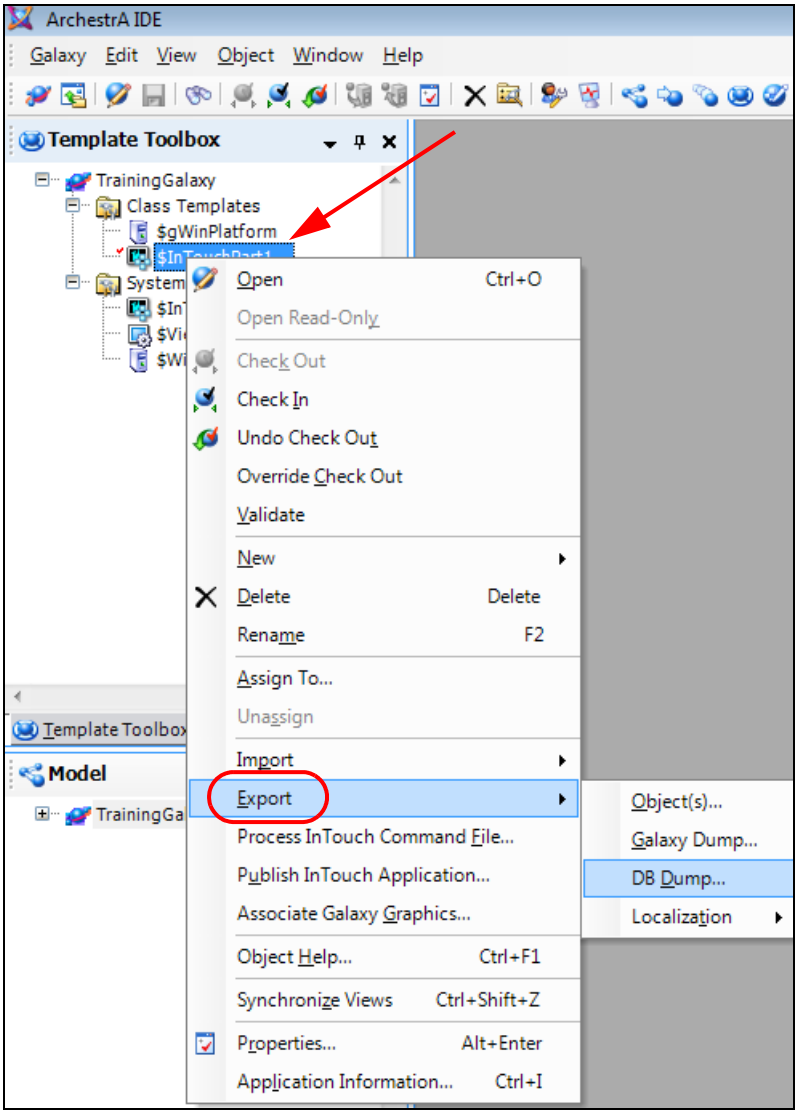
1. Close WindowViewer and WindowMaker.
2. Enter an appropriate comment.
3. Click **OK**.

   The application is automatically checked in.

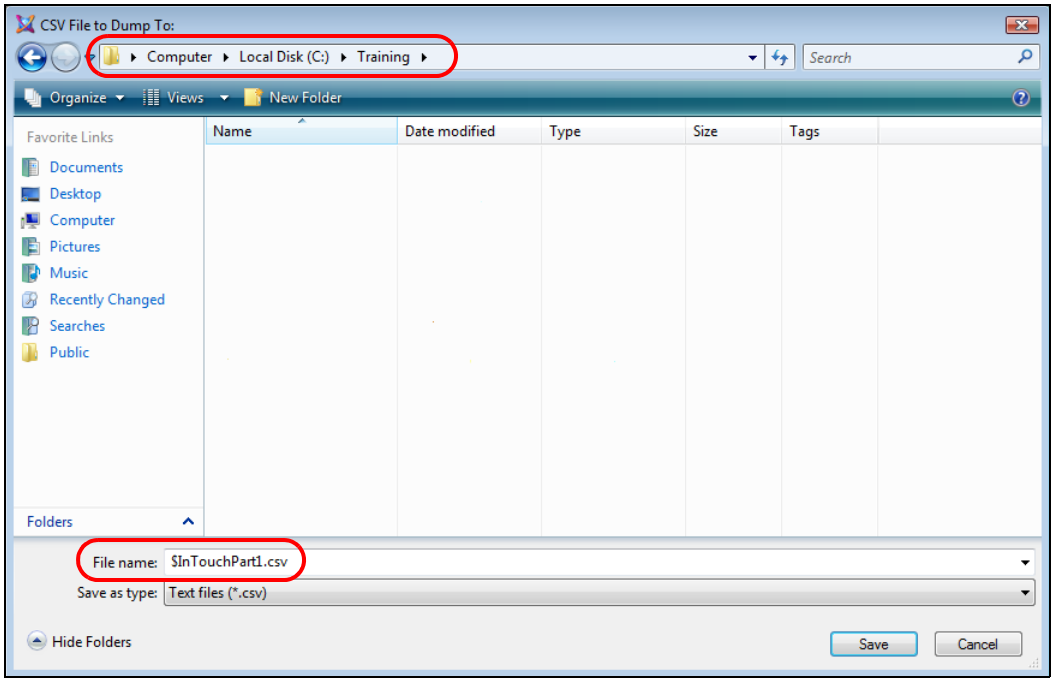


4. Click **Close**.

5. In ArchestrA IDE, **Template Toolbox**, right-click **Class Templates\$InTouchPart1** and select **Export | DB Dump**.

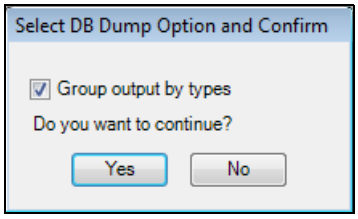


6. Navigate to **C:\Training** and keep the default **File name**, **$InTouchPart1.csv**.



7. Click **Save**.

   The **Select DB Dump Option and Confirm** dialog box appears.



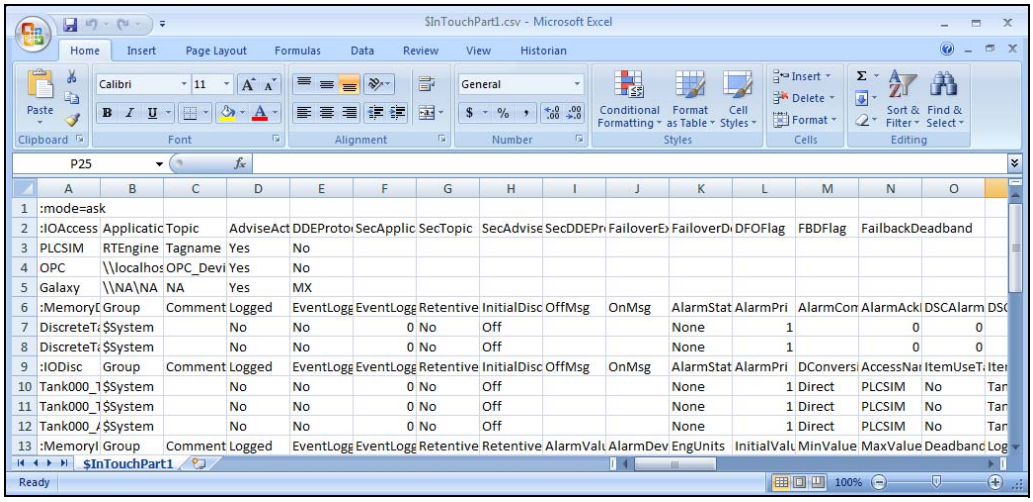
8. Leave the default selection and click **Yes**.

   You will receive a message when the dump is successful.

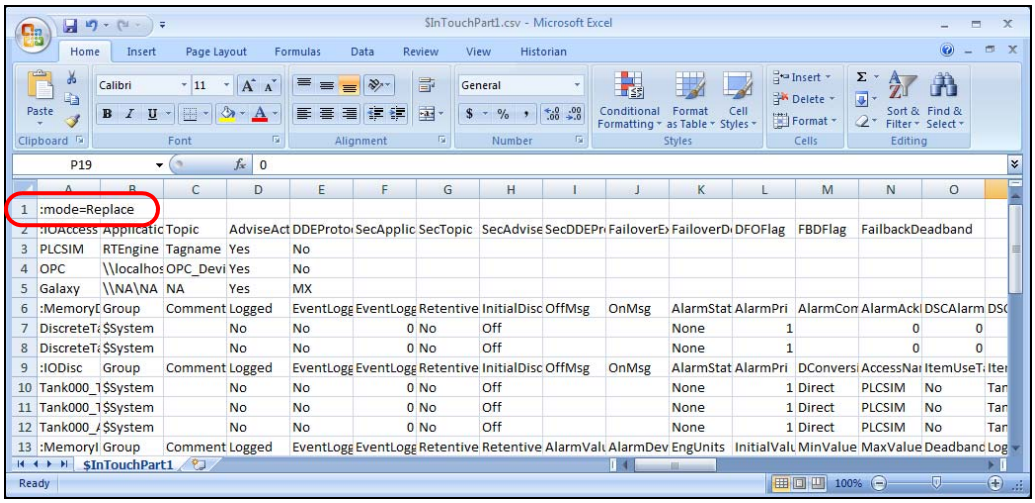
9. Click **OK**.

10. Open Excel, then open the **C:\Training\$InTouchPart1.csv** file.

Note: In the Excel **Open** dialog box, make sure **All Files (*.*)** is selected so you can see the .csv file.



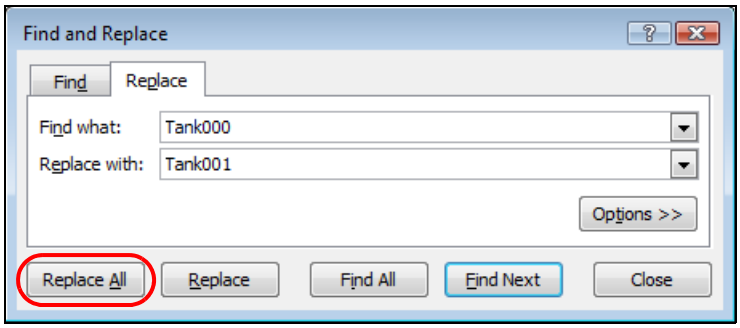
11. Change **:mode=ask** to **:mode=Replace**.



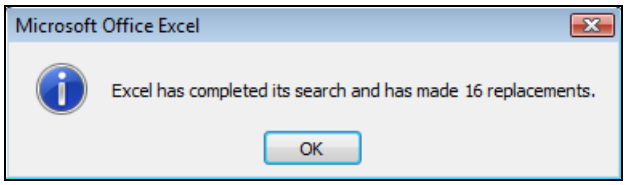
Next, you will search for all tags that start with Tank000 and replace them with Tank001. This will create 8 new tags when imported.

12. In Excel, press **Ctrl+F** to open the **Find and Replace** dialog box.
13. Click the **Replace** tab.
14. In the **Find what** field, enter **Tank000**.

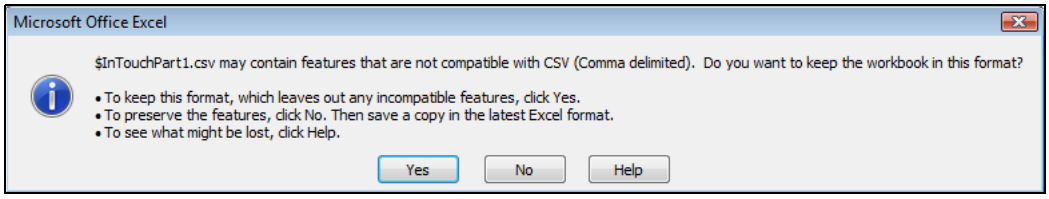
15. In the **Replace with** field, enter **Tank001**.
16. Click **Replace All**.



17. When the process is complete, click **OK**.

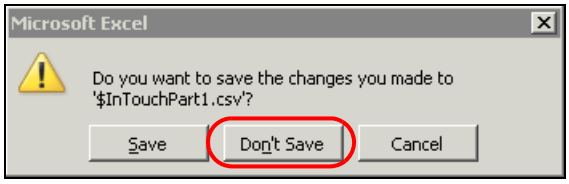


18. Click **Close**.
19. In Excel, click **Save** .
20. Click **Yes** to keep the **CSV** file format.



21. Close Excel.

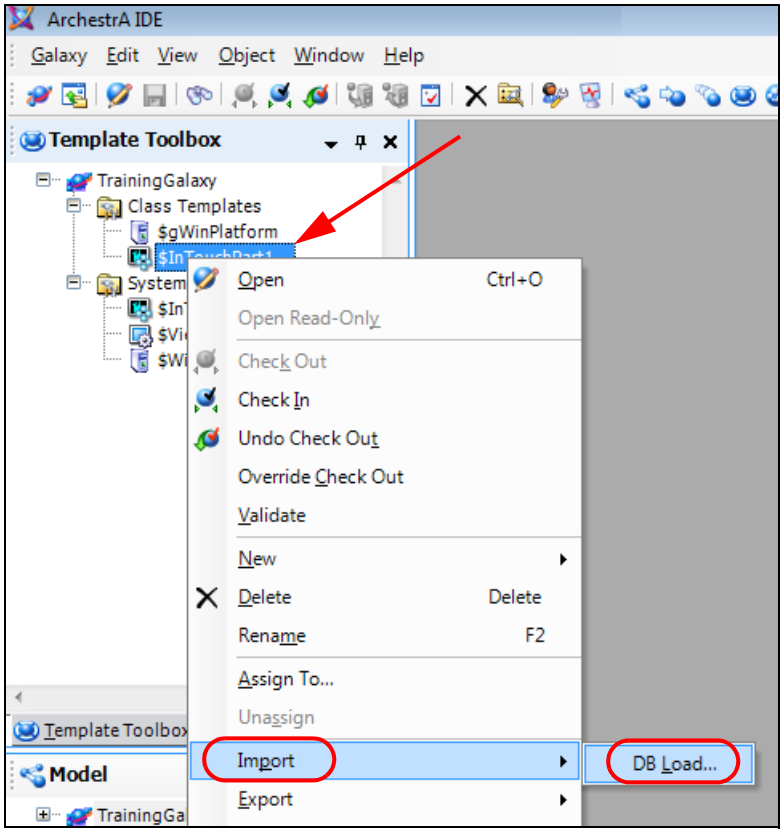
    The following message appears:



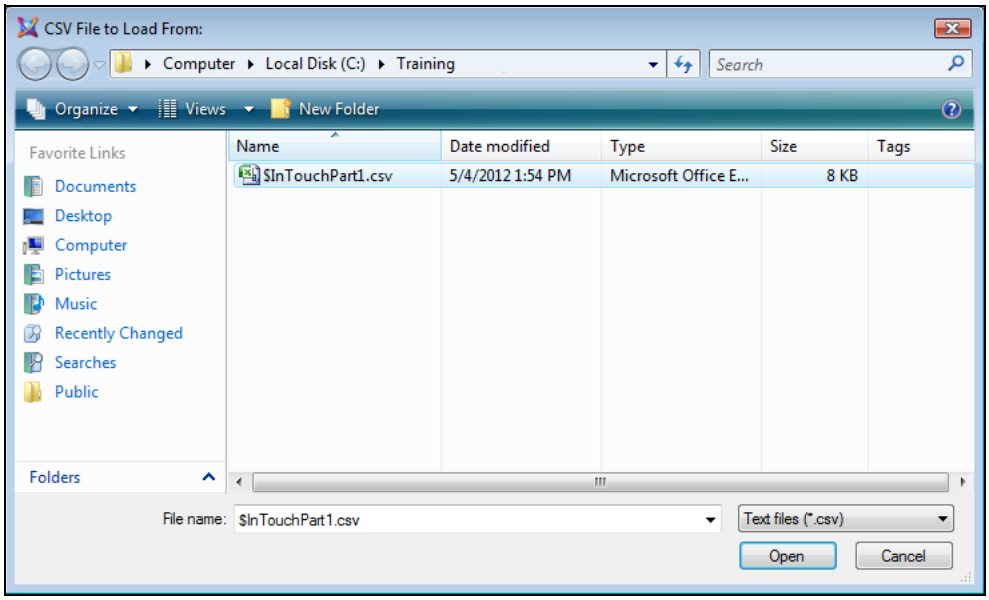
22. Click **Don't Save**.

Import New Tags

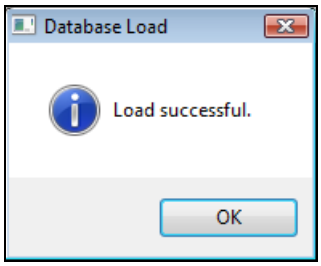
23. In ArchestrA IDE, right-click your InTouch application and select **Import | DB Load**.



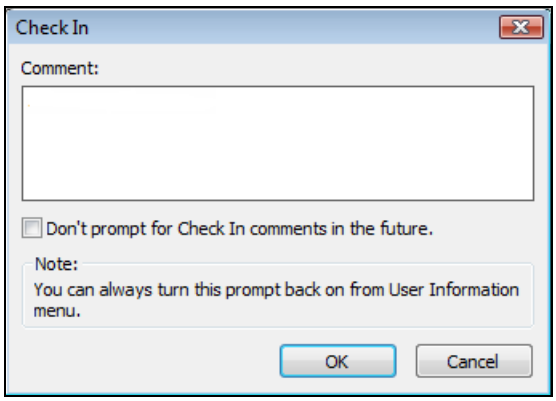
24. Locate and select the **.csv** file in the **Training** folder, then click **Open**.



A message displays when the load is successful.



25. Click **OK**.
26. Enter an appropriate check-in comment.



27. Click **OK**.

    The **Tank001** tags are added to the **Tagname Dictionary**.

28. Click **Close**.

Review Changes in Tagname Dictionary

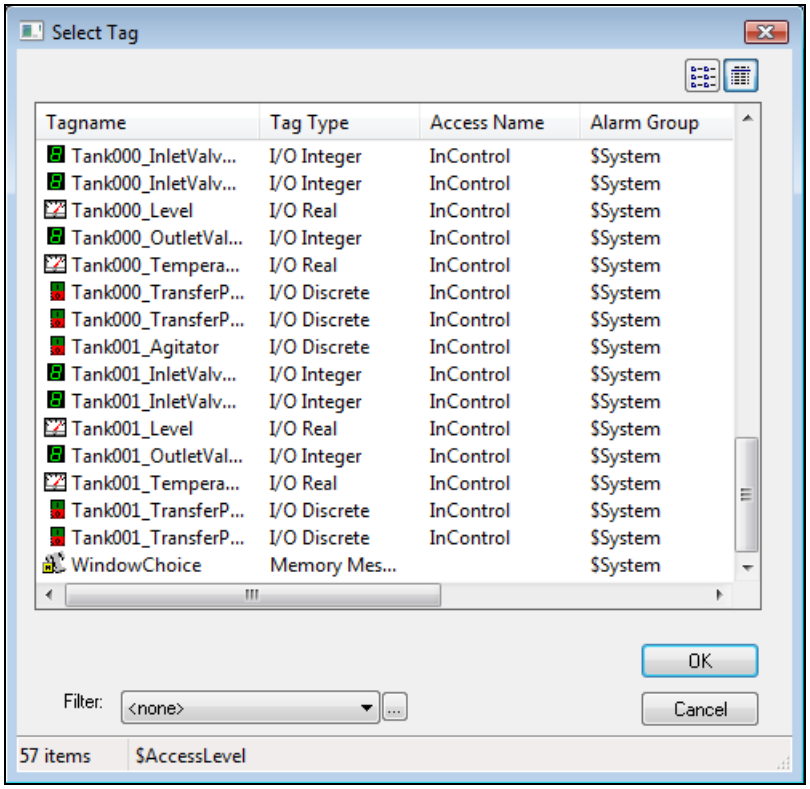
29. Double-click your InTouch application to start WindowMaker.

    The **Windows to Open** dialog box opens.

30. Click **Cancel**.

31. Open **Tagname Dictionary**, then click **Select** to view the list of tags.

32. Scroll down the list to locate **Tank000** and **Tank001**.



Modify Tags

Now you will make a modification to one of the new tags in the Tagname Dictionary.

33. Select **Tank001_Level**, then click **OK**.

34. Change **Max EU** to **1500**.

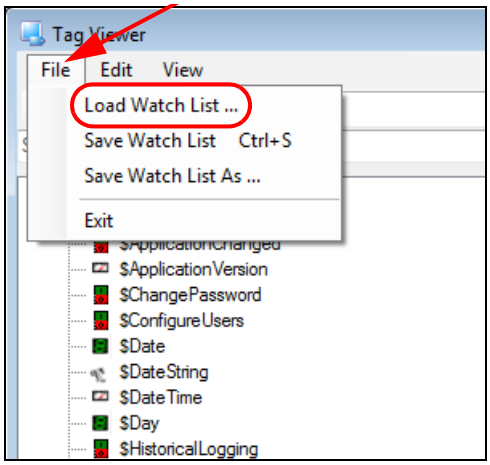
35. Click **Close**.

Create Watch List

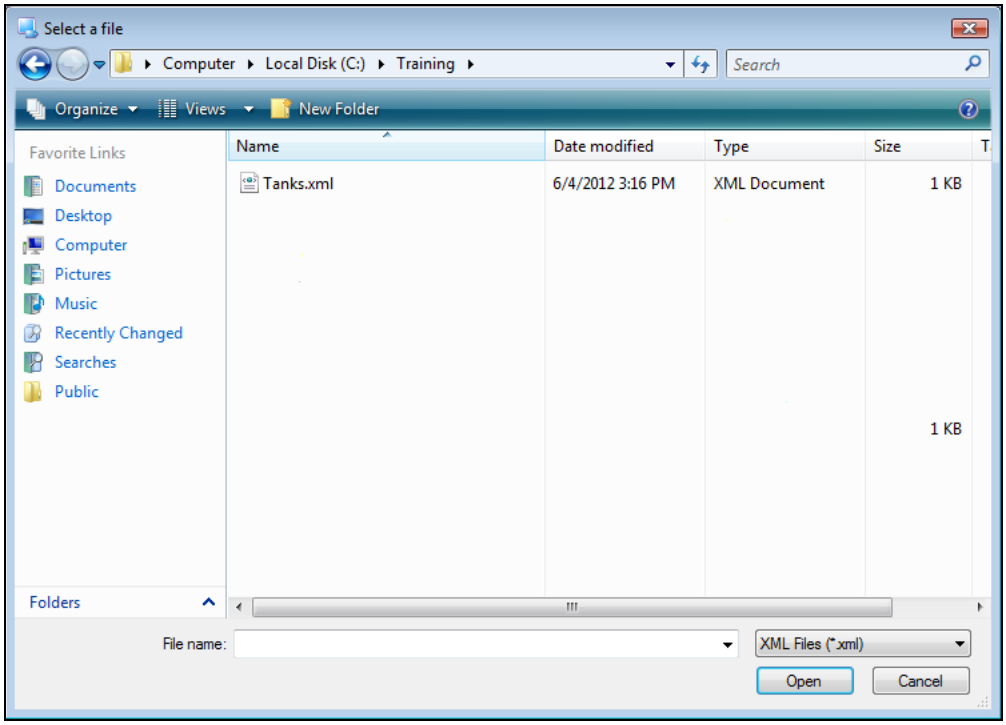
Next, you will create a new Watch List.

36. Click **Runtime**.
37. On the **Special** menu, click **Tag Viewer**.

    The **Tag Viewer** window appears.
38. On the **File** menu, click **Load Watch List**.

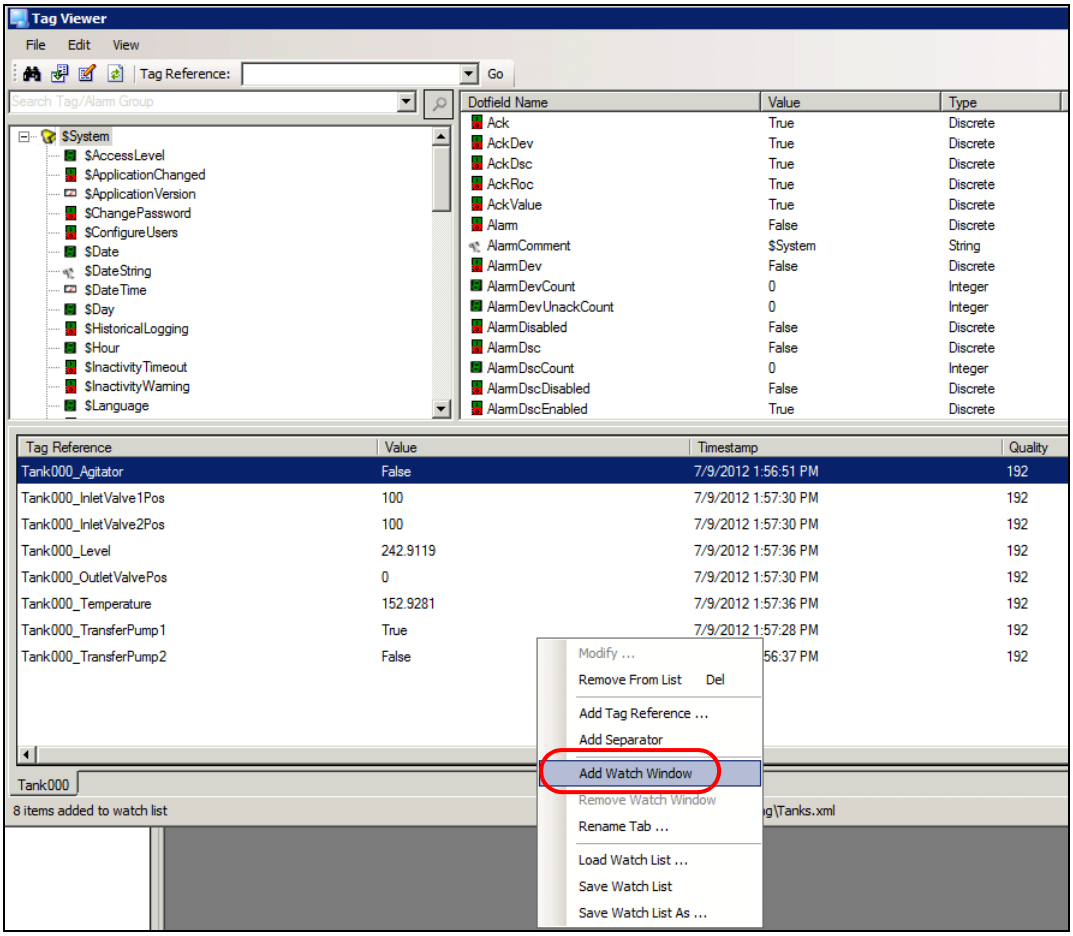


39. Navigate to **C:\Training** and double-click **Tanks.xml**.



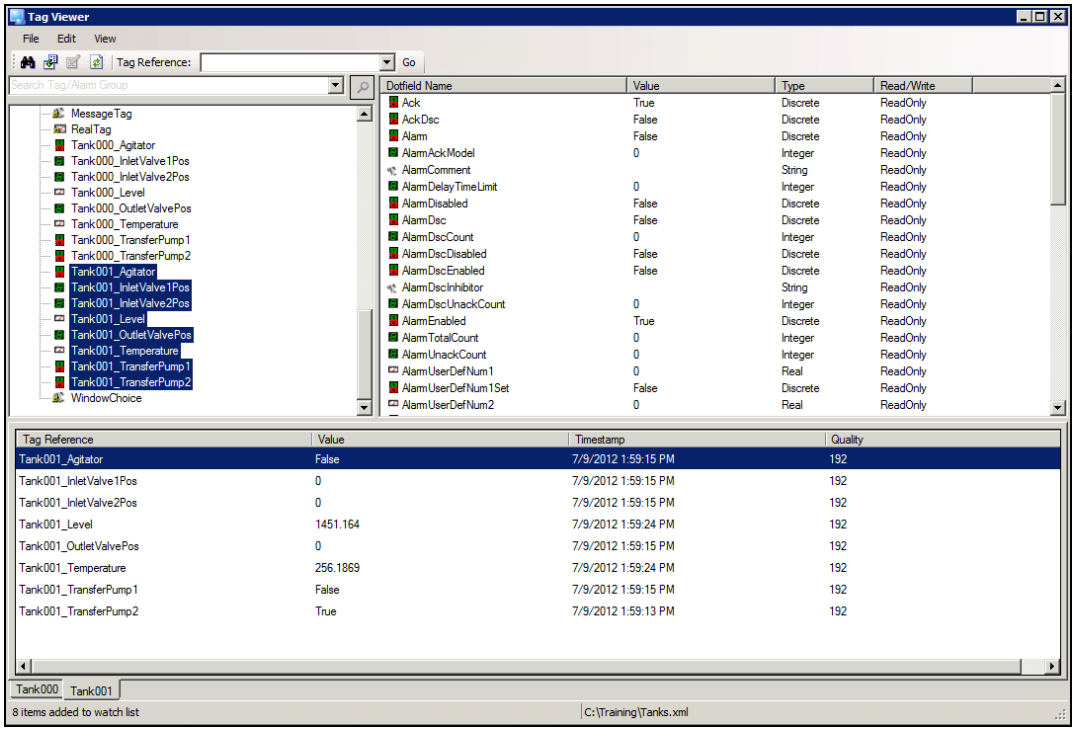
The **Watch List** opens in **Tag Viewer**.

40. In the **Watch List** window, right-click anywhere and select **Add Watch Window**.



41. Rename the Watch List **Tank001**.
42. Click **OK**.
43. Locate the **Tank001** tags in the top area.
44. Hold the **Shift** key and select the following tags.
    - **Tank001_Agitator**
    - **Tank001_InletValve1Pos**
    - **Tank001_InletValve2Pos**
    - **Tank001_Level**
    - **Tank001_OutletValvePos**
    - **Tank001_Temperature**
    - **Tank001_ TransferPump1**
    - **Tank001_ TransferPump2**

45. Drag the items to the **Tank001 Watch List**.



Note: A **Quality** of **192** indicates the newly created Tank001 I/O tags are healthy and communicating with the PLCSim Access Name.

46. In the **Watch List** window, right-click and select **Save Watch List**.
47. **Close** Tag Viewer.
48. In WindowViewer, click **Development!** to return to WindowMaker.

Section 8 – Finding and Deleting Tags

There are several ways to maintain and edit links and tags. This section discusses the various methods such as the InTouch Cross Reference Utility, updating user counts, and deleting, substituting, and converting tagnames.

Overview

Earlier in this module you learned about the Tagname Dictionary and data sources. You also learned about the Tag Viewer and monitoring tags. In the last lab, you use a tag database utility to create and modify tags. This section discusses tag additional tag database utilities used to create, modify, and delete tags.

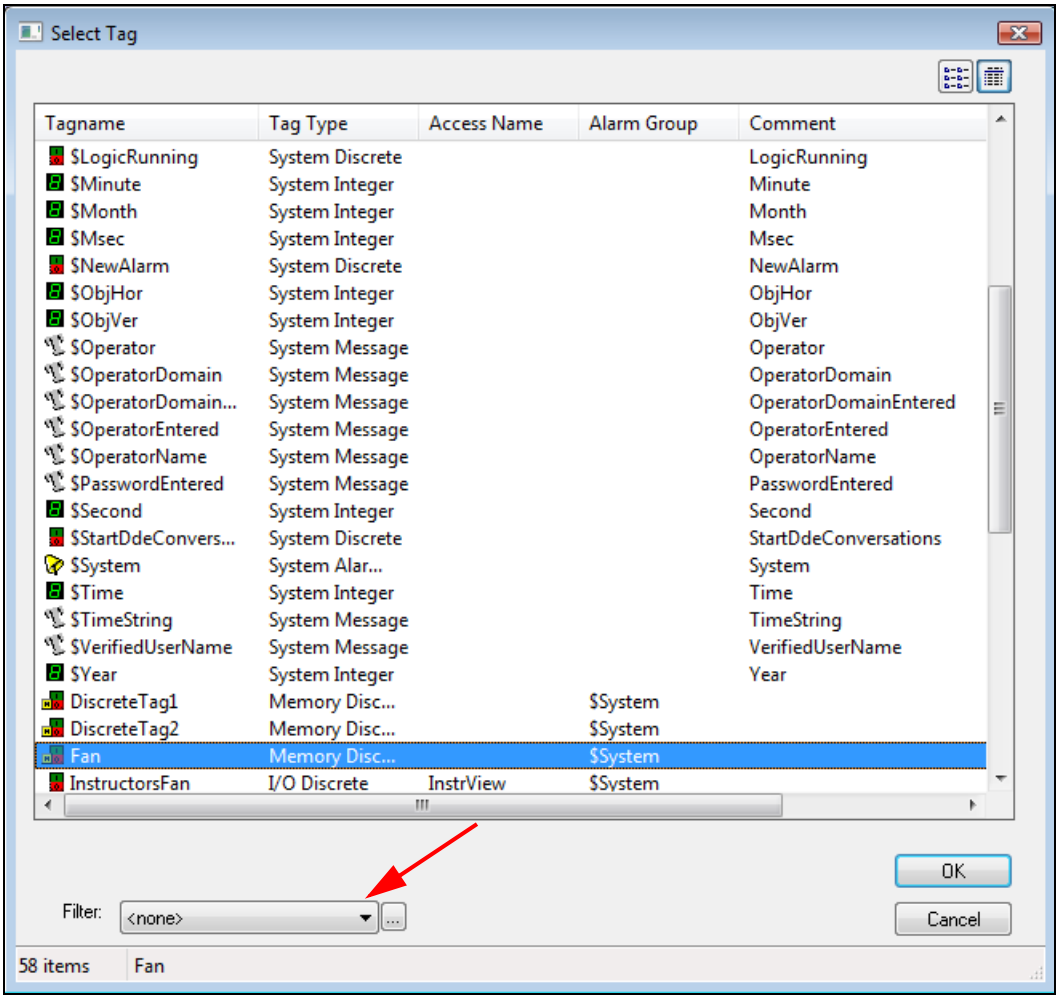
You can delete tags you no longer need, or unused tags. However, the delete tag functionality is deliberately difficult. You cannot delete a tag that is used in any script or animation link. Use the Cross Reference utility to determine tag usage, and then delete all links and scripts containing the tagname you want to delete. You can also print the Tagname Dictionary for review.

The Use Counts utility maintains a use count for each tagname in the database. This count is not updated automatically for certain operations such as changing and deleting tagnames in links or scripts. Run the utility to update the use count prior to deleting tags. You must close WindowViewer prior to deleting tags.

Once these steps have been completed, you can delete individual tags or use an option to delete all unused tags.

Tag Filter

If you have a large number of tags in your Tagname Dictionary, you can configure a filter to view a subset of the tags. This option is available from the Tagname Dictionary browser. As discussed earlier, on the **Tagname Dictionary** dialog box, click **Select** to open the **Select Tag** dialog box, and choose a **Filter** option.



You can select a defined filter, or click the ellipsis next to the **Filter** field to open the **Define Tag Filter** dialog box. You can enter a filter name and filter options such as a specific tag time, access name, or alarm group.

You can also use the following wildcard expressions in your filter:

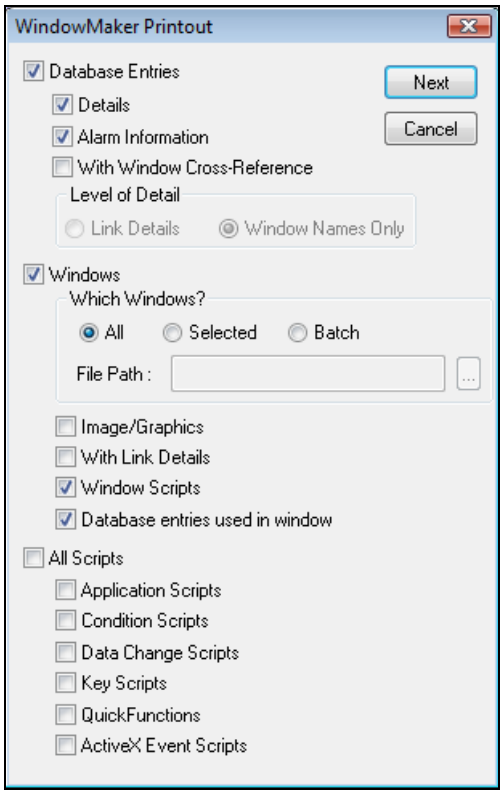
- The multiple character wildcard is the asterisk (*). For example, “Asyn*” searches for all tagnames beginning with the characters “Asyn”.
- The single character wildcard is the question mark (?). For example, the “Tag?” filter searches for all four-character tagnames that begin with “Tag”. The “Tag*” filter searches for all tagnames that begin with “Tag”.
- Any sequence of valid tagname characters, together with the two wildcard characters, is acceptable in a filter. Valid tagname characters are: A-Z, a-z, 0-9, !, @, -, #, $, %, _ and &.

You can also delete a search filter you no longer need by selecting it from the **Filter** list and clicking **Delete**. The filter is immediately deleted. The filter is reset with each reuse.

Printing the Tagname Dictionary

Printing the Tagname Dictionary details can help determine tagname usage. In addition to printing a .CSV (comma-separated values) file, you can print listings of the Tagname Dictionary details, alarm information, link details, and scripts.

To print the Tagname Dictionary, on the **File** menu, click **Print** to open the **WindowMaker Printout** dialog box.



Check **Database Entries** to print all database information. When selected, you can define the amount of detail you want to print. Check **Windows** to print listing of the document entries used in the application windows. You can further refine what you print based on the type(s) of tags you want to see, such as **Application Scripts** or **Data Change Scripts**.

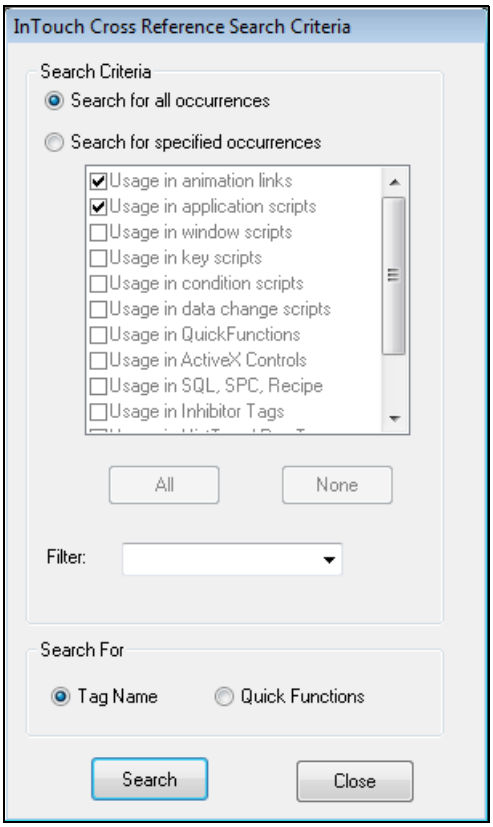
InTouch Cross Reference Utility

The InTouch cross reference utility allows you to determine your tagname and SuperTag usage in animation links, InTouch QuickScript, QuickFunctions, ActiveX controls, scripts, and the following InTouch add-on programs: SPC Pro, SQL Access Manager and Recipe Manager.

For all objects such as animation links, wizards, and ActiveX controls, it displays the window name and the coordinates of all objects linked to the tagname. It also allows you to view any QuickScript or QuickFunctions in which a tag reference is found.

The InTouch cross-reference tool can remain running in WindowMaker while other tasks are performed.

You can access this utility from the **Special** menu, **Cross Reference**. The **Cross Reference Search Criteria** dialog box allows you to limit your search to specific tags or scripts.



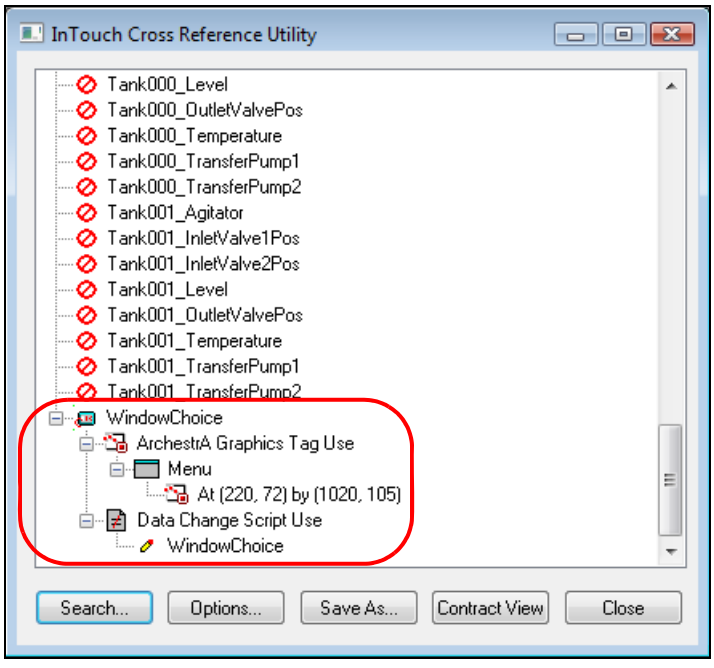
The **InTouch Cross Reference Search Criteria** dialog box contains options to limit your search criteria and filter the results of your tag search. These options are as follows:

Option	Purpose
Search Criteria	Enables you to limit the scope of your search. The scope can be determined by selecting only the options you need: • Search for all occurrences: Search for all uses of a tagname (or SuperTag) in animation links, InTouch QuickScript and all add-on programs such as SPC, SQL Access Manager, Recipe Manager, etc. • Search for specific occurrences: Search for only the tagnames (or SuperTag) in defined usage(s). **Note:** To search for the usage in window scripts, select **Usage in window scripts**.
Filter	Enter a unique name to identify the filter that you wish to define or select an option from the **Filter** right-click drop-down. When you define filters, the name you enter is added to the Filter list.
Search	Begins the cross-reference search based on the defined view criteria.

Cross Reference Search Results

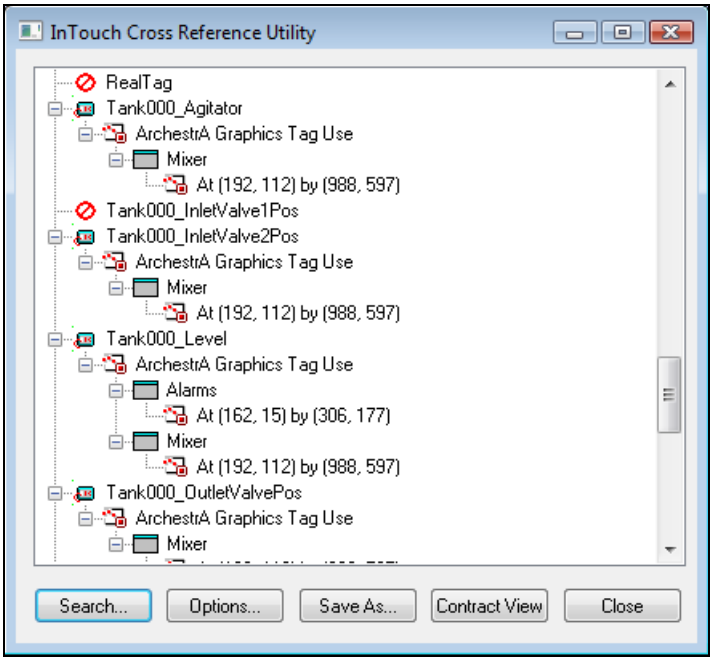
When you perform a cross reference search, the **InTouch Cross Reference Utility** dialog box appears listing all instances of usage found for the **Filter** that you specified.

If no filter is used, all tagnames defined in the current application's Tagname Dictionary are displayed.



The icons adjacent to the filename indicate tag type and associated animation or script. The circle and slash icon indicates that the tagname or SuperTag is not assigned to an object.

After completing a search, the **InTouch Cross Reference Utility** dialog box opens displaying the results of the search.



Here, you can change the search criteria by clicking **Options** and reentering search parameters to start your search again. The Options list at the bottom of the dialog box provides you with two options. You can display the search results in either one of the following views.

Cross-Reference by Tagname

Cross-referencing by tagname, alphabetically lists all tagnames found for the search criteria defined. This is the default view.

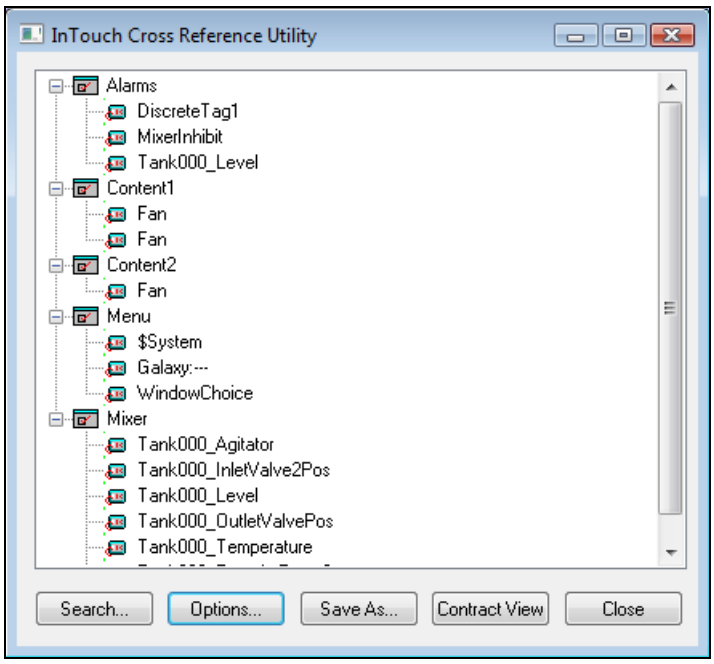
Note: The above graphic shows the Cross-Reference utility filtered by tagname.

Based on your defined search criteria, this view allows you to display the usage of all tagnames found in windows, animation links, scripts, and add-on applications.

- To expand the view, either double-click a displayed tagname and then double-click **Animation Links**, or use the **Expand View** button.When the view is expanded, the window name and the location (coordinates) of the object(s) linked to the tagname are displayed.
- Click **Contract View** in the **InTouch Cross Reference Utility** dialog box to return the dialog box to its default mode.
- The drop-down list field at the top of the dialog box displays all scripts associated with the selected tag. Open the list to select another script.
- For **Application**, **Window**, **Key**, and **Condition** scripts, the list contains the names of all scripts that see this tagname.
- In the case of **Data Change** scripts, only the tagname is listed.
- For **QuickFunctions**, the list will contain the names of all QuickFunctions (read-only view).

Cross-Reference by Window Name

Cross-referencing by window name, sorts the display by window name and then by the tagnames used in the window.



The **Options** button can be found in the **InTouch Cross Reference Utility** dialog box (on the **Special** menu, select **Cross Reference | Search | Options**).

When selecting **By Window** view, the windows display alphabetically. Expand each window view to see all tags associated with it.

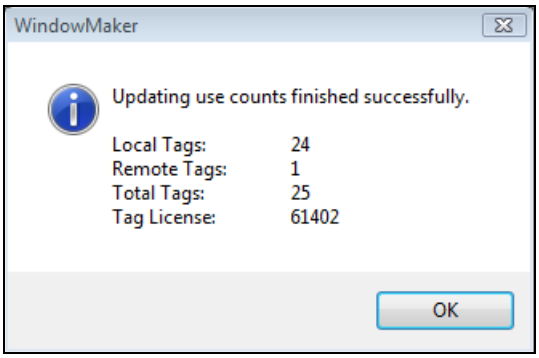
Use Counts and Unused Tagnames

InTouch maintains a use count for each tagname in the database. This count is not updated automatically for certain operations, such as deleting a window, changing or deleting tagnames in links or scripts, etc. In these cases, InTouch continues to consider the tagname as being used in the application and does not allow it to be deleted. Therefore, you need to update your use count to enable deletion of one tagname, or multiple tagnames.

Updating Use Counts

Since InTouch maintains a use count for each tagname in the database, you will need to update the use counts to set all unused tagnames to zero before InTouch will allow you to delete them.

After closing all windows, on the **Special** menu, select **Update Use Counts**, and confirm when prompted.



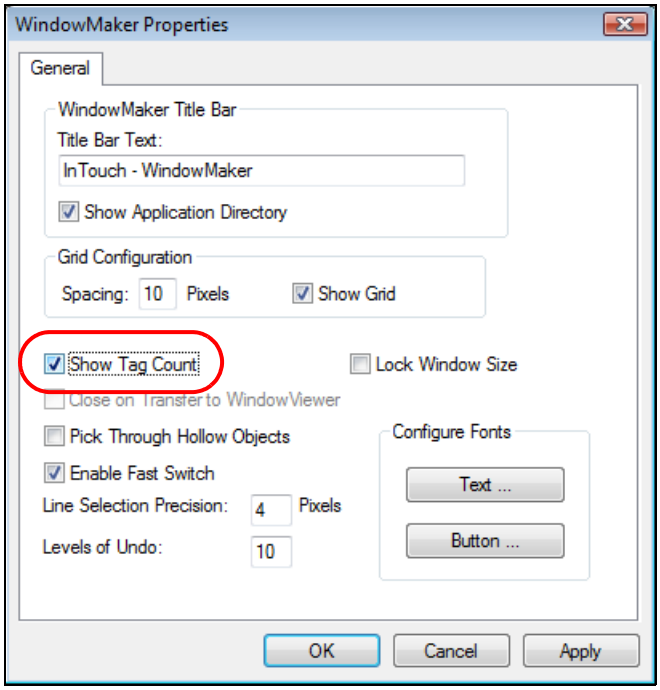
The message lists all the tags in use, as well as the number of tags licensed.

Displaying the Tag Use Count

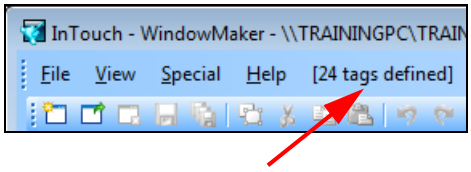
The number of local tags defined in the Tagname Dictionary can be displayed in the WindowMaker menu bar. The tag count does not include internal system tags or remote tagname references.

Note: System Tags (37) do not count toward licensed tags.

To display the number of local tags defined in the Tagname Dictionary, display the **WindowMaker** dialog box on the **Special** menu, by selecting **Configure | WindowMaker**. Use the options there to display the tag count.



Open WindowMaker to view the total number of local tagnames defined in the Tagname Dictionary.



Deleting Tags

When a tag is referenced by any script or animation, it cannot be deleted. Use the **Cross Reference** tool to determine everywhere the tag is applied.

The **Delete** button is not available if WindowViewer is running or if InTouch has flagged the tagname as being used in the application. Therefore, techniques must be applied to free tag usage before a tag can be deleted, because deleting tags is made deliberately difficult in InTouch.

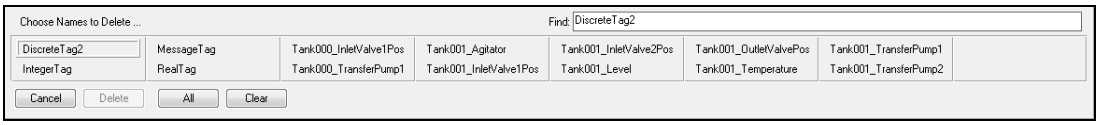
- When WindowViewer is running, tags cannot be deleted.
- When a tag is referenced by any script or animation link, it cannot be deleted.
- Use the Cross Reference tool to determine tag usage.

In other words, to delete an unused tag from the Tagname Dictionary, you must complete the following tasks:

- Shut down WindowViewer.
- Determine tag use with the Cross Reference utility.
- Delete the tag from all windows, animation links, Failover configuration, ActiveX and .Net Controls and scripts containing the tagname.
- Update Use Counts under **Special | Update Use Counts**.
- Delete the tag from the Tagname Dictionary or use **Special | Delete Unused Tags**.

Deleting Multiple Unused Tagames

After you have updated the use count, InTouch allows you to delete all unused tags. You can either delete them one-by-one in the Tagname Dictionary, or delete several at once. Display the **Choose the Names to Delete** dialog box where you can select and delete unwanted tags.



References to a tag inside of quotes (“ “), such as within a script or animation link, excludes this tag from the use counts check. Using a tag inside of quotes is common in **Indirect Tag** scripting. Therefore, know what you can and cannot delete from the **Delete Unused Tags** list, as many an unwary application developer has accidentally deleted tags which are used in this particular way, thus breaking key application functionality.



invensys
Learning Services

Module 3 – Data Presentation

Section 1 – Working with ArchestrA Symbols	3-3
Lab 6 – Working with ArchestrA Symbols	3-11
Section 2 – Using the ArchestrA Symbol Editor	3-37
Lab 7 – Modifying Your ArchestrA Symbol	3-47
Section 3 – ArchestrA Symbol Custom Properties	3-71
Section 4 – ArchestrA Symbol Animation	3-75
Lab 8 – Animating Symbols	3-95
Section 5 – ArchestrA Symbol Status and Quality	3-115
Lab 9 – Monitoring Custom Symbol Status and Quality	3-119

Module Objectives

- Link tags to ArchestrA Symbols
- Build a custom symbol
- Link custom properties to an embedded symbol
- Animate symbols
- Describe symbol quality and status

Section 1 – Working with ArchestrA Symbols

This section describes how to embed an ArchestrA graphic in a window in your InTouch application. Then it describes how to link tags and fields to graphics and configure custom properties to streamline how they are viewed in WindowMaker.

Overview

You have learned to insert ArchestrA symbols and embed them into a window of your IDE-managed InTouch application using the ArchestrA symbol library. You now learn how to connect them to InTouch tagnames.

Using ArchestrA symbols you can:

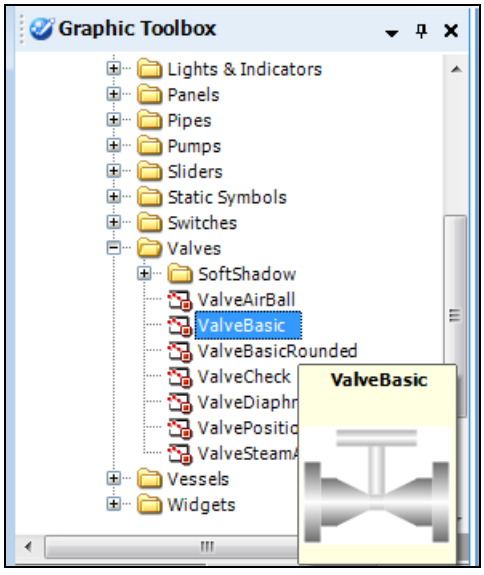
- Embed ArchestrA symbols into a window of a Managed InTouch application.
- Resize and reposition embedded ArchestrA symbols.
- Configure the Custom Properties of embedded ArchestrA symbols.
- Test the ArchestrA symbols in WindowViewer.

Graphic Toolbox

In the Graphic Toolbox, you can create a folder hierarchy to organize your symbols. You can also move symbols around within the folder hierarchy. These folders are called “graphic toolsets”.

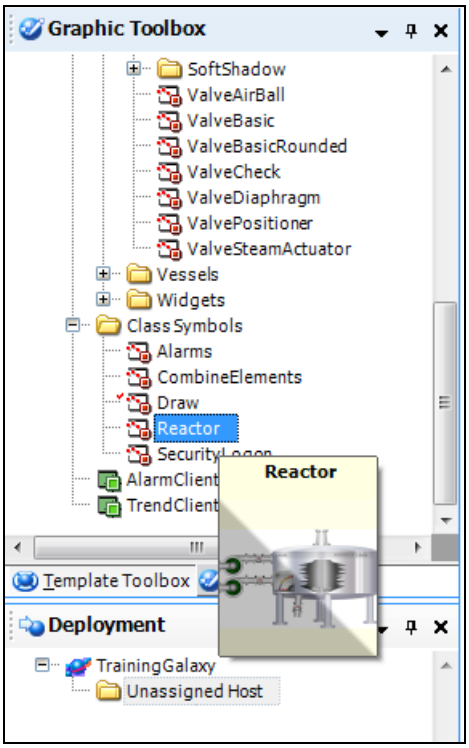
The power of ArchestrA Graphic Symbols is reusability. Design symbols that you can reuse over and over again with your InTouch application, or even better design them to be used in all of your InTouch Applications. For example, use the Graphic Toolbox to create a generic valve that you will use in numerous process graphics.

When you create a new symbol using the Graphic Toolbox, the symbol is listed in the Graphic Toolbox. After you create a new symbol, you can move the symbol or open the ArchestrA Symbol Editor to edit it.



Changes made to graphic symbols affect all the embedded instances of those symbols, wherever they are used, and in any InTouch application within your Galaxy. This powerful design approach allows you to maintain tight compliance in design and propagate modifications everywhere without having to modify the individual copies of the symbol being used.

A strong design approach involves combining common generic symbols into assemblies. For example, embedding pipes, valves, pumps, motors, and a vessel into a single mixing process graphic. This assembly may then be reused in your InTouch application to represent many similar mixer processes throughout your production facility.



The ArchestrA Symbol Editor

ArchestrA IDE includes the ArchestrA Symbol Editor, used to create and modify your symbols. These symbols may then be embedded into a Managed InTouch application.

An ArchestrA basic graphical object is called an element. Typical elements include lines, rectangles, ellipses, and curves, as well as buttons and Windows control objects. To draw a graphic element, from the **Tools** panel, select the button representing the graphic element you want to insert into your symbol.

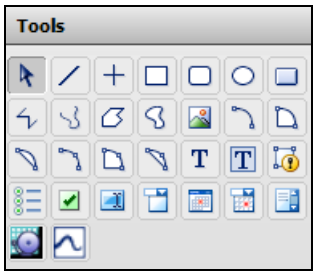
There are two methods to change the appearance of a drawn element, as follows:

- Change the drawing's properties listed in the Properties Editor
- Use the tools in the graphics toolbars

The ArchestrA Symbol Editor is comprised of the following components:

Tools Panel

The Tools panel contains elements you can select to draw your symbol on the canvas.

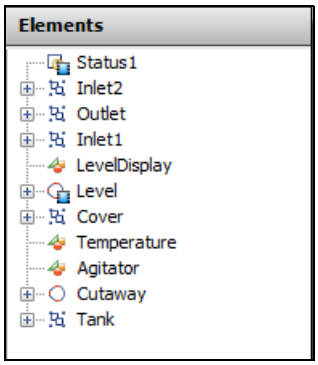


The Tools panel contains the following components:

- A pointer tool to select and move elements on the canvas
- Basic geometrical objects such as lines, rectangles, polygons, arcs, etc
- A status element you can use to show quality and status of selected ArchestrA attributes
- Windows common controls, such as combo boxes, calendar controls, radio button groups, etc

Elements List

The Elements list contains a catalog of all the elements on the canvas, as well as their sub-components (when graphic elements are drawn and then grouped together as a single element).

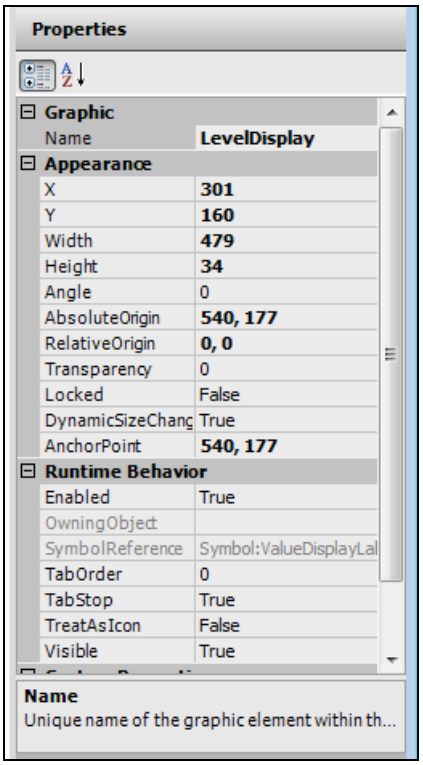


The Elements list is particularly useful for selecting one or more elements that are visually hidden by other elements on the canvas. You can use the Elements list to:

- See a list of all elements, groups of elements, embedded symbols, and client controls on the canvas
- Select elements or groups of elements to work with them
- Rename an element or a group of elements

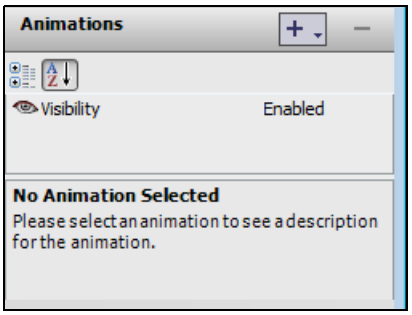
Properties Editor

You can use the Properties Editor to view and configure properties for the selected element or group of elements. These properties include size, location, color, and fill style, as well as runtime behavior.



Animation Summary

You can use the animation summary to review, select, and configure the animation behavior of a selected element on the canvas.



Canvas

The canvas is your drawing area. You use it as you would in other image editing software by drawing elements and changing them to your requirements.

Connecting Elements to Tags

As discussed earlier, a tag is simply defined as a value. The value represents a real-time value in your processes. Each tag has a variety of attributes such as tagname, min/max values, and access name.

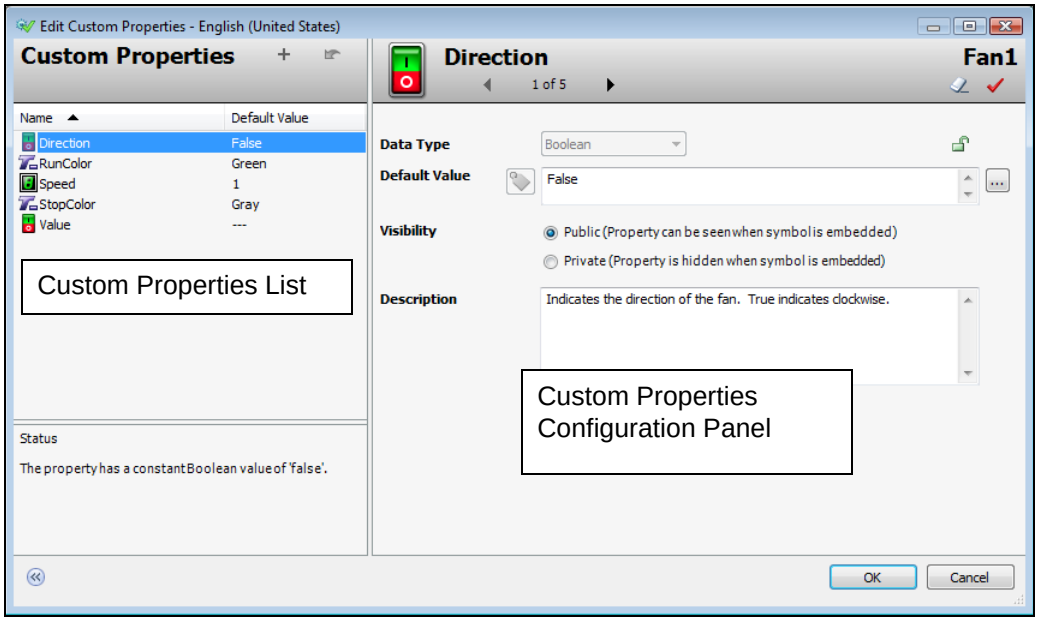
When working with an IDE-Managed InTouch application, you can link ArchestrA symbol elements, animations, and scripts to tagnames from the **Tagname Dictionary**. This can be done using two methods:

- Using the syntax **InTouch:Tagname** in the Symbol Editor
- After embedding an ArchestrA symbol, link tagnames to its **Custom Properties**

Using **InTouch:Tagname** syntax, hardcodes the link to a specific tag. If this tag is not available in the InTouch application where the symbol is embedded, it will not work. This reduces the reusability of the symbol with all InTouch applications, or requires certain tags be created in every InTouch application where the symbol will be used. The exception is when linking to $System tags such as **InTouch:$Second**, since $Second is available in every InTouch application.

Custom Property Visibility

You can manage all custom properties of a symbol in the **Edit Custom Properties** dialog box.



When you edit the custom properties of a symbol embedded from the ArchestrA Symbol Library, each property listed has a default value data type, default value, default visibility, and default description. The description is useful because it indicates what the symbol can do and to a degree how to configure it. You can see the name of the symbol and the custom property in the header of the right side of the dialog box.

Adding Custom Properties

To add a custom property, display the **Edit Custom Properties** dialog box, click the **Add** button **+**, and enter a name. Next, select the **Data Type** list of the custom property. Data types include the following:

- **Boolean** (InTouch Discrete)
- **Double** (InTouch Real)
- **Float** (InTouch Real)
- **Integer** (InTouch Integer)
- **String** (InTouch Message)
- **Elapsed Time / Time** (ArchestrA Object Based Design not supported by Tag-Based InTouch)

In the **Default Value** field, enter a literal value, reference, or expression, or browse for a reference using the **Browse** icon.

Where supported, you can select the **T** icon to toggle the Default Value field between an expression (literal value or constant) and a reference value (tag)

For **Visibility**, select either:

- **Public**, so that the custom property is visible and can be used in a parent symbol if the symbol is embedded or inserted.
- **Private**, so that the custom property is not visible outside of the defining symbol and cannot be used if the symbol is embedded or inserted.

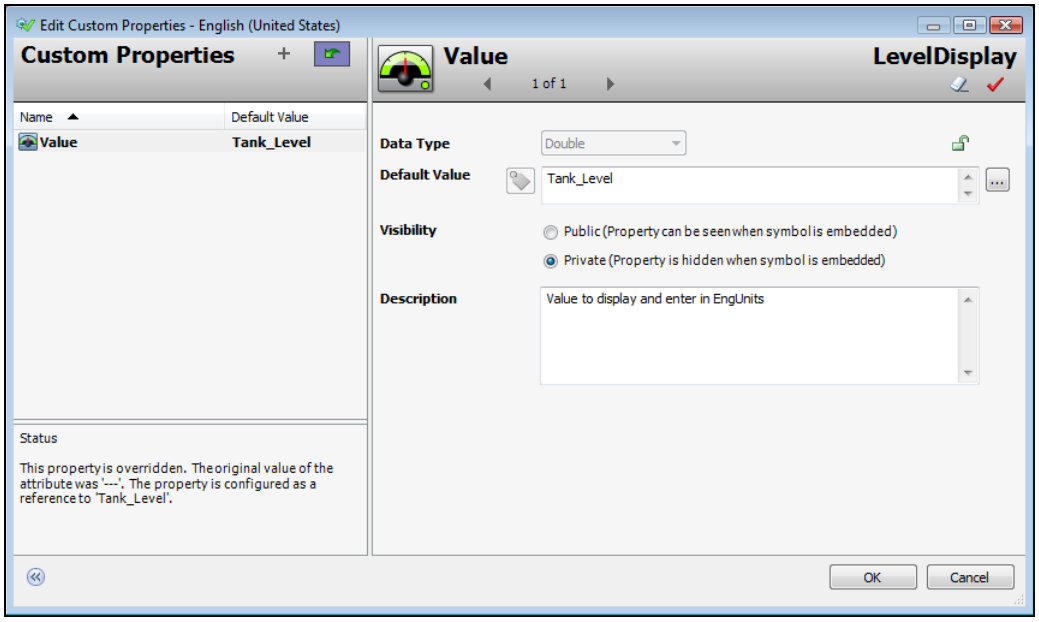
In the **Description** field, enter instructions or guidance for using this property, such as an example or short list of a legitimate values.

Note: For additional information see *Creating and Managing ArchestrA Graphics User Guide*.

Linking Custom Properties to External Sources

You can link Custom Properties of a symbol directly to external sources by configuring a special InTouch reference syntax in the **Default Value** field. When you insert the symbol in an InTouch window, the referenced InTouch tagnames connect to the tagnames in InTouch.

To connect an ArchestrA symbol to an InTouch tagname, right-click the symbol and select **Edit Custom Properties**, and then select the custom property you want to connect to and enter the InTouch tagname in the **Default Value** field, or click the **Browse Galaxy** button to select a tagname from the **Select Tag** dialog box.



Any animation in the ArchestrA symbol configured with that custom property will now interact with the InTouch tagname.

Editing Custom Properties

Edit an ArchestrA symbol using the **Edit Custom Properties** dialog box to do the following:

- View an ArchestrA symbol's custom properties
- Find out about a symbol's settings and behaviors
- Connect ArchestrA symbols with InTouch tagnames

Lab 6 – Working with ArchestrA Symbols

Introduction

In this lab, you will create a Graphic Toolset to organize your ArchestrA symbols. Next you will create a Reactor symbol by embedding and grouping symbols from the ArchestrA Symbol Library. Finally you will create a new window to host your Reactor symbol, link tags, and test in runtime.

Objectives

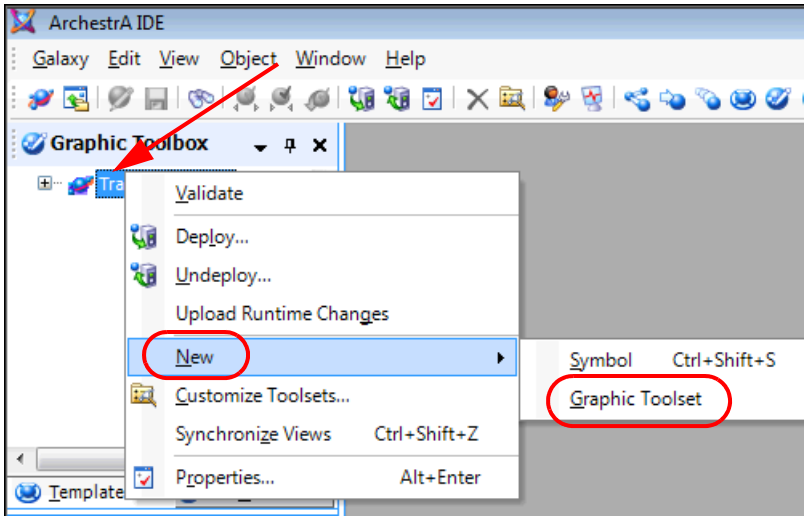
Upon completion of this lab, you will be able to:

- Create a Graphic Toolset
- Create an ArchestrA symbol
- Rename and group elements
- Link tagname Dot Fields

Create a Graphic Toolset

First, you will create a Graphic Toolset to organize your work and then create a new symbol.

1. In ArchestrA IDE, **Graphic Toolbox**, right-click **TrainingGalaxy** and select **New | Graphic Toolset**.

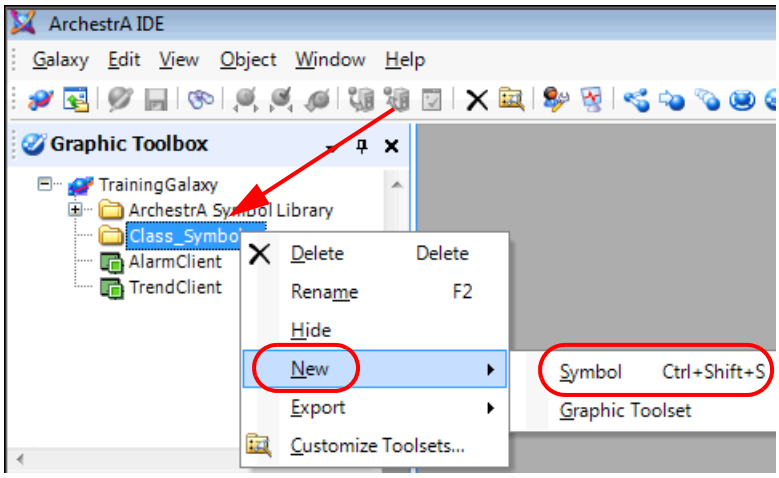


2. Change the name to **Class_Symbols**.

Create a Reactor Symbol

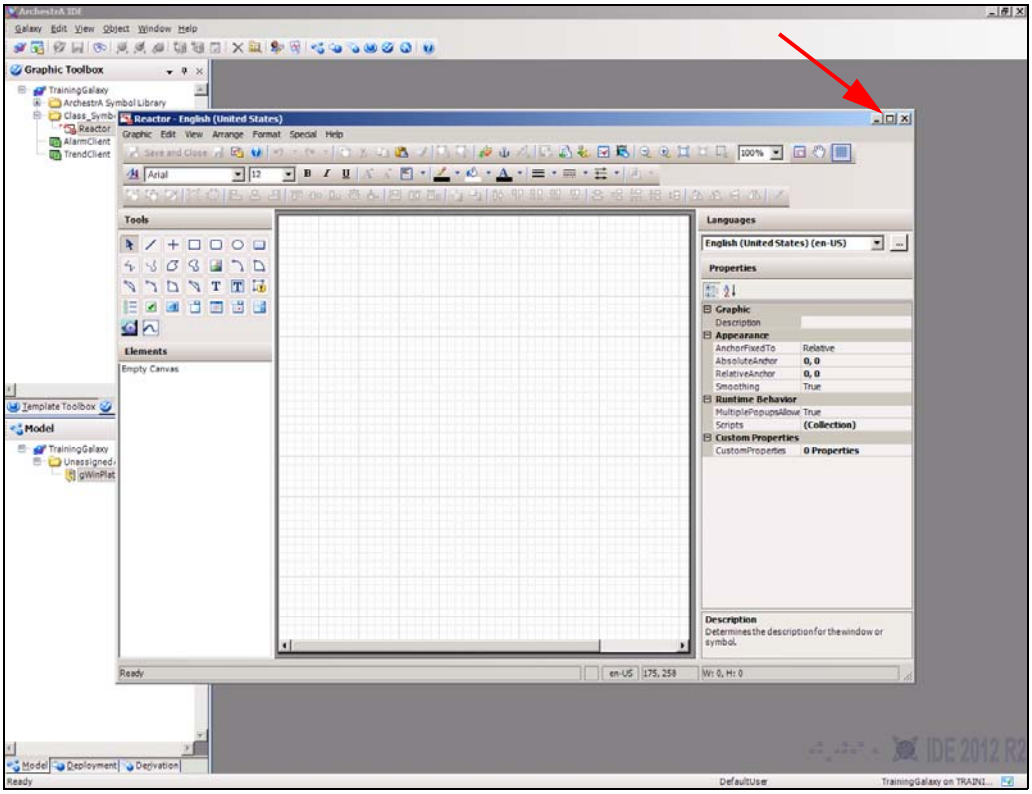
Next, you will create graphical elements using existing symbols in the ArchestrA Symbol Library.

3. Right-click **Class_Symbols** and select **New | Symbol**.

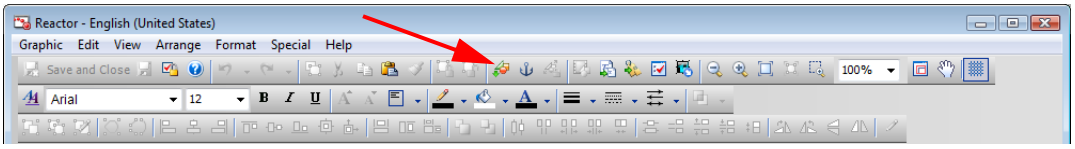


4. Name the new symbol **Reactor**.

5. Double-click **Reactor** to open it for editing.
6. Maximize **ArchestrA Symbol Editor** to provide as much work space as possible.

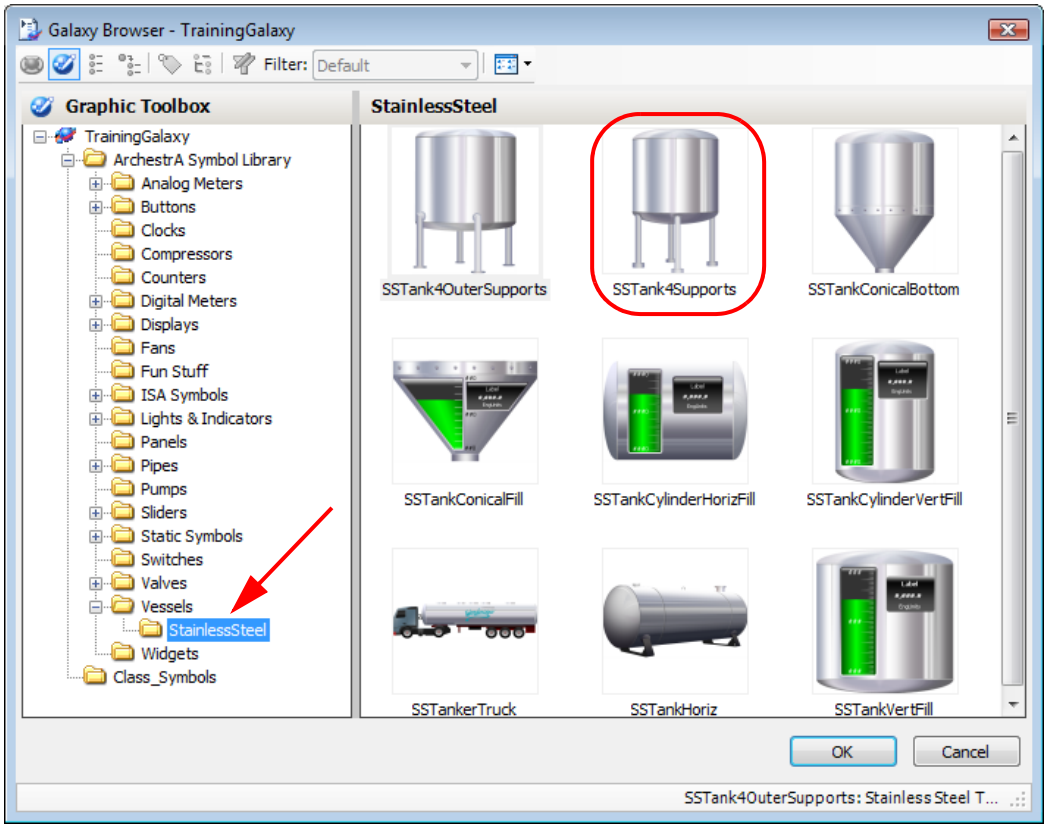


7. Click **Embed ArchestrA Graphic** .



The **Galaxy Browser** appears.

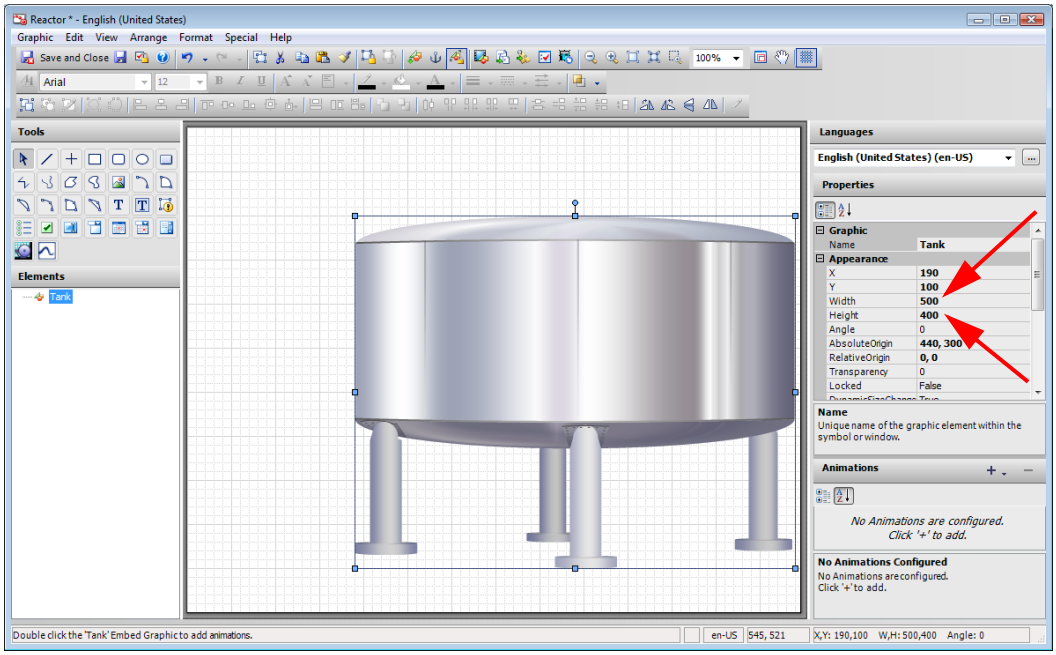
8. From **Graphic Toolbox**, expand **Vessels** and click **StainlessSteel**, then double-click **SSTank4Supports**.



9. On the canvas, place **SSTank4Supports**.

10. In **Properties** pane, **Name** field, change **SSTank4Supports1** to **Tank**.
11. Next, change the **Appearance** properties as follows:

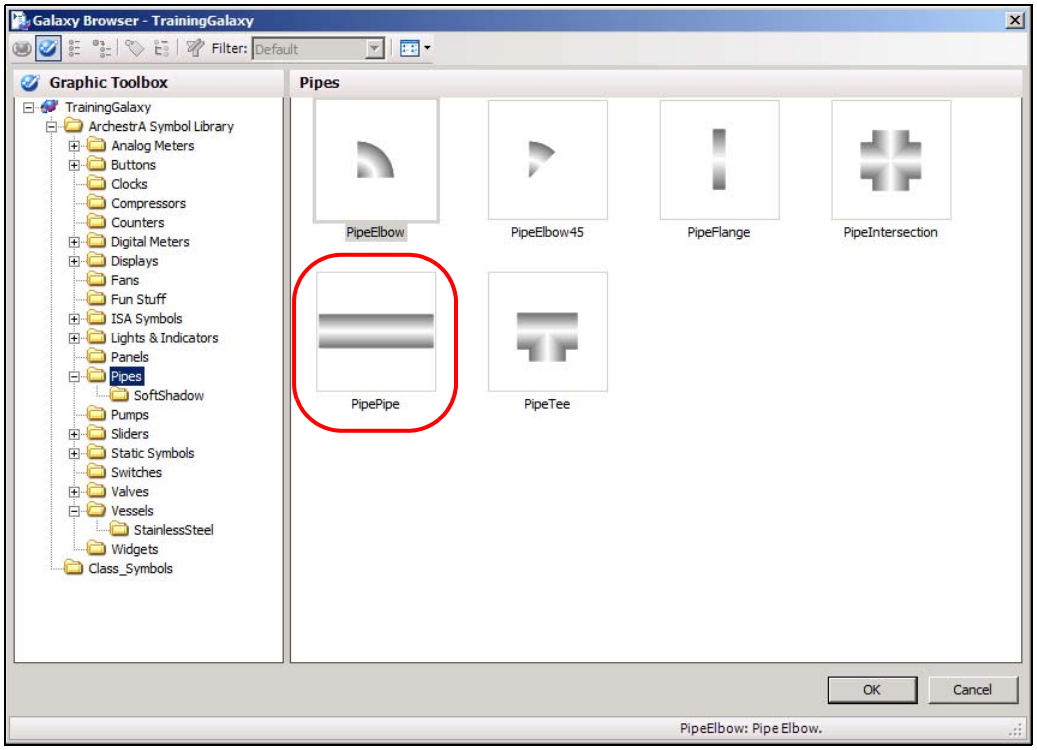
    Width: 500
    Height: 400

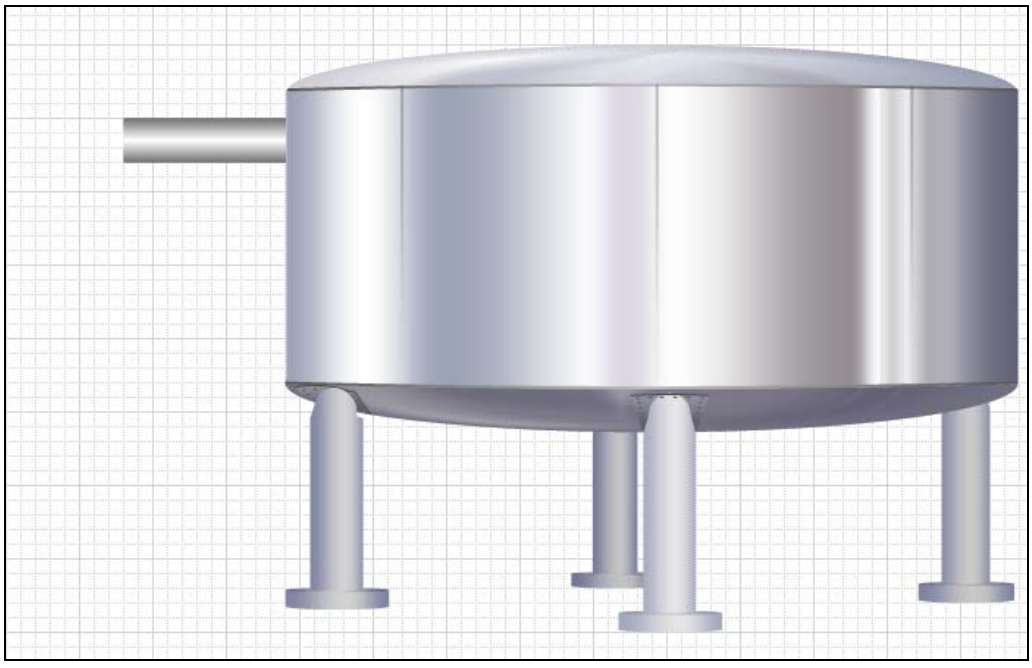


12. Position the tank on the right side of your canvas.

Next, you will add some features to your Reactor.

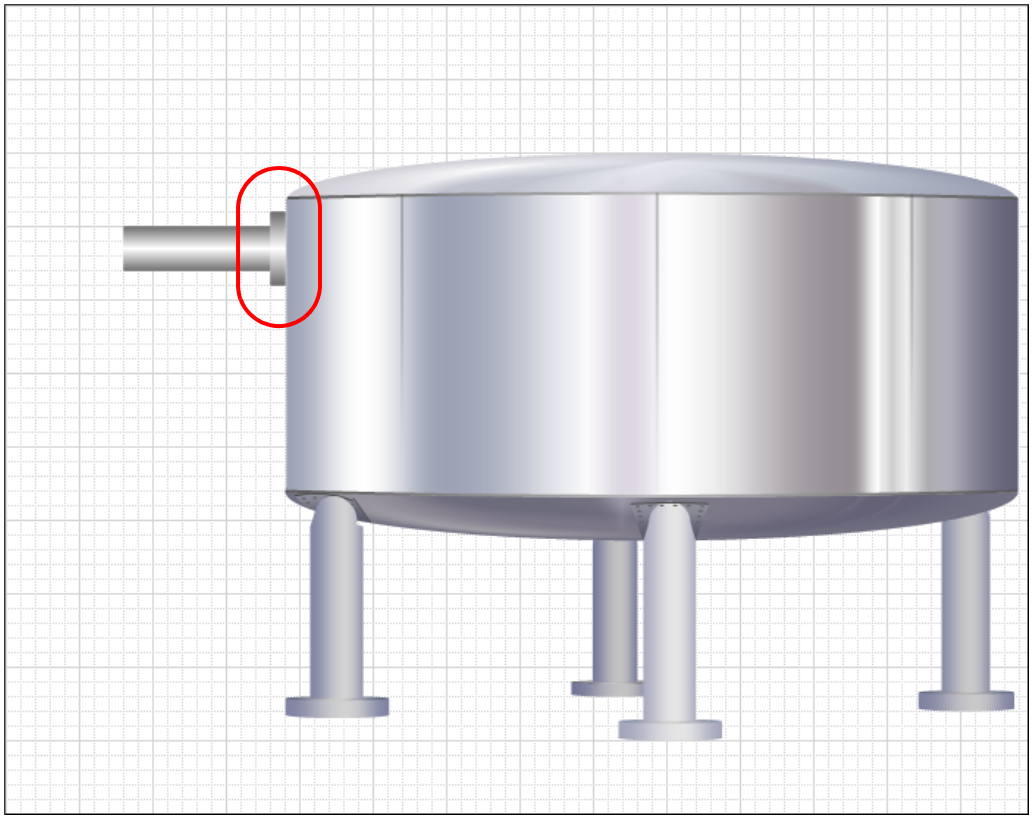
13. Click **Embed ArchestrA Graphic** .
14. Click **Pipes**, then double-click **PipePipe** and place it on the upper-left corner of the tank.





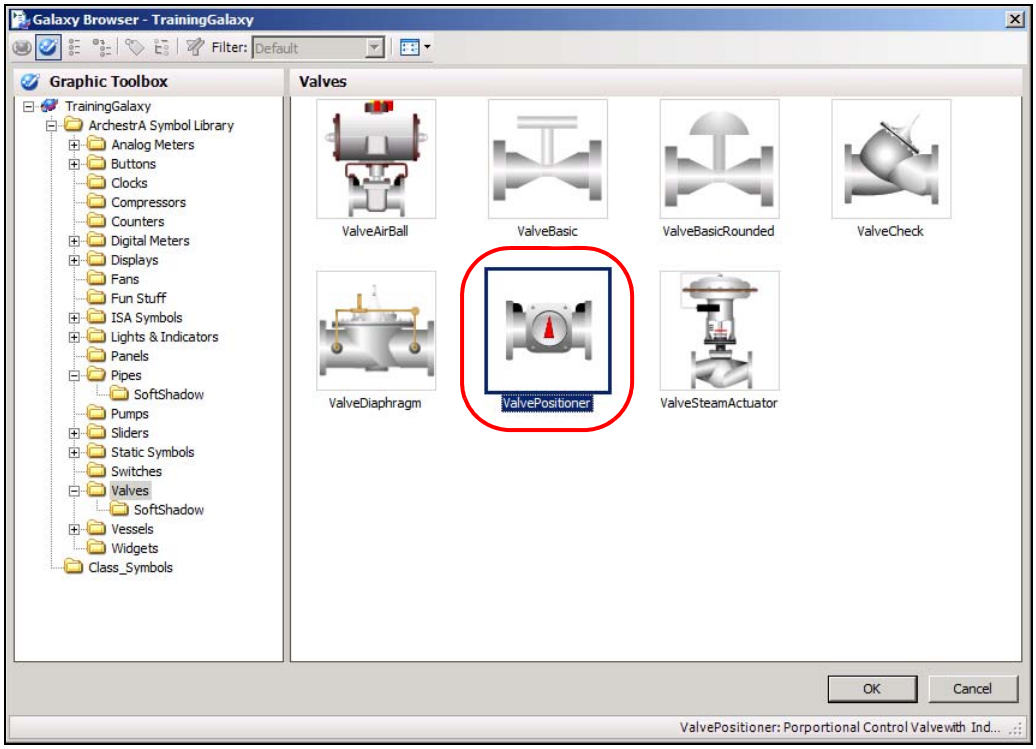
15. In **Properties** pane, rename the element **Pipe**.

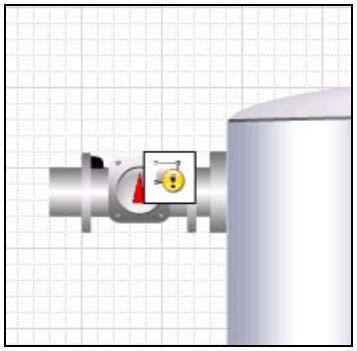
16. Embed **Pipes\PipeFlange** to the right of the pipe.



17. Name the element **Flange**.

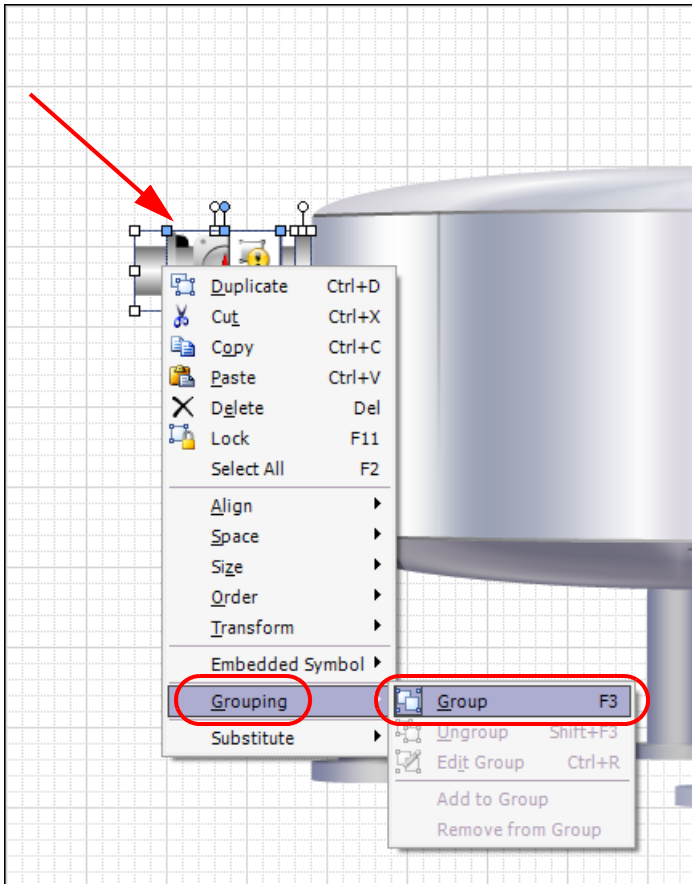
18. Embed **Valves\ValvePositioner** on the top of the pipe.





19. Rename the element **Valve**.

20. With **Valve** already selected, hold the **Shift** key and click **Pipe** and **Flange** to select all the three elements, then right-click and select **Grouping | Group**.

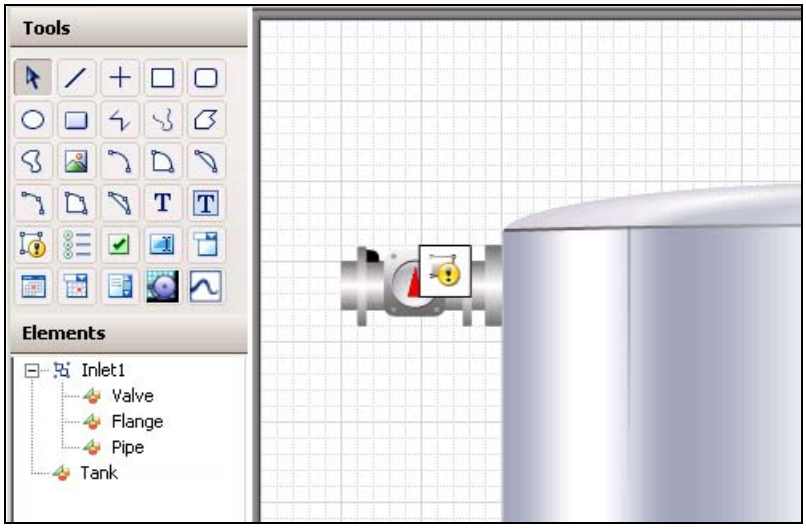


21. Name the new group **Inlet1**.

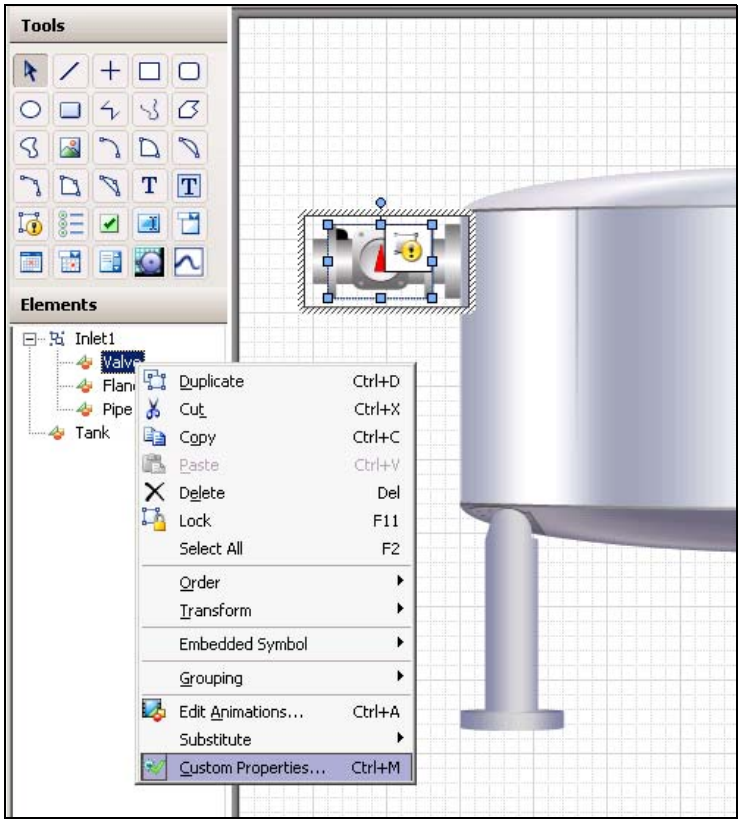
As a best practice, configure the custom properties of embedded symbols before duplicating them to save significant development time. In a later lab, you will spend time creating and using custom properties. For now, you will control the visibility of some unused properties.

22. In **Elements** pane, click **+** to expand **Inlet1**.

    The list of three embedded symbols that comprise the **Inlet1** group; **Valve**, **Flange**, and **Pipe** are listed in order of most recently created.

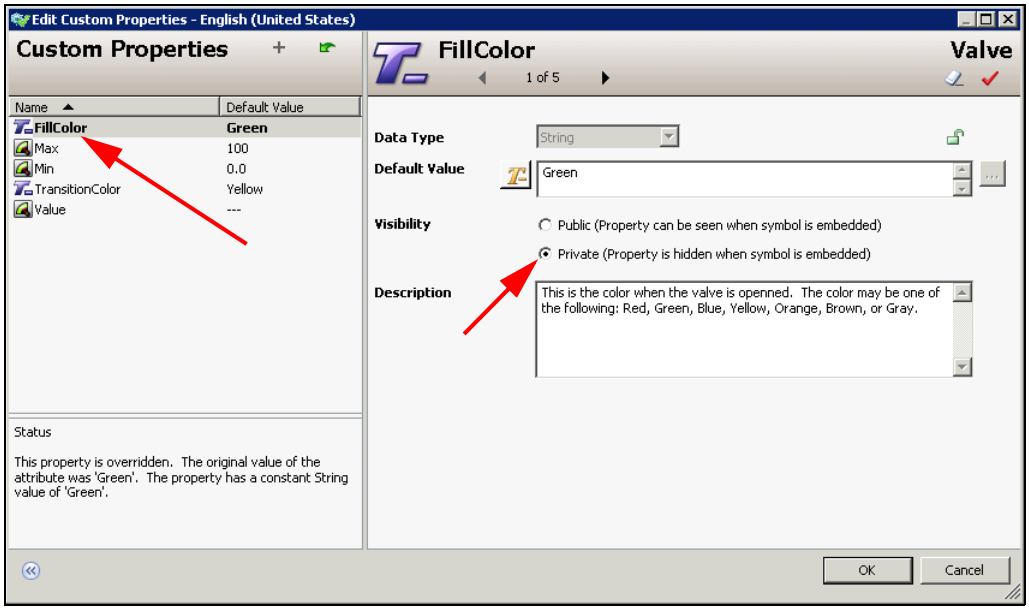


23. Right-click **Valve** and select **Custom Properties**.



    The **Edit Custom Properties** dialog box appears.

24. Select **Fill Color**, then set the **Visibility** to **Private**.



25. Select **Max**, then set the **Visibility** to **Private**.
26. Select **Min**, then set the **Visibility** to **Private**.
27. Select **TransitionColor**, then set the **Visibility** to **Private**.

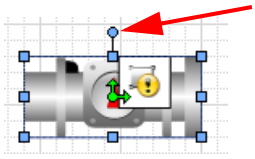
Note: Later in this lab you will link an InTouch **Tagname** to the **Value** custom property of the **Valve**, so you will leave the **Value** property marked as **Public** (the default).

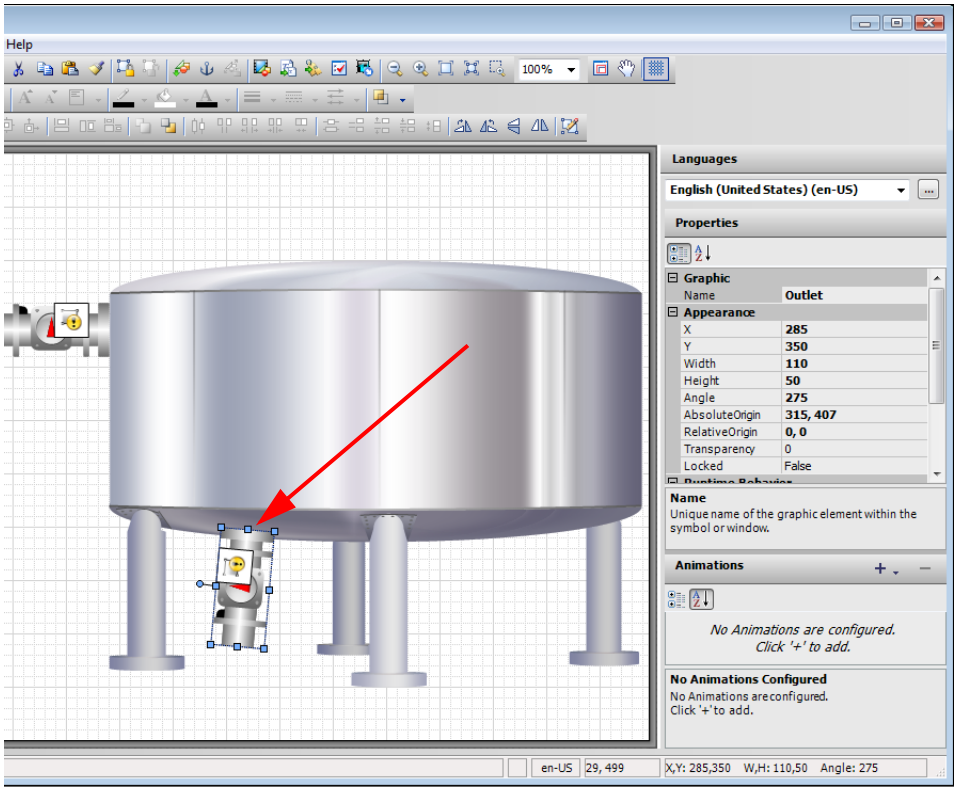
---

28. Click **OK** to close the **Edit Custom Properties** dialog box.
29. Open the **Edit Custom Properties** dialog box for each of the following and set the **Visibility** to **Private**:

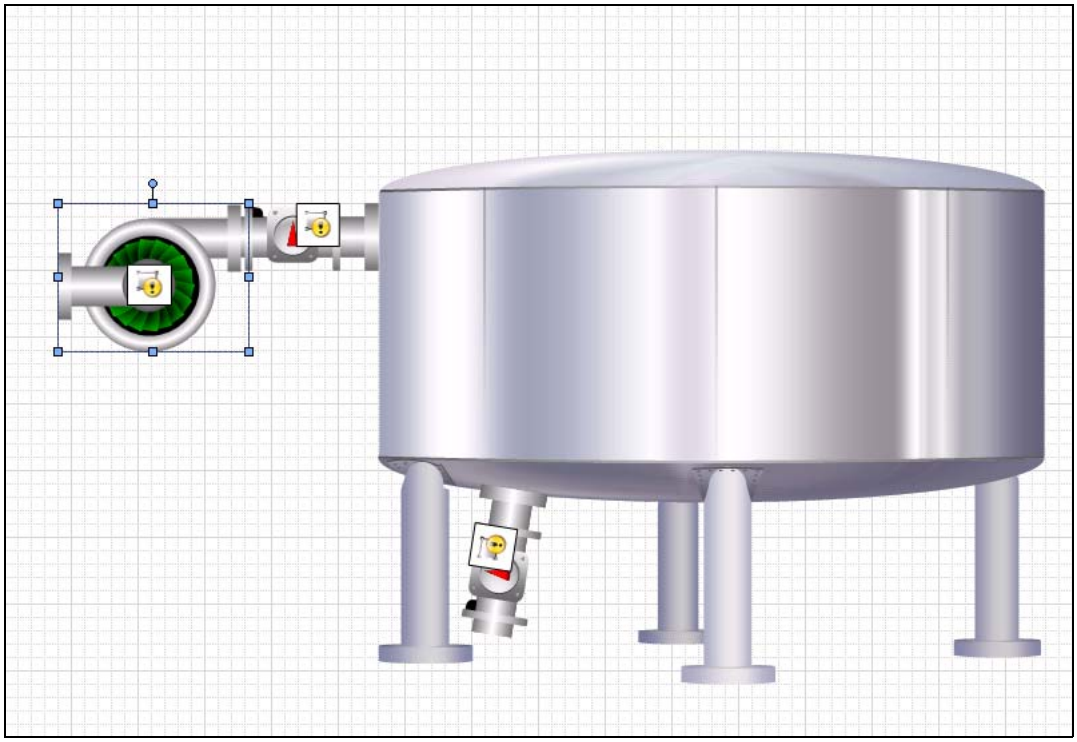
Element	Custom Property to mark Private
Flange	• FillColor • Value
Pipe	• FillColor • Value

30. In **Elements** pane, right-click **Inlet1** and select **Duplicate**, then rename the duplicate group **Outlet**.
31. Move **Outlet** to the bottom of the tank.
32. Click and drag the rotation handle counter-clockwise to align the **Flange** with the bottom of the tank.





33. Embed **Pumps\PumpCentrifugal** at the left edge of **Inlet1**.



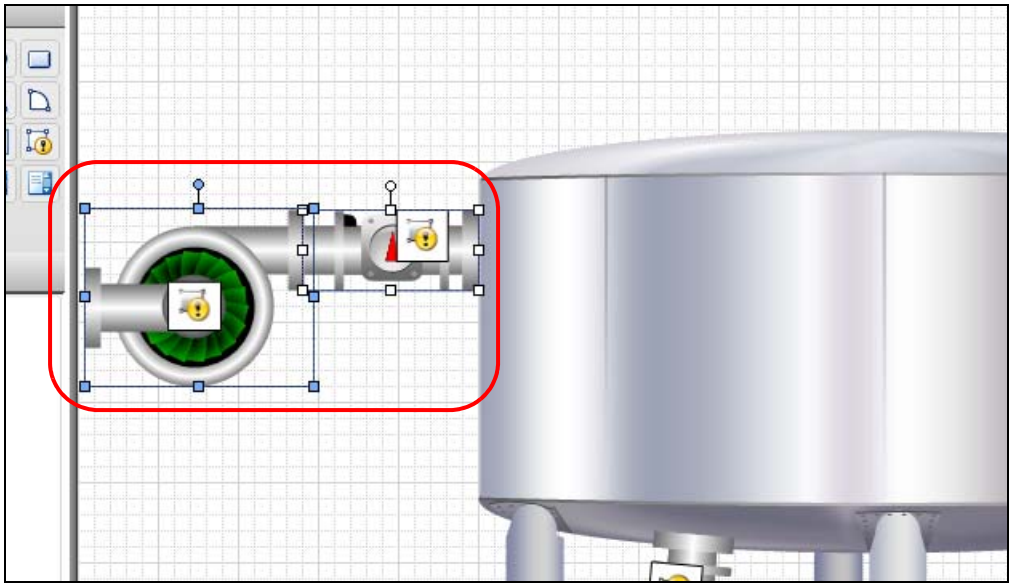
34. Rename the element **Pump**.
35. Right-click **Pump** and select **Custom Properties**.

    The **Edit Custom Properties** dialog box appears.
36. Set the **Visibility** to **Private** for the following:

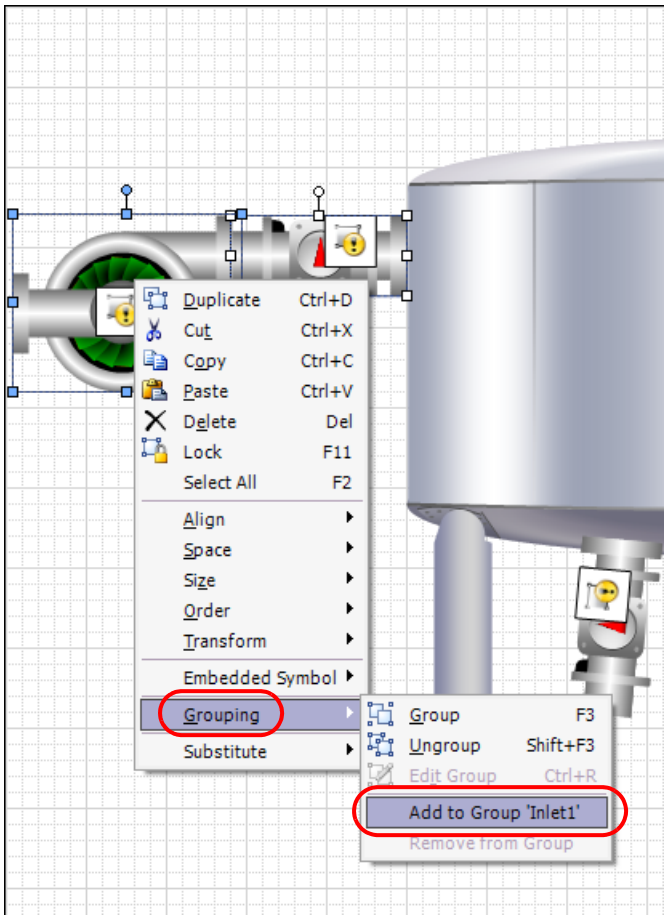
Element	Custom Property to mark Private
Pump	• BladeColorOff • BladeColorOn • FillColor • InletActive

37. Click **OK**.

38. On the canvas, click the **Inlet1** group, then press the **Shift** key and click **Pump** to add it to your selection.

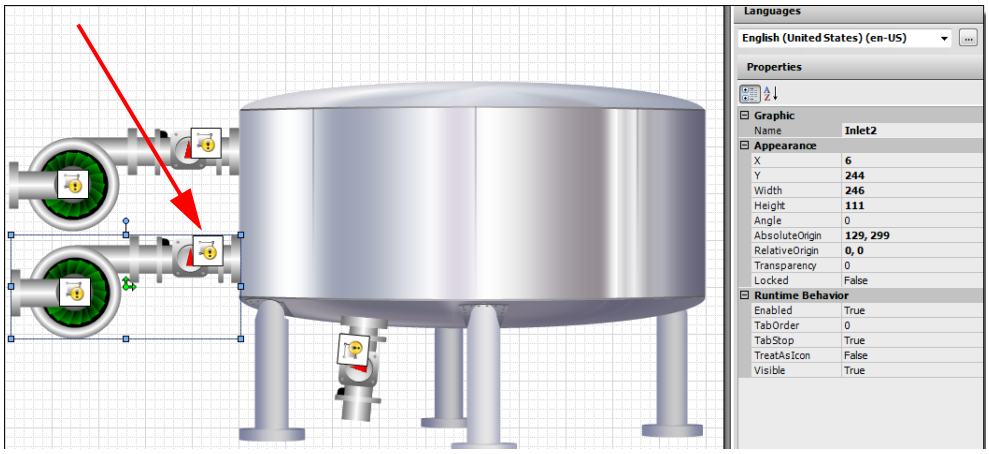


39. Right-click and select **Grouping | Add to Group 'Inlet1'**.



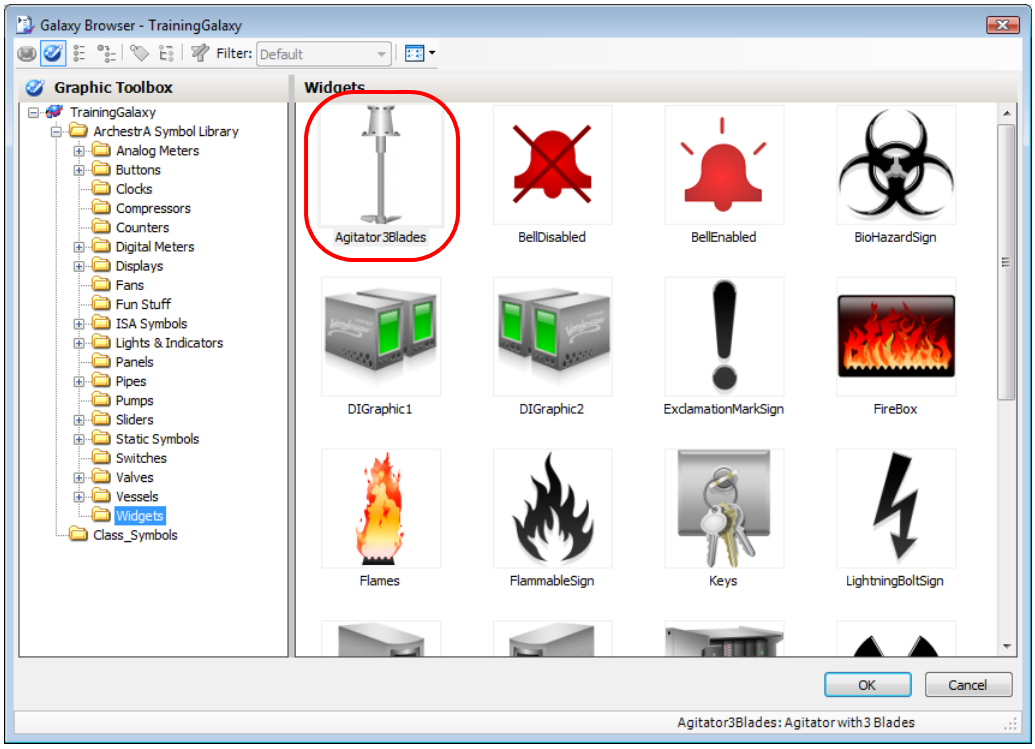
The **Pump** is now part of the group.

40. Duplicate **Inlet1** and name the new grouping **Inlet2**.
41. Move **Inlet2** to the lower-left of the tank.



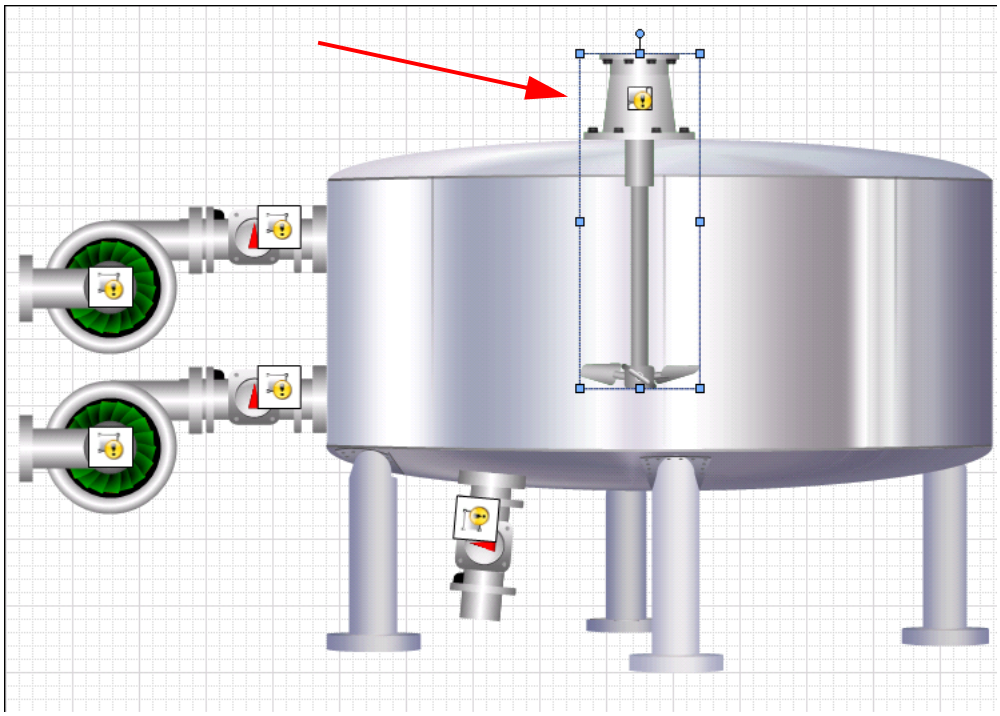
Next, you will add an agitator to your symbol.

42. Embed **Widgets\Agitator3Blades** on top of the tank.



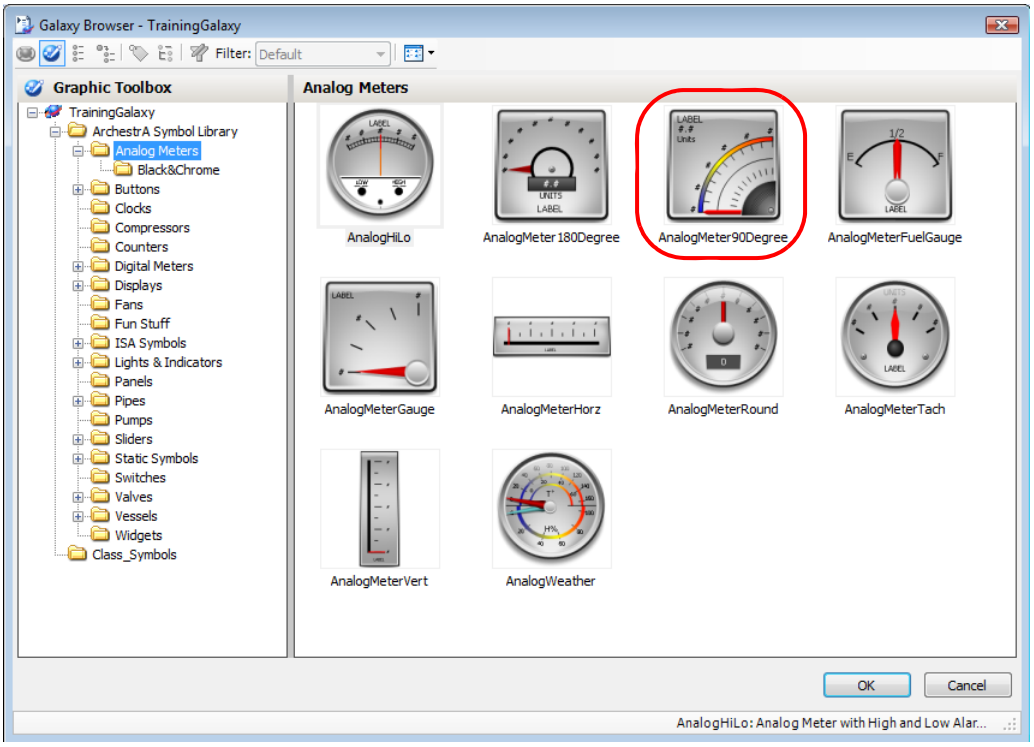
43. With the agitator selected, hold the **Shift** key down to maintain the agitator's aspect ratio, then click and drag the corner handle to resize the agitator.

44. Position the agitator motor at the top center of the tank.

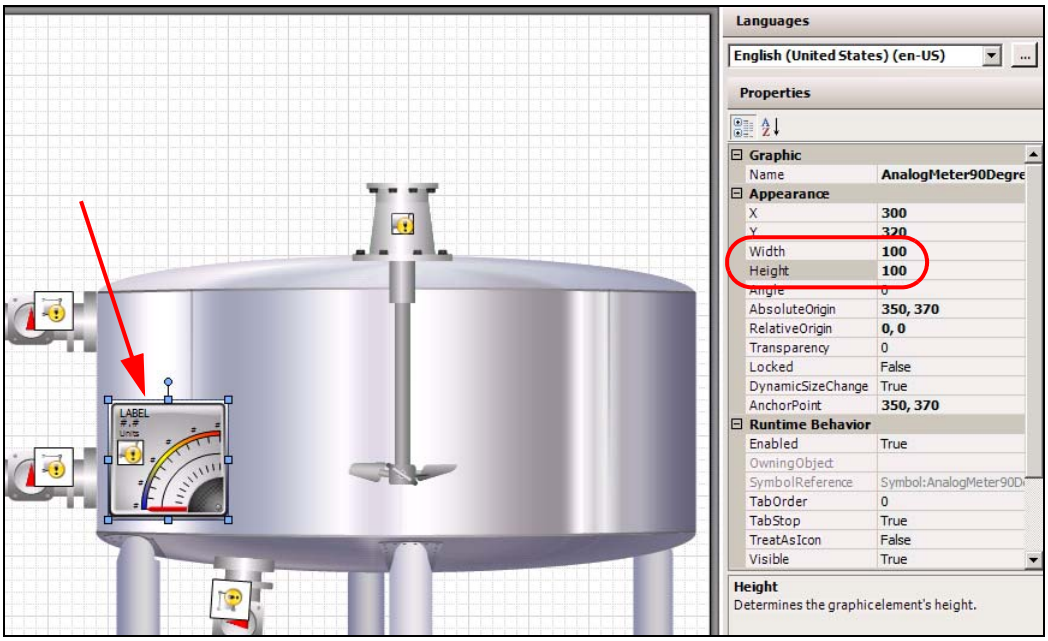


45. Rename the element **Agitator**.

46. Embed **Analog Meters\AnalogMeter90Degree** on the **Tank**.



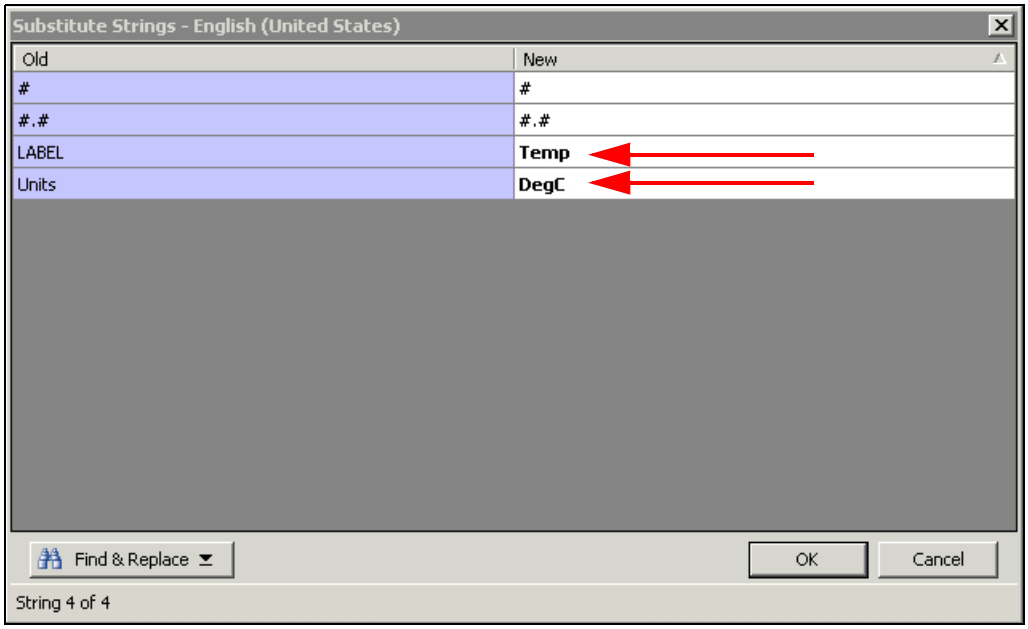
47. Resize the meter to a **Height** and **Width** of **100**.



48. Rename the element **Temperature**.

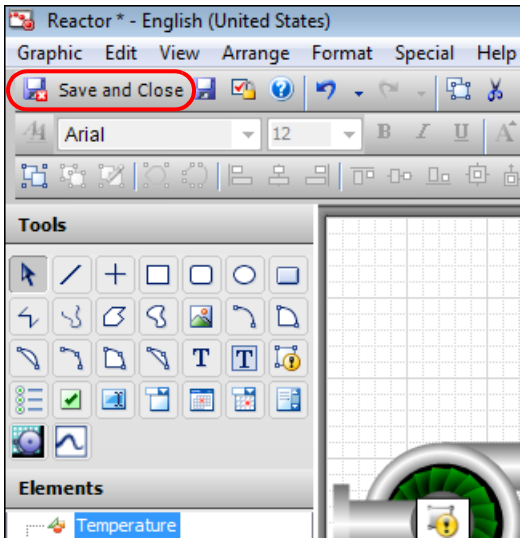
49. Right-click the **Temperature** element and select **Substitute | Substitute Strings** to change the labels on the embedded graphic as follows:

Old	New
LABEL	Temp
Units	DegC

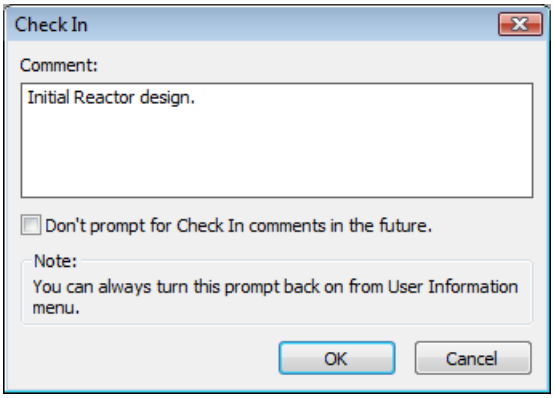


50. Click **OK**.

51. Click **Save and Close** to check in the symbol.



52. In the **Comments** field, enter **Initial Reactor design**.



53. Click **OK**.

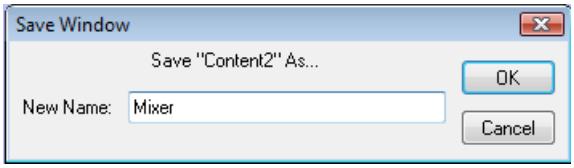
    The symbol is checked into the **Galaxy** database.

Create the Mixer Window

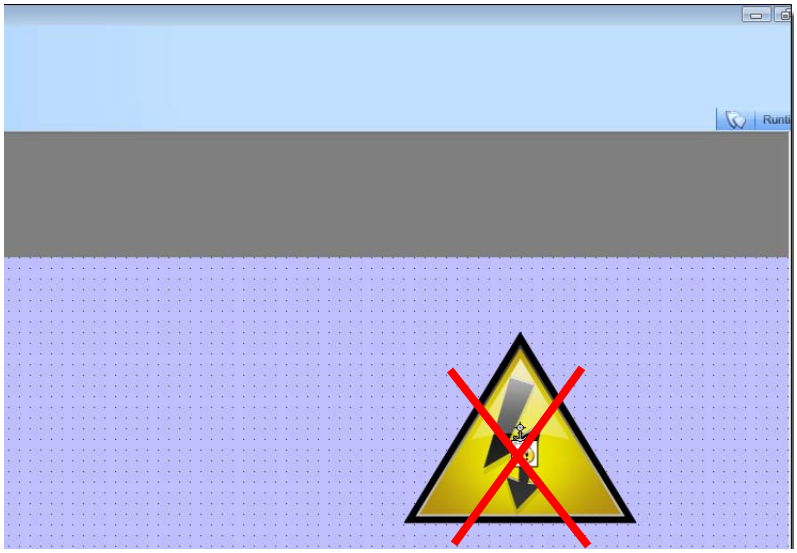
Next, you will create a window for the Reactor symbol.

54. Return to WindowMaker and open **Content2**.
55. In **Windows** pane, right-click **Content2** and select **Save As**.

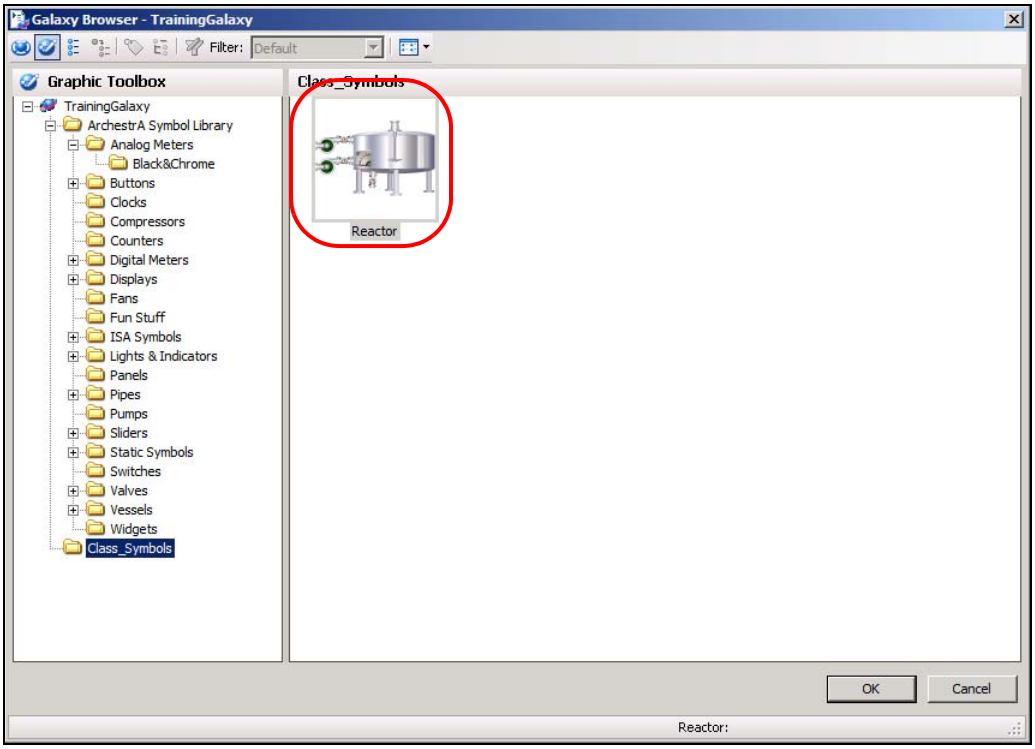
    The **Save Window** dialog box appears.
56. In the **New Name** field, enter **Mixer**.



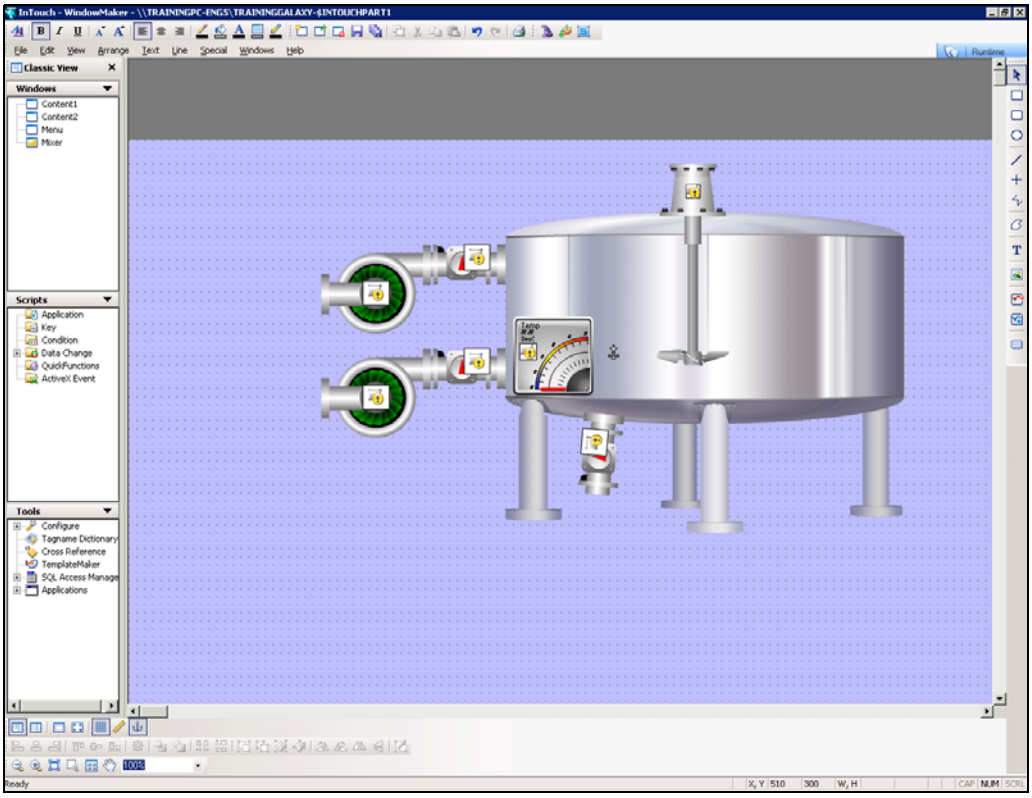
57. Click **OK**.
58. Delete the element from the new **Mixer** window.



59. Click **Embed ArchestrA Graphic**, click **Class_Symbols**, then double-click **Reactor**.



60. Place the **Reactor** symbol on the **Mixer** window.



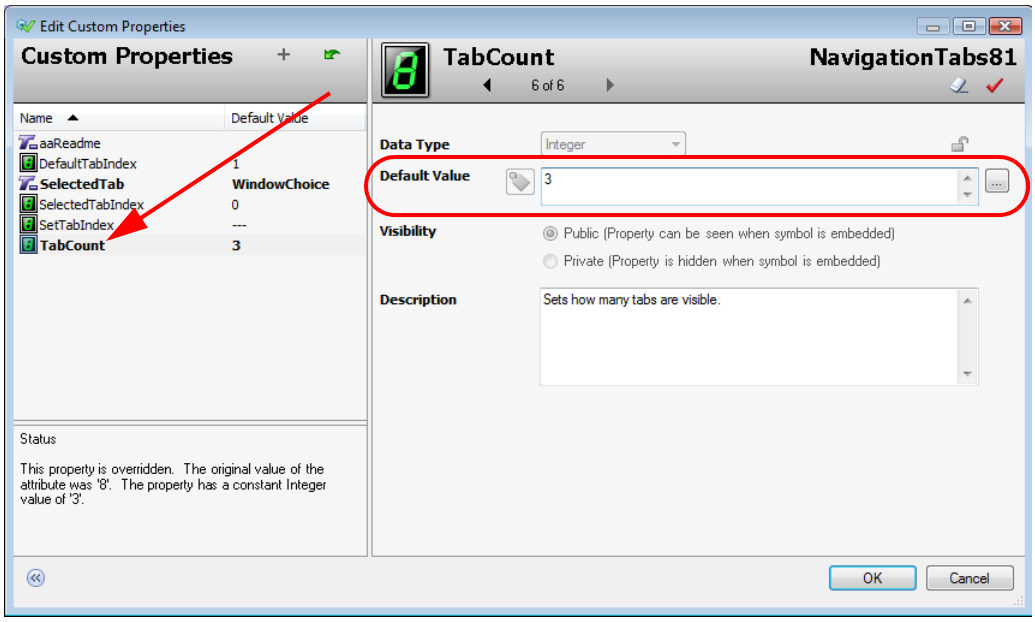
Add Navigation for the Mixer Window

Next, you will add navigation for the Mixer window.

61. Open the **Menu** window, then double-click the **NavigationTabs8** graphic.

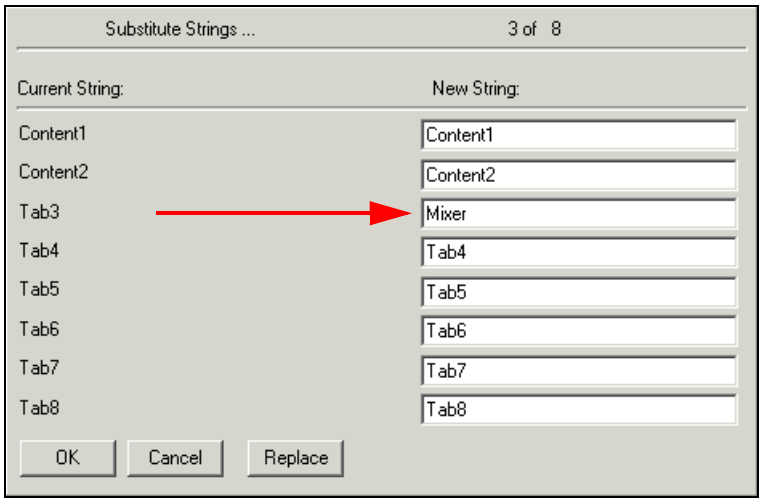
    The **Edit Custom Properties** dialog box appears.

62. Click **TabCount** and set property to **3**.



63. Click **OK**.

64. Right-click **NavigationTabs8** and select **Substitute | Substitute Strings**, then change **Tab3** to **Mixer**.

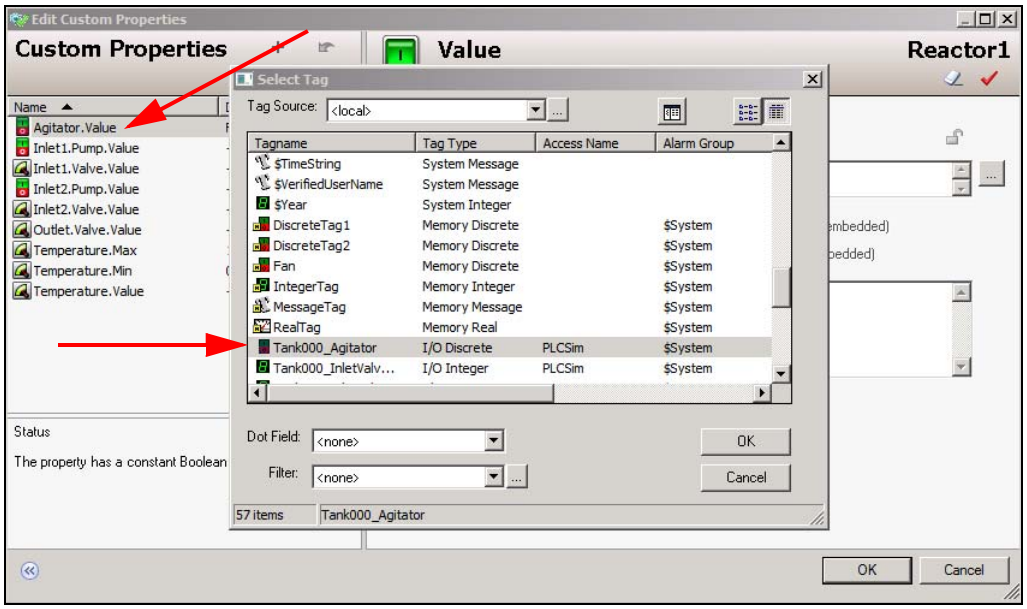


65. Click **OK**.

Configure and Link Tags

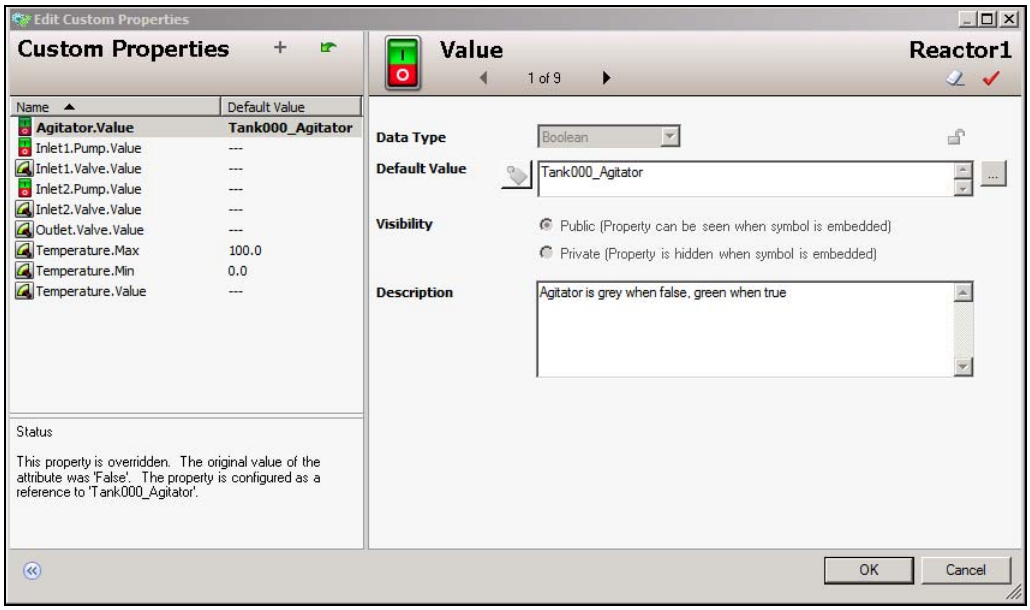
Next, configure the symbol and link custom properties to tags.

66. Double-click the **Reactor** symbol to open the **Edit Custom Properties** dialog box.
67. With the **Agitator.Value** selected, double-click the **Default Value** field to open the **Select Tag** dialog box.
68. Click **Tank000_Agitator**.



69. Click **OK**.

Tank000_Agitator is assigned to the **Agitator.Value** custom property.



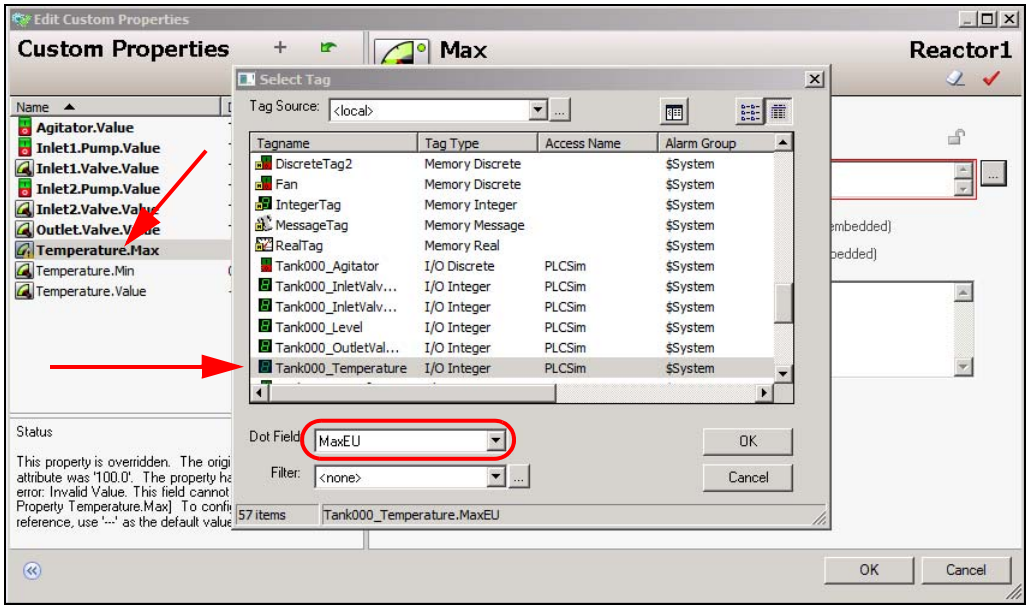
70. Assign **Default Value** for the following **Custom Properties**:

Custom Properties	Default Value
Inlet1.Pump.Value	Tank000_TransferPump1
Inlet1.Valve.Value	Tank000_InletValve1Pos
Inlet2.Pump.Value	Tank000_TransferPump2
Inlet2.Valve.Value	Tank000_InletValve2Pos
Outlet.Valve.Value	Tank000_OutletValvePos

For the Temperature.Max and Temperature.Min custom properties, you need to assign Dot Field attributes MaxEU and MinEU of the Tank000_Temperature tag as follows:

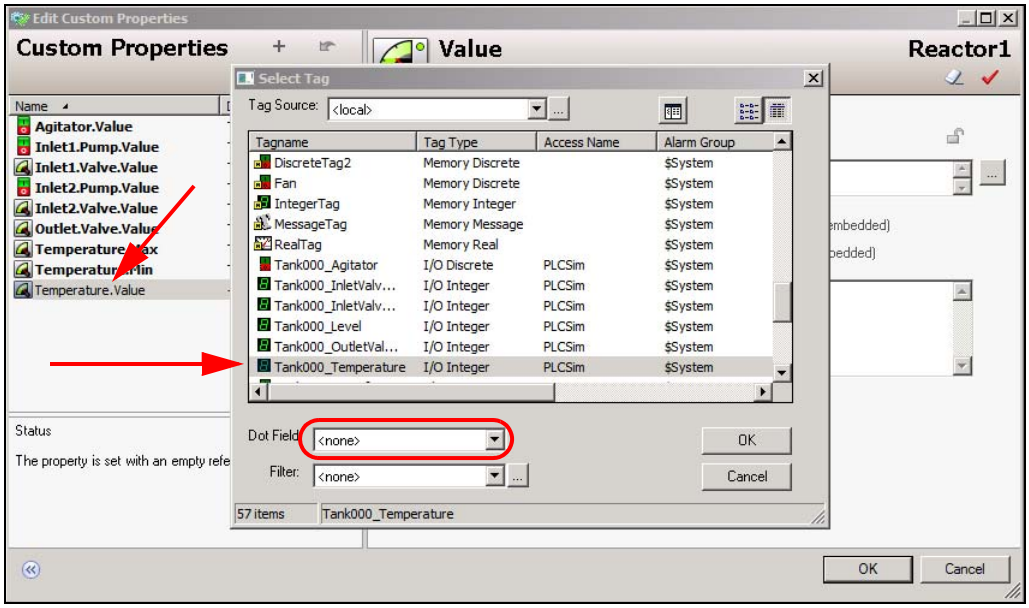
71. Click **Temperature.Max**, then for the **Default Value** select **Tank000_Temperature**, leaving the **Select Tag** dialog box open.

72. In the **Dot Field** drop-down, select **MaxEU**.



73. Click **OK**.

74. Repeat the previous steps to first assign the tagname Dot Field **Tank000_Temperature.MinEU** to the custom property **Temperature.Min**.
75. Click **Temperature.Value**, then for the **Default Value** select **Tank000_Temperature**, leaving the **Select Tag** dialog box open.
76. In the **Dot Field** drop-down, select **<none>**.



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

Test What You Have Built

79. Switch to **Runtime**.
80. In the **Mixer** tab, verify your **Reactor** window functions as follows:

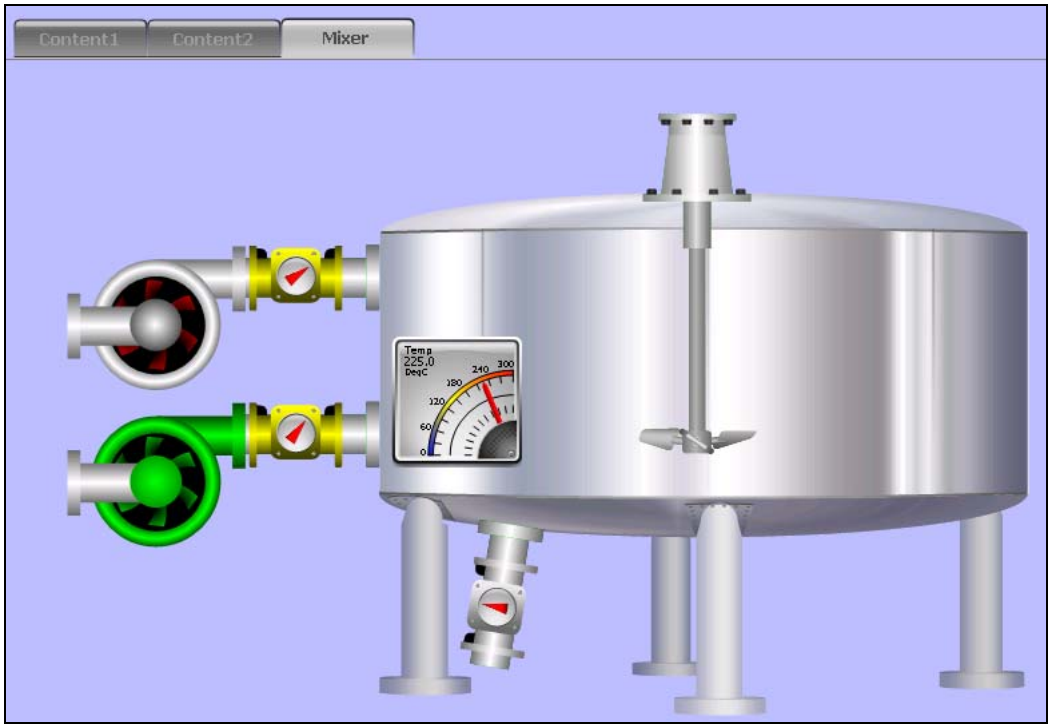
Note: When you start WindowViewer you may see the process at an advanced stage of the sequence.

- The process begins with **InletValve1** opening and **TransferPump1** running.
- The **Temperature** begins to increase.
- After about 20 seconds, **InletValve1** closes and **TransferPump1** stops running.
- After about 10 seconds, **InletValve2** opens and **TransferPump2** runs.
- The **Temperature** continues to increase.
- **InletValve2** closes and **TransferPump2** stops running.
- **Agitator1** then runs for 15 seconds.

Note: The agitator animates by turning green when running, it does not spin.

- **OutletValve** then opens and the **Temperature** decreases.
- After about 30 seconds the **OutletValve** closes.
- The PLC process runs continuously, repeating the sequence every 1.5 minutes.

81. Observe the animation for one entire sequence.



Section 2 – Using the ArchestrA Symbol Editor

This section discusses symbol duplication and editing. It also describes the ArchestrA symbol editor features.

Overview

The ArchestrA Symbol Editor is used to create symbols that represent production processes. You can create a new symbol using graphic elements, or select an existing graphic from the ArchestrA Symbol Library. You can also place symbols directly into Managed InTouch applications.

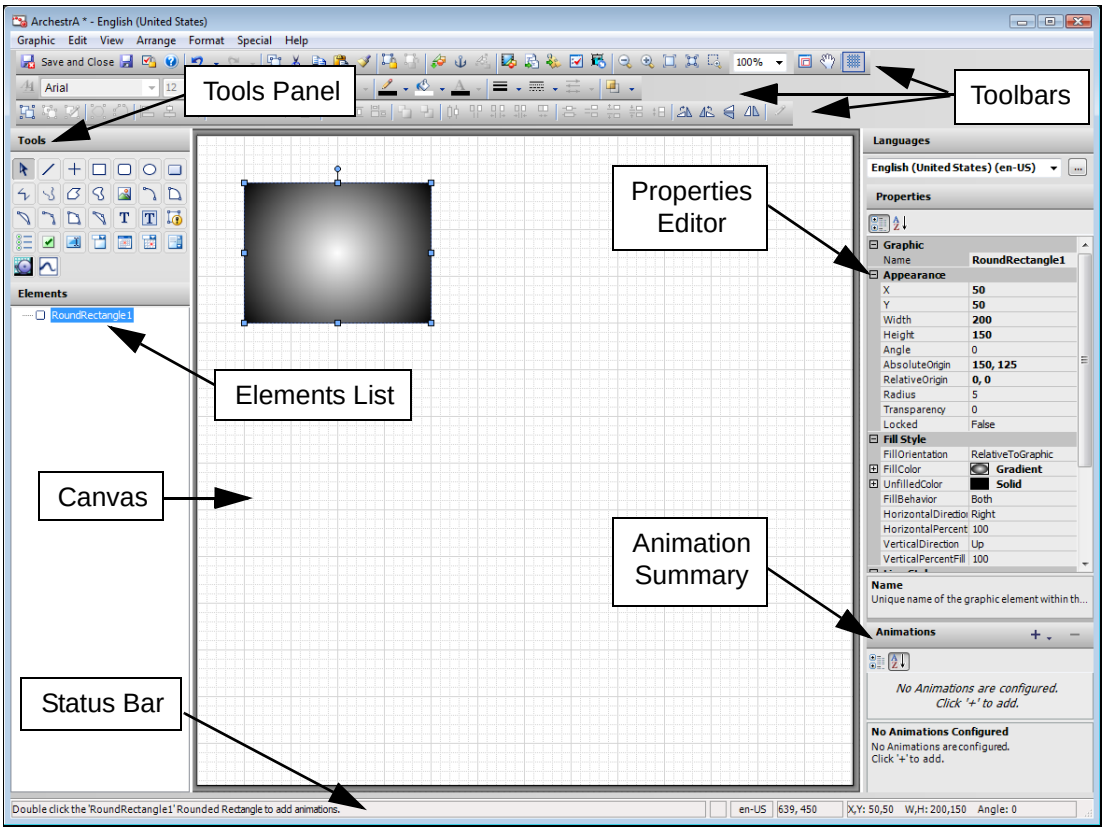
Open a symbol to start the ArchestrA Symbol Editor.

It is important to use caution when editing an existing symbol. To leave existing symbol functionality intact, duplicate the symbol. Changes made in the duplicate symbol do not impact the original symbol.

You can control the appearance of an element, a group of elements, or multiple elements with functions on the toolbar and the Properties panel. You can layer graphic elements in front or back of other objects, and combine other graphical elements to give the illusion of depth or highlight to an object.

The ArchestrA Symbol Editor

The InTouch HMI is coupled with the ArchestrA IDE. The ArchestrA IDE includes the ArchestrA Symbol Editor, used to create symbols that represent production processes. These symbols can be placed directly into the Managed InTouch applications.



Working with Graphic Elements

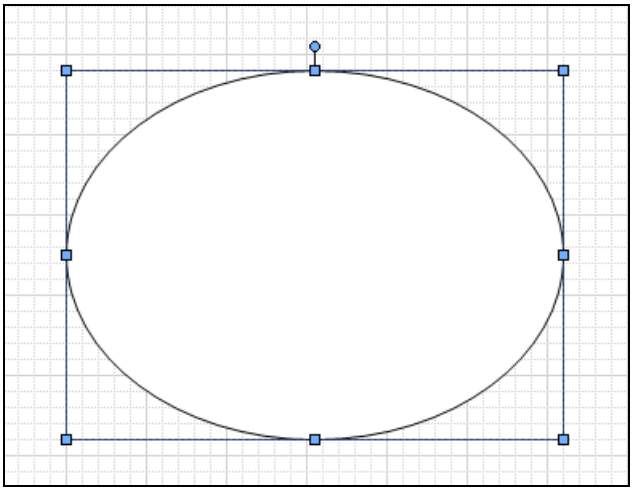
Graphic elements are basic shapes and controls you can use to create a symbol to your specifications. You can:

- Draw an element using tools from the **Tools** panel, and then configuring its properties
- Select one or more elements on the canvas with the mouse or from the **Element** list
- Edit certain elements in a special way called inline editing
- Copy, cut, paste, and duplicate elements
- Move and resize elements
- Align elements to each other
- Change the z-order of elements to change which elements appear on top when they overlap
- Rotate elements.
- Change the origin of elements to specify around which point the elements are rotated
- Flip elements on their horizontal or vertical axis
- Lock elements to restrict changes
- Undo and redo changes
- Group elements to bind them together
- Create a path graphic from multiple open line elements

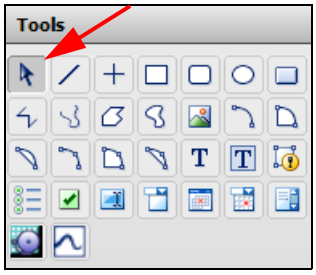
Drawing and Dragging Elements

You can use the drawing tools and your mouse to create new shapes or embed symbols from the **ArchestrA Graphics Library** to become part of your new symbol.

You can create elements such as lines, curves, circles, squares, and so on. You can combine these elements to create complex drawings of all the equipment in your manufacturing environment. After you draw an element, you can modify its properties.



After you draw an element, the pointer tool is selected again by default. To draw multiple elements of the same type, double-click the element in the **Tools** panel. It remains selected after you draw your first element of that type.



You can press the **Esc** key to return to the pointer tool again.

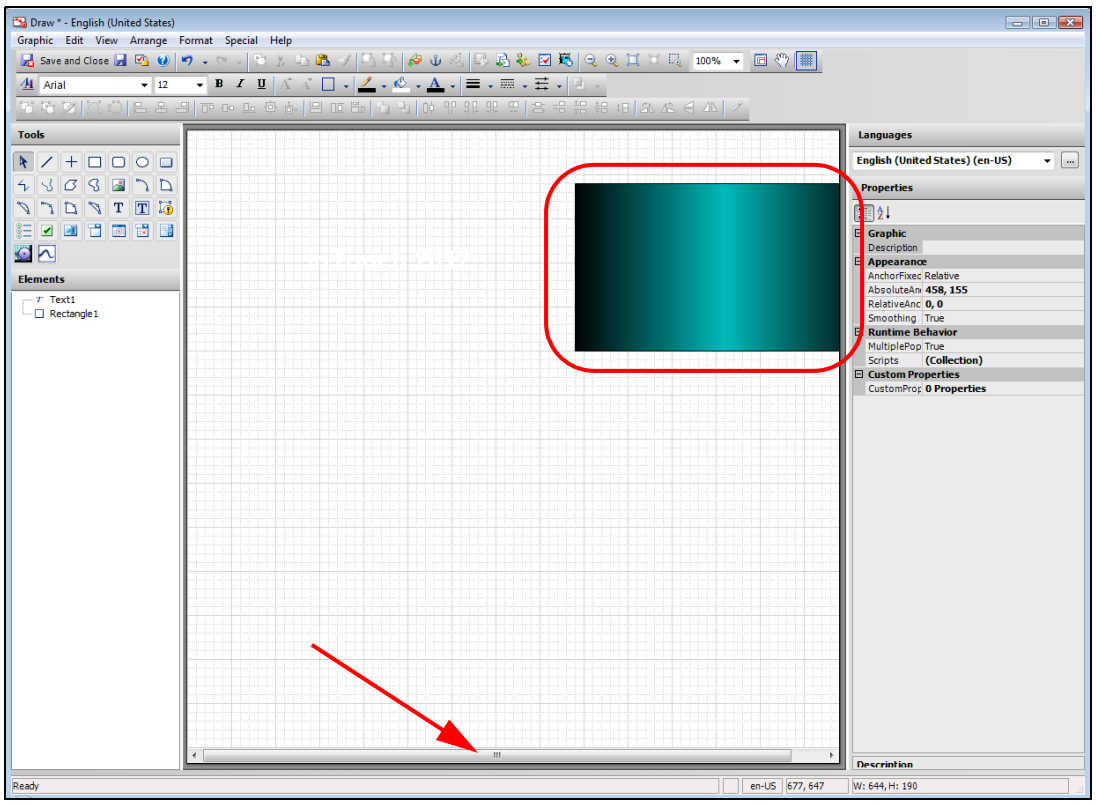
Please see the online help for a detailed description of each element in the **Tools** panel.

Drawing Objects

You can draw simple elements using rectangles, ellipses, and lines on the canvas. You can also use polylines, polygons, 2- and 3-point arcs, and 2-point pies and point chords, and combine elements to create sophisticated objects. If you are drawing a closed element, it automatically closes when you are done drawing.

You can change the shape of these elements anytime by editing their start and sweep angles.

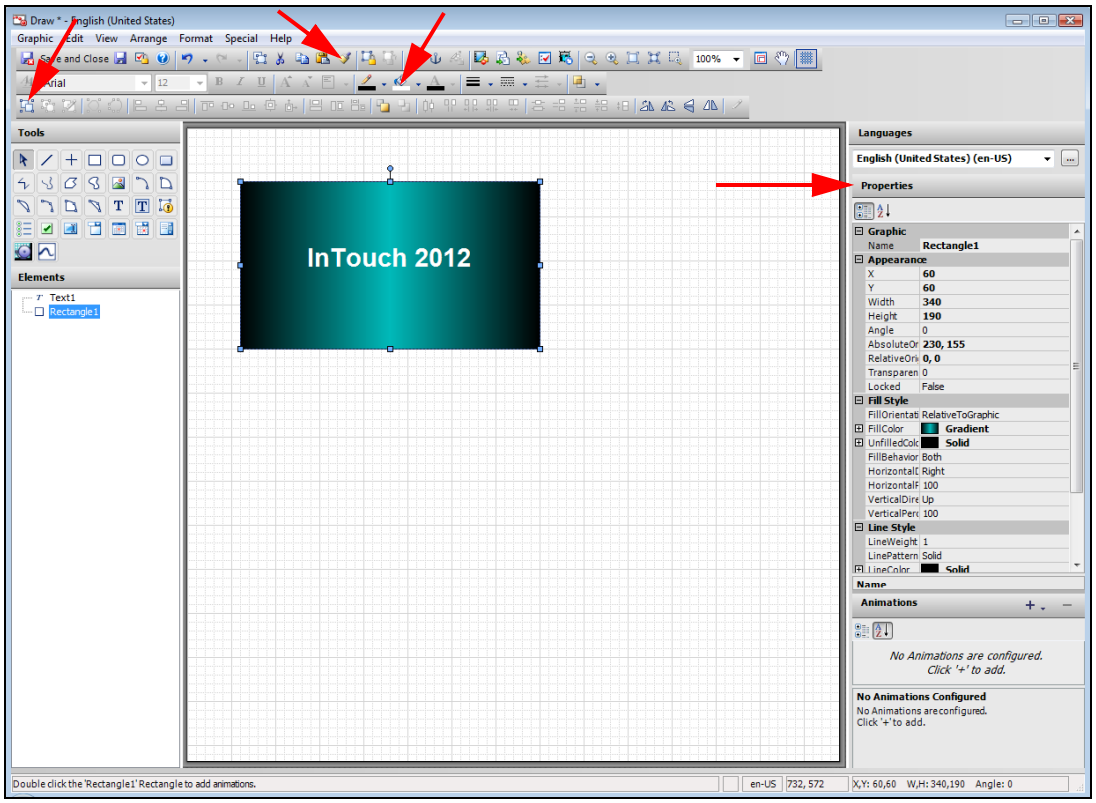
If you draw or drag an element outside of the visible canvas area to the right or bottom, horizontal and/or vertical scroll bars appear but the visible area does not follow the mouse.



You can later use the scroll bars to scroll the canvas and see the element you drew or moved.

Beyond using the basic drawing tools, there are strategies that you can use to create more sophisticated graphics. This includes:

- Combining multiple shapes to create more complex graphical elements
- Using the Properties panel to numerically modify the appearance, size, angle, or location of a graphic element
- Creating complex fill colors by using multiple colors, modified distribution shape, gradients, and transparency
- Layering objects in front or back of other objects combined with transparency and gradient fills to give the illusion of depth or highlight
- Using the Format Painter tool to transfer complex appearance properties from one element to another



Adjusting the Z-Order of Elements

The z-order of elements specifies which element appears on top of other elements when the elements overlap on the canvas. The z-order also determines how the elements of a path graphic connect. When you place new elements on the canvas, they are placed at the top and can cover all other elements.

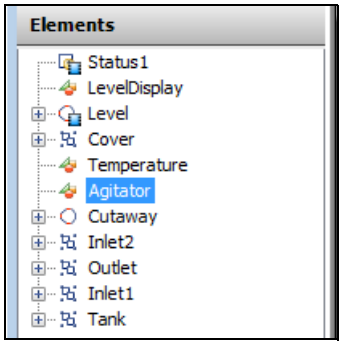
However, you might want to bring certain elements forward so that they are always visible or overlap certain other elements. Or you may want to use a large background element behind all other elements. You can:

- Bring one or more elements to the very front
- Send one or more elements to the very back
- Bring one or more elements one level forward
- Send one or more elements one level backward

Select an object and use the **Bring to Front** and **Send to Back** buttons on the toolbar.



You can also use the **Elements** list to see or change the z-order of the elements. Click an element you want to move and use the **F8** and **F9** keys to move elements up and down the list. The z-order changes according to element positioning in the list.



- **F8** moves a selected item down one level
- **Shift-F8** moves a selected item up one level
- **F9** moves a selected item to the bottom of the list
- **Shift-F9** moves a selected item to the top of the list

Rotating Elements

You can rotate elements to any orientation (0-359 degrees):

- Graphically with the rotation handle
- Numerically by typing the orientation angle in the **Properties** editor
- Using the 90-degree rotation tools in a clockwise or counter-clockwise direction



The element is rotated around its point of origin. By default, the point of origin is in the center of the element. You can move the point of origin to any other location, even outside of the object itself. Use the rotate handle to rotate an object. To move the point of origin, display it by using the **Properties** editor and selecting one of the following properties under the **Appearance** category:

- **Angle**
- **AbsoluteOrigin**
- **RelativeOrigin.**

You can then click and drag the point of origin, or enter the new location in the AbsoluteOrigin and/or RelativeOrigin properties.

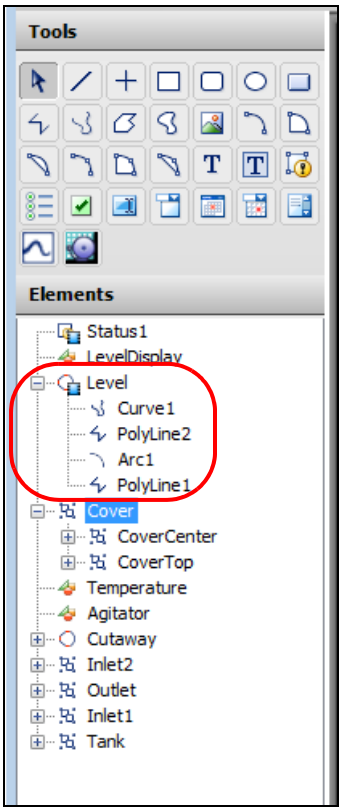
Create a Group

Grouping allows you to relate elements together as a unit. Groups can contain elements and other groups. Groups are shown in the **Elements** list with a default name, such as Group1. Groups can be renamed just as you rename elements.

After selecting the elements you wish to group, click the **Group** button to cluster them.



The group is displayed as branches in the element hierarchy.



Editing Grouped Components

You can work with the elements of a group as one set of elements or, by selecting the elements in the **Elements** list you can work with the individual elements in the group without having to ungroup them. This is referred to as inline editing.

An advantage of inline editing is that you can select an individual element graphically without having to know its element name.

In the **Elements** list, expand the group that contains the element that you want to edit. When the element appears, select in the group and the group is outlined with a diagonal pattern.

Edit the element with the **Properties** editor, by mouse, or by menu according to your requirements.

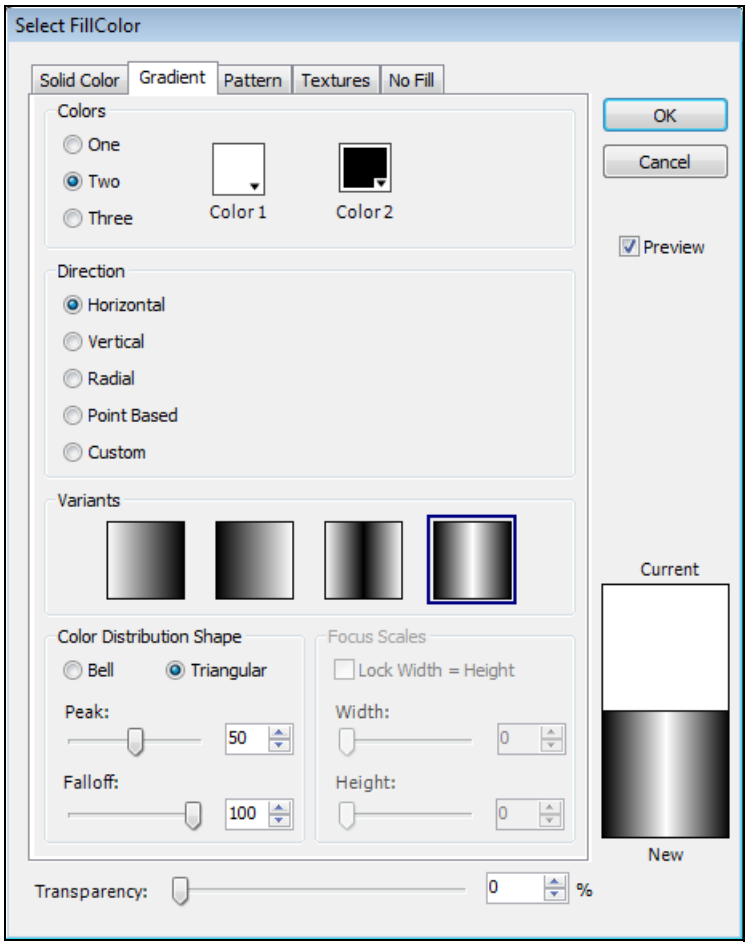
Fills and Gradients

Use the **Fill Color** button on the toolbar to bring up the **Gradient** options and select a gradient color similar to the one shown below.

You can configure gradients by the:

- Number of **Colors** - 1, 2 or 3.
- **Direction** - Horizontal, Vertical, Radial, Point Based, or Custom.
- **Variant** - depending on your selection for the number of colors and direction.
- **Color Distribution Shape** - Bell or Triangular with options to configure the Peaks and Falloff.
- **Focus Scales** - Width and Height.
- **Transparency** - 0 to 100%.

You set a gradient on the **Gradient** tab in the **Select FillColor** dialog box.



Using a **Point Based** gradient with a **Height** of **100** and a **Bell Color** distribution allows the gradient shading to more accurately follow the shape of a complex object.

Object Transparency

An image can be defined with a transparent color so background objects or graphics behind it will be visible through the transparent areas. By defining a transparent color, the graphic's background or any objects behind the image will show through in the areas where the transparent color is used.

To make an image transparent, change the **Fill Color** for the container to **No Fill**, change the **Line Color** to **No Line**. Right-click the image and select the **Select Image Transparent Color** option. Use the dropper-shaped **Color Picker** to hover over the color you wish to make transparent and select it by clicking. The image changes to reflect the transparent color selection.

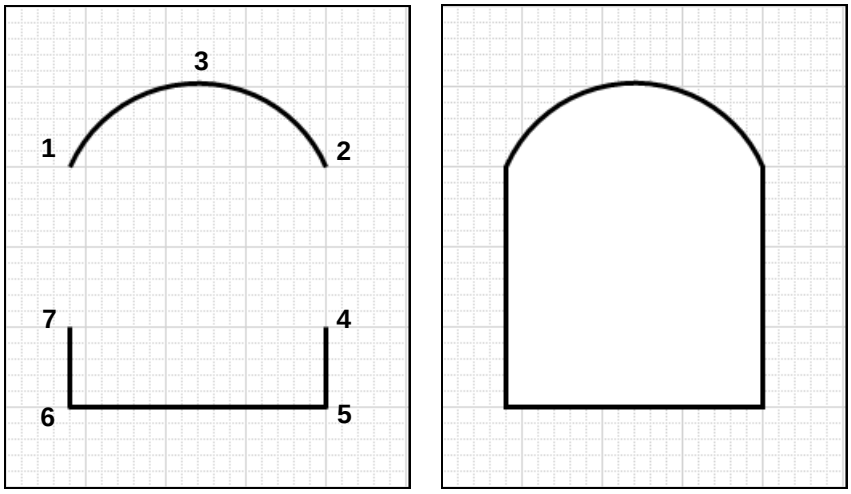
You can also adjust the **Transparency** slider on the **Select FillColor** dialog box to adjust element transparency.

Path Graphics

Path graphics are elements that combine selected open elements, such as lines, H/V lines, polylines, curves, and arcs to a single closed graphic element.

The path graphic depends upon:

- The order in which you drew its elements. Each element is linked to the next element by z-order. The z-order of the elements is the order shown in the **Element Browser** list.
- The direction in which you drew its elements. The end point of one element is connected to the start point of the next element.



The properties of the elements contained within a path graphic are retained, so that when you break the path graphic, the elements of which it consists appear as they did before the path graphic is created.

The path graphic has the same properties as a rectangle, ellipse, or polyline. It loses these properties when you break the path.

- Use the **Path Combine** button to join any shape with two open points.
- Use the **Path Break** button to break a path graphic.

Placing Text

The text element has no border and no background fill. The text does not wrap. When you type the text, the size of the Text element expands. You can also drag the handles of the Text element to resize it.

Using the **Text** button, locate and size it, type a single line of text and do one of the following:

- Press **Enter** to close the text element and create a new line of text immediately below it.
  OR
- Click the canvas outside the text element to deselect the text element.

Drawing Text Boxes

You can draw text boxes on the canvas. Use the **Text Box** button on the toolbar to locate and size it. Upon completion, the text appears in edit mode, type the text and press **Enter** to continue.

- Text boxes can have borders and background fill
- You can also configure the text to wrap in the text box and create multi-line text boxes

Editing Text Objects

The font, font style, font size, justification, and rotation of any selected text object can be customized in the ArchestrA symbol editor by using the **Properties** dialog box.

Symbol Dynamic Size Propagation

You can control the way that size changes of the parent symbol are propagated to its child symbols, which are inserted ArchestrA symbols. It is important to consider the impact of graphical changes as they are propagated to your runtime environment. For example, a size change is:

- Resizing one of the elements in the parent symbol so that the symbol boundary changes
- Adding elements to, or removing elements from the parent symbol so that the symbol boundary changes

This feature is called dynamic size change and can be either enabled or disabled for each embedded symbol.

Note: For more information about dynamic size propagation, see *Creating and Managing ArchestrA Graphics User's Guide*.

Editing Element Properties

You can control the appearance of an element, a group of elements, or multiple elements with functions on the toolbar and/or properties in the **Properties** panel.

Often you can edit an element by changing the values of its properties instead of using the mouse to perform the same function. This is useful when you want very exact editing, such as when you want to resize an element to a specific width.

The Properties panel shows the properties common to all selected elements.

- Read-only properties appear in gray.
- Non-default values appear in bold.

Properties are organized in categories so you can find them more easily.

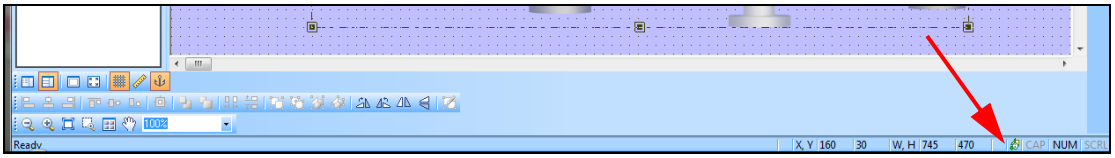
When you create a new symbol using the **Graphic Toolbox**, it appears in the list. You can then open the symbol for editing using one of the following methods:

- Double-click the symbol in the **Graphic Toolbox**
- Right-click the symbol and select **Open**

It is important to note that any changes to the parent ArchestrA symbol are propagated to all embedded copies of that symbol. This affects the ArchestrA symbols in WindowMaker. If WindowMaker is open and one of its windows contains an inserted ArchestrA symbol, any change made to the symbol is indicated with a refresh/update symbol icon that appears at the right end of the status bar.

There are three ways to update a symbol:

- Close and reopen the window
- Click the refresh icon at the bottom of the window
- Switch to Runtime to obtain the latest revisions



Duplicating a Symbol

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

Deleting a Symbol

- You can delete a symbol you no longer need. Deleting a symbol removes it from the ArchestrA Symbol Library. You can delete a symbol from the **Graphic Toolbox** by right-clicking the symbol and selecting **Delete**
- When you delete a symbol, you are shown where the symbol is used. This lets you understand the impact of deleting the symbol before you actually delete it
- You cannot delete symbols that someone else has open for editing or left checked out
- If you delete a symbol that is used in an InTouch window, it is not deleted from that window. However, the ArchestrA animation links no longer work and you cannot edit the symbol with the ArchestrA symbol editor

To delete a symbol or group of symbols, do one of the following:

- Open the **Graphic Toolbox**, right-click the symbol and select **Delete**.
- Select the element or group of elements, right-click and select **Delete**.

Lab 7 – Modifying Your ArchestrA Symbol

Introduction

In this lab, you will make custom modifications to the Reactor symbol you created in the previous lab.

Objectives

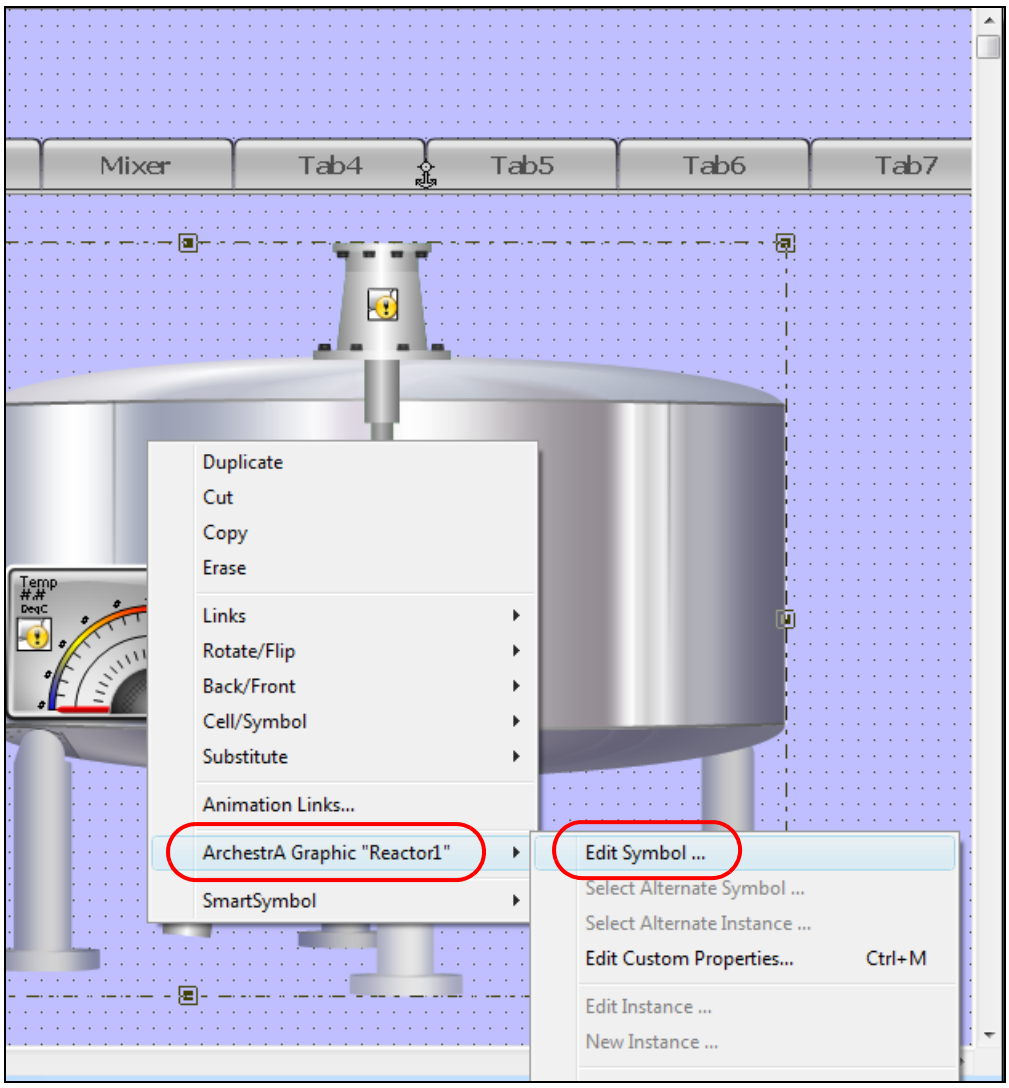
Upon completion of this lab, you will be able to:

- Convert an embedded symbol to a group for editing
- Use the Graphic tools to create custom graphics
- Refresh a symbol in WindowMaker to obtain the latest content

Create a Custom Symbol

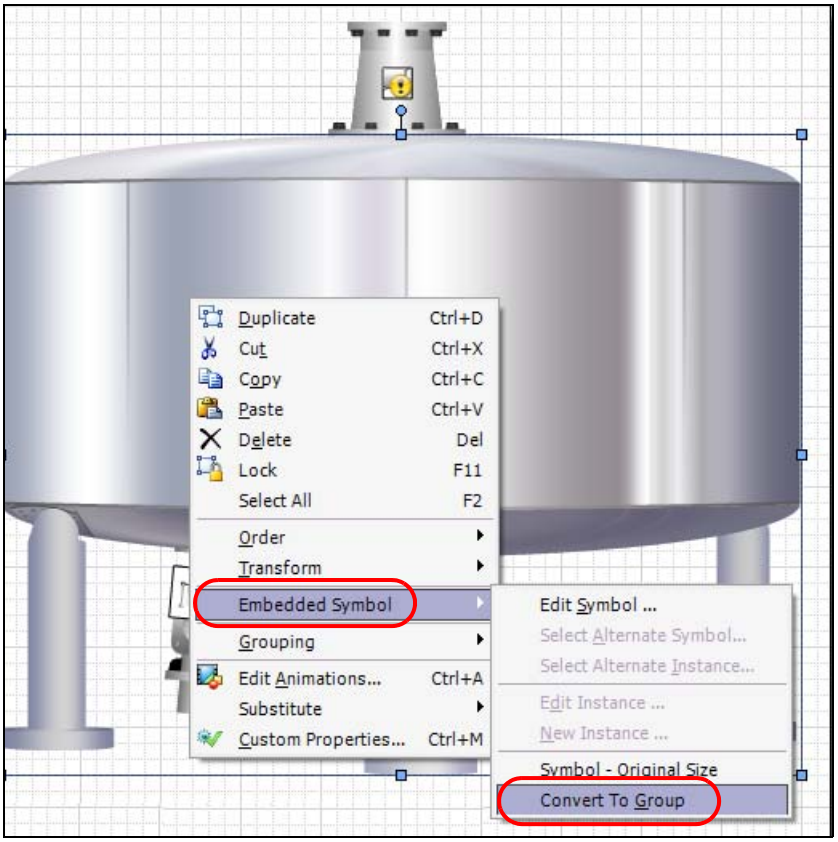
First you will duplicate the tank symbol to reuse some of its components as an overlay to hide the shaft of the Agitator.

1. Return to WindowMaker, **Mixer** window.
2. Right-click the **Reactor** symbol and select **ArchestrA Graphic "Reactor1" | Edit Symbol**.



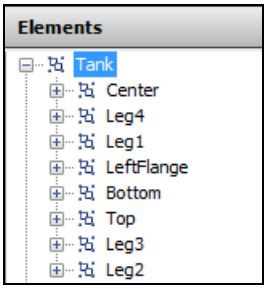
The **Reactor** symbol opens for editing in the **ArchestrA Symbol Editor**.

3. Right-click **Tank** and select **Embedded Symbol | Convert To Group**.

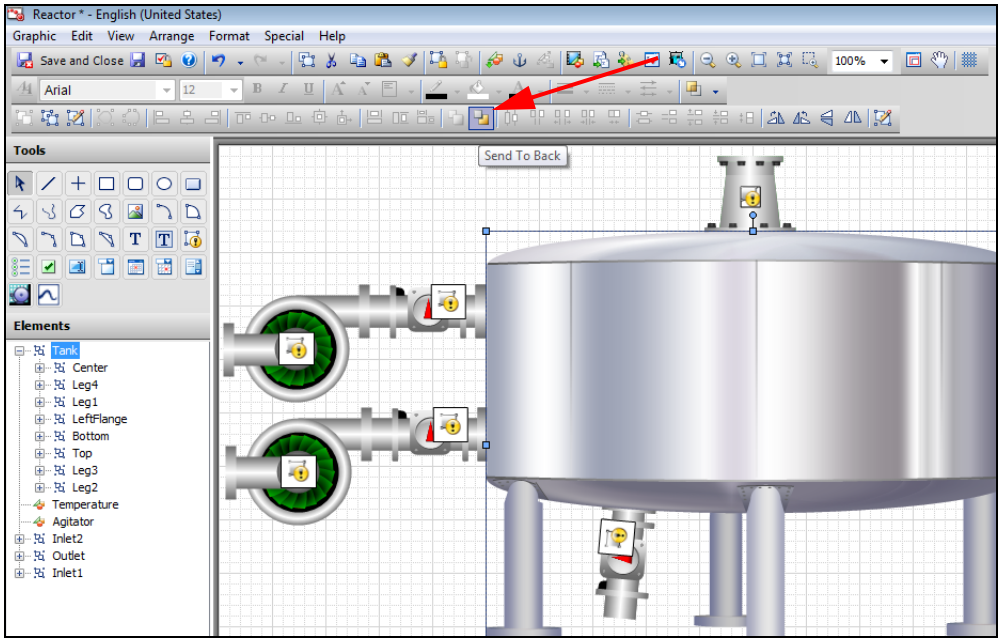


The embedded symbol is now a group named **Tank** containing all components of the original tank symbol. You can now modify **Tank** components.

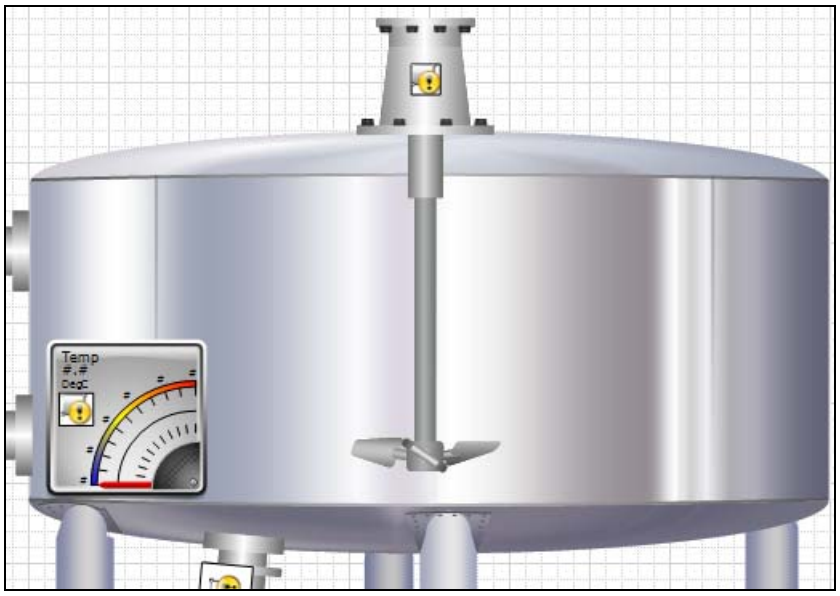
4. In **Elements** pane, expand **Tank**.



5. Ensure **Tank** is selected, then click the **Send to Back** button on the toolbar.



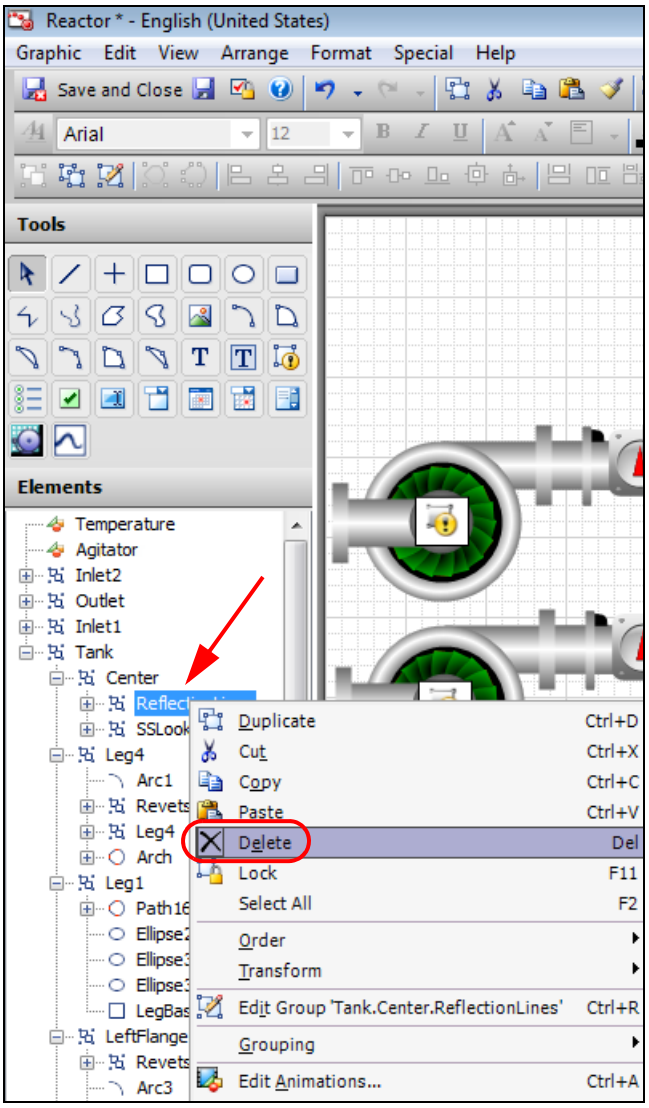
Notice the **Agitator** and **Temp** symbol are now in front of the **Tank**.



Next, you will create an effect to cover a part of the agitator with parts of the tank.

In the tank graphic, you will create a visual seam on your graphic. To improve the graphic appearance, you will remove the reflection lines.

6. In **Elements** pane, expand the **Tank** group, then expand the **Center** elements.
7. Right-click **ReflectionLines** and select **Delete**.

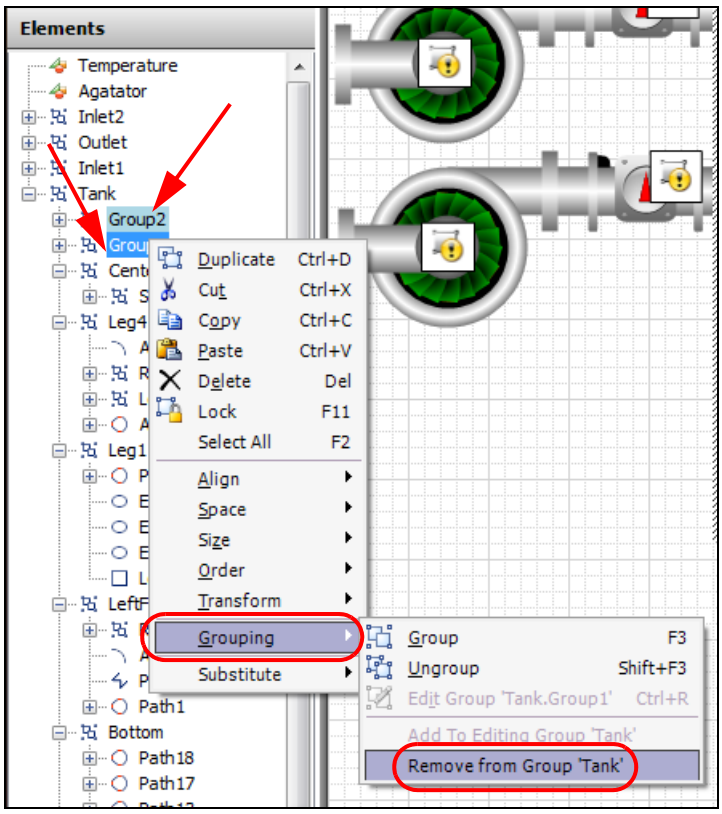


The **ReflectionLines** group is removed from your graphic.

8. In **Elements** pane, select both the **Center** and **Top** groups.
9. Right-click and select **Duplicate**.

Next, you will layer the graphics to cover the Agitator.

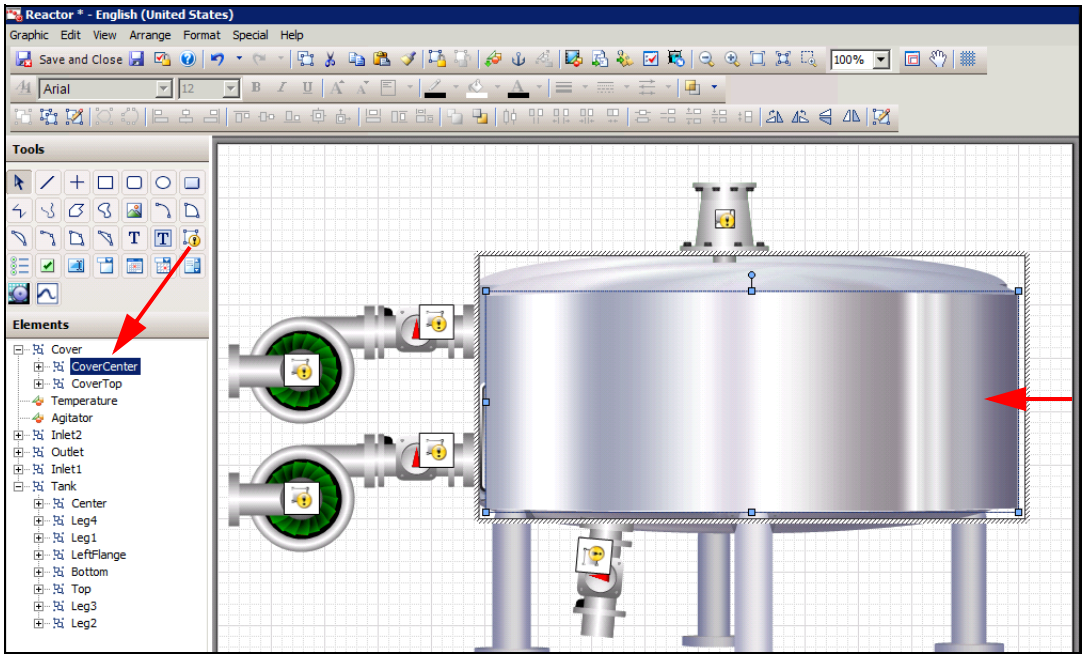
10. With **Group1** and **Group2** selected, right-click and select **Grouping | Remove from Group 'Tank'**.



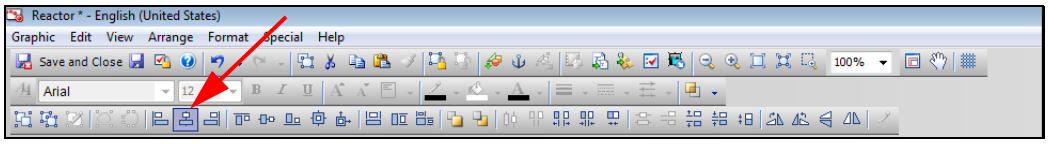
11. Rename **Group1** to **CoverTop**.
12. Rename **Group2** to **CoverCenter**,
13. Select **CoverTop** and **CoverCenter**.
14. Right-click and select **Grouping | Group**.
15. Name the new group **Cover**.

16. In **Elements** pane, expand **Cover** and click **CoverCenter**.

    Notice the **CoverCenter** element is selected in the graphic.

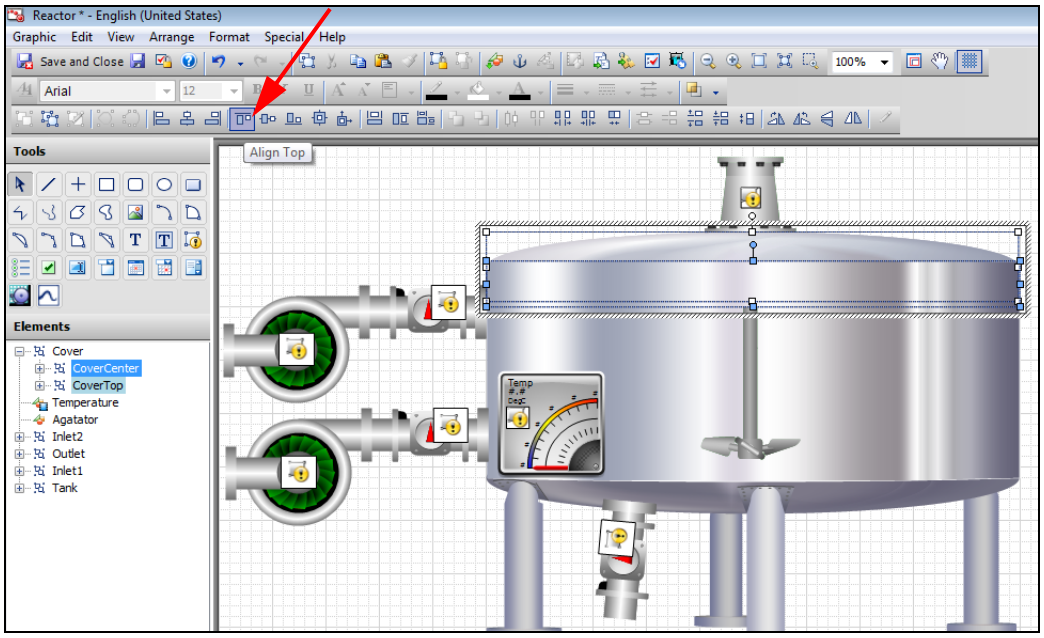


17. In **Properties** pane, change the **Height** to **43**.
18. In **Elements** pane, select both **Cover** and **Tank**.
19. On the toolbar, click the **Align Center** button.



20. On the toolbar, click **Align Top**.

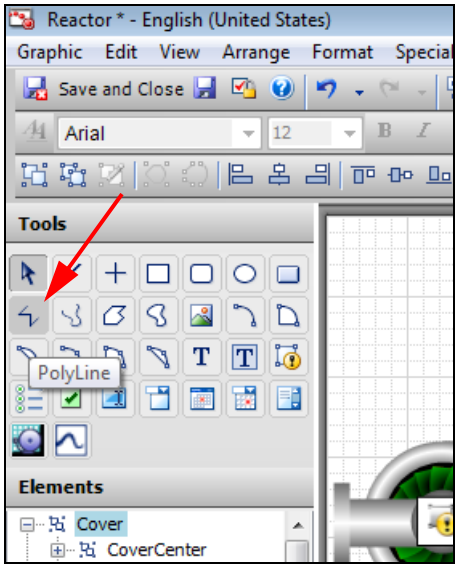
    Notice the **Cover** aligns with the **Tank**.



Create a Custom Element

Now you will create a tank cutaway to reveal the agitator.

21. In **Tools** pane on the left, click the **PolyLine** tool.

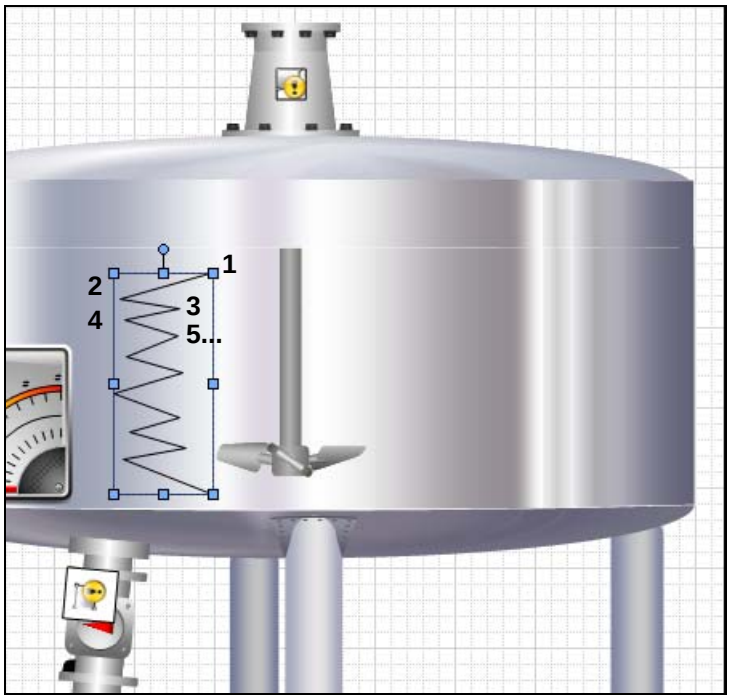


22. Click and move your mouse to draw a zigzag pattern on the tank as shown below.

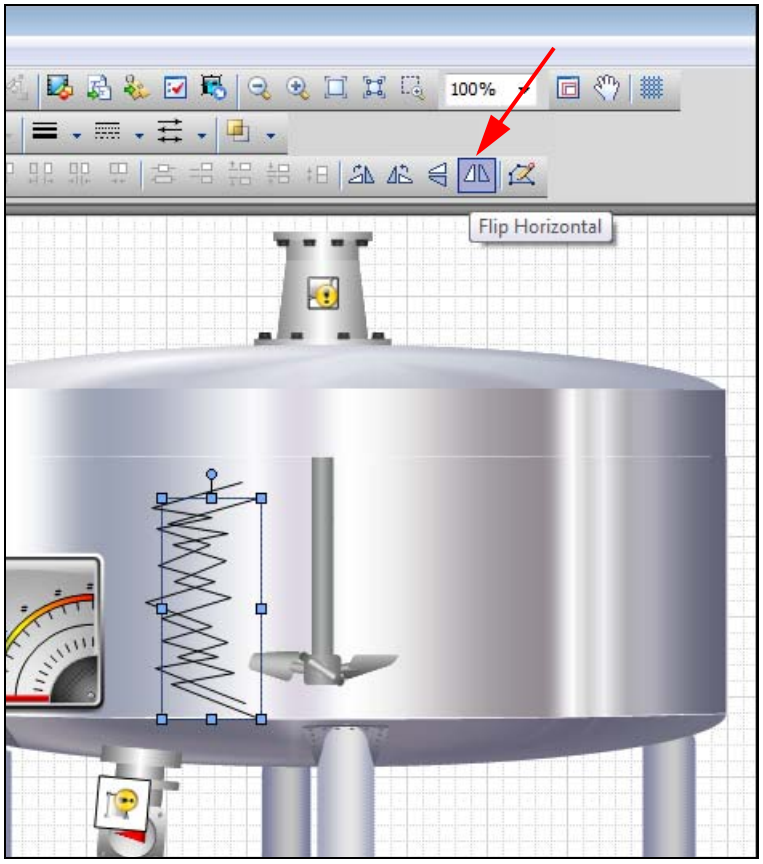
    Each time you click creates an anchor point.

    Continue until the zigzag pattern is near the bottom edge of the tank, and double-click to end.

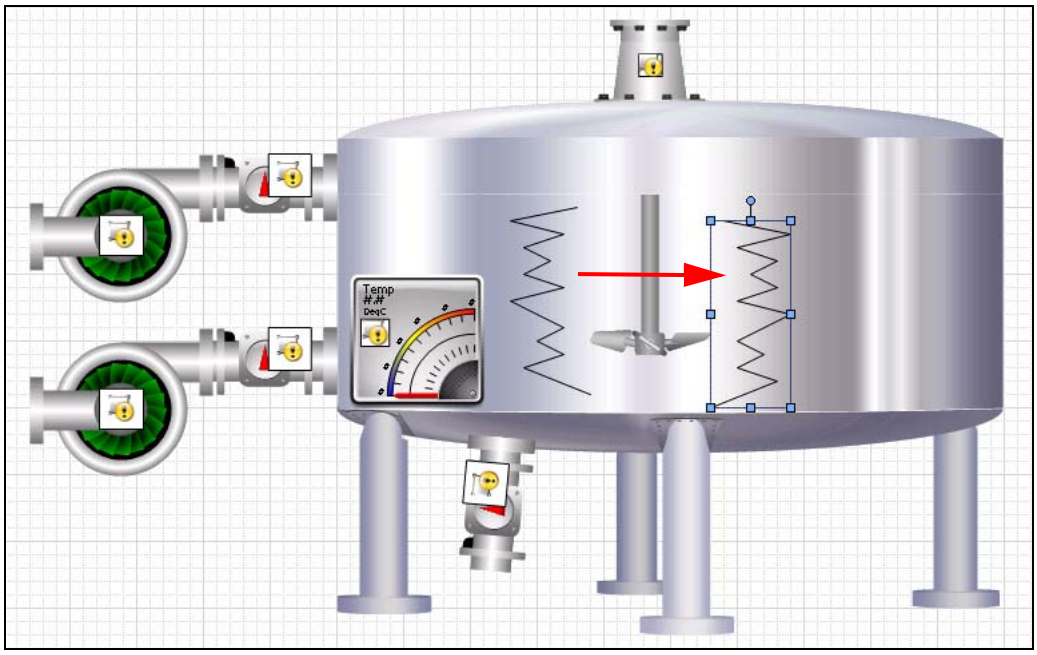
Important: Draw the polyline, top to bottom as shown.



23. Duplicate the element and click the **Flip Horizontal** button.

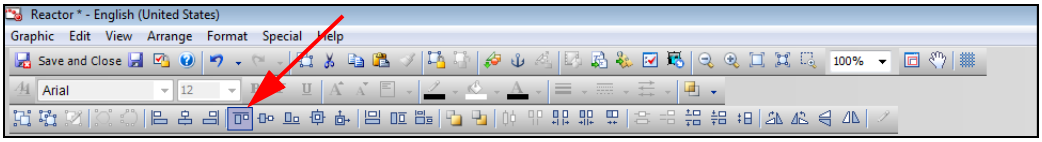


24. Move **PolyLine2** to the opposite side of the **Agitator** shaft.

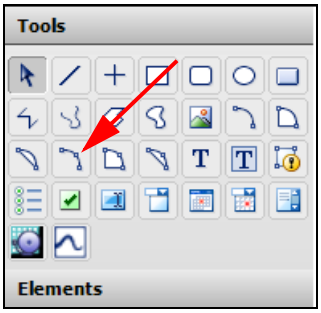


25. While **Polyline2** is selected, hold the **Shift key** and select **Polyline1**.

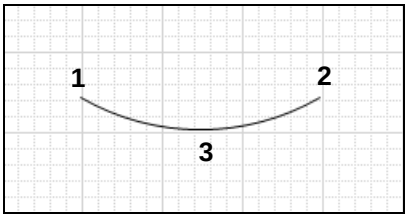
26. Click **Align Top**.



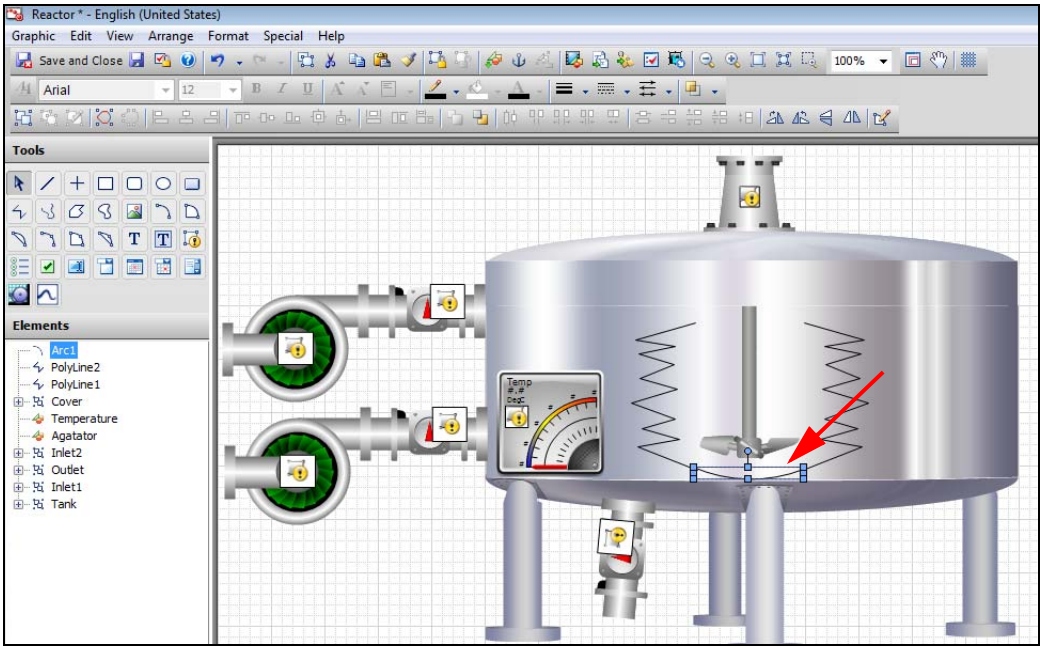
27. Click the **3 Point Arc** tool.



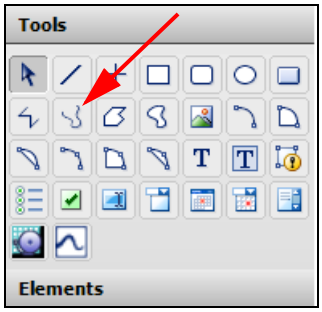
28. Click the bottom of **PolyLine1** to start the **3 Point Arc**.
29. Click the bottom of **PolyLine2**.
30. Click the bottom center edge of the tank and follow the point order as shown below.



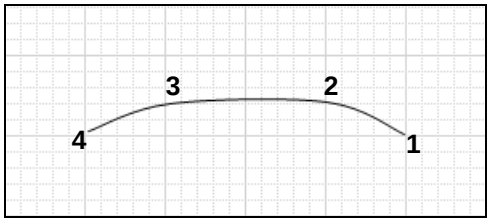
The completed arc is shown below.



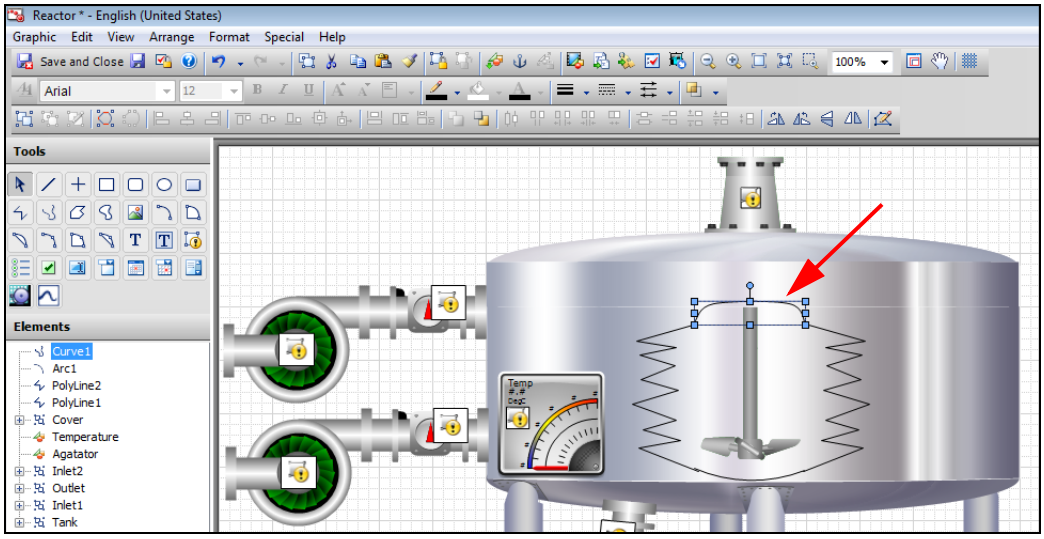
31. Click the **Curve** tool.



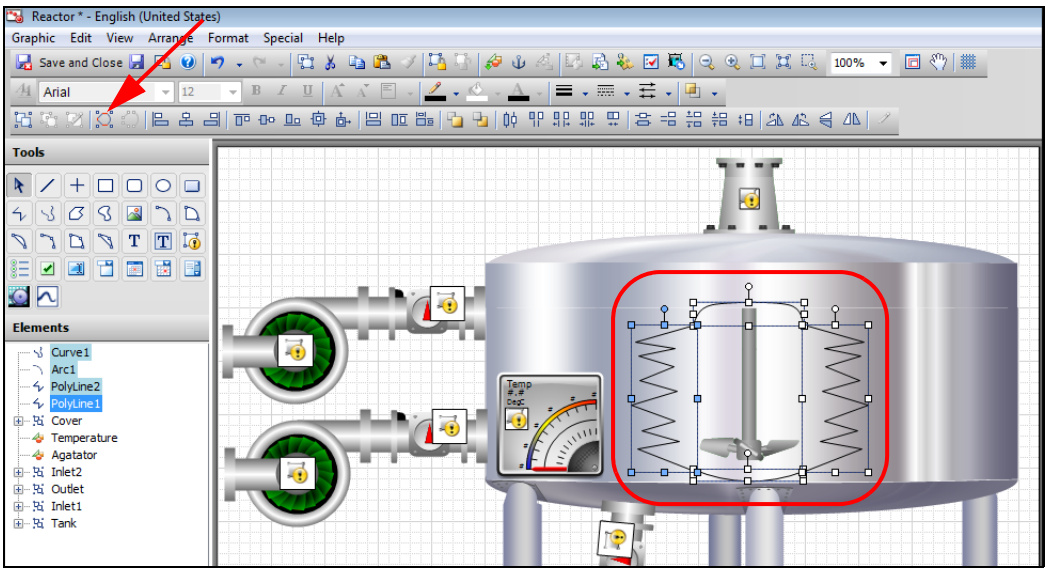
32. Click the top of **PolyLine2** to start the **Curve**.
33. Click the top edge of the tank to the left of **PolyLine2**.
34. Click the top edge of the tank to the right of **PolyLine1**.
35. Double-click the top of **PolyLine1**. Follow the point order for the curve as shown below.



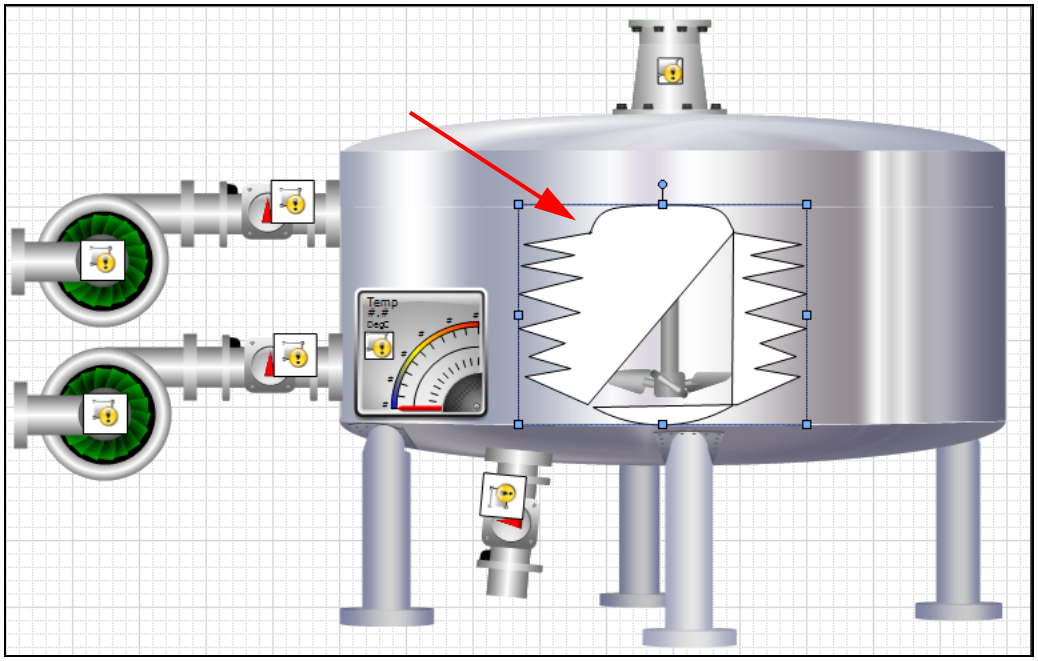
The completed curve is shown below.



36. On the toolbar, select all four new elements and join them using the **Path Combine** tool.



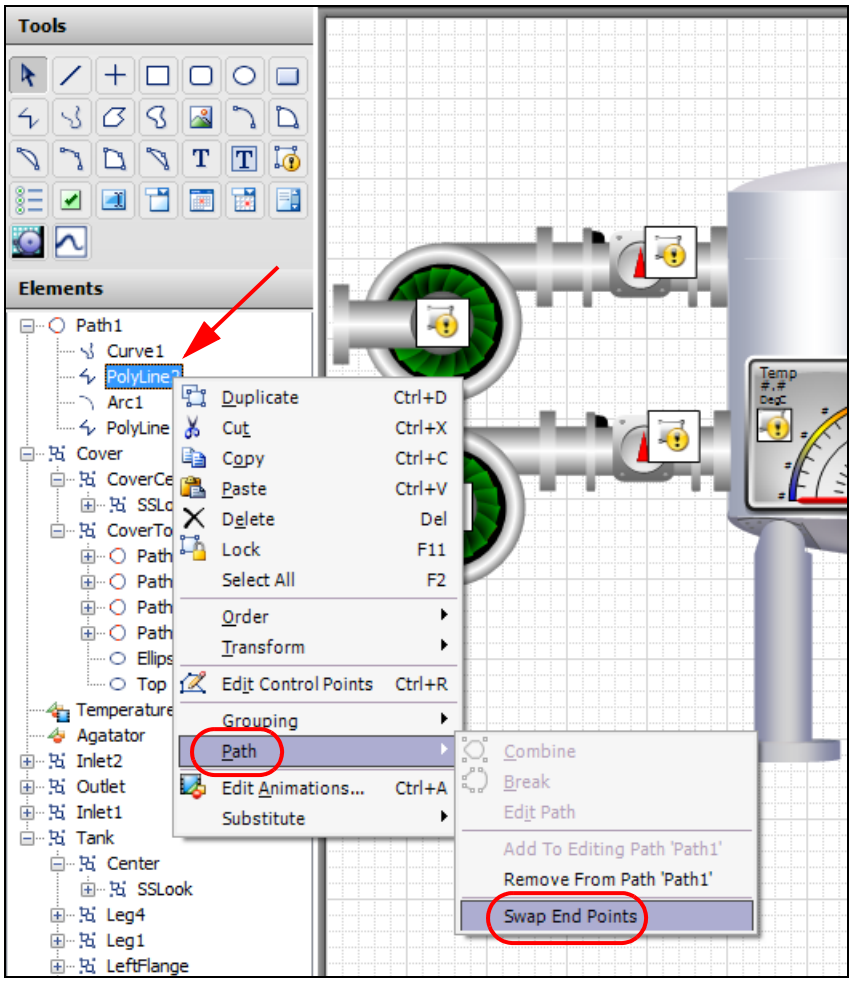
Notice the crossover between **Polyline1** and **Curve1** that occurred when you duplicated and flipped the polyline. The new element did not close properly due to the order and method the elements were drawn.



37. In **Elements** pane, expand **Path1**.
38. Click on an element, then use **Shift+F8** to order the elements as follows:
    - **Curve1**
    - **PolyLine2**
    - **Arc1**
    - **PolyLine1**

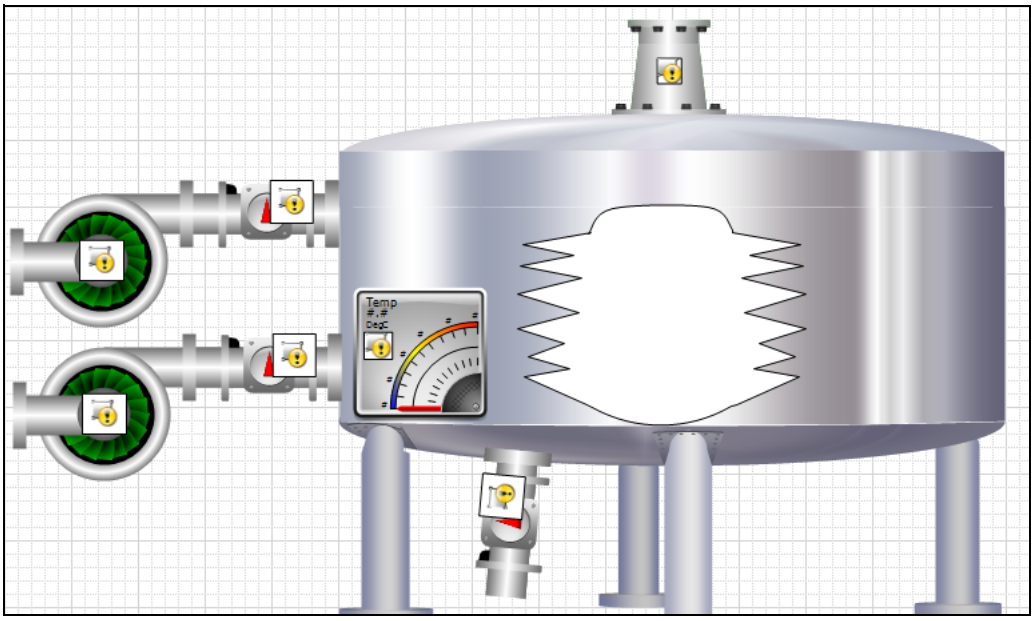
Now you will use Swap Endpoints to properly close the element.

39. Right-click **PolyLine2** and select **Path | Swap End Points**.

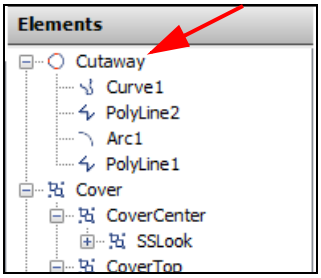


40. Reposition and resize as necessary.

    The completed cutaway is shown below.



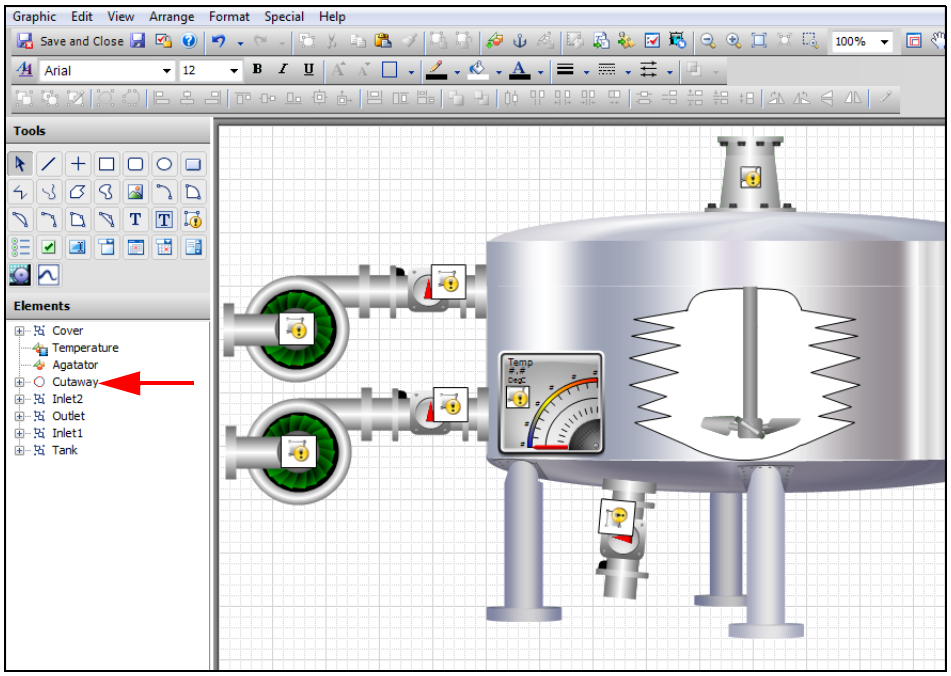
41. Rename **Path1** to **Cutaway**.



Next, you will position the tank cutaway on top of the Agitator and make the shaft visible.

42. In **Elements** pane, select **Cutaway** and press **F8** until it is directly below the **Agitator**.

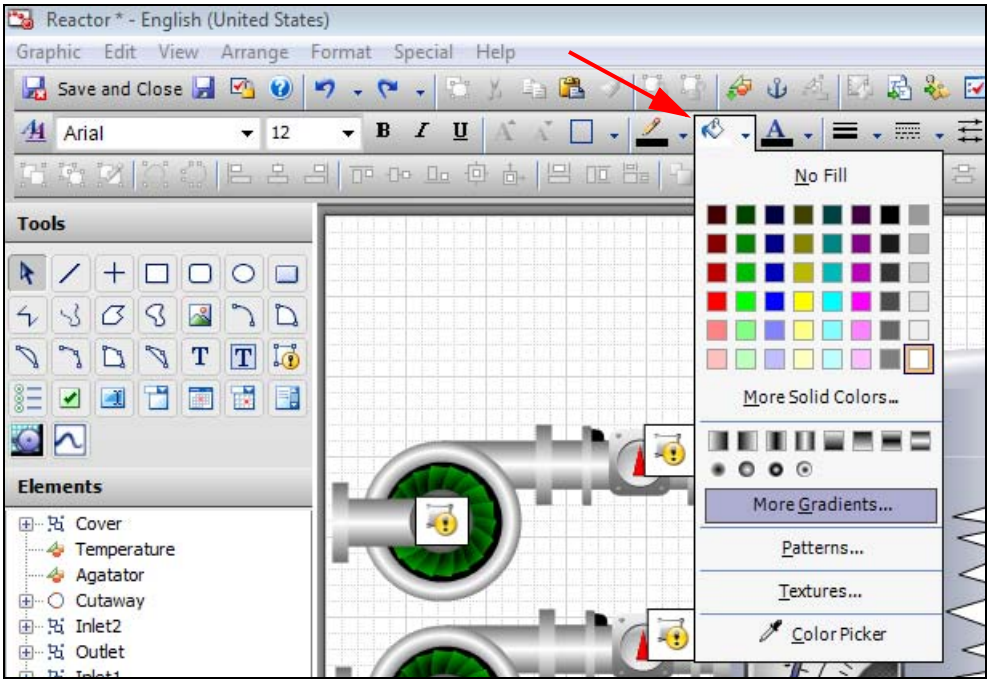
Note: Turn off **Snap to Grid** to refine the position and size of the **Cutaway** element. The **F12** key is the shortcut to turn **Snap to Grid** on and off.



Add Graphical Features

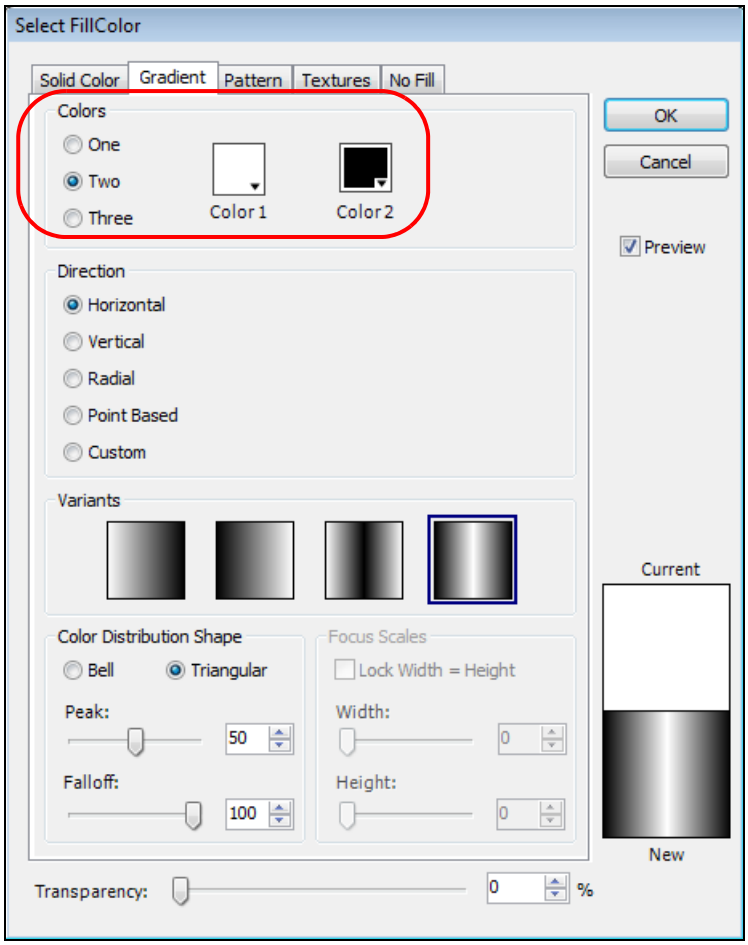
You can further improve the graphics with fill color and gradients.

43. Select **Cutaway**.
44. On the toolbar, click the **Fill Color** drop-down list, and select **More Gradients**.



45. On the **Gradient** tab, **Colors** area, click **Two**.

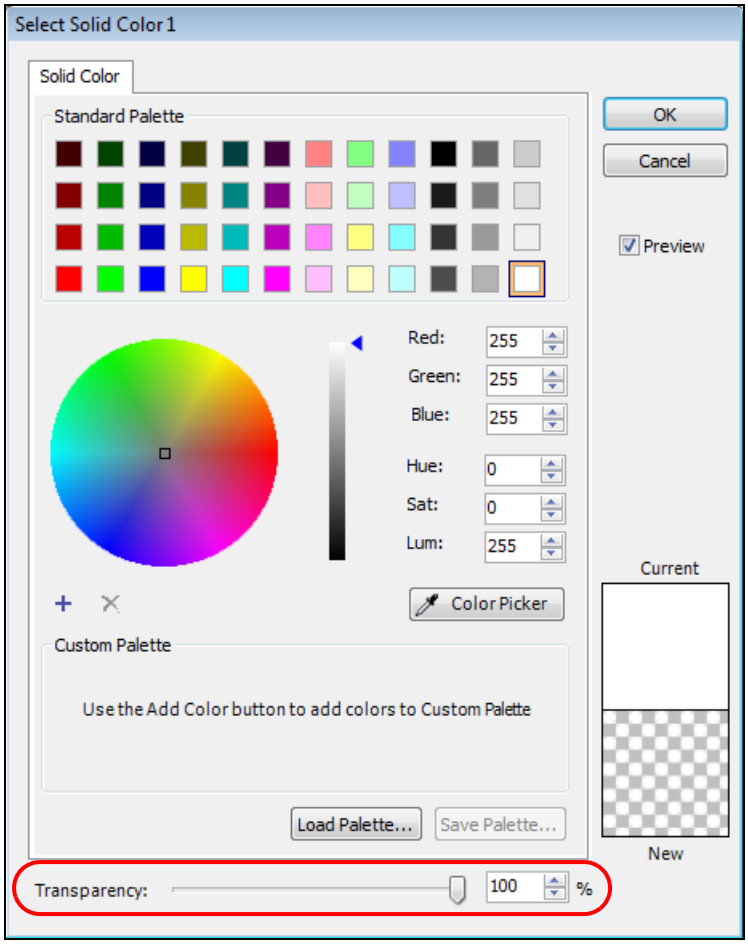
    This enables two color options.



46. Click **Color 1**.

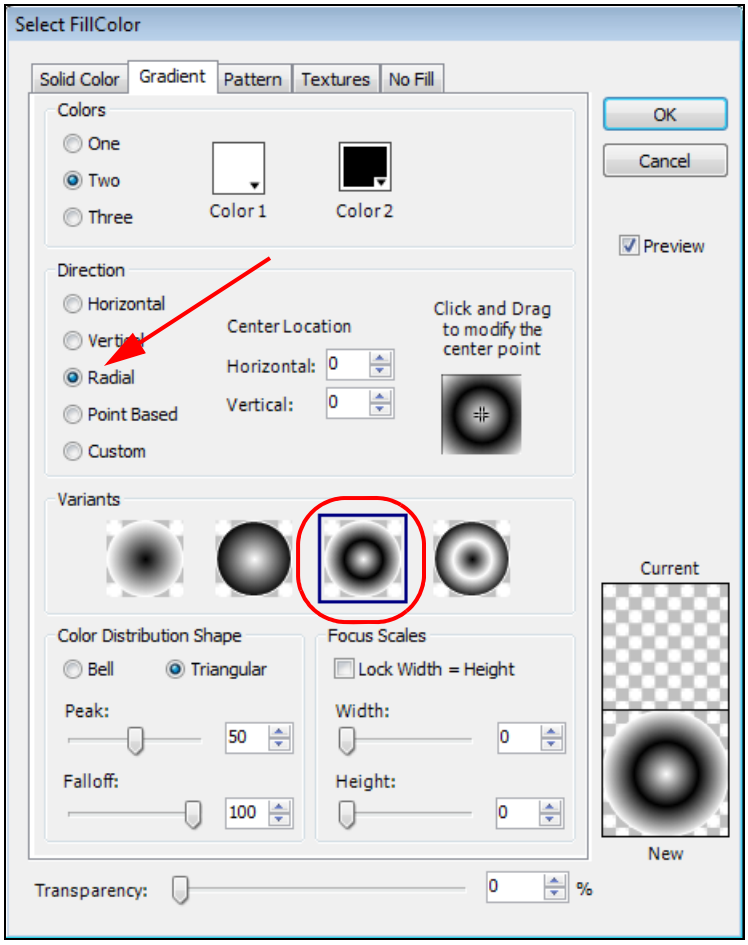
    The **Select Solid Color** dialog box appears.

47. At the bottom of the dialog box, change the **Transparency** to **100%**.

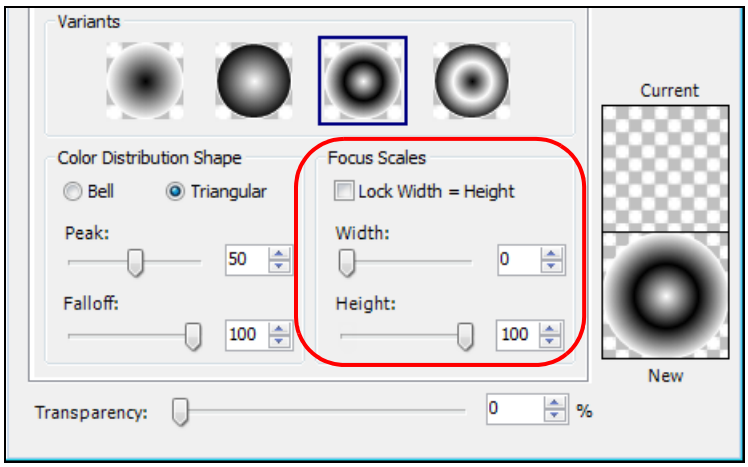


48. Click **OK**.
49. Leave **Color 2** as **Black**.

50. In the **Direction** area, click **Radial**, then in the **Variants** list select the third variant.



51. In the **Focus Scales** area, change the **Height** to **100%**.

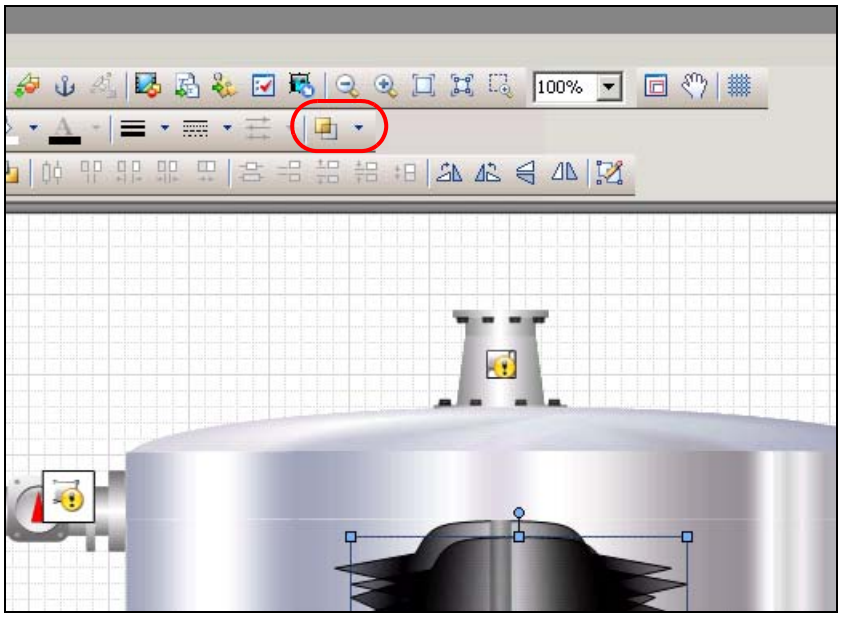


52. Click **OK**.

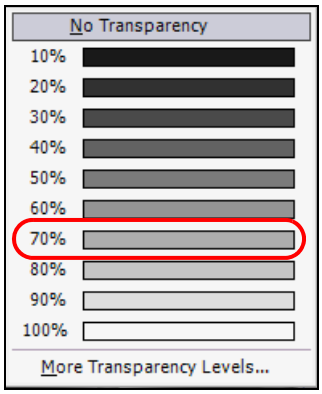
Create Level Indicator

Next, you will duplicate the cutaway. In a later lab, you will use this cutaway to provide a level indicator using an animation to fill and empty the tank.

53. Right-click the **Cutaway** and select **Duplicate**.
54. Rename the duplicate **Level**.
55. With **Level** selected, on the toolbar click on the **Transparency** drop-down arrow.

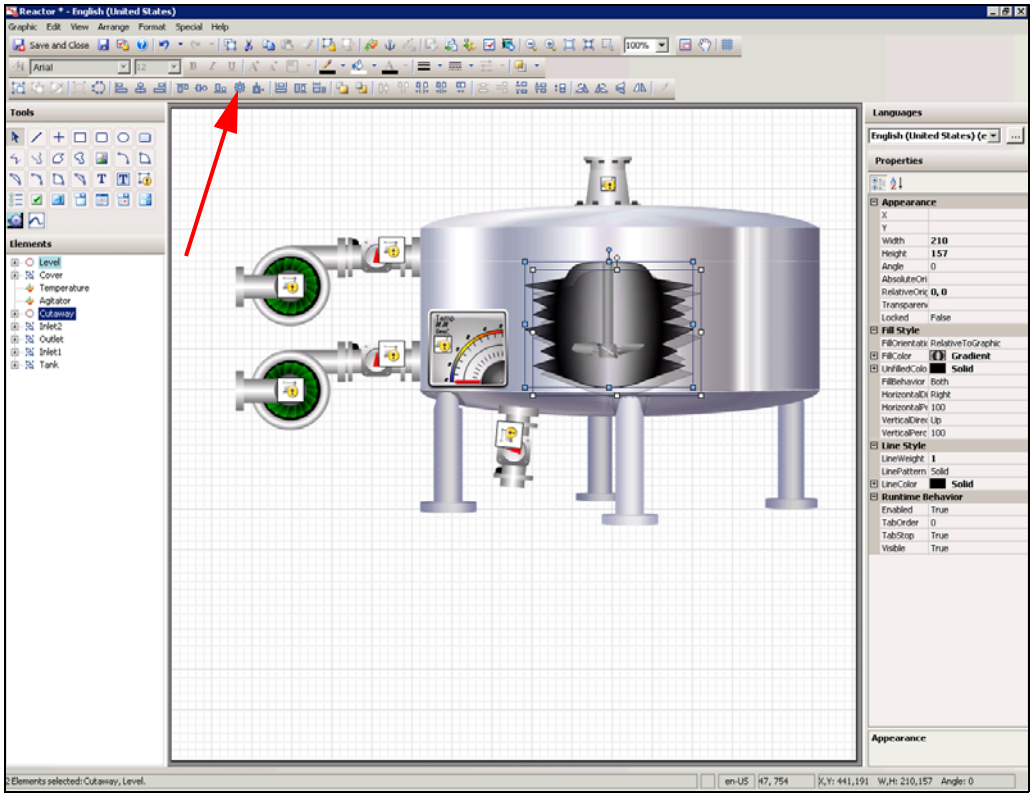


56. Select the **70%** transparency level.



Now align the two cutaway elements.

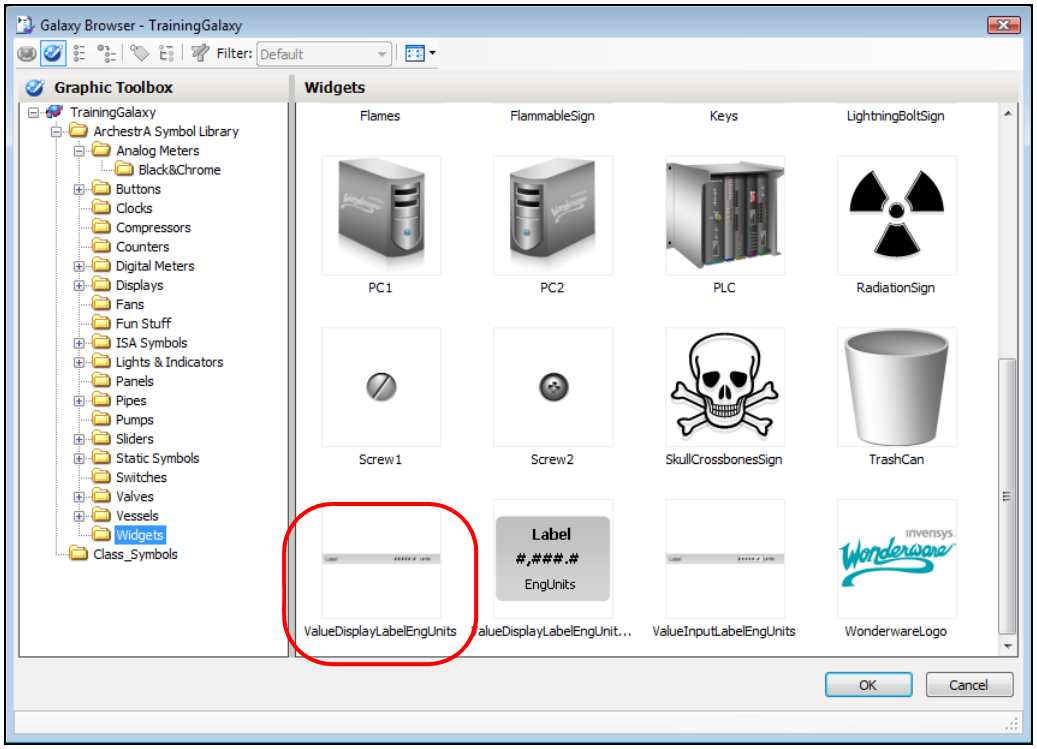
57. In **Elements** pane, press **Ctrl** and select **Level** and **Cutaway**, then on the toolbar click **Align Centers**.



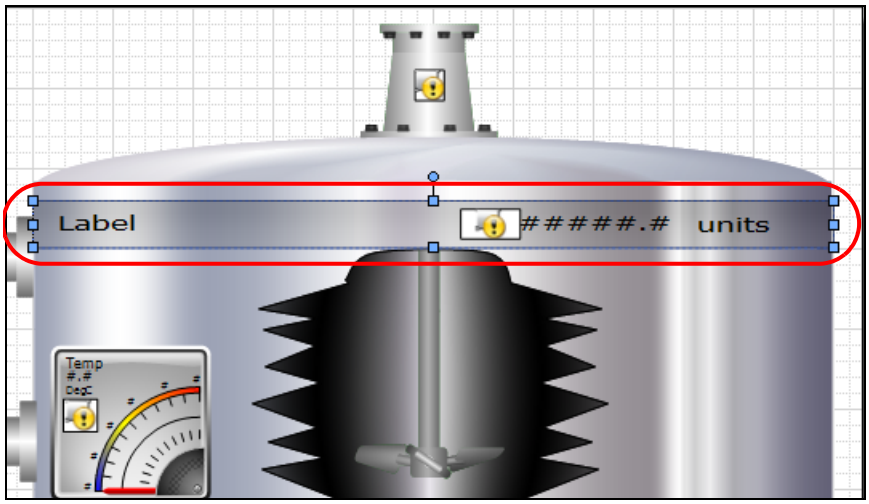
Create a Tank Level Display

Finally, you will create a digital display used to view the Tank level

58. Embed **Widgets\ValueDisplayLabelEngUnits** on the tank.

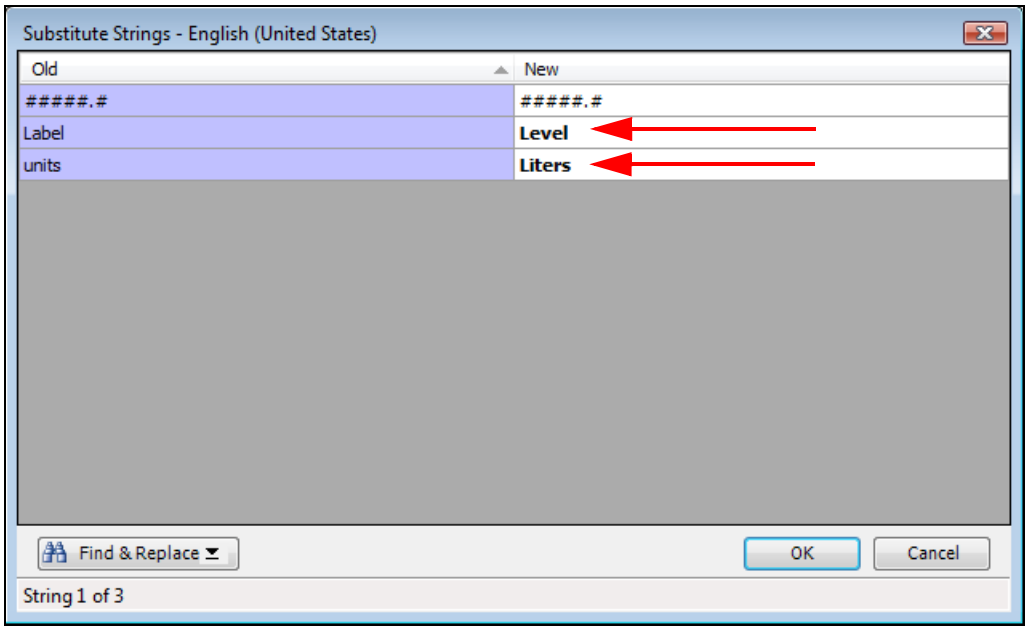


59. Stretch the symbol to fit the width of the tank.



60. Rename the new embedded graphic **LevelDisplay**.

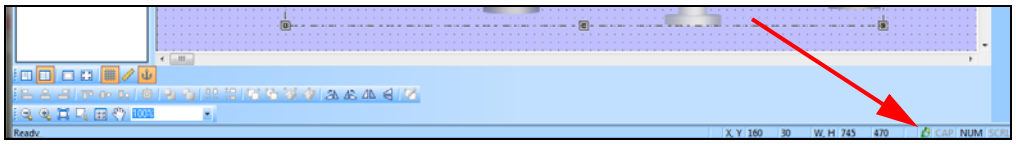
61. Right-click the element and select **Substitute | Substitute Strings**.
62. Change **Label** to **Level**.
63. Change **units** to **Liters**.



64. Click **OK**.
65. Click **Save** but do not close the **ArchestrA Symbol Editor**.

    You will continue editing this symbol in the next lab.
66. Switch to WindowMaker.

    Notice that the **Agitator** is still in front of the tank and none of the recent changes are displayed. WindowMaker is retaining the previous symbol used in the library.
67. In WindowMaker, double-click the **Refresh Embedded Symbols** icon in the status bar.



    The symbol is refreshed.

Section 3 – ArchestrA Symbol Custom Properties

This section describes custom ArchestrA symbol animation and the flexibility available when using custom properties.

Overview

Custom Properties are additional user-defined properties you can associate with an element.

Custom Properties allow you to make the most use of your symbols by making them customizable when embedded. You can use Custom Properties to extend the functionality of a symbol. A custom property can contain:

- A value that can be read and written to
- An expression that can be read
- A property of an element or symbol
- A custom property of a symbol

Custom Properties can be set to either be Private or Public:

- **Private**: Property is not exposed when the symbol is embedded
- **Public**: Property can be customized when the symbol is embedded

What are Properties?

Properties determine the appearance and behavior of an element or of the symbol.

There are two types of properties:

- **Predefined** properties
- **Custom** properties

Groups have special properties, some of which they propagate to their contained elements, some of which they do not.

Predefined Properties

Properties are specific to the selected element and may vary between elements of different types.

All elements have the following property groups:

- **Graphic** – the name of the element (or group)
- **Appearance** – element dimension, location, rotation, and transparency

You can view specific properties for a specific kind of element or group by clicking a drawing tool and drawing an element. These properties can be changed at design time and more importantly also at runtime. You can use the properties of elements in scripting to dynamically manipulate the appearance of any element of the ArchestrA symbol.

Custom Properties

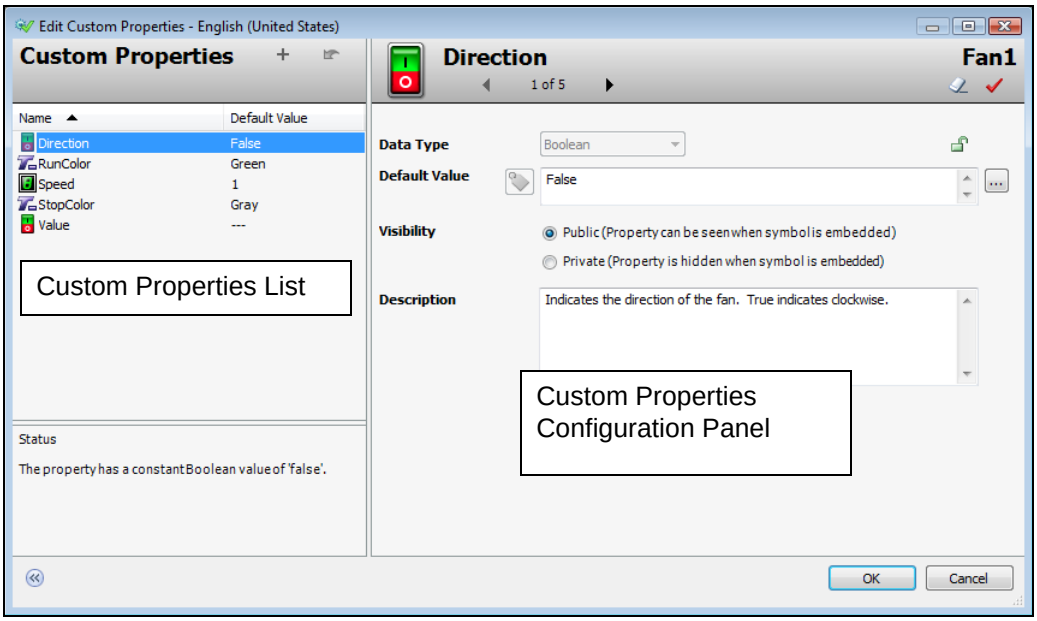
You can use Custom Properties to extend the functionality of a symbol. A custom property can either contain:

- A value, that can be read and written to
- An expression that can be read
- A property of an element or symbol
- A custom property of a symbol
- A reference to an InTouch tag

Custom Properties appear in the **Properties Browser**. You can either edit default values of custom properties in the **Properties** pane, or in the **Edit Custom Properties** dialog box.

Managing Custom Properties

You can manage all custom properties of a symbol in the **Edit Custom Properties** dialog box.



When you edit the custom properties of a symbol embedded from the ArchestrA Symbol Library, each property listed has a default value data type, default value, default visibility, and default description. The description is useful because it indicates what the symbol can do, and to a degree how to configure it. You can see the name of the symbol and the custom property in the header of the right side of the dialog box.

Adding Custom Properties

To add a custom property, display the **Edit Custom Properties** dialog box, click the **Add** button **+**, and enter a name. Next, select the **Data Type** list of the custom property. Data types include the following:

- **Boolean** (InTouch Discrete)
- **Double** (InTouch Real)
- **Float** (InTouch Real)
- **Integer** (InTouch Integer)
- **String** (InTouch Message)
- **Elapsed Time / Time** (ArchestrA Object Based Design not supported by Tag-Based InTouch)

Use the **Lock** icon  to make the property read-only. This will make the property read-only at design time to prevent further changes to it when the symbol is embedded into another symbol, and to prevent the value from being changed at run-time.

In the **Default Value** field, type a literal value, reference, or expression, or browse for a reference using the **Browse** icon.

If the selected data type is **String**, **Time**, or **Elapsed Time**, you can:

- Select the **T** icon to indicate that the default value is a static value.
- Select the **Label** icon to indicate that the default value is a reference to a value.

For **Visibility**, select either:

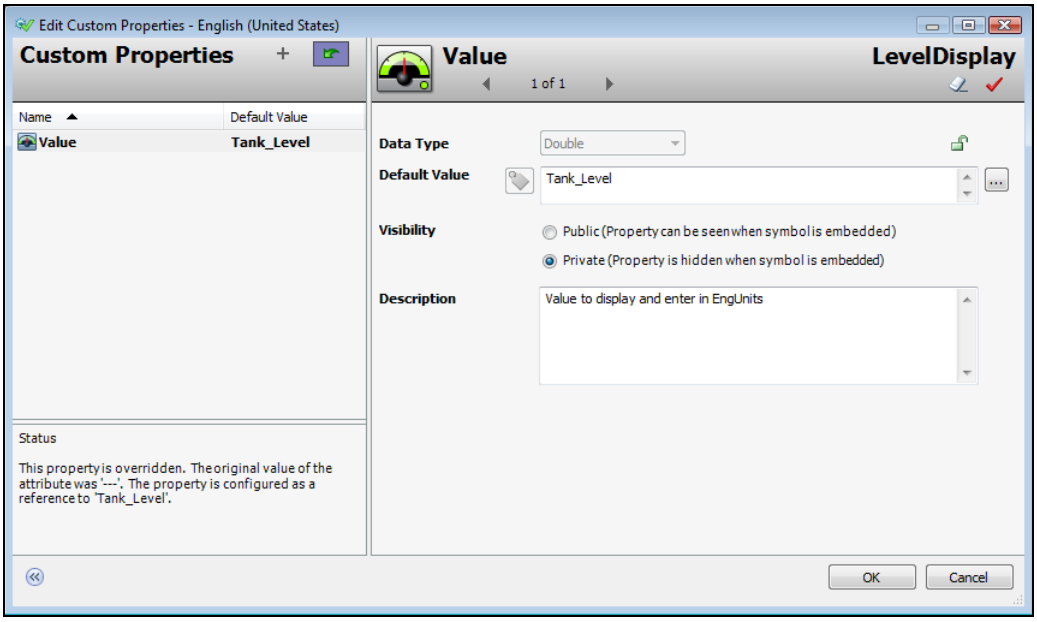
- **Public**, so that the custom property is visible and can be used in a parent symbol if the symbol is embedded or inserted.
- **Private**, so that the custom property is not visible outside of the defining symbol and cannot be used if the symbol is embedded or inserted.
- In the **Description** field, type instructions or guidance for using this property, such as an example or short list of a legitimate values.

Note: For additional information see *Creating and Managing ArchestrA Graphics User Guide*.

Linking Custom Properties to External Sources

You can link Custom Properties of a symbol directly to external sources by configuring a special InTouch reference syntax in the **Default Value** field. When you insert the symbol in an InTouch window, the referenced InTouch tagnames connect to the tagnames in InTouch.

To connect an ArchestrA symbol to an InTouch tagname, right-click the symbol and select **Edit Custom Properties**, and then select the custom property you want to connect to and enter the InTouch tagname in the **Default Value** field, or click the **Browse Galaxy** button to select a tagname from the **Select Tag** dialog box.



Any animation in the ArchestrA symbol configured with that custom property will now interact with the InTouch tagname.

Editing Custom Properties

Edit an ArchestrA symbol using the **Edit Custom Properties** dialog box to do the following:

- View an ArchestrA symbol's custom properties
- Find out about a symbol's settings and behaviors
- Connect ArchestrA symbols with InTouch tagnames

Section 4 – ArchestrA Symbol Animation

This section describes best practices in ArchestrA symbol animation, including the use of $System tags present in all InTouch applications.

Overview

You can use animations to specify how the element appears at runtime. Animations are driven by data that comes from an ArchestrA object's attribute values and expressions, as well as the element properties.

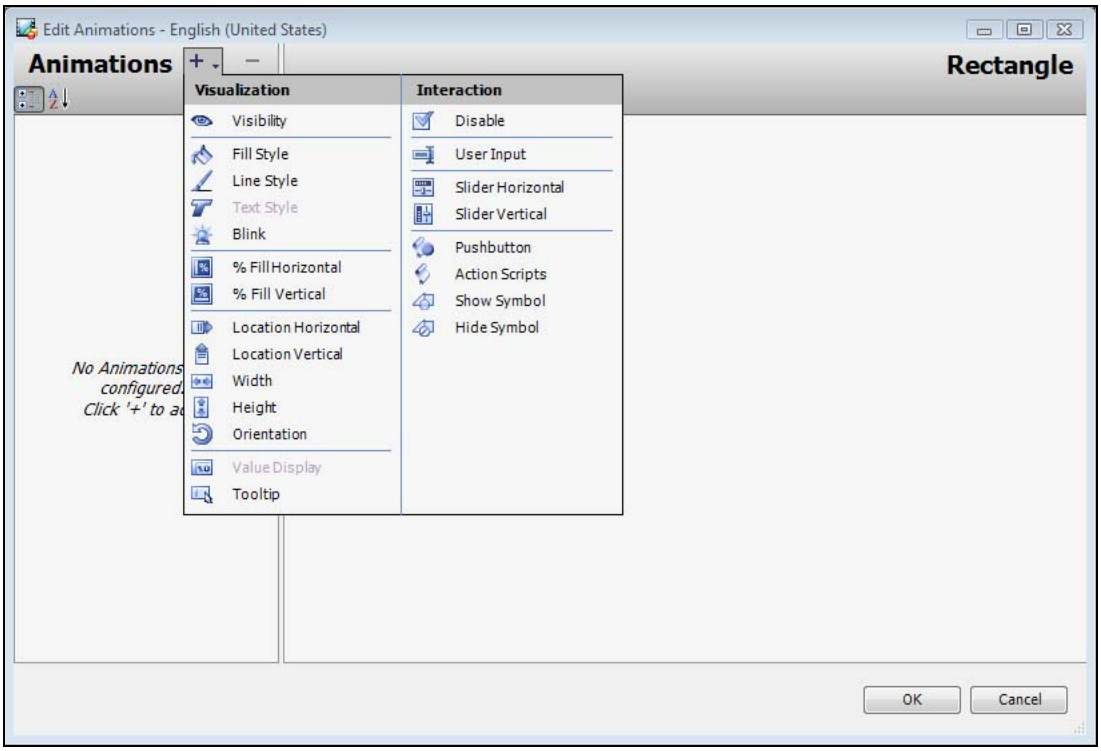
Several different categories of animations are available:

- **Visualization animations**
  - Visibility
  - Fill, line and text styles
  - Blinking
  - Horizontal and vertical percent fill
  - Horizontal and vertical location
  - Width and height
  - Orientation
  - Value display
  - Tooltip
- **Interaction animations**
  - Disablement
  - User input
  - Horizontal and vertical sliders
  - Pushbutton
  - Action Script
  - Show and hide symbols
- **Element-specific animations**
  - Status
  - Windows Common Controls
- **Client Controls**

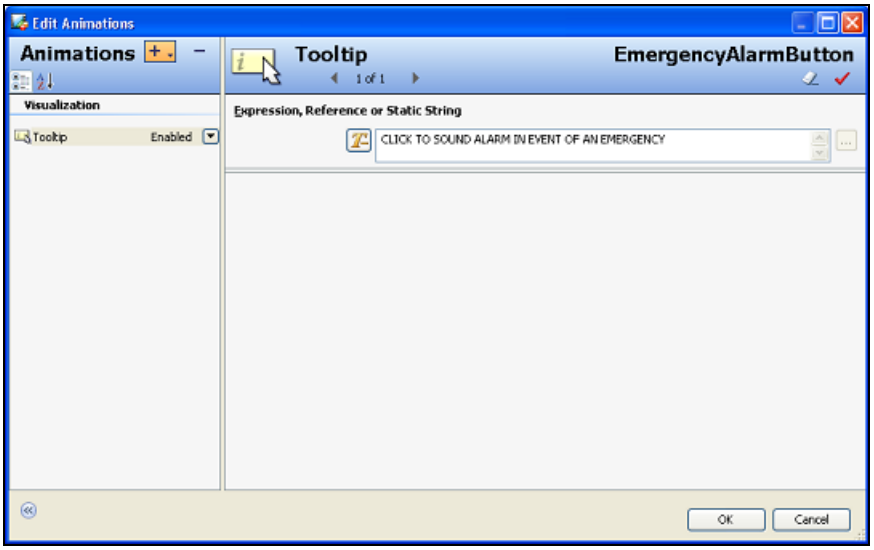
You can disable and enable individual animations for troubleshooting and debugging purposes. You can also cut, copy, and paste animations between elements regardless of their type. Only animations that are supported by the target element are pasted. In addition, you can substitute references and strings in animations.

Edit Animations

You can add, remove, and view animations for an element using the **Edit Animations** dialog box accessed from the **ArchestrA Symbol Editor | Edit | Animations**. You can also access this dialog box from the **Animations** pane, or by double-clicking an element on the canvas.

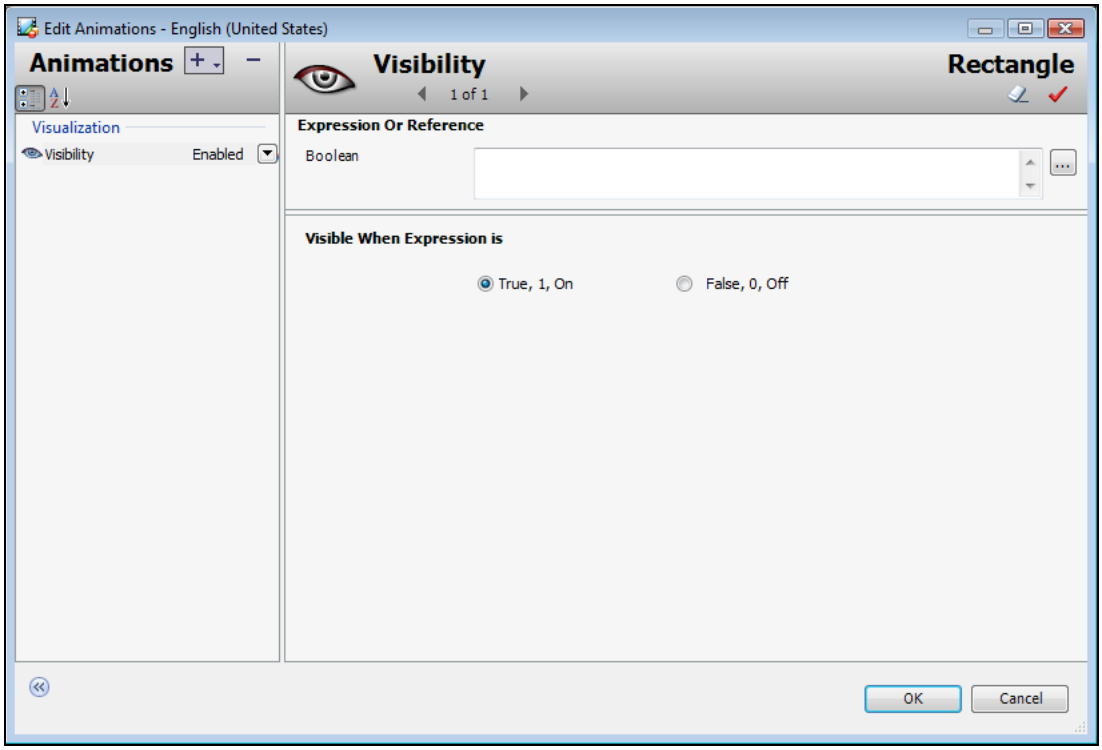


Use the **Tooltip** animation to specify an **Expression**, **Reference**, or **Static String** to display as a tooltip when the mouse is moved over the element at runtime.



Interaction

After an animation is selected, its parameters can be configured.



In the **Edit Animations** dialog box, you can also:

- Validate the Configuration of an Animation
- Connect Animations with Element Properties
- Connect Animations with Custom Properties

Validate the Configuration of an Animation

If the configuration has an error, an exclamation mark is shown next to the animation icon  .

Use the **Validate** button to validate the currently selected animation and see possible errors highlighted.

Errors include for example:

- Animation is disabled
- Syntax errors, such as data mismatches
- Required values not specified
- Specified values out of valid range

Connecting Animations with Element Properties

You can connect the element behavior and appearance with a property of any element on the canvas.

You can use the **Browse Galaxy** button to browse the properties of all elements on the canvas in the **Galaxy Browser**. Elements can be accessed in the **Element Browser** tab.

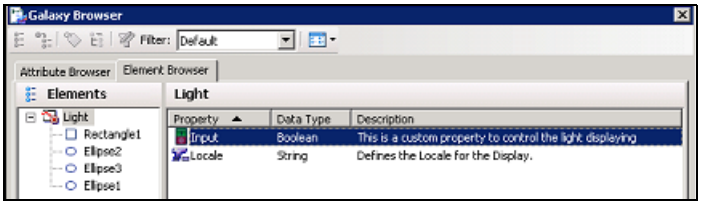
Note: You cannot connect animations to properties of elements that are part of an embedded symbol on the canvas, only to **Custom Properties** (more on this later) of the embedded symbol itself.

Connecting Animations with Custom Properties

You can connect the element behavior and appearance with a custom property of either:

- The current symbol
- An embedded symbol on the canvas

You can use the **Browse Galaxy** button to browse the custom properties of all elements on the canvas in the **Galaxy Browser**.



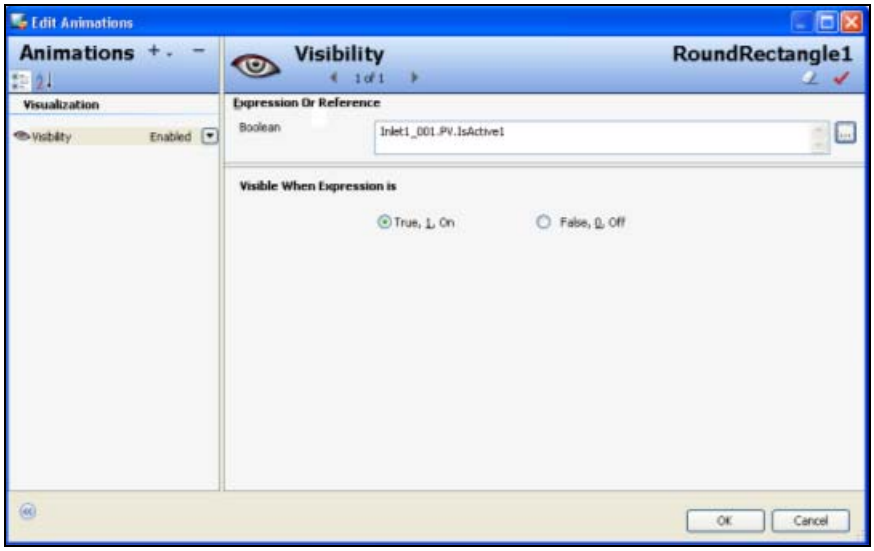
Configure Common Types of Animations

This section will show you a sample of each type of animation and what references it can use.

Visualization Animations

Visibility

Use the **Visibility** animation to specify a **Boolean** expression that determines if an element will be visible or not at runtime.

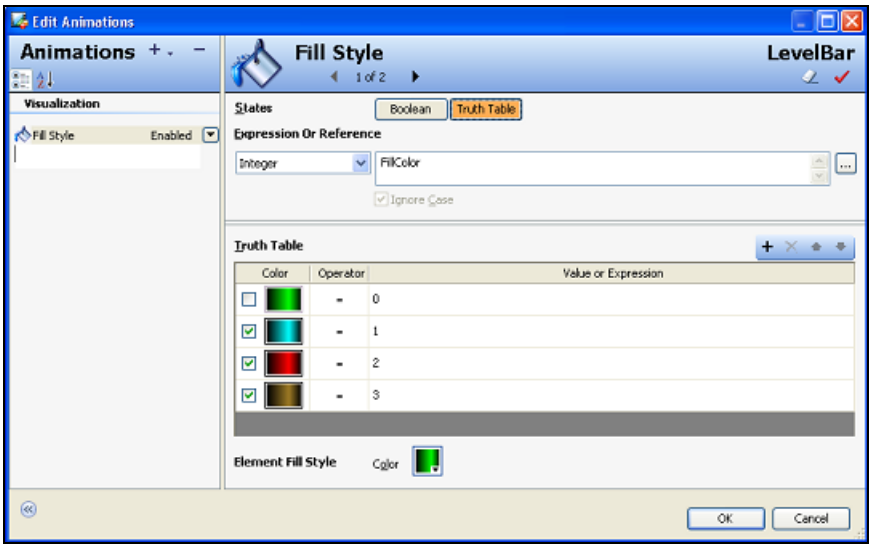


Fill, Line, and Text Style

Use the **Fill**, **Line**, or **Text Style** animations to specify an expression that determines the fill style of an element at runtime, based on a **Boolean** condition or a **Truth Table** of values.

The Fill, Line, or Text Style animations can specify the interior fill style of:

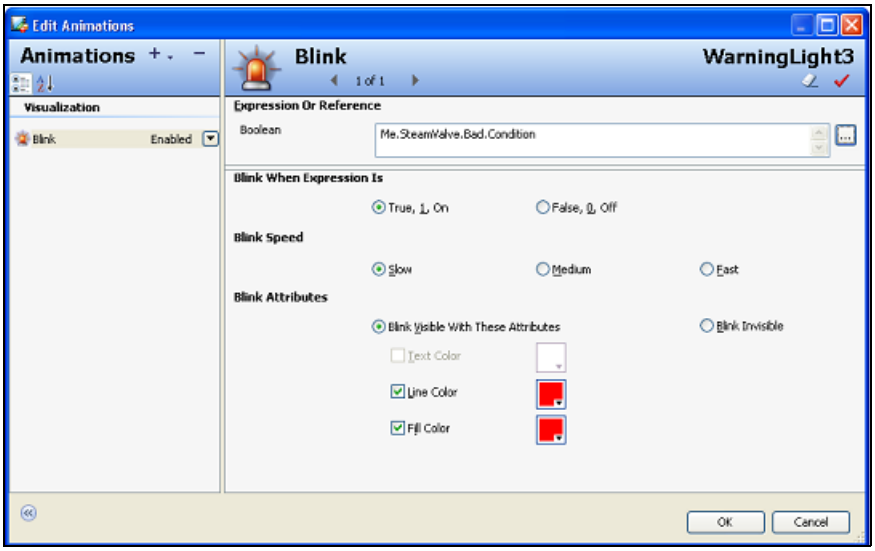
- Selected elements on the toolbar
- **Style** properties in the **Properties Editor**
- Nested style properties, such as just one color of a multi-colored gradient



Blink

Use the **Blink** animation to specify a **Boolean** expression that determines the blink behavior of an element at runtime. You can specify:

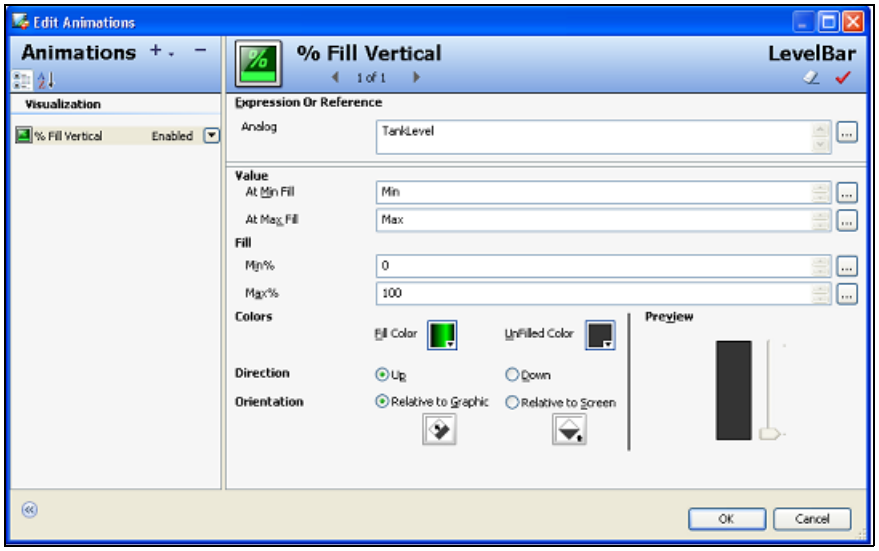
- **Blink When Expression Is**: True or False
- **Blink Speed**: Slow, Medium, or Fast
- **Blink Attributes**: If the element should blink invisibly or if it should blink with specified colors



% Fill Vertical and % Fill Horizontal

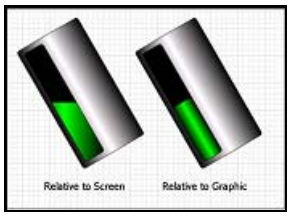
Use the **%Fill Horizontal** animation to specify an **Analog** expression or reference that determines the horizontal fill percentage and direction (left to right or right to left) of an element at runtime.

Use the **%Fill Vertical** animation to specify an **Analog Expression** or **Reference** that determines the vertical fill percentage and direction (bottom to top or top to bottom) of an element at runtime.



You can also specify:

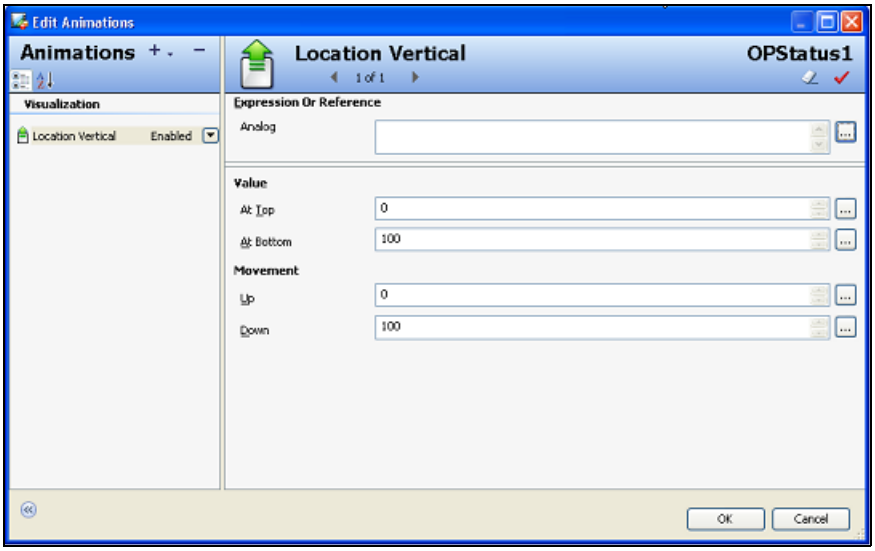
- **Unfilled Color**: the style of the background when the element has 0% filling
- **Fill Orientation**: if the filling is in relation to the element or to the screen



Location Vertical and Location Horizontal

Use the **Location Vertical** animation to specify an **Analog** expression that determines the vertical offset of the element's position at runtime.

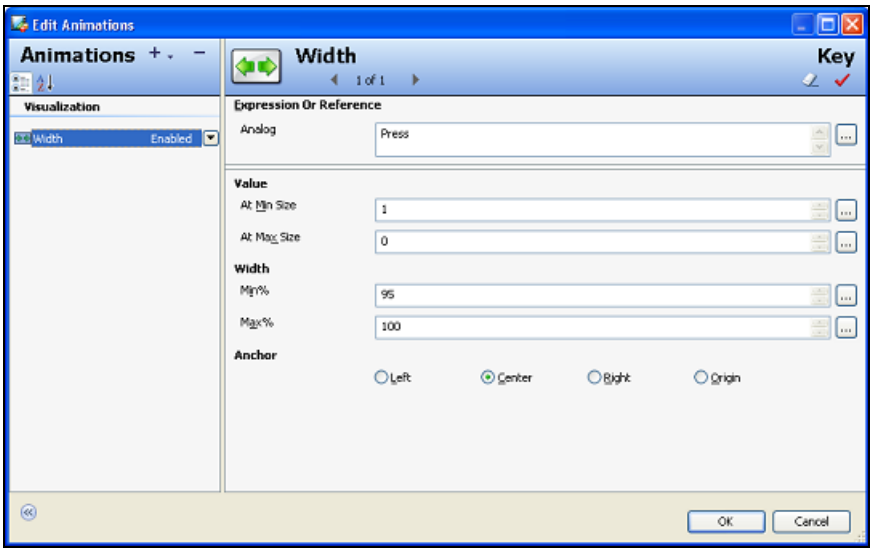
Use the **Location Horizontal** animation to specify an **Analog** expression that determines the horizontal offset of the element's position at runtime.



Width and Height

Use the **Width** animation to specify an **Analog** expression that determines the width (increased or decreased) of an element at runtime.

Use the **Height** animation to specify an **Analog** expression that determines the height (increased or decreased) of an element at runtime.

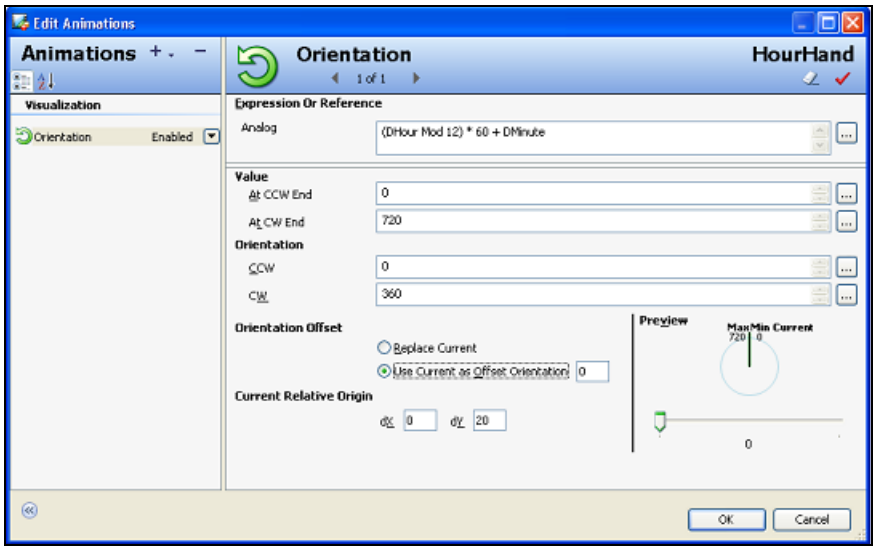


Orientation

Use the **Orientation** animation to specify an **Analog** expression that determines the orientation of an element at runtime.

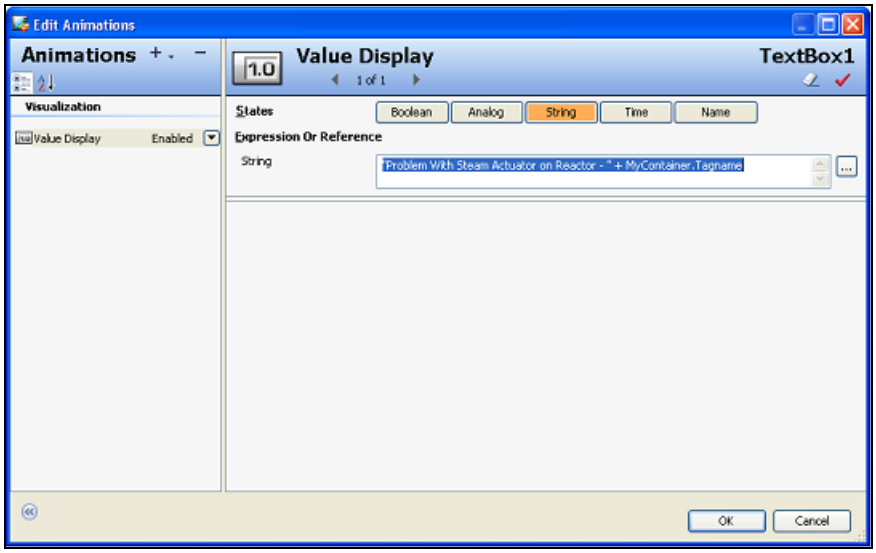
You can also:

- Specify a different orientation origin
- Ignore or accept the design-time orientation of the element on the canvas
- Preview the orientation at run time with a slider



Value Display

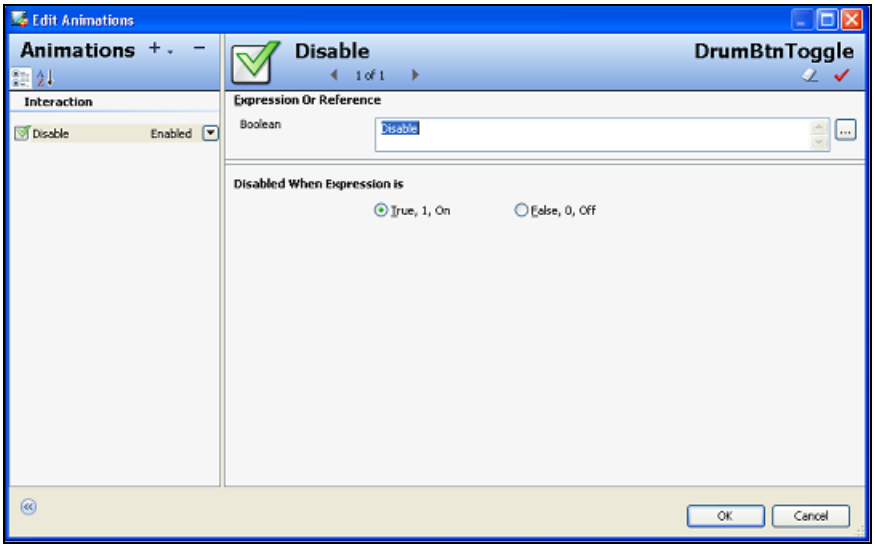
Use the **Value Display** animation to specify a **Discrete**, **Analog**, **String**, **Time**, or **Name** expression to display at runtime.



Interaction Animations

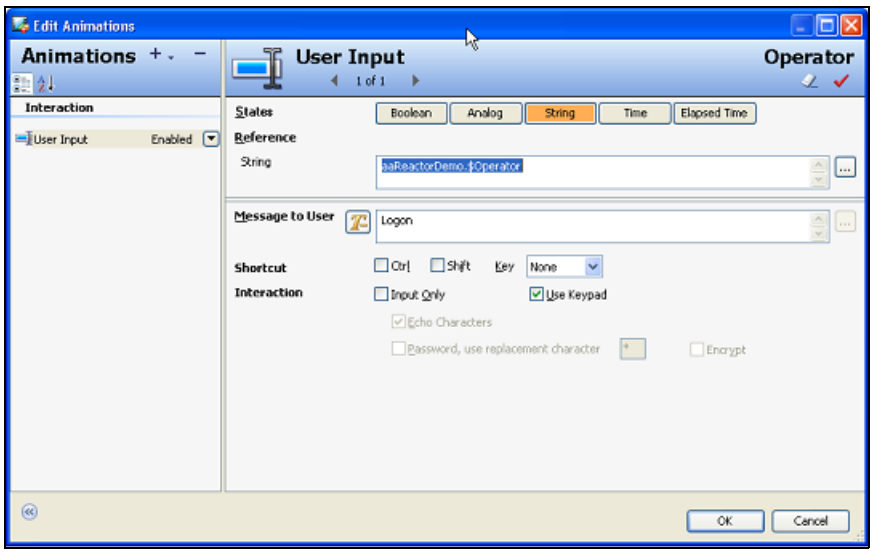
Disable

Use the **Disable** animation to specify a **Boolean** expression or reference that determines if an element's interaction animations (animations that execute a task when the object is touched with the mouse) are disabled at runtime.

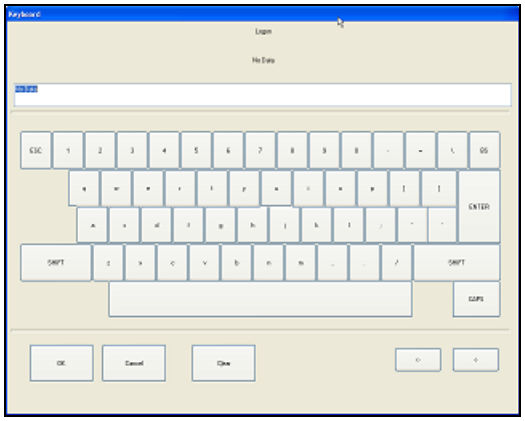


User Input

Use the **User Input** animation to specify an **Analog**, **Boolean**, **String**, **Time**, or **Name Reference** that is assigned to an attribute or property at runtime.



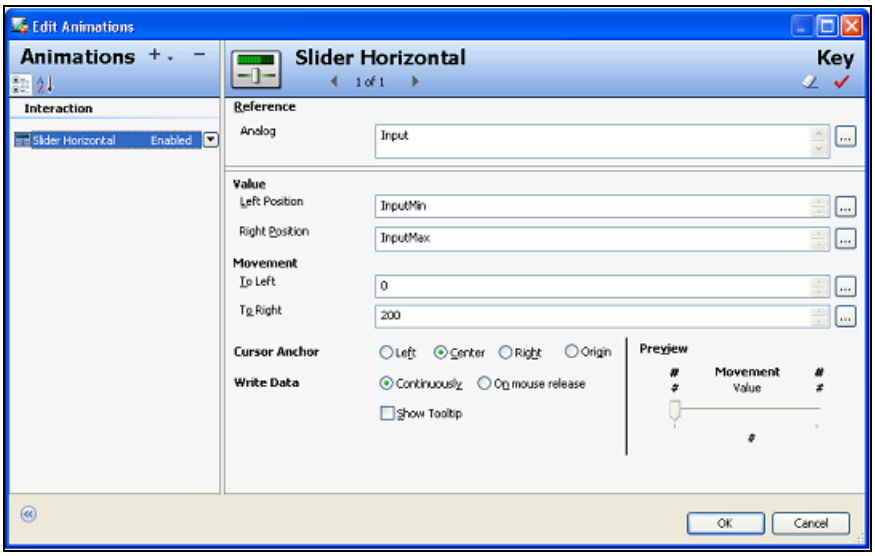
You can also provide a keypad for the user. When the user clicks the object in runtime, the input **Keyboard** is presented. A numeric keypad will display if the reference is **Analog**.



Slider Horizontal and Slider Vertical

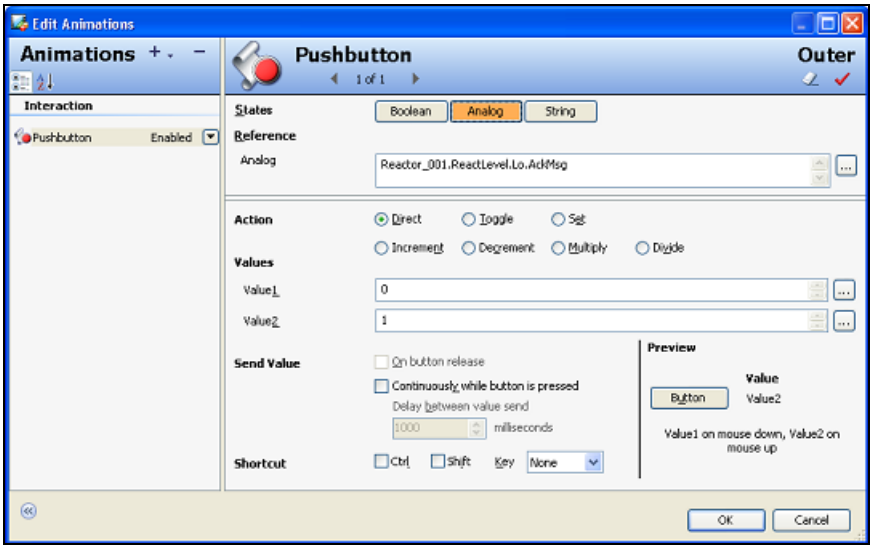
Use the **Slider Horizontal** animation to allow the runtime user to drag the element left or right, and write back the offset to an analog attribute or property.

Use the **Slider Vertical** animation to allow the runtime user to drag the element up or down, and write back the offset to an analog attribute or property.



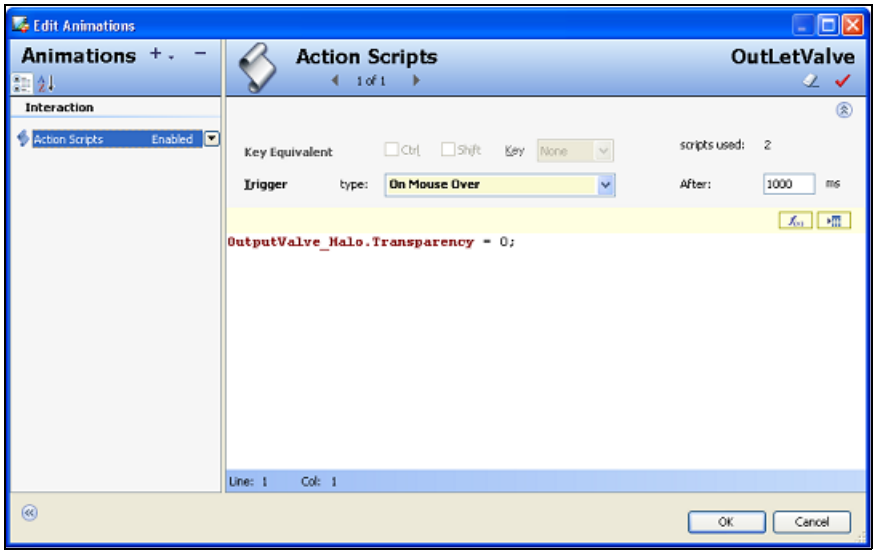
Pushbutton

Use the **Pushbutton** animation to write predetermined values to an **Analog**, **Boolean**, or **String** attribute or property when the runtime user clicks on the element.



Action Scripts

Use the Action Scripts animation to run an Action Script when the user performs the defined **Trigger** on the element at runtime (Click, Double-Click, Key Equivalent, etc.).



Configuring an Action Script Animation

You can configure an element with an Action Script animation.

You can assign multiple Action Scripts to one element to be activated in different ways, such as:

On Primary Click / Key Down	to activate the Action Script when the primary mouse button or a specific key is pressed.
While Primary Click / Key Down	to activate the Action Script when the primary mouse button or a specific key is pressed and held.
On Primary Click / Key Up	to activate the Action Script when the primary mouse button or a specific key is released.
On Primary Double Click	to activate the Action Script when the primary mouse button is double-clicked.
On Secondary Down	to activate the Action Script when the secondary mouse button is pressed.
While Secondary Down	to activate the Action Script while the secondary mouse button is pressed and held.
On Secondary Up	to activate the Action Script when the secondary mouse button is released.
On Secondary Double Click	to activate the Action Script when the secondary mouse button is double-clicked.
On Center Down	to activate the Action Script when the center mouse button is pressed.
While Center Down	to activate the Action Script when the center mouse button is pressed and held.
On Center Up	to activate the Action Script when the center mouse button is released.
On Center Double Click	to activate the Action Script when the center mouse button is double-clicked.
On Mouse Over	to activate the Action Script when the mouse pointer is moved over the element.
On Mouse Leave	to activate the Action Script when the mouse pointer is moved out of the element.
While Mouse Over	to activate the Action Script while the mouse pointer is over the element.

Note: To expand the available space for your script, you can use the expansion buttons  to hide the script header and/or the animation list.

Configuring a Show Symbol Animation

You can configure an element with a Show Symbol animation. A Show Symbol animation shows a specified symbol in a new window, when the element is clicked on.

You can configure:

- The symbol which appears (behaves like a window)
- If the window has a title bar, and if so, if it has a caption
- If the window is modal or modeless
- If the window has a close button
- If the window can be resized
- The initial window position
- The size of the window

Select the element to be configured with show symbol animation, on the **Special** menu, select **Edit Animations**. At the **Edit Animation** dialog box, select to **Show Symbol** animations. The Show Symbol animation is added to the animation list.

The Show Symbol animation configuration options are as follows:

Option	Purpose
Reference	Type a symbol name, or navigate to one using the **Browse** button.
Has Title Bar	Adds a title to the symbol.
Use Symbol Name for Window Title	Uses the symbol name as the window title.
Window Type	The options are **Modal** or **Modeless**.
Has Close Button	Adds a **Close** button to the symbol.
Resizeable	Adds resize controls to the symbol.
Position list	Select where you want the window to appear by selecting a position in the list. You can select one of the following: ● **Center Screen** to position the symbol in the center of the screen. ● **Top Left Mouse Position** places the symbol's top left corner at the mouse pointer's current location. ● **Top Right Mouse Position** places the symbol's top right corner at the mouse pointer's current location. ● **Bottom Left Mouse Position** places the symbol's bottom left corner at the mouse pointer's current location. ● **Bottom Right Mouse Position** places the symbol's bottom right corner at the mouse pointer's current location. ● **Center Mouse Position** places the symbol's center point on the mouse pointer. ● **Top Left X,Y Positio**n places the symbol's top left corner at a specific location. ● **Center X,Y Position** places the symbol's center point at a specified position. ● **Top of Screen** places the symbol at the top center of the screen. ● **Left Side of Screen** places the symbol at the left middle of the screen. ● **Right Side of Screen** places the symbol at the right middle of the screen. ● **Bottom of Screen** places the symbol at the bottom center of the screen. ● **Top Left Corner of Screen** places the symbol at the top left corner of the screen. ● **Top Right Corner of Screen** places the symbol at the top right corner of the screen. ● **Bottom Left Corner of Screen** places the symbol at the bottom left corner of the screen. ● **Bottom Right Corner of Screen** places the symbol at the bottom right corner of the screen.

Option	Purpose
Size list	Select how large you want the window to be in the **Size** list. You can select either one of the following: ● **Symbol Size** (1x) to make the window size the same as the size of the symbol. ● **Symbol Size plus 25%** (1.25x) to make the window size 25% larger than the size of the symbol. ● **Symbol Size plus 50%** (1.5x) to make the window size 50% larger than the size of the symbol. ● **Symbol Size plus 75%** (1.75x) to make the window size 75% larger than the size of the symbol. ● **Double Symbol Size** (2x) to make the window size twice the size of the symbol. ● **Triple Symbol Size** (3x) to make the window size three times the size of the symbol. ● **Quadruple Symbol Size** (4x) to make the window size four times the size of the symbol. ● **25% Screen Size** to make the window size cover 25% of the screen. ● **50% Screen Size** to make the window size cover 50% of the screen. ● **75% Screen Size** to make the window size cover 75% of the screen. ● **Full Screen Size** to make the window size cover the entire screen. ● **Custom Width and Height** to specify a width and height.
Stretch symbol to fit window size	Select if you want the window to be stretched to fit the window size.
Stretch window to screen width	Select if you want the window to be stretched to fit the screen height. If the size is Custom Width and Height, type in a value for the in the **H** box.
Stretch window to screen height	Select if you want the window to be stretched to fit the screen height. If the size is Custom Width and Height, type in a value for the width in the **W** box.
Shortcut area	If you want the symbol window to appear by pressing a key to key combination, select a shortcut key in the **Key** list, and **Ctrl** and/or **Shift** to combine the shortcut key.

Configuring a Hide Symbol Animation

You can configure an element with a Hide Symbol animation. The Hide Symbol animation lets you either:

- Close a symbol that is shown by a specified element
- Configure an element with a Hide Symbol animation

To configure an element for Hide Symbol animation, select the element you want to configure. On the **Special** menu, select **Edit Animations** to open the **Edit Animations** dialog box. Use the **Add** button and then select the **Hide Symbol** animation from the list.

At the **Hide Symbol** configuration panel, select either one of the following:

- **Current Symbol**, if you want to close the currently shown symbol or window
- **Symbol shown by an element**, if you want to close a symbol that is shown by that element (type the element name in the adjacent box)

If you want the symbol window to be hidden by pressing a key or key combination, in the **Shortcut** area select a shortcut key in the **Key** list, and select **Ctrl** and/or **Shift** to make the shortcut a combination of keys.

Element-Specific Animations

Some elements have their own unique animation type that can only be used for that element type. You cannot remove their unique animation, but depending on the element you can add and remove other common animations.

The elements with specific animations are:

- **Status Graphic**
- Windows **Common Controls**, such as Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, and List Box Client Controls (more on this later)

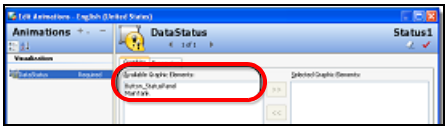
Configuring a Status Graphic Animation

You can configure the Status Graphic element to indicate quality and status from:

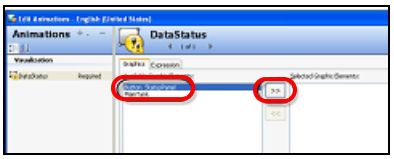
- ArchestrA attributes used in elements with animation.
- ArchestrA attributes directly.

The appearance of the Status Graphic element depends on the settings in the **Configure Quality and Status Display** dialog box.

To start, select the element(s) that you want to configure and use **Special | Edit Animations** to display the **Configure Quality** and **Status Display** dialog box:



From the **Available Graphic Elements** list, highlight the element(s) to be configured to monitor quality and status. Notice that, upon selecting an element(s), the **Move Data to the Right** >> button becomes enabled:



- Click >> button to add the selected elements to the **Selected Graphic Elements** list. Notice that now both directional move buttons are enabled.
- Click << button to remove any elements you do not want to configure.

Select the **Expression** tab to display the **Value or Expression** list.

In the **Value or Expression** area, enter a value or expression. It can be either literal, or a reference, or an element property.

Add a Value or Expression

From the **Expression** toolbox, select **Add** (as illustrated below). When a new row appears, do one of the following:

- Enter the value, reference, or animation link in the row

  OR

- In the **Galaxy Browser**, use **Browse** to display the ABGalaxy's attribute and element dialog box, navigate to the appropriate reference, value, or animation link to select it.



You can configure a **Windows Common Control** to write the data either:

- Immediately when it is selected in the control at runtime.
- When a specified **Boolean** expression becomes true.

The Boolean expression is a trigger that determines when the value is written from the control to the tag or attribute. If the value changes in the tag or attribute, then the value is written to the control, regardless of the trigger setting or condition.

You can set the **Reference** of the Animation to the **Symbol StatusPanel**.

Radio Button Group

The Radio Button Group animation is only used by the **Radio Button Group** element.

You can either create a:

- **Static** radio button group - uses static captions and values that you define in the configuration panel.
- **Array** radio button group - uses captions and values that are contained in an Automation Object array. Used with ArchestrA Object, not supported by Tag-Based InTouch.
- **Enum** radio button group - uses captions and values that are contained in an enum data type of an Automation Object. Used with ArchestrA Object, not supported by Tag-Based InTouch.

Check Box

The Check Box animation is only used by the **Check Box** element. You can set the caption of the check box at runtime, with the **Override Caption at Runtim**e.

Combo Box

The **Combo Box** animation is only used by the **Combo Box** element. You can use Combo Box-specific methods in scripting to perform various functions at runtime. You can browse these methods in the Galaxy Browser with the Combo Box selected.

You can either create a:

- **Static Combo Box** - uses static captions and values that you define in the configuration panel.
- **Array Combo Box** - uses captions and values that are contained in an AutomationObject array. Used with ArchestrA objects, not supported by Tag-Based InTouch.
- **Enum Combo Box** - uses captions and values that are contained in an enum data type of an AutomationObject. Used with ArchestrA objects, not supported by Tag-Based InTouch.

Edit Box

The Edit Box animation is only used by the **Edit Box** element. You can set the caption of the check box at runtime, with the **Override Caption at Runtime**.

You can determine the configuration of the Edit Box:

- **Multiline** to wrap the text in the Edit Box.
- **Read-Only** to use the edit box to only show text and not allow text input.
- **Maximum Length** to limit the maximum numbers of characters you can type in the Edit Box control. You can specify the maximum number in the **Characters** box.

Calendar Control

The Calendar Control animation is only used by the **Calendar Control** element. The Calendar Control date format depends on the regional settings of the operating system. The Calendar Control is used to browse and select a date. You can adjust the available date range and bold special dates. Bolded dates use an **Array** data type supported by the ArchestrA System Platform for objects and not supported by Tag-Based InTouch.

DateTime Picker

The **DateTime Picker** animation is only used by the **DateTime Picker** element. The DateTime Picker appears as a single-line drop-down control taking less space than the Calendar Control. A calendar selector appears when the user clicks the control, and hides again after the user selects a date. Unlike the Calendar Control, the DateTime Picker may have its width resized and provides greater flexibility in date formatting. It also adds a time component.

List Box

The List Box animation is only used by the **List Box** element.

You can either create a:

- **Static List Box** - uses static captions and values you define in the configuration panel.
- **Array List Box** - uses captions and values contained in an AutomationObject array. Used with ArchestrA objects, not supported by Tag-Based InTouch.
- **Enum List Box** - uses captions and values contained in an enum data type of an AutomationObject. Used with ArchestrA objects, not supported by Tag-Based InTouch.

You can also use List Box-specific methods in scripting to perform various functions at run time. You can browse these methods in the **Galaxy Browser** with the **List Box** selected.

Lab 8 – Animating Symbols

Introduction

In this lab, you will create a custom animation and animate the Tank cutaway to show the fill level. You will also create custom properties and then associate these properties with graphic elements.

Objectives

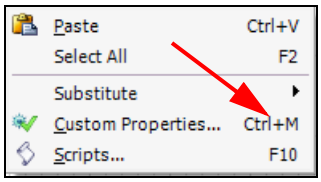
Upon completion of this lab, you will be able to:

- Add custom properties to a symbol
- Add animations to an element
- Use the Color Picker for color selection

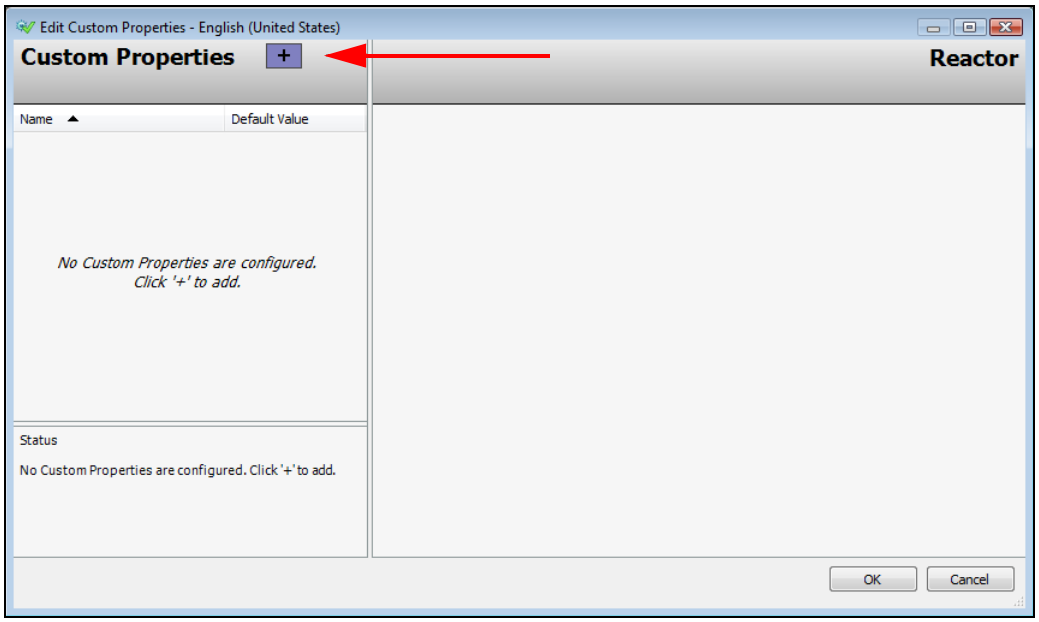
Create Custom Properties

Now you will create new custom properties for the Reactor symbol.

1. Return to **ArchestrA Symbol Editor**.
2. On the **Reactor** canvas, right-click on a blank area and select **Custom Properties**.

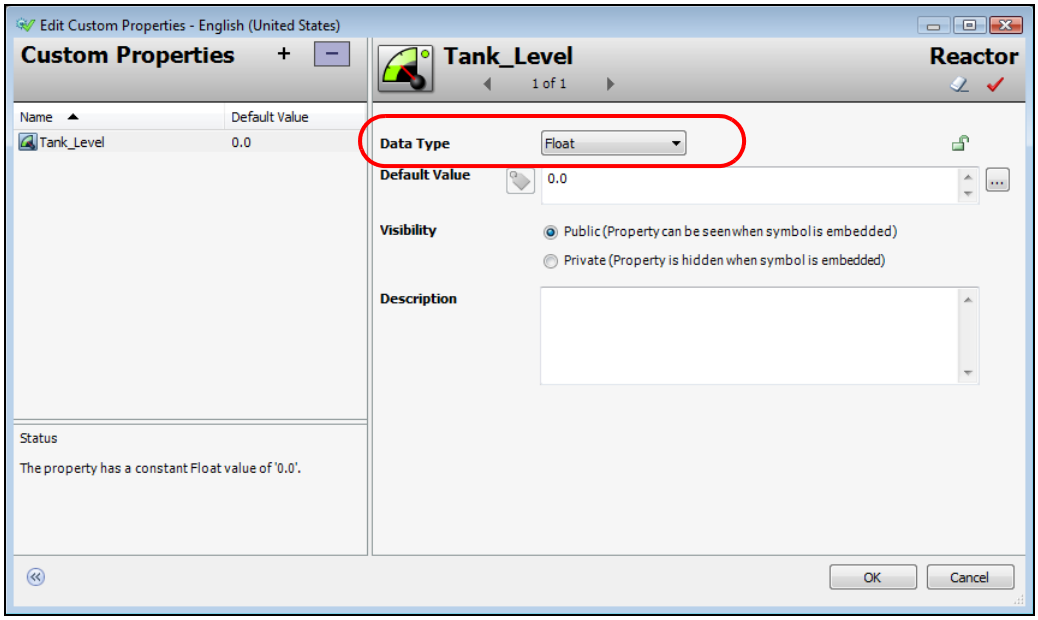


3. Click **+** to add a **Custom Property**.



4. In the **Name** field, enter **Tank_Level**.

5. From the **Data Type** drop-down, select **Float**.



By default, the **Default Value** is set to **0**, and the **Visibility** is **Public**.

6. Add the following **Custom Properties**:

Custom Property	Data Type
Tank_Level_Max	Float
Tank_Level_Min	Float

7. Click **OK**.

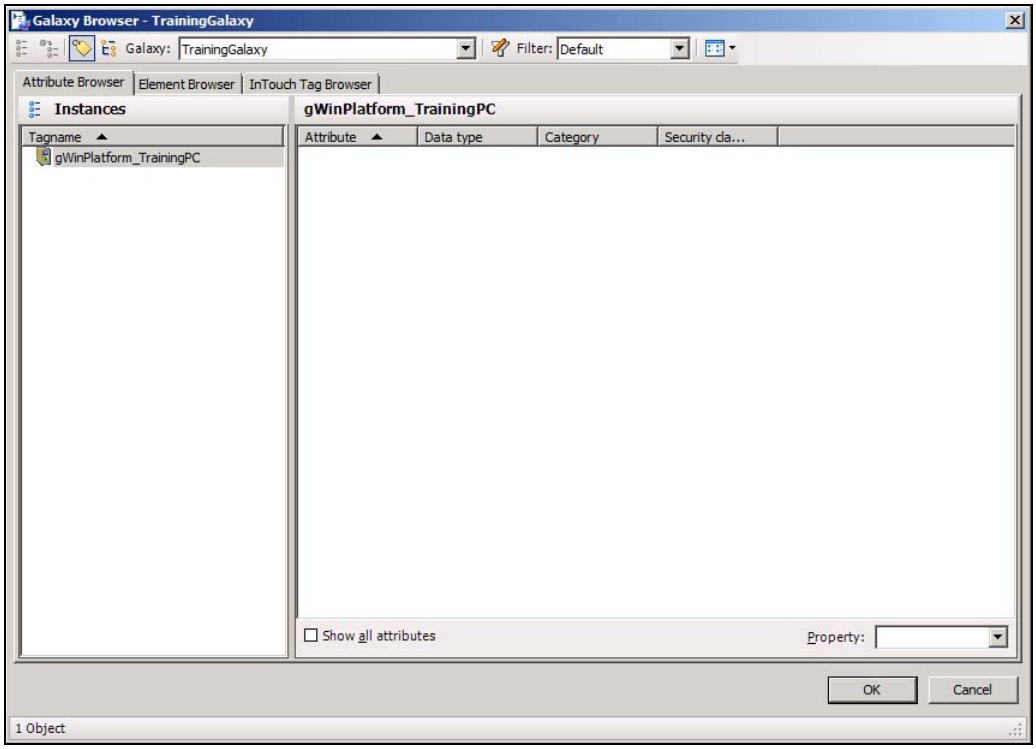
8. In **Elements** pane, right-click **LevelDisplay** and select **Custom Properties**.

   The **Edit Custom Properties** dialog box opens with the **Value** property selected.

9. In the **Default Value** field, remove **---**.

10. To the right of the **Default Value** field, click the **Browse Galaxy** button [...].

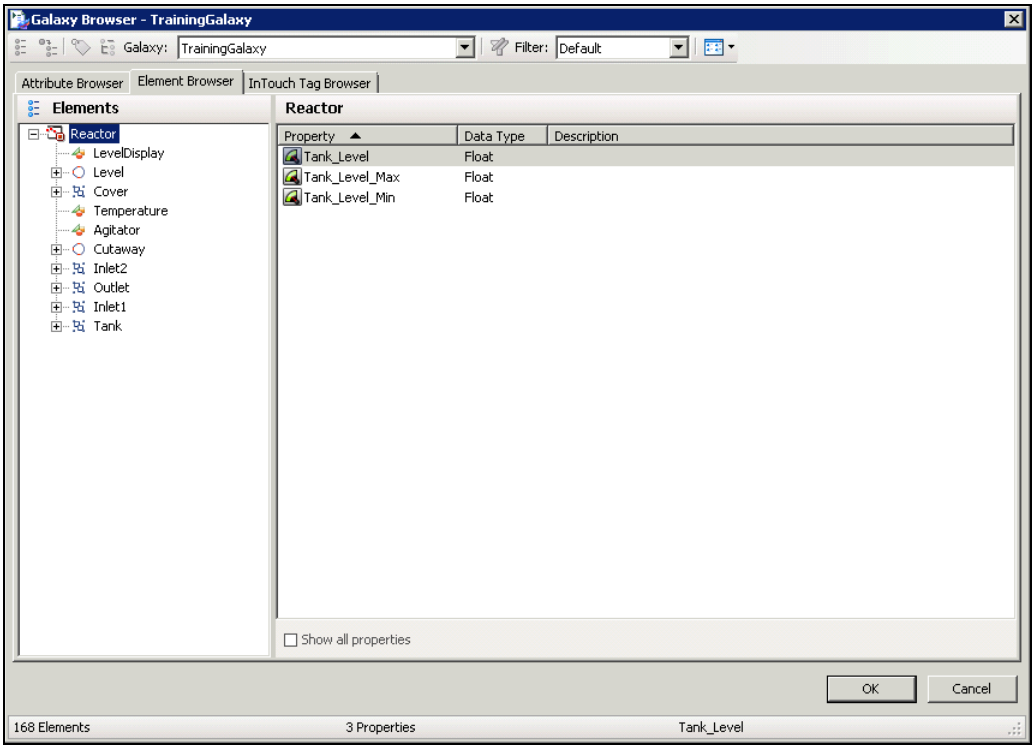
    The **Galaxy Browser** appears.



11. Click the **Element Browser** tab.

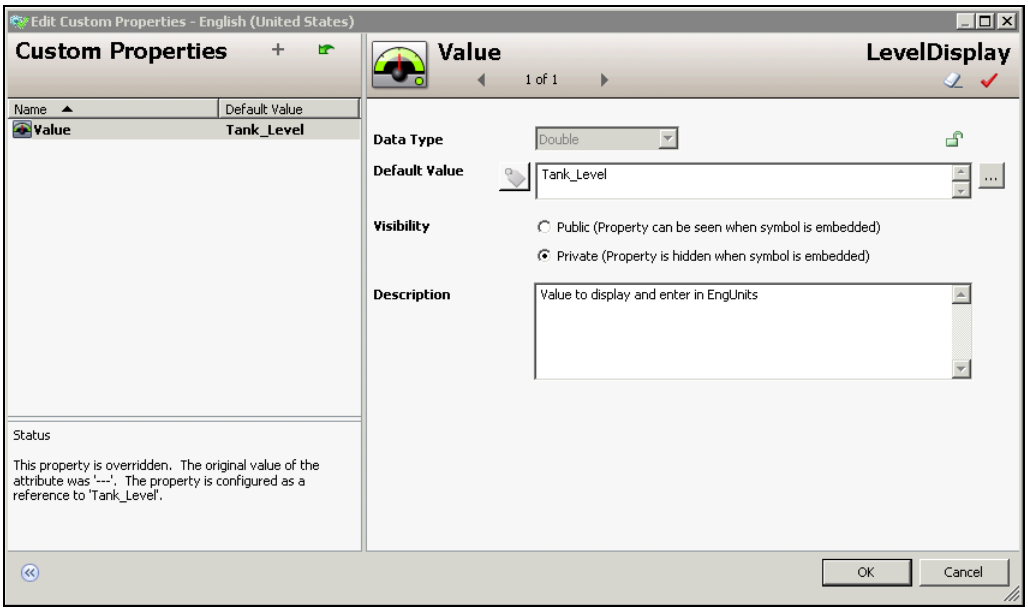
12. In **Elements** pane, click **Reactor** to display the list of custom properties.

13. In **Reactor** pane, click **Tank_Level**.



14. Click **OK**.

15. In the **Edit Custom Properties** dialog box, set **Visibility** to **Private**.

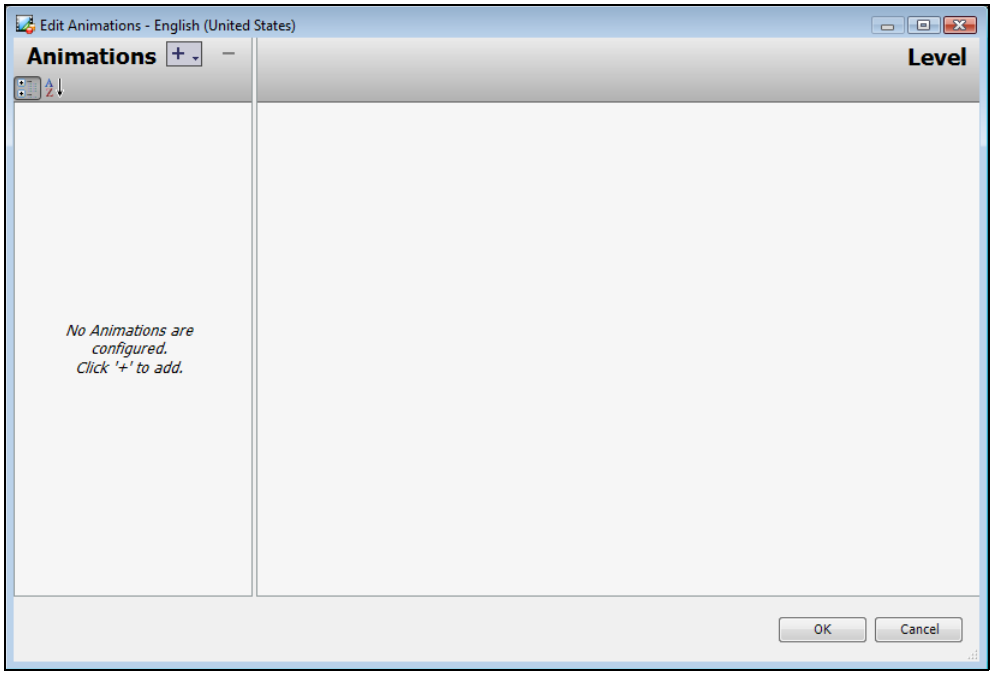


16. Click **OK**.

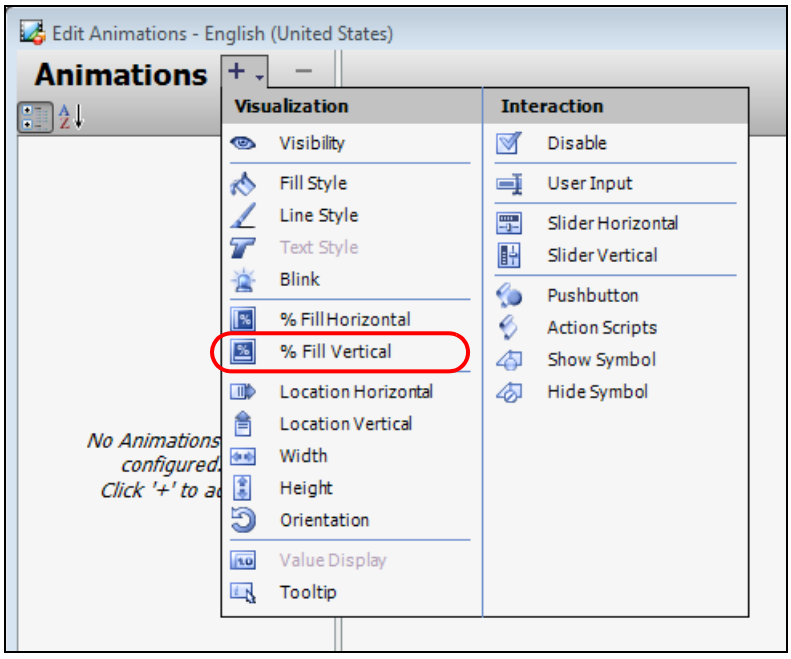
Add Animations to Level

Next, you will add fill animations to graphic elements.

17. In **Elements** pane, double-click **Level** to open the **Edit Animations** dialog box.



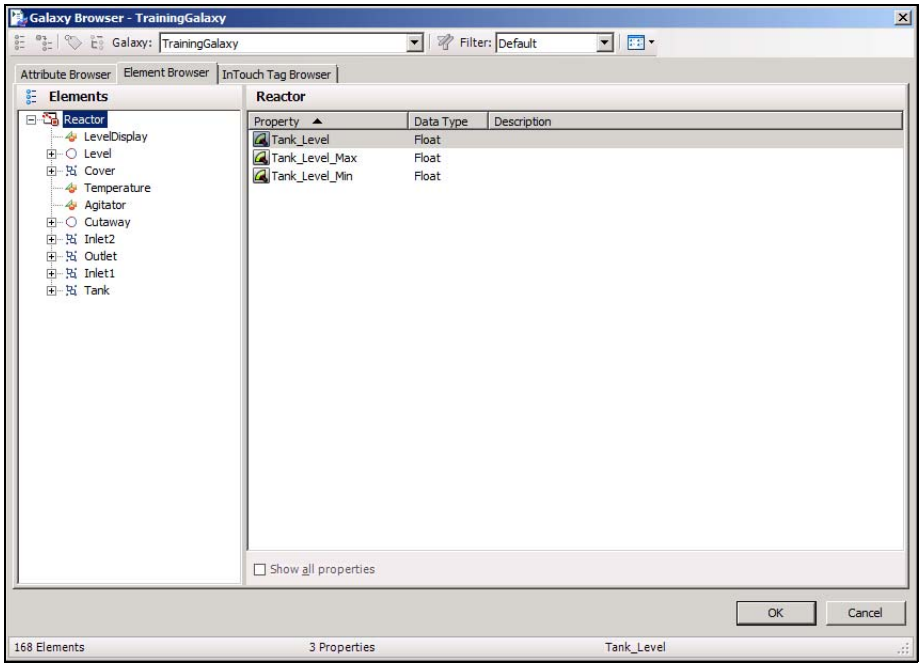
18. Click **+** and select **% Fill Vertical**.



19. In **Visualization** pane, click **% Fill Vertical**.
20. In the **Expression Or Reference** area, double-click the **Analog** field.

    The **Galaxy Browser** appears.

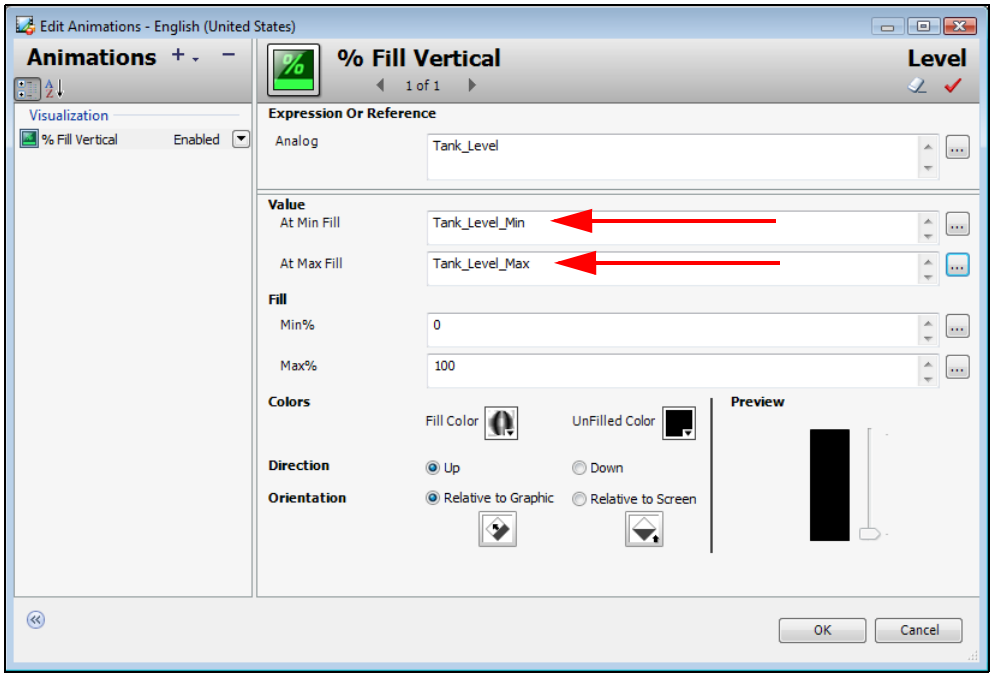
21. On the **Element Browser** tab, in **Elements** pane, click **Reactor** to display the list of custom properties.
22. In **Reactor** pane, click **Tank_Level**.



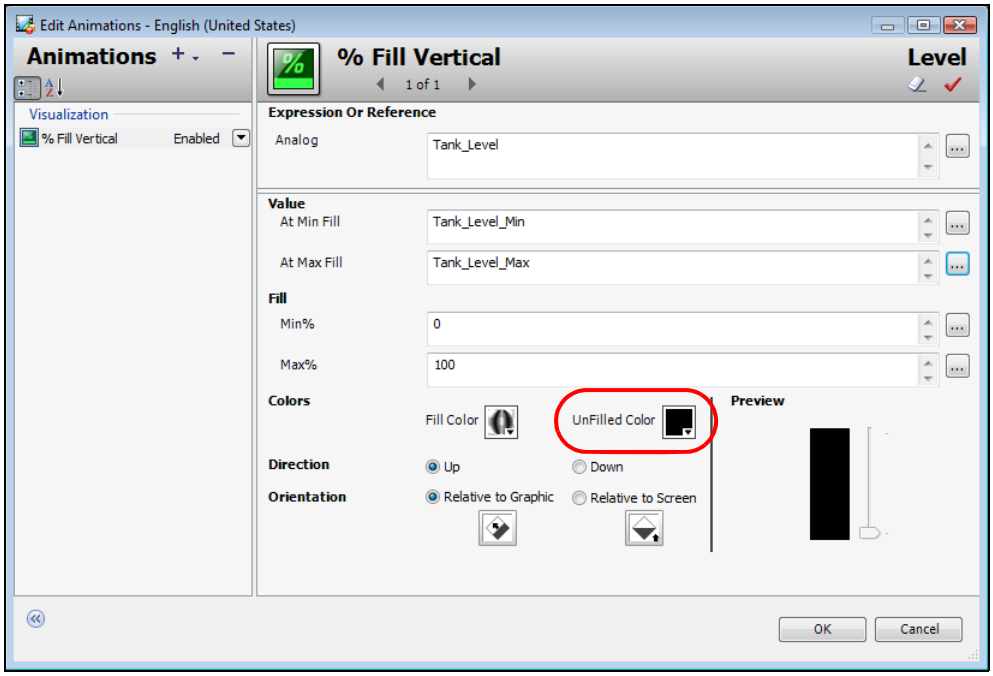
23. Click **OK** to return to the **Edit Animations** window.
24. Repeat the previous steps to configure the **Value** fields as follows:

At Min Fill: Tank_Level_Min

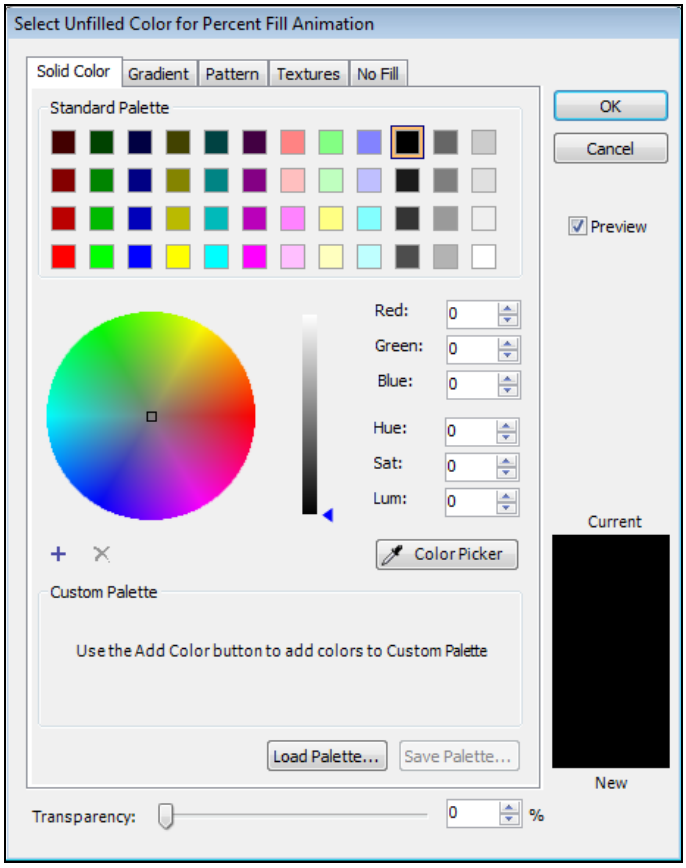
At Max Fill: Tank_Level_Max



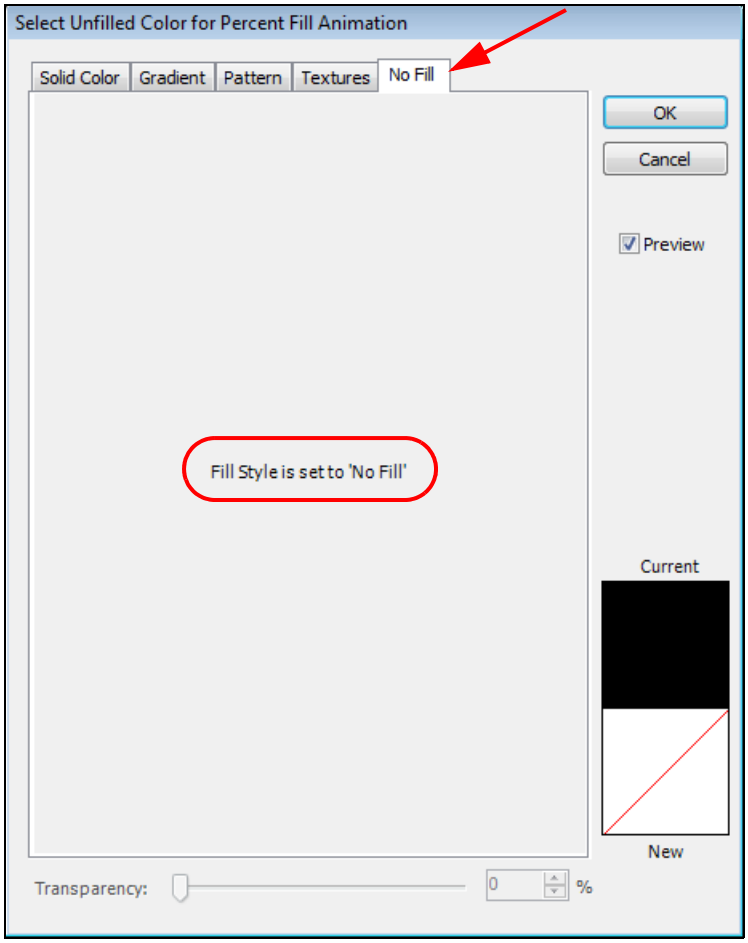
25. Click **UnFilled Color**.



The **Select Unfilled Color for Percent Fill Animation** dialog box appears.



26. Click the **No Fill** tab.

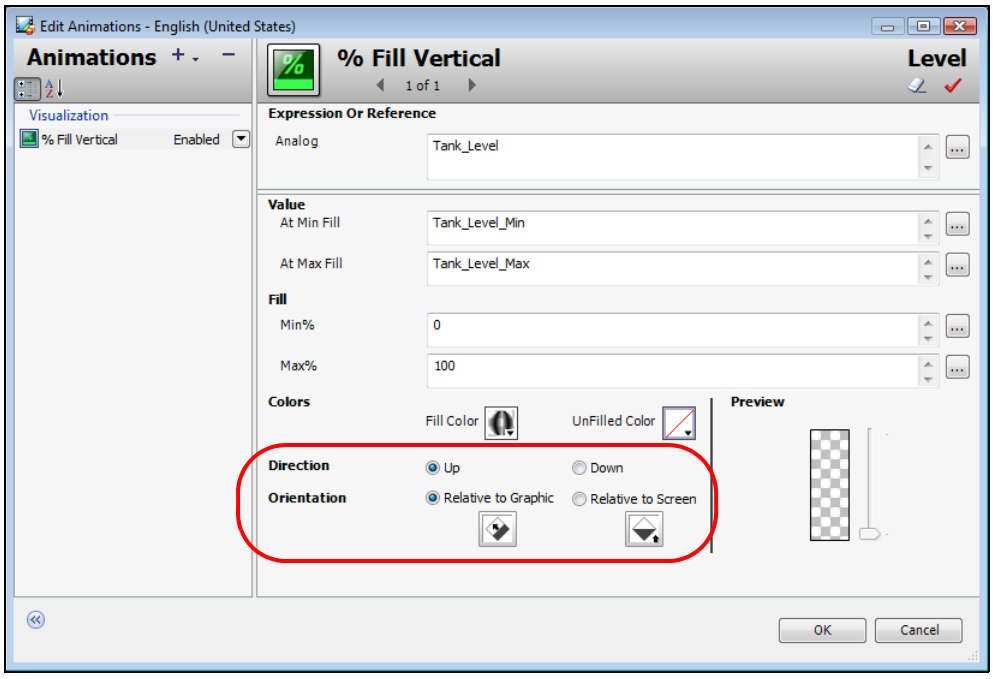


27. Click **OK**.

28. Ensure **Direction** is set to **Up**.

    This setting directs the fill animation to start at the bottom of the **Cutaway** and travel to the top.

29. Ensure **Orientation** is set to **Relative to Graphic**.

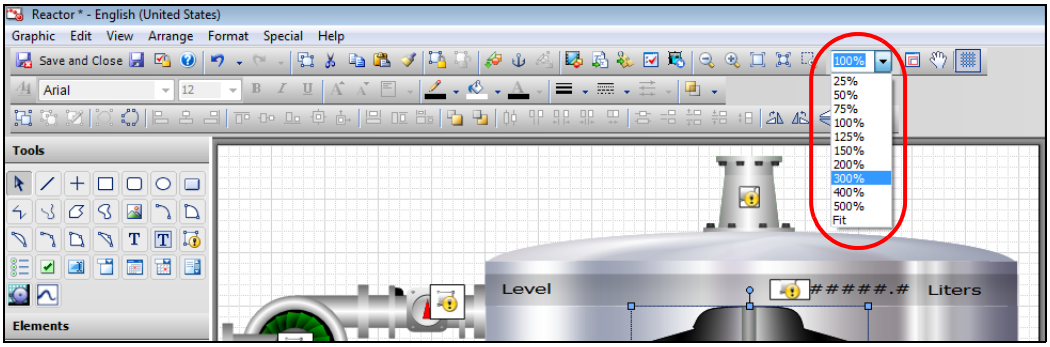


    The **Relative to Graphic** option sets the orientation of the fill perpendicular to the object angle. The **Relative to Screen** option sets the orientation of the fill parallel to the screen.

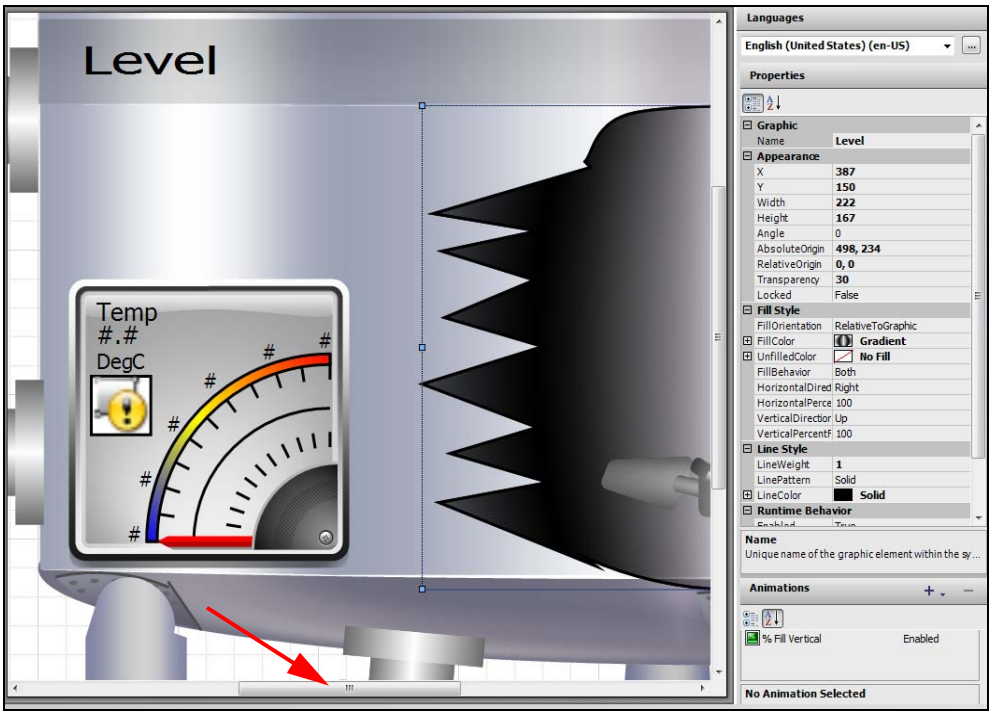
30. Click **OK**.

Next, add an animation to your cutaway based on the level in the tank to show a color gradient as the tank fills and empties, so the level graphic also doubles a temperature indicator. You will use four colors selected directly from the Temperature graphic.

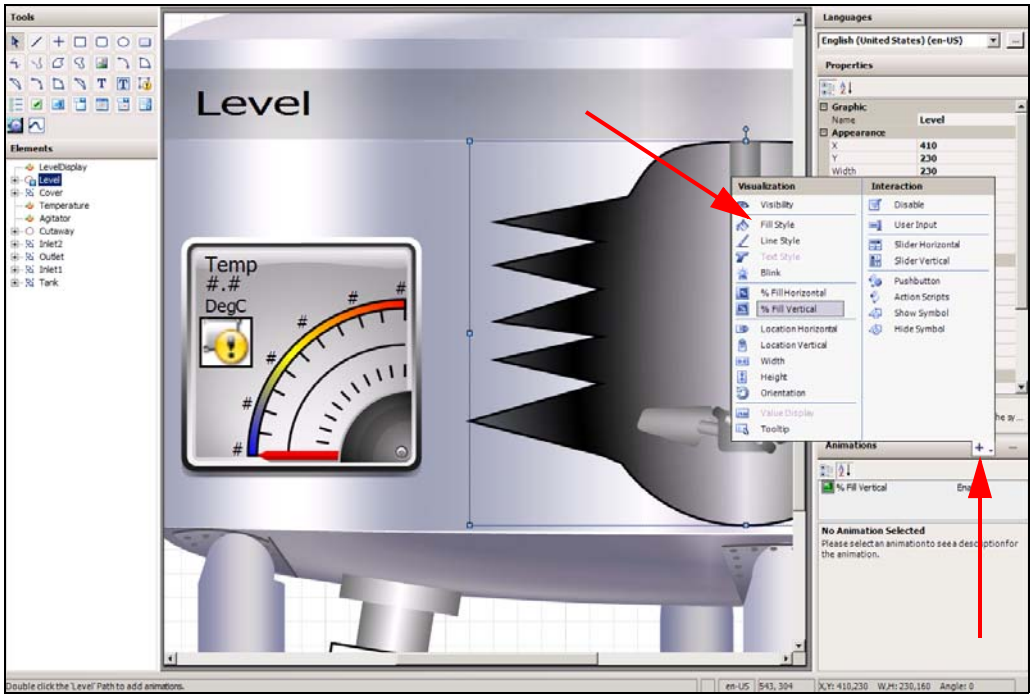
31. On the toolbar, click **Zoom** and select **300%**.



32. Scroll until you can see the **Temperature** element on the screen.

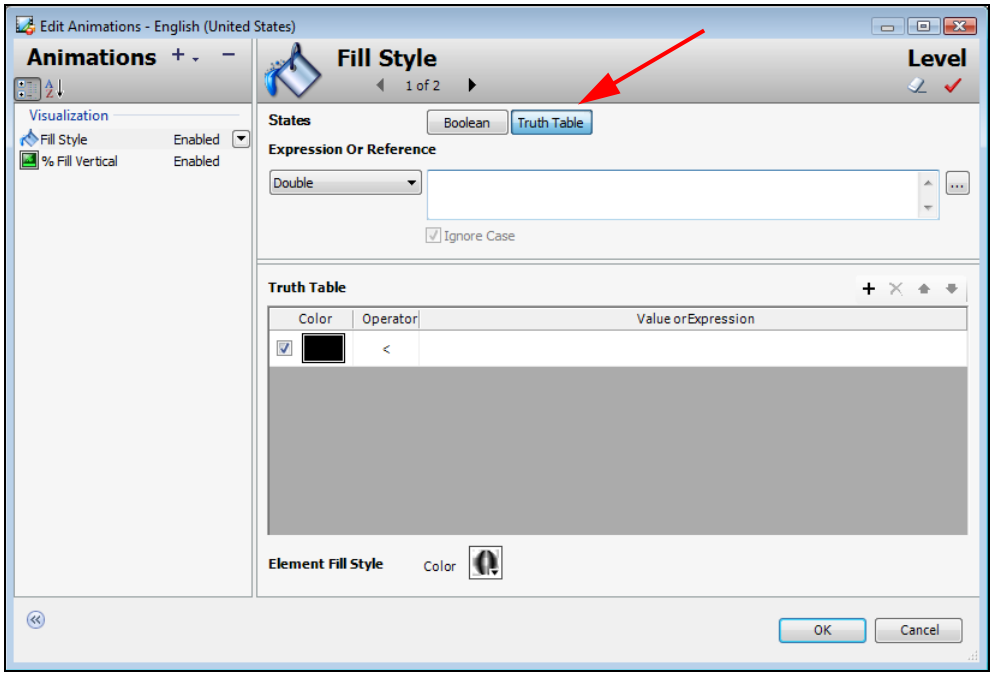


33. In **Elements** pane, select **Level**, then in the **Animations** pane click **+** and select **Fill Style**.

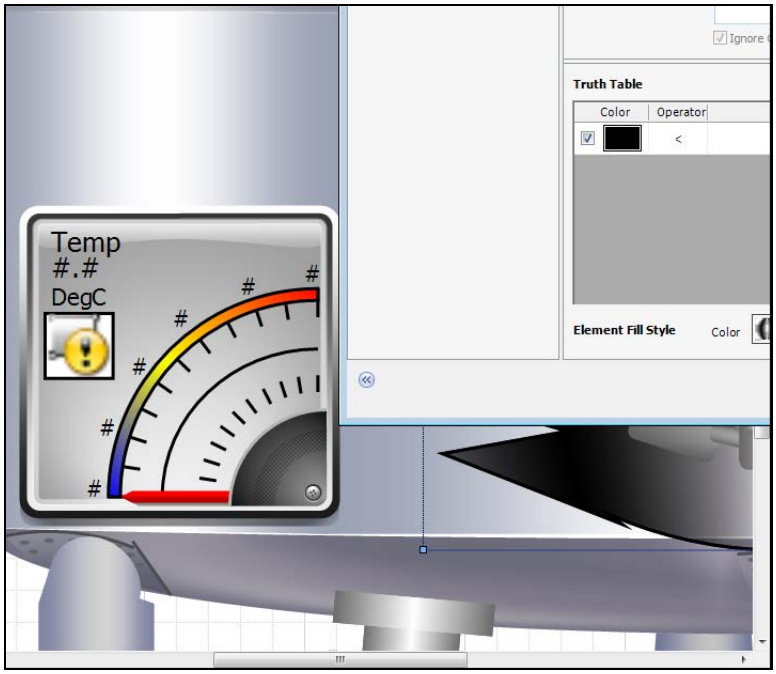


The **Edit Animations** dialog box appears.

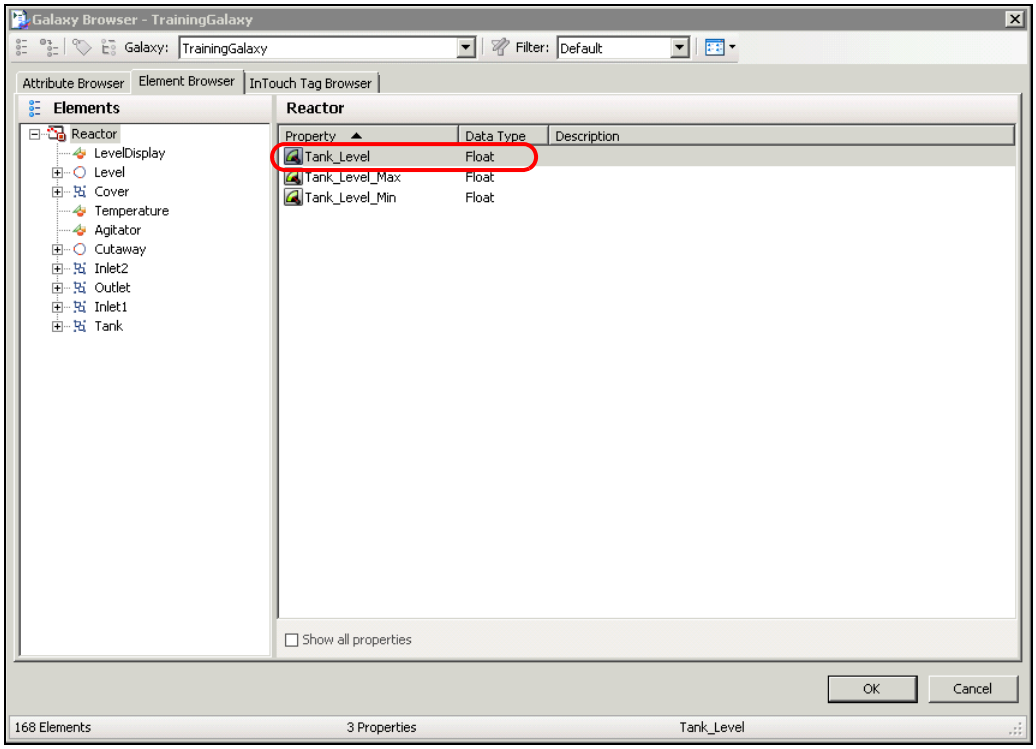
34. Click **Fill Style**, then click **Truth Table** to define a range of fill color values.



35. Move the **Edit Animations** dialog box to reveal the **Temperature** element.

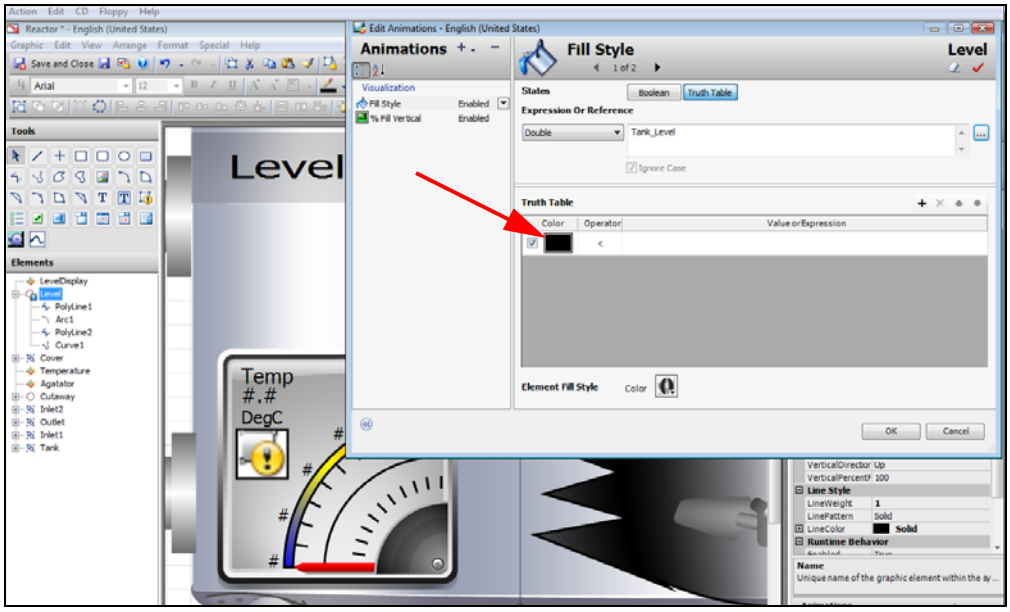


36. In the **Expression Or Reference** field, double-click to open the **Galaxy Browser**.
37. Click the **Element Browser** tab.
38. In the **Elements** list, click **Reactor**.
39. In **Reactor** pane, double-click **Tank_Level**.



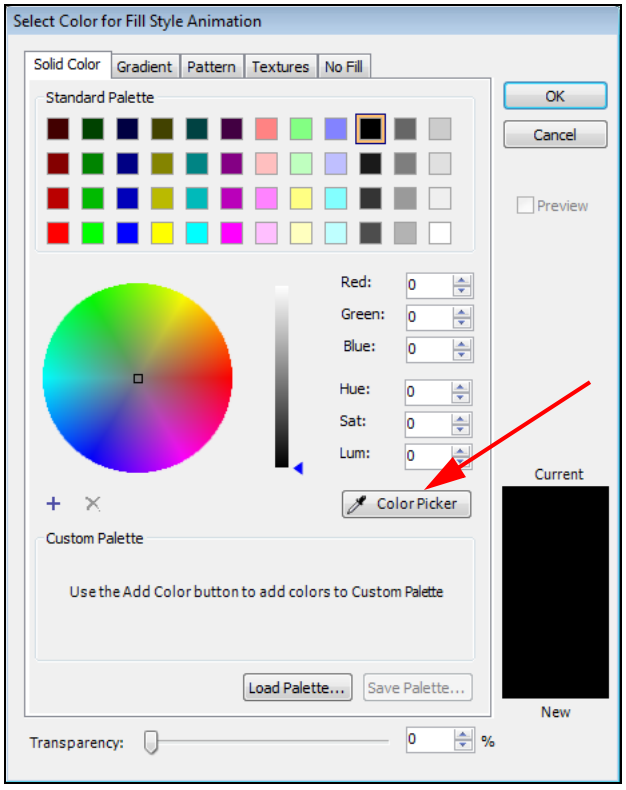
Now you will build a Truth Table of colors based on a range of Tank_Level expressions.

40. In **Truth Table** (lower pane), click **Color**.



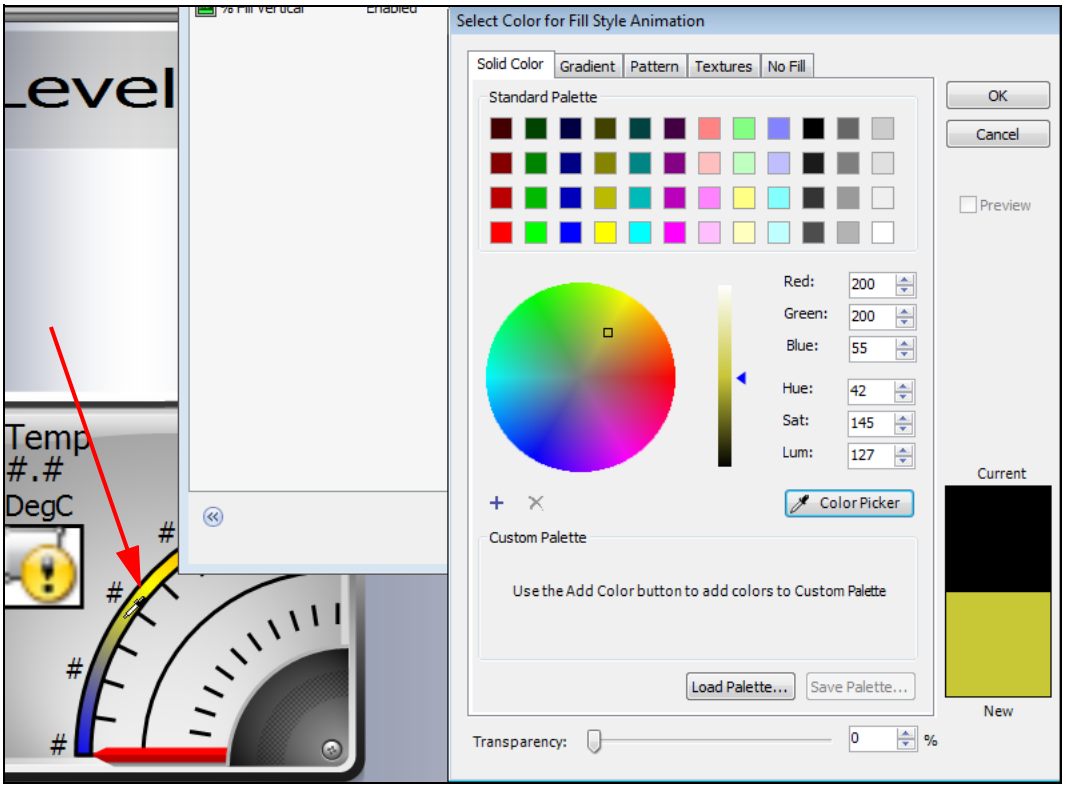
The **Select Color for Fill Style Animation** dialog box appears.

41. Click **Color Picker**.



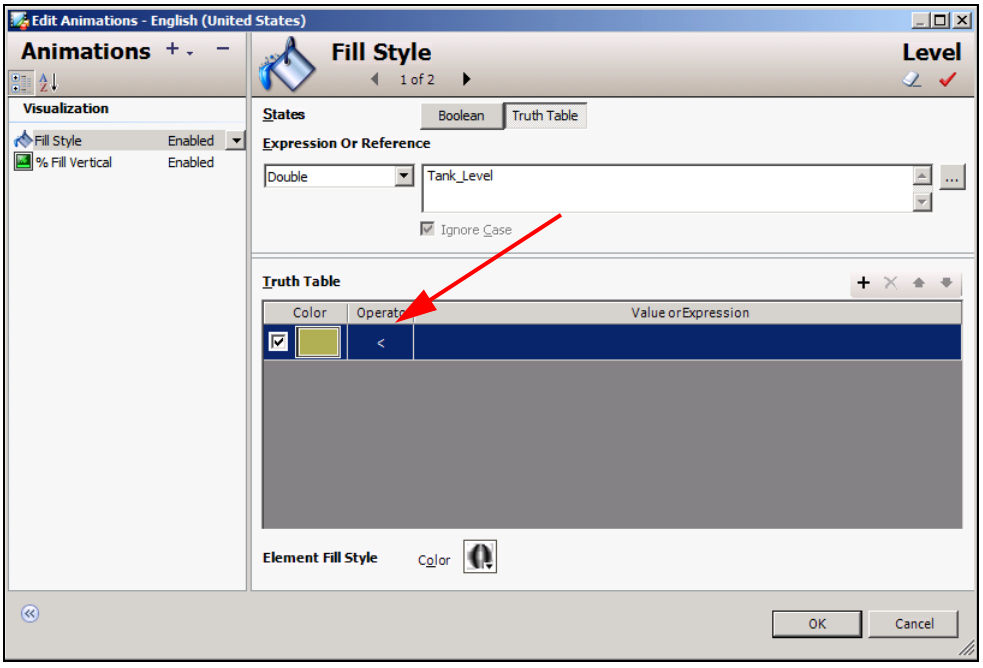
42. Click the yellow-green area on the **Temperature** symbol.

    You may have to move the overlapping dialog boxes.

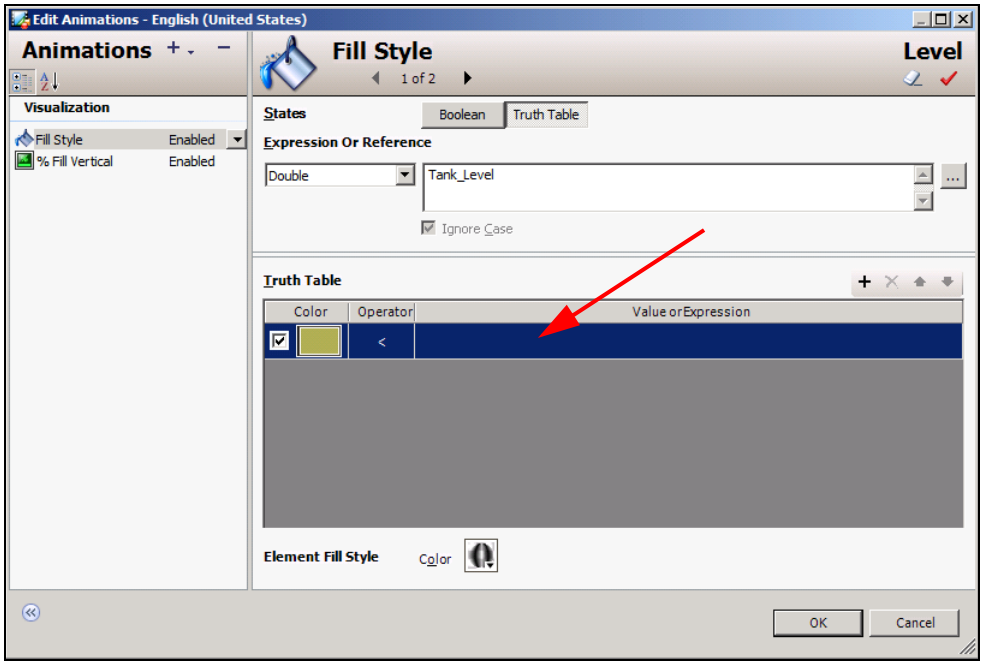


43. Click **OK**.

44. In the **Operator** column, verify that **<** (Less Than) is selected.

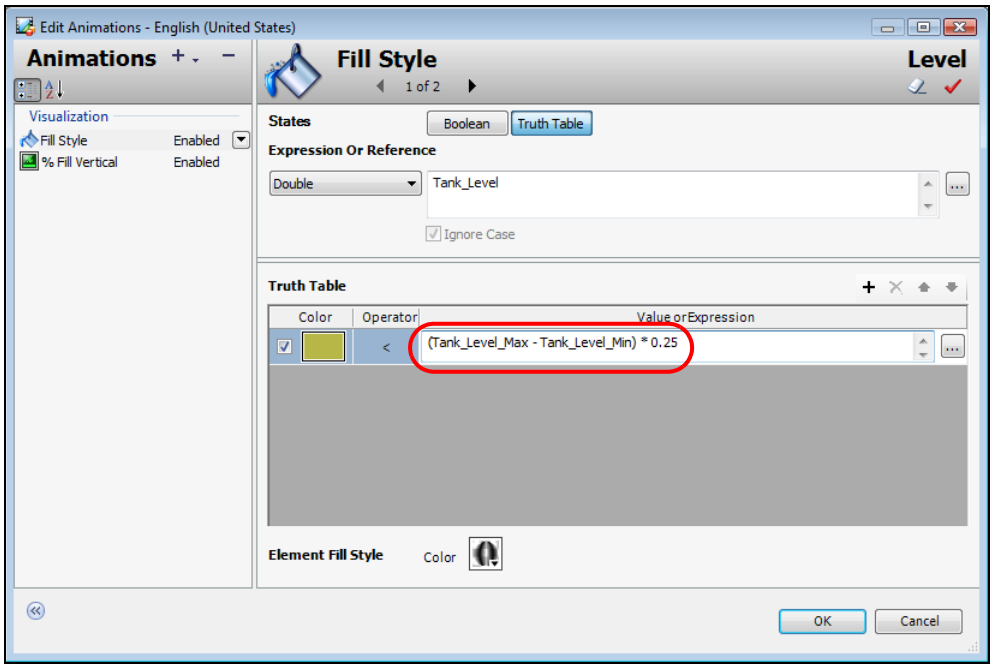


45. Click the area directly below the **Value or Expression** column to place your cursor in the expression field.



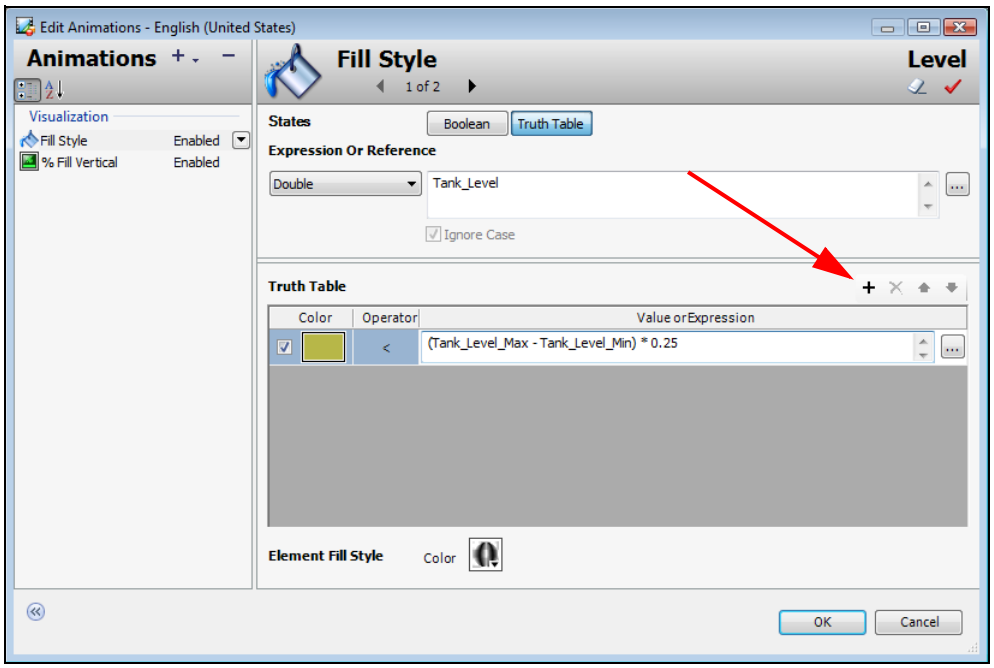
46. In the **Value or Expression** field, complete the expression as shown below:

(Tank_Level_Max – Tank_Level_Min) * 0.25

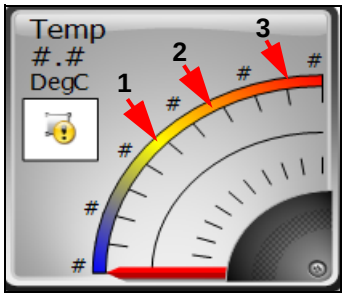


Note: The parentheses guarantees the order of mathematical operations in an expression. First, **Tank_Level_Min** is subtracted (**-**) from **Tank_Level_Max**, then the result is multiplied (*****) times **0.25**. This will make the **Level** element animate a yellow-green color when the level in the tank is in the bottom **25%**.

47. In the **Truth Table** area, click **+** to add an additional row to the truth table..

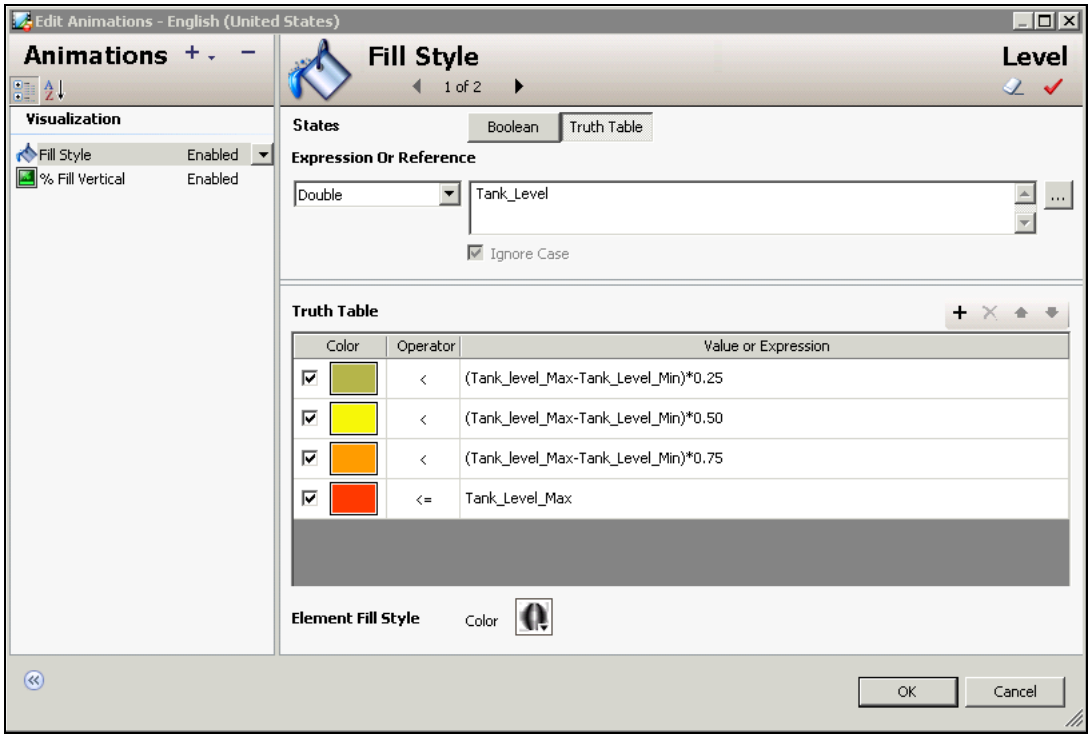


48. Use the **Color Picker** to select the colors indicated in the table below, and configure the operator and expression for each row added to the **Fill Style Truth Table** as indicated below.

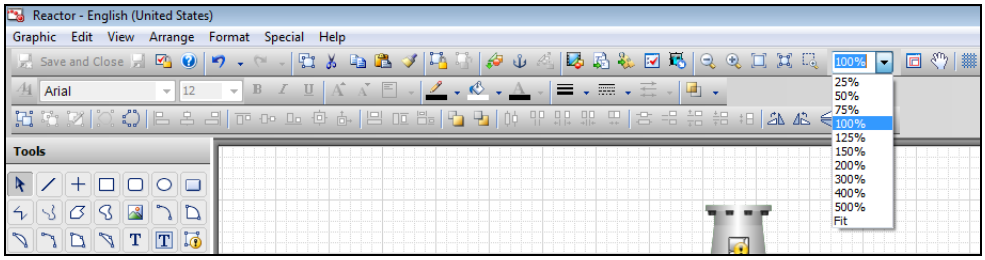


Color	Operator	Expression or Value
Yellow (1)	<	(Tank_Level_Max - Tank_Level_Min) * 0.50
Orange (2)	<	(Tank_Level_Max - Tank_Level_Min) * 0.75
Red (3)	<=	Tank_Level_Max

The completed **Fill Style** animations look like this.



49. Click **OK**.
50. On the toolbar, click **Zoom** and select **100%**.

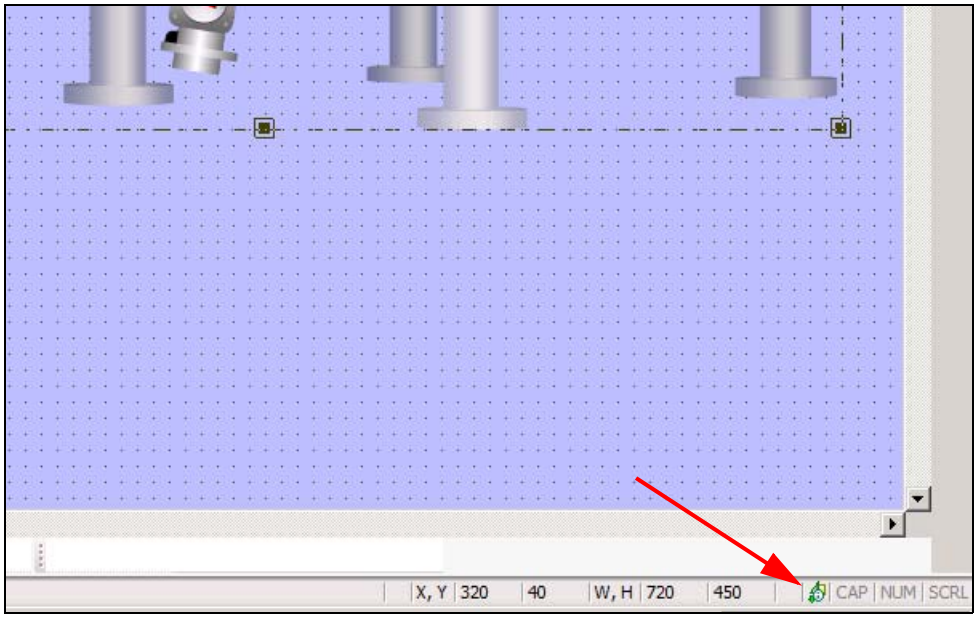


51. Click **Save** .

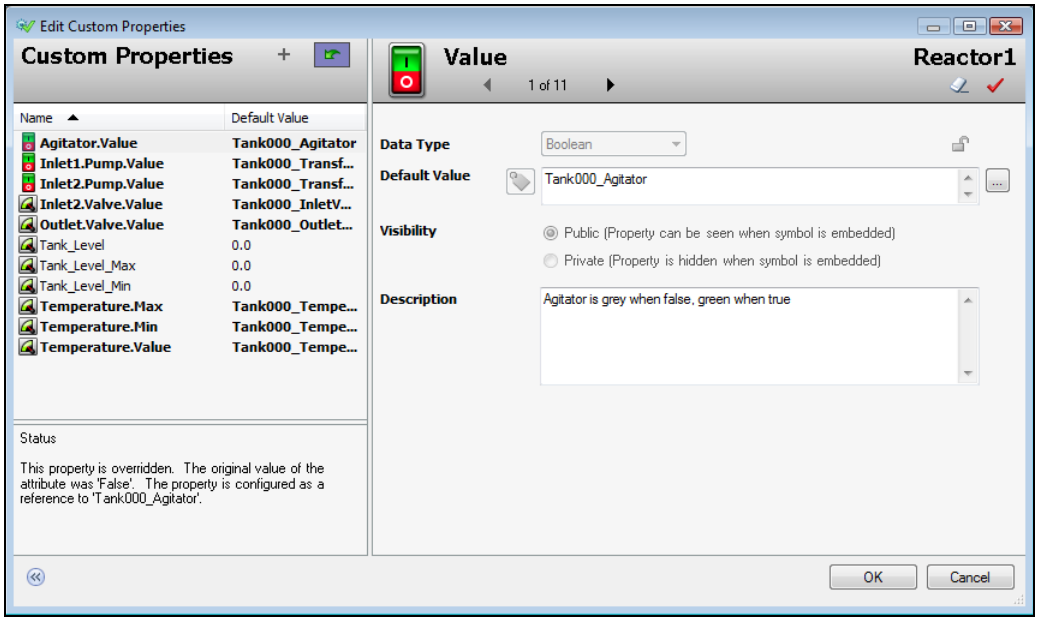
Link Properties to Tags

Now you will link custom properties to tags you previously set up in WindowMaker.

52. In WindowMaker, double-click the **Refresh Embedded Symbols** icon in the status bar.



53. In the **Mixer** window, double-click the **Reactor** symbol to open the **Edit Custom Properties** dialog box.



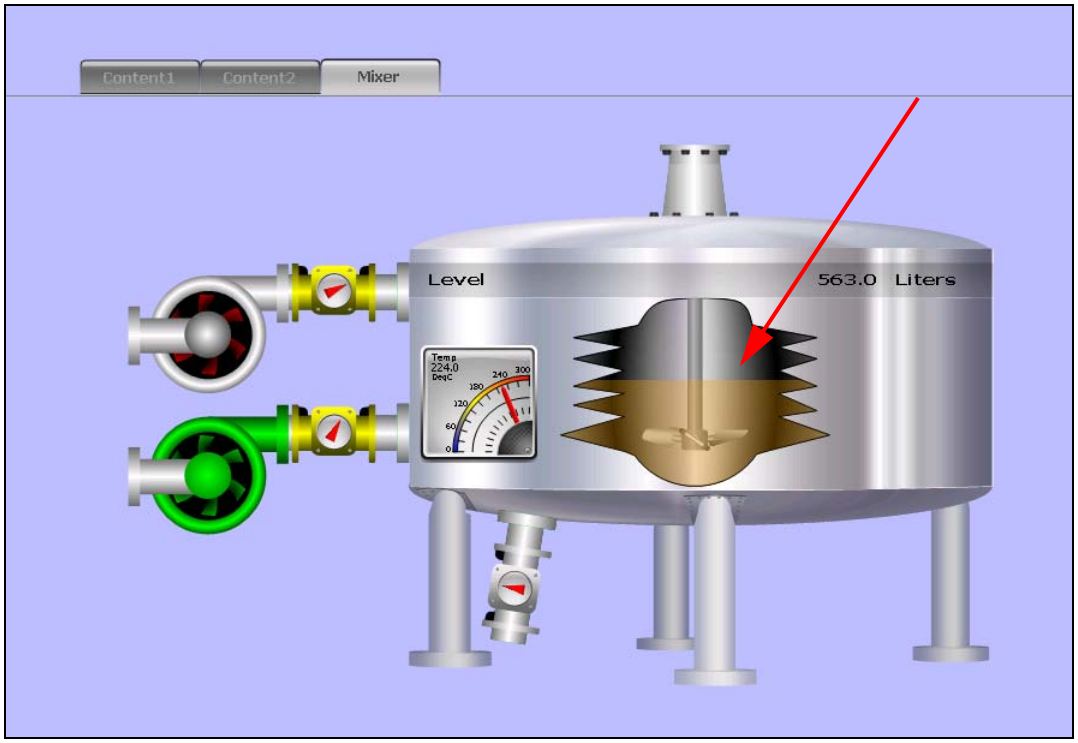
54. For each **Custom Property** in the table below, link the **Default Value** to the **Tagname** and **Dot Field** as follows:

Custom Property	Default Value
Tank_Level	Tank000_Level
Tank_Level_Max	Tank000_Level.MaxEU
Tank_Level_Min	Tank000_Level.MinEU

55. Click **OK**.

Test What You Have Built

56. Switch to **Runtime** and open the **Mixer** window.



Notice the color changes as the **Tank** fills. Also notice that the **Agitator** can be viewed through the transparency.

57. Click **Development!** to return to WindowMaker.

Section 5 – ArchestrA Symbol Status and Quality

This section describes methods for monitoring the graphical representation of quality and data status.

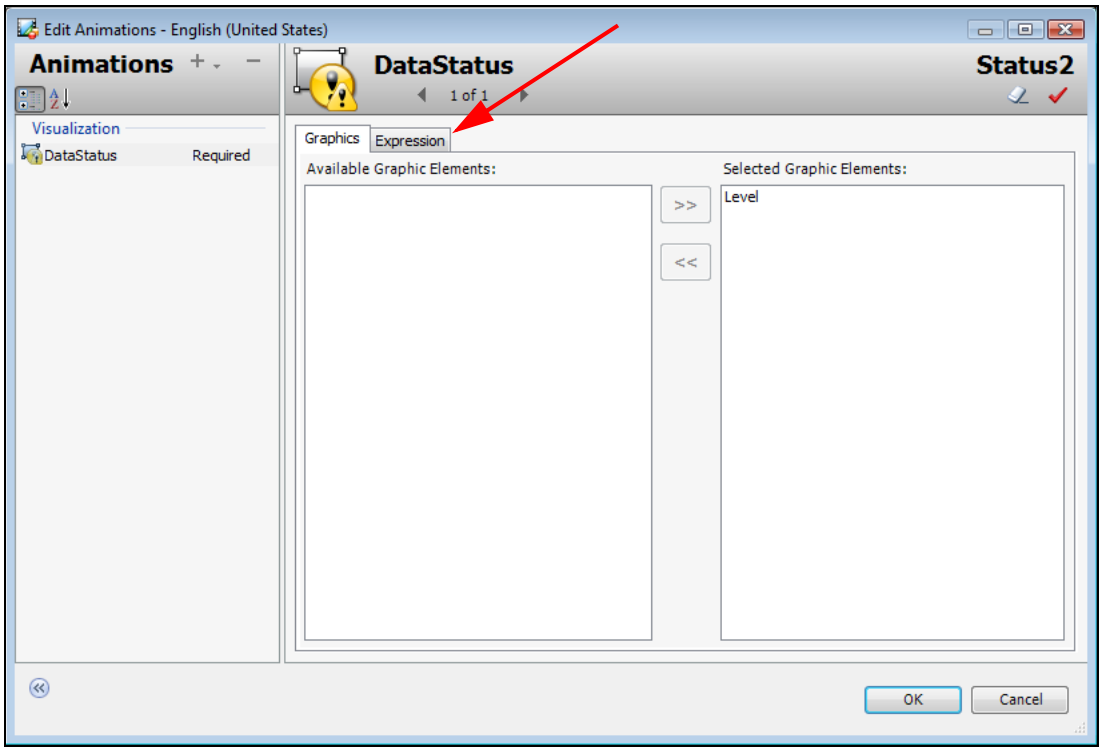
Overview

Built-in quality and status associated with objects are available in the ArchestrA Symbol Library. You can use the status element to indicate specific quality and status conditions of attributes.

Ensure the integrity of data sent between your InTouch applications and the external data source such as an I/O or DAServer using quality Dot Fields and **Tag Viewer**. The set of quality Dot Fields represent the quality state of an item's data value. This quality attribute makes it fairly easy to monitor the integrity of InTouch data sent between network nodes.

Configure the **Status Graphic** element to indicate quality and status from ArchestrA attributes used in elements with animation, and directly from ArchestrA attributes.

You can also place a **Status** button on the canvas, and then associate an expression or tag with the element.



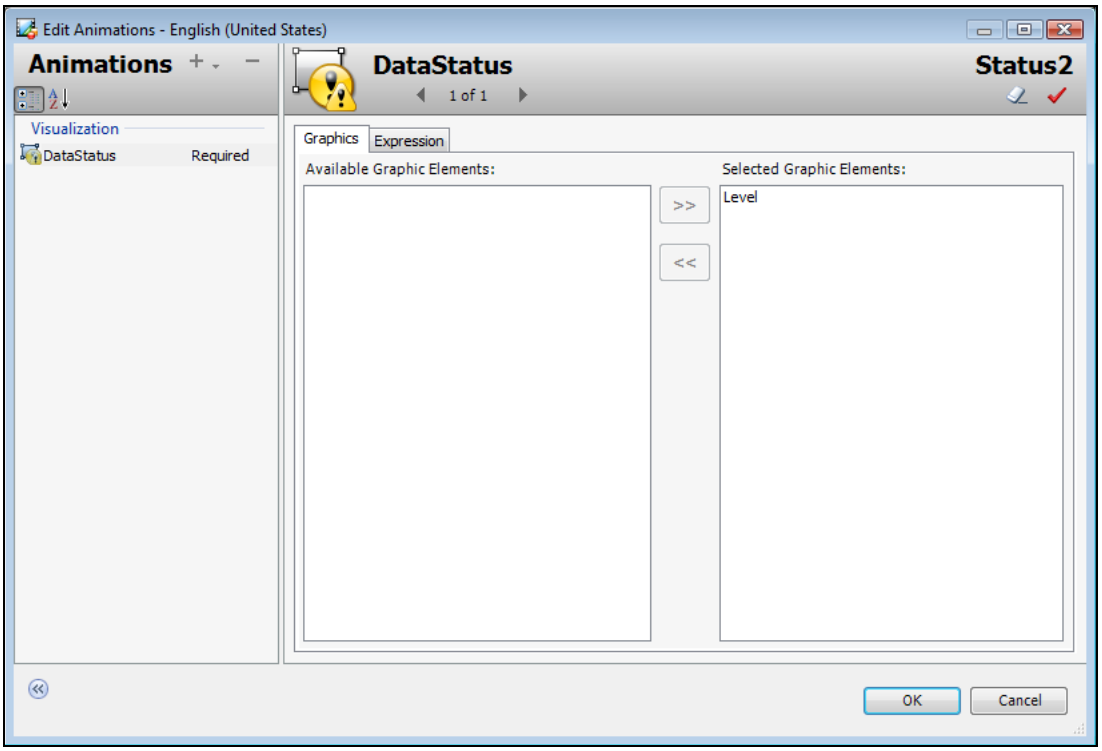
Configuring a Status Graphic Animation

You can configure the **Status Graphic** element to indicate quality and status from:

- ArchestrA attributes used in elements with animation.
- ArchestrA attributes directly.

The appearance of the Status Graphic element depends on the settings in the **Configure Quality and Status Display** dialog box. The Status Graphic animation is only used by the **Status** element and cannot be removed from the Status element.

After placing a **Status Display** rectangle on the symbol, the **DataStatus** configuration appears. This can be accessed for editing subsequently using **Special | Edit Animations** to display the **Configure Quality and Status Display** dialog box.



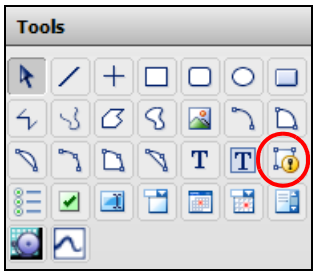
Note: For more information on **Quality and Status** configuration, please see the *InTouch User Guide*.

Element-Specific Animations

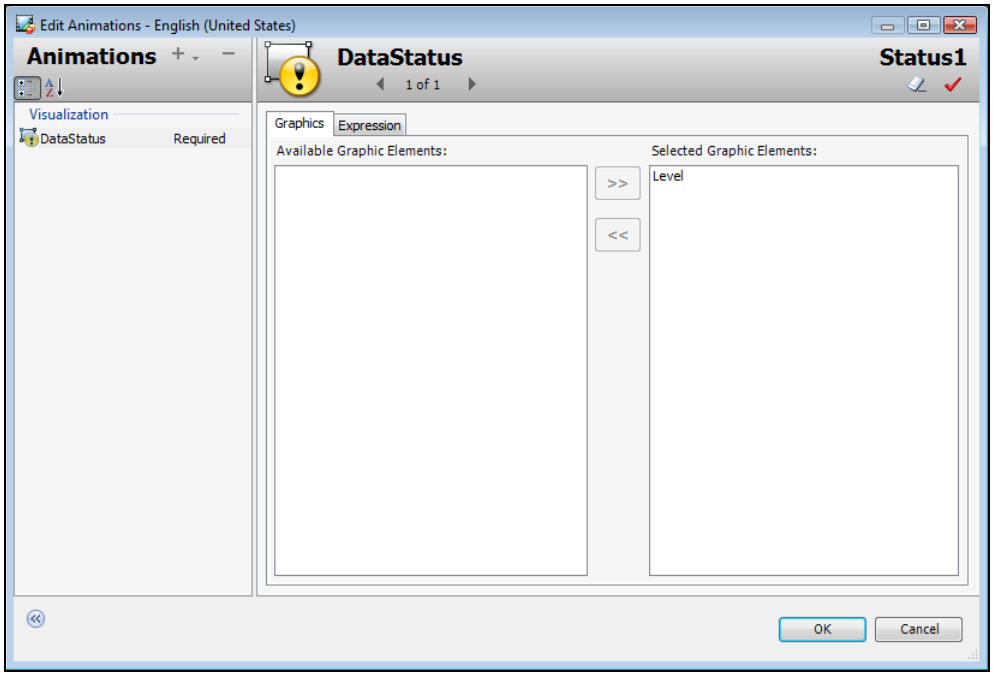
Instead of overriding the appearance of elements on the canvas, you can use a status element to show the quality and status of monitored attributes.

In addition, some elements have their own unique animation type that can only be used for that element type. You cannot remove their unique animation, but depending on the element you can add and remove other common animations.

Access the status element to indicate specific quality and status conditions of attributes. You select this option from the Tools pane and place in on the canvas.



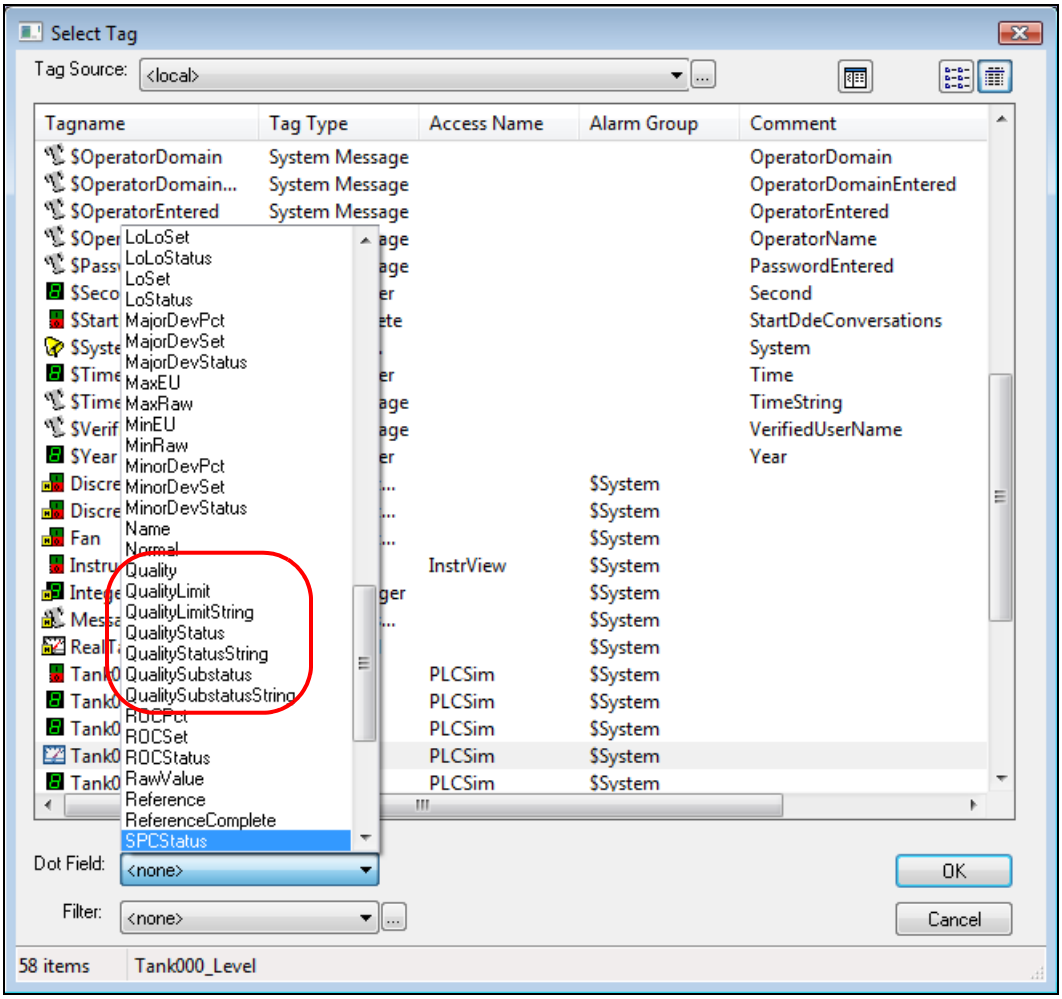
At the **Edit Animations** dialog box, assign the graphics and expressions you want to monitor. When you add the status element to an object, quality and status of the object is monitored based on associations you define for the associated tagname.



Quality and Status Dot Fields

The InTouch application tracks the current values and other status information from the component properties assigned to tags. Status can be tracked using Dot Fields such as Min, Max, Hi and Lo Status, and Time, You can also use a set of quality Dot Fields to ensure the integrity of data sent between an I/O Server and your InTouch applications. Quality dotfields represent the quality state of an item's data value. This quality attribute makes it fairly easy to monitor the integrity of InTouch data sent between network nodes.

The Quality Dot Fields indicate the quality of data values the last time data was received. This allows you to observe a quality change when a new data value is received. Some I/O Servers can send current data values with updated quality when the quality associated with the data changes.



Lab 9 – Monitoring Custom Symbol Status and Quality

Introduction

In this lab, you will configure and link graphics to monitor the Quality and Status of I/O communications.

Objectives

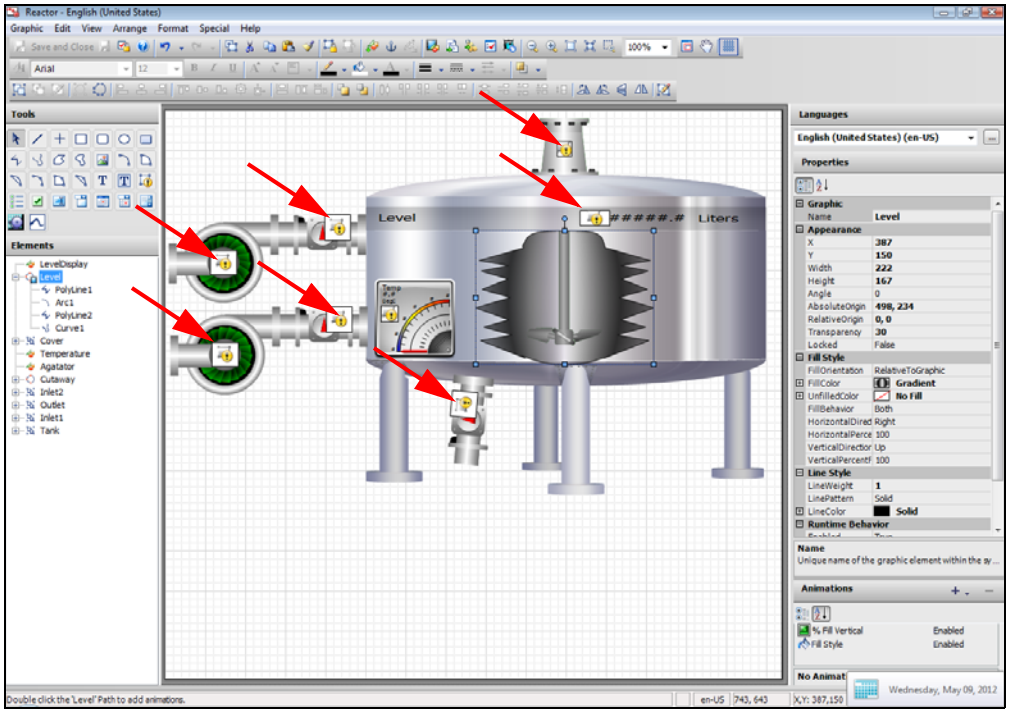
Upon completion of this lab, you will be able to:

- Add a Quality and Status elements to a symbol
- Link animations to quality and status
- Identify I/O communication and linking problems

Monitor the Health of I/O Communications

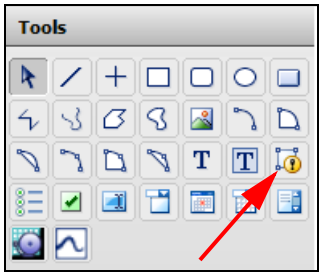
You will now add a Quality and Status element to your Level animation.

1. In ArchestrA IDE, switch to the **Reactor** symbol.

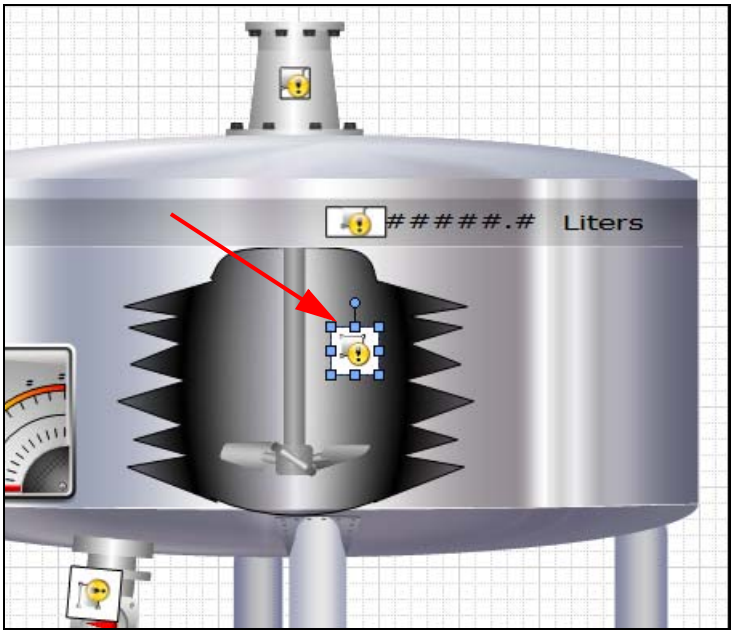


Notice that the embedded symbols you included already have the **Quality and Status** element. You will now add **Quality and Status** to the **Level** element you created.

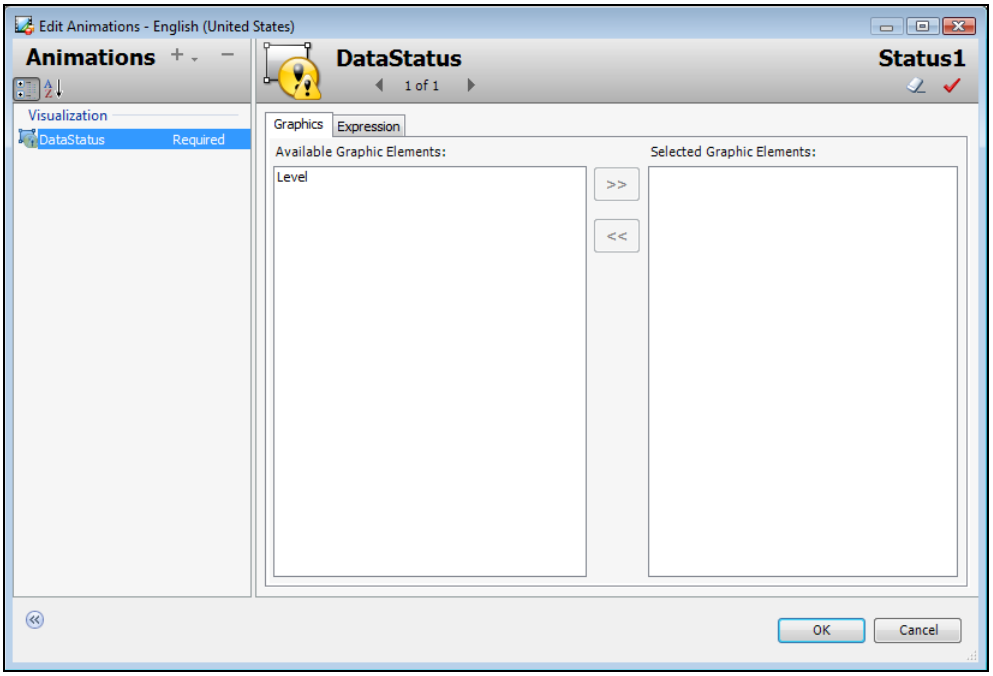
2. In **Tools** pane, click **Status**.



3. On the **Cutaway**, click and drag the **Quality and Status** element to create a small square.



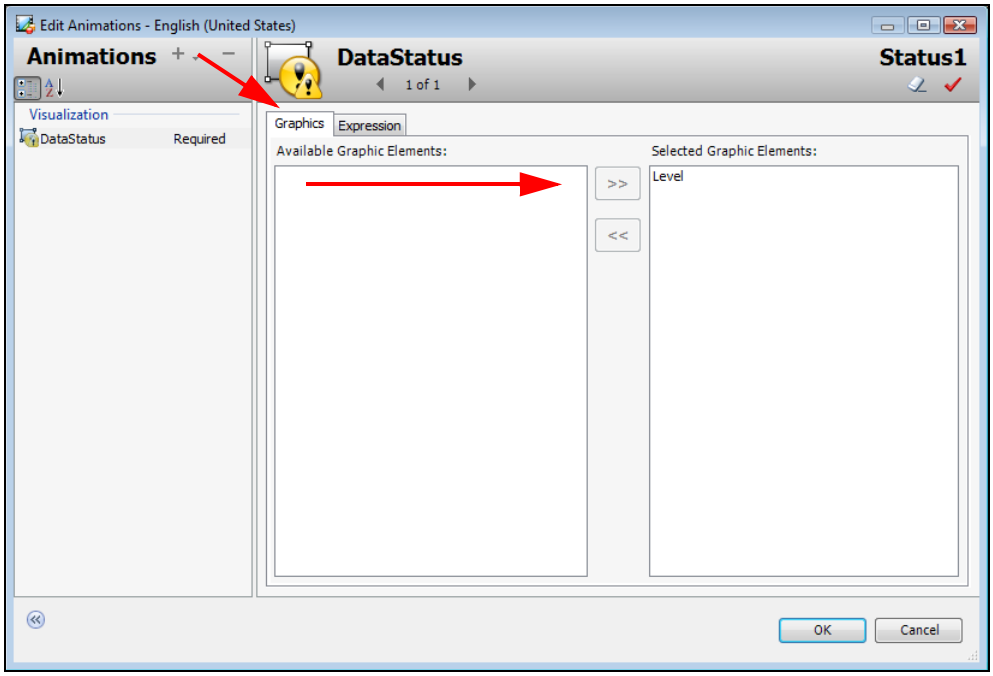
When you release your mouse, the **Edit Animations** window appears.



4. On the **Graphics** tab, in the **Available Graphic Elements** list, click **Level**.

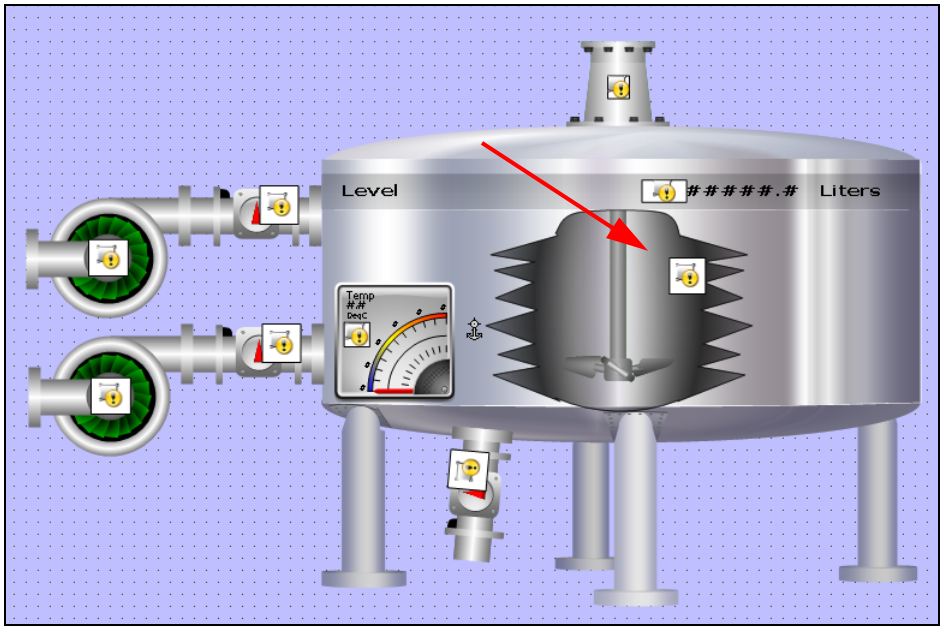
5. Use the double-arrow button [>>] to move the **Level** element to the **Selected Graphic Elements** list on the right.

   The **Status** element will monitor the **Level**.



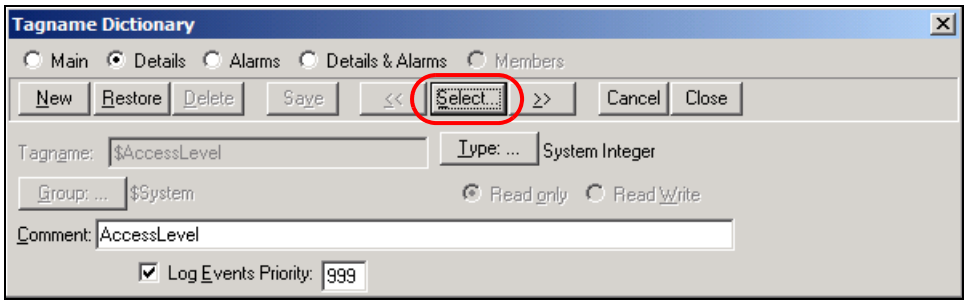
6. Click **OK**.
7. **Save and Close** the **Reactor** symbol.
8. Enter an appropriate **Check In** comment and click **OK**.
9. In WindowMaker, refresh the symbol.

   The **Quality and Status** element now appears.

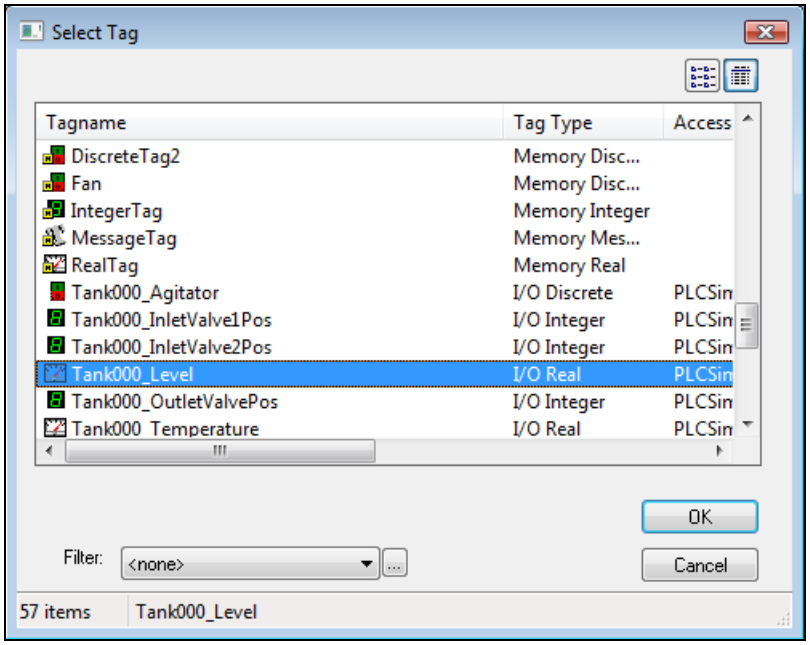


Next, you will simulate a configuration error by intentionally misspelling a PLCSim item name.

10. Open the **Tagname Dictionary**.
11. Click **Select**.



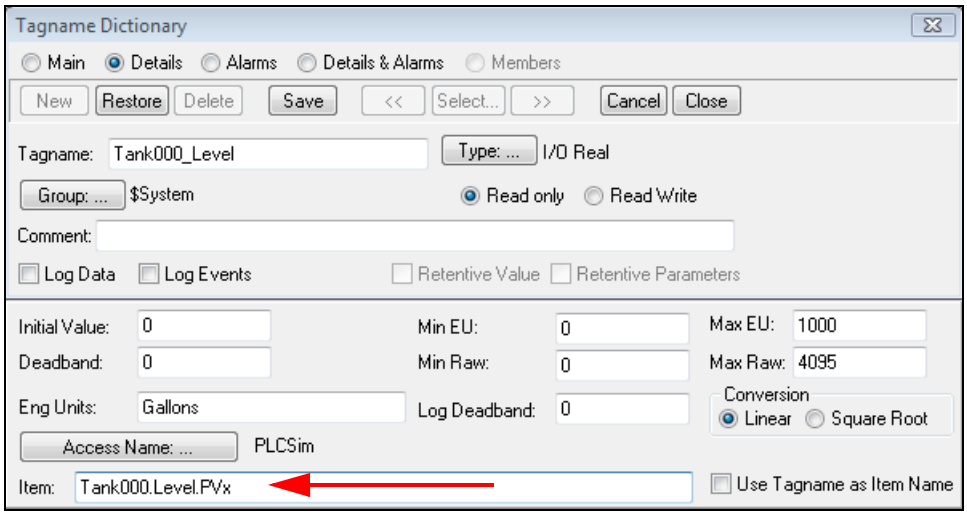
12. Select **Tank000_Level**.



13. Click **OK**.

14. In the **Item** field at the bottom of the dialog box, add an extra letter **x** to **Tank000.Level.PV**.

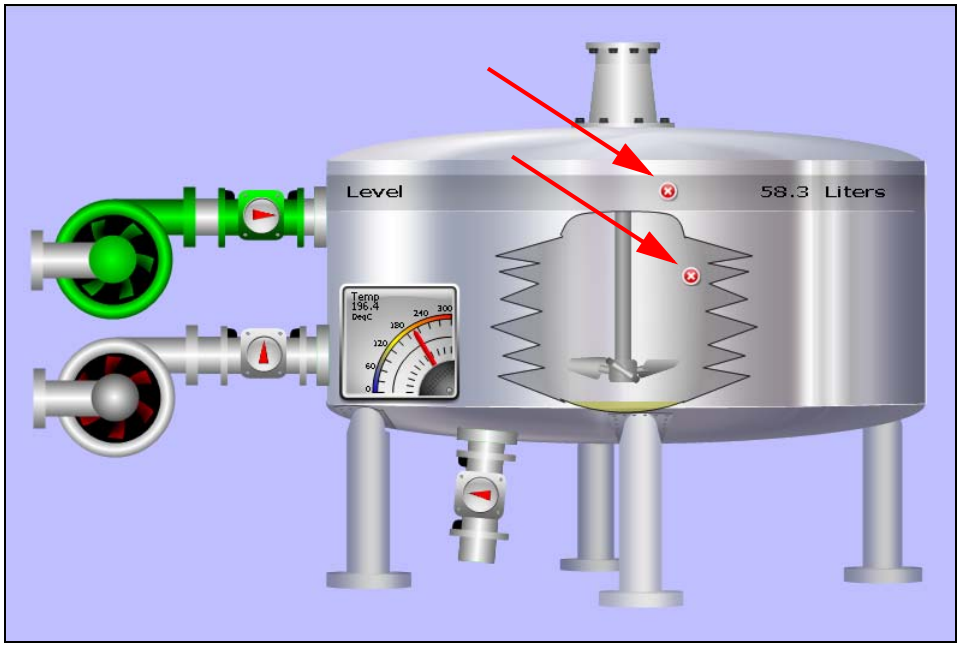
    The new item name will be: **Tank000.Level.PVx**



    This will make the reference invalid since that item name does not exist in the PLC simulation.

15. Click **Close** to save the tag and close the **Tagname Dictionary**.
16. Switch to **Runtime**.
17. Select the **Mixer** window.

    Notice the **Quality and Status** icon changes to a white **x** in a red circle, indicating a **Bad** quality.



18. Switch back to **Development**, and remove the extra character in the **Item** name of your **Tank000_Level** tag.
19. Review the change in **Runtime** to confirm the error indicators are gone and the animations are working again.



Learning Services

Module 4 – Alarms and Events

Section 1 – Enabling and Configuring Alarms	4-3
Lab 10 – Configuring Alarms for Discrete and Analog Tags	4-7
Section 2 – Alarm Client	4-29
Lab 11 – Implementing an Alarm Client	4-31
Section 3 – Alarm DB Logger	4-47
Lab 12 – Viewing Alarm History with the Alarm DB Logger	4-51
Section 4 – Alarm Printing	4-67

Module Objectives

- Introduce alarms and events
- Enable alarms for tags
- Create and configure an alarm display
- Enable and configure the alarm DB Logger
- Analyze alarms and events in the alarm display

Section 1 – Enabling and Configuring Alarms

This section describes methods to enable and configure alarms for discrete and analog tags.

Overview

InTouch has two types of notifications to inform operators of process activity: **Alarms** and **Events**.

Alarms represent warnings of process conditions that could cause problems and require an operator response. You can configure alarms to require an acknowledgment even if the condition causing the alarm has passed. This ensures that an operator is aware of events that caused a temporary alarm state but have returned to normal.

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.

Events represent normal system status messages, and do not require an operator response. A typical event is triggered when a certain system condition takes place, such as an operator logging into InTouch. If configured to do so, InTouch can log an event to the alarm database and print it out to a printer.

Any tag can be configured to do event monitoring while you are defining it in the Tagname Dictionary. When you define a tag to do event monitoring, an event message is logged to the alarm system each time the tag value changes. The event message logs how the value changed, whether the operator, I/O, scripts, or the system initiated the change.

Alarms are enabled in the Tagname Dictionary on a tag-by-tag basis. As a supervisory system, InTouch is frequently used to monitor discrete alarm outputs from a PLC or DCS system. Then InTouch is used primarily to track operator information based on who acknowledges, or who is logged into the system when an alarm occurs. InTouch can add value to alarm monitoring with three types of analog alarm monitoring that may not exist in an existing hardware based system:

- **Limit Alarms** – used to track Lo, Lo Lo, Hi, and Hi Hi value limits which can also be changed in Runtime through InTouch Dot Fields.
- **Rate of Change Alarms** – used to compare a previous alarm value and its timestamp, and the current value and timestamp, to determine the value has changed by too much during that period.
- **Minor and Major Deviation Alarms** – Used to monitor that the current alarm value does not drift too far from the target value specified. Events are informational data, triggered when a tag changes is enabled by checking the log events option for any tag type. Events can be used, for example, to monitor the start-stop cycles on a piece of equipment to determine the maintenance period. Events can be disabled globally for all tags through WindowMaker configurable enabled events. Events are not displayed in a live alarm display and are only shown in Historical Display Option.

Alarm Groups

Alarm groups are created to organize alarms into parent-child relationships. In the example below, a valve alarm will trigger an in-alarm state for the **Valves** group, the **Line1** group, and the **Production** groups as these are nested.

Below is an example showing an alarm group:

- Production
  - Line1
    - Pumps
    - Temps
    - Pressures
    - Valves

In this example, in the **Production** alarm group, if a valve alarm occurs there will be an alarm condition for **Line1**, as well as **Production**.

Alarm and Event Properties

When an alarm or event is configured for a tag, a priority between 1 and 999 must be provided, where 1 is the highest priority.

The table below shows a suggested range for critical, major, minor, and informational alarms. This range might be specified by regulations:

Critical:	1-249
Major:	250-499
Minor:	500-749
Informational:	750-999

Alarm Acknowledgment Models

Acknowledged alarms can be processed in a variety of methods, based on the industry regulations and network environments. InTouch is configured to automatically acknowledge an alarm which returns to normal. However, regulated industries such as nuclear and pharmaceutical should disable this feature.

Alarm Acknowledgment Models include the following:

- **Condition** – This is the default state and handles alarms as an instance. That is, only the most current alarm state is shown and may be acknowledged. For example, a tag configured with both a Lo and a Lo Lo state, that is currently in a Lo Lo alarm implies that it is also in a Lo alarm as the value must cross through the Lo state in order to reach the Lo Lo state. Thus, when the Lo Lo instance is acknowledged the Lo state is automatically acknowledged.
- **Event Oriented** – This alarm state is similar to Condition acknowledgment, except it is designed to deal with race conditions frequently occurring in high latency networks, such as acknowledgment from an operator station connected over the Internet.
- **Expanded Summary** – This is the alarm model typically used for regulated industries. This option requires each state and sub-state be recorded as acknowledged separately.

Alarm Annunciation

InTouch provides numerous ways to provide Operator notification, and to annunciate that an alarm has occurred. InTouch provides system tags, alarm groups, tagname Dot Fields, script functions, symbol properties, animation links, and popup windows – any or all of which can be used to monitor and annunciate alarm conditions. By leveraging ArchestrA symbols, script functions and animations in nearly infinite number of possibilities to alert Operators can be achieved.

Alarm Inhibition

Alarms can be prevented from annunciation such as appearing in the Alarm Client using Alarm Inhibition Alarms. For example, an Alarm Group called Production1 can have all of its Hi and Hi Hi alarms inhibited. Inhibition requires a discrete tag be created and linked to the inhibition field of a tag or group.

In runtime, when this discrete tag is set to 1, all of the alarm states the inhibition tag was linked to, are inhibited. For each distinct inhibition, additional discrete tags must be created and linked. Alarm inhibition can be hidden from the Alarm Client display while still being tracked and recorded.

Lab 10 – Configuring Alarms for Discrete and Analog Tags

Introduction

In this lab, you create an Alarms window and configure alarms for Discrete and Analog tags.

Objectives

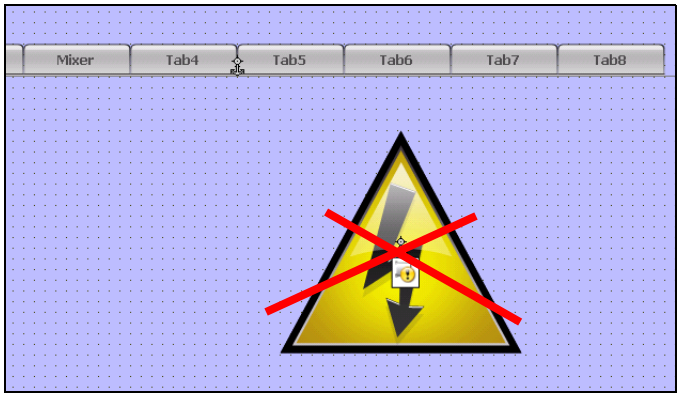
Upon completion of this lab, you will be able to:

- Configure tags for alarming
- Animate alarms
- Inhibit alarms

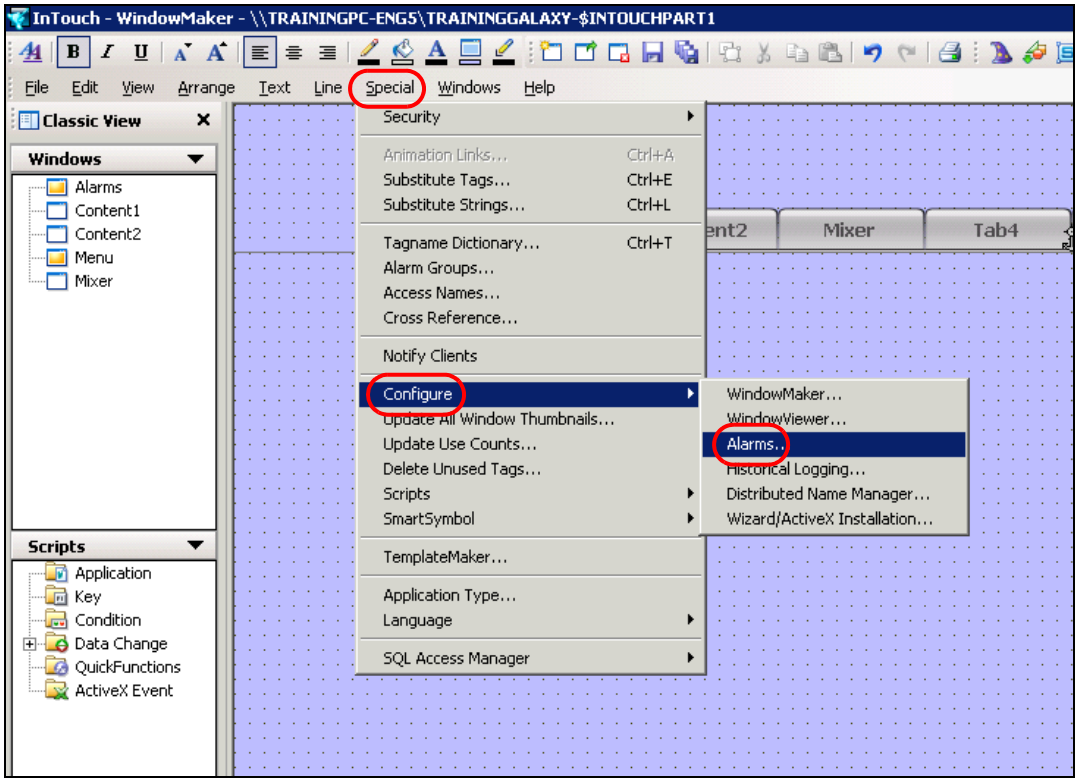
Create an Alarms Window

First, you create an Alarms window and make changes to the alarm configuration.

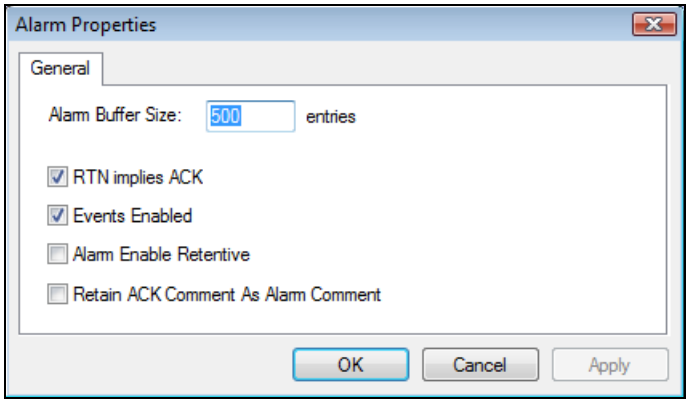
1. In WindowMaker, open the **Content2** and **Menu** windows.
2. Right-click **Content2** and select **Save As**.
3. Change the window name to **Alarms**.
4. Delete the warning light on the new **Alarms** window.



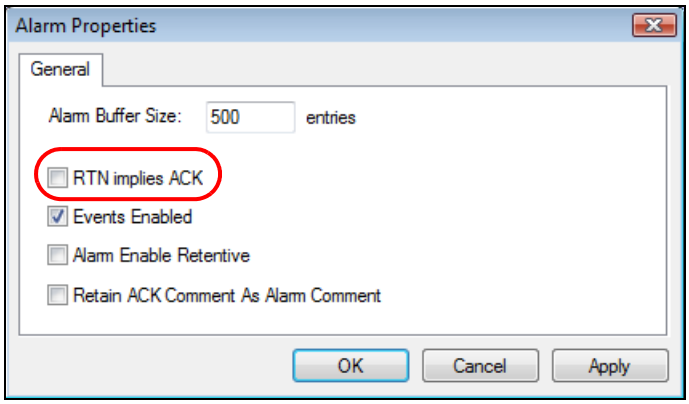
5. On the **Special** menu, click **Configure | Alarms**.



The **Alarm Properties** dialog box appears.

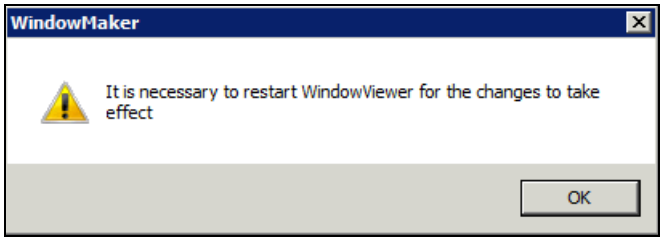


6. Uncheck **RTN implies ACK**.



Note: Most regulated industries do not permit automatic acknowledgment. That is, operators are required to acknowledge every alarm state.

7. Keep the remaining defaults and click **OK**.
8. A message to restart WindowViewer appears.

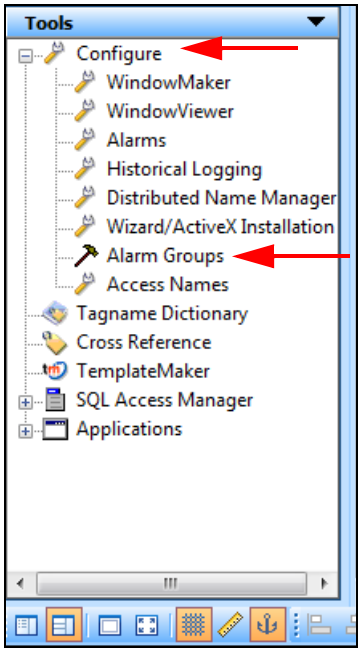


9. Click **OK**.
10. Close WindowViewer.

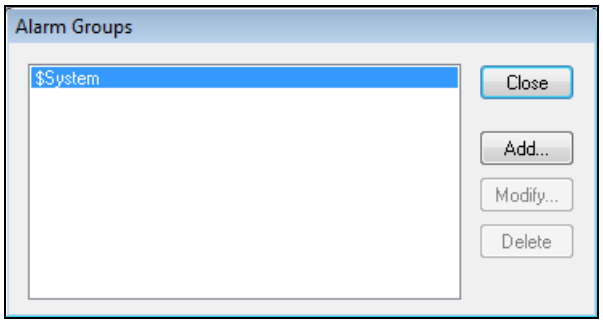
Create an Alarm Group

Next, create a new alarm group from the Alarm Groups option.

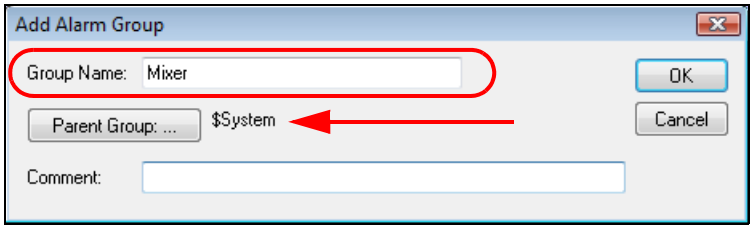
11. In **Tools** pane, expand **Configure** and double-click **Alarm Groups**.



The **Alarm Groups** dialog box appears.

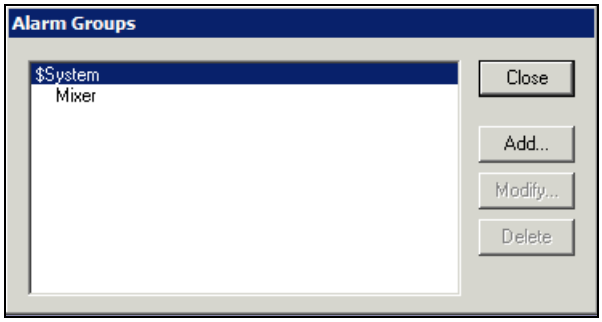


12. Click **Add**.
13. In the **Group Name** field, enter **Mixer**.



14. Confirm the parent group is **$System**.
15. Click **OK**.

16. In the **Alarm Groups** dialog box, click **Close**.

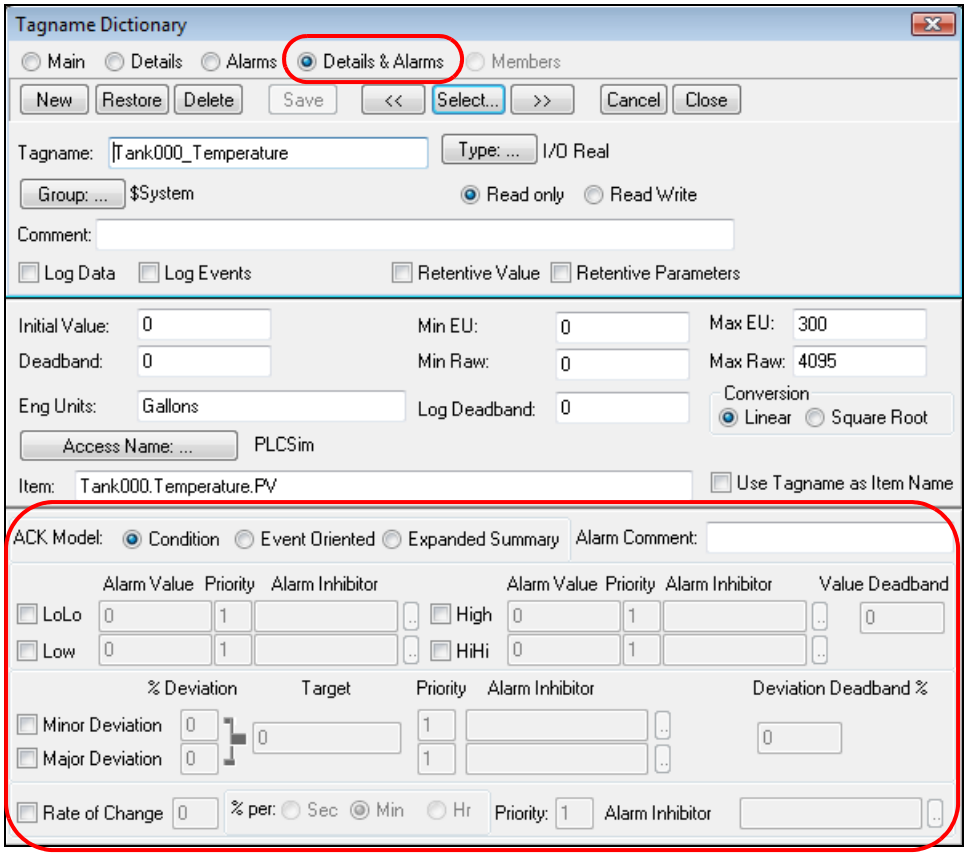


Configure Tags for Alarming

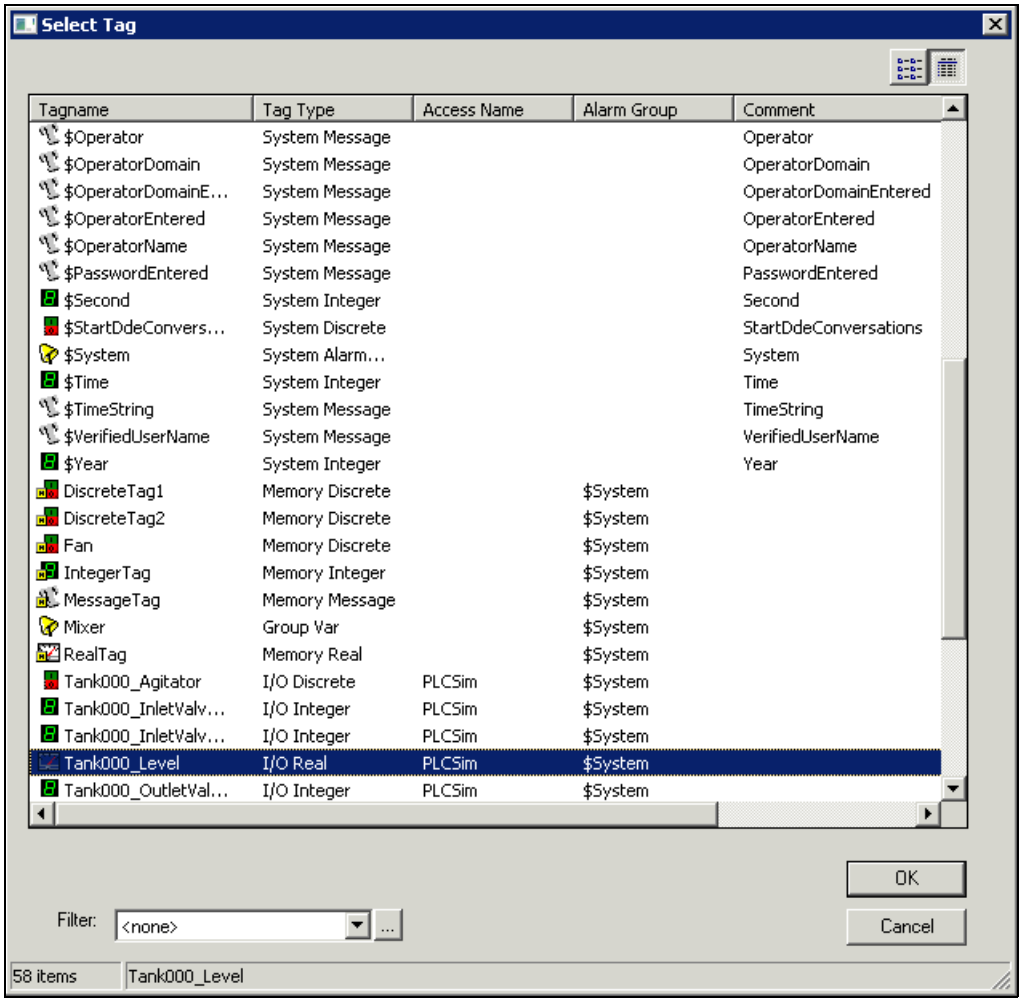
Next, set up alarm types for your tags.

17. In **Tagname Dictionary**, select **Details & Alarms**.

    This displays the alarm configuration area of the **Tagname Dictionary**.

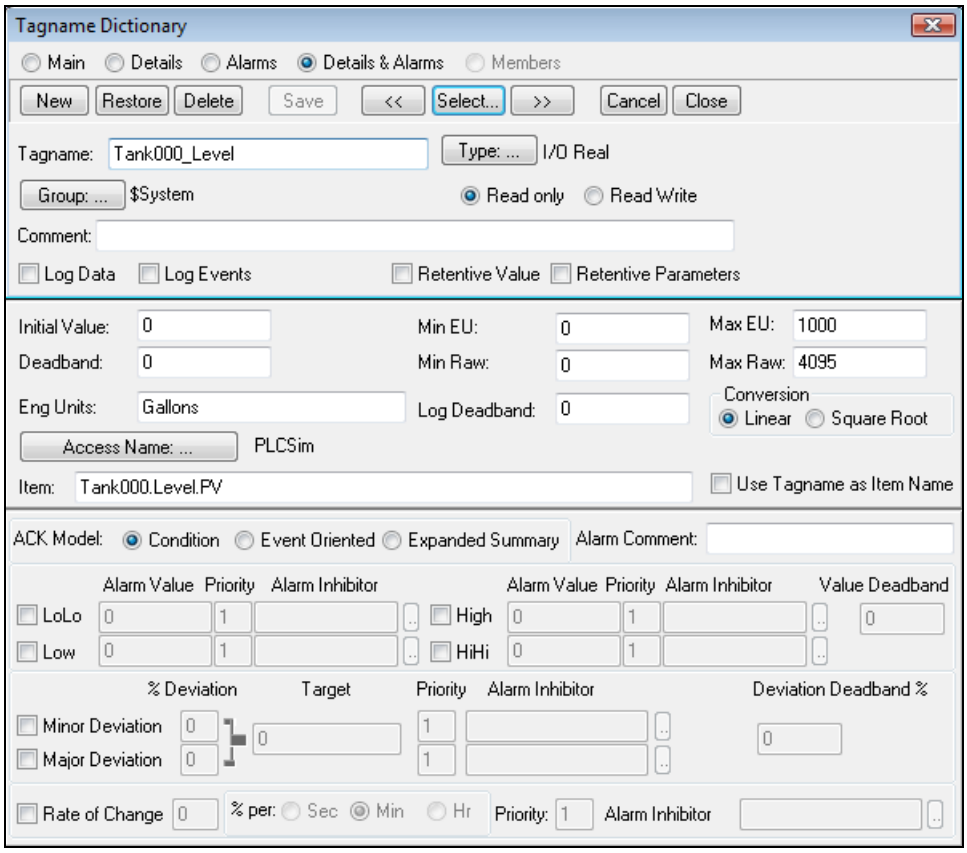


18. Click **Select**, and select **Tank000_Level**.



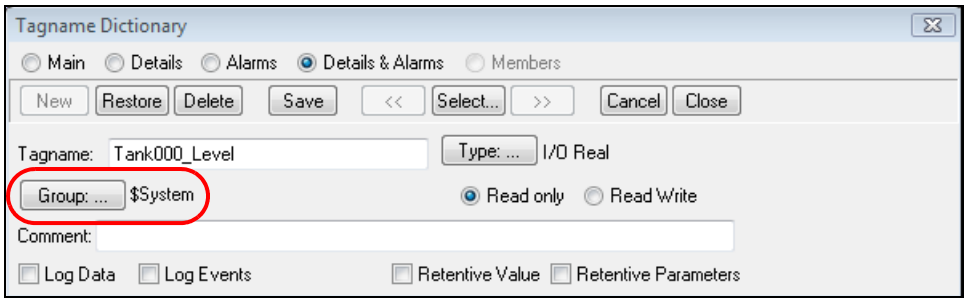
19. Click **OK**.

    Notice the **Alarm** options for the tag in the bottom portion of the dialog box.

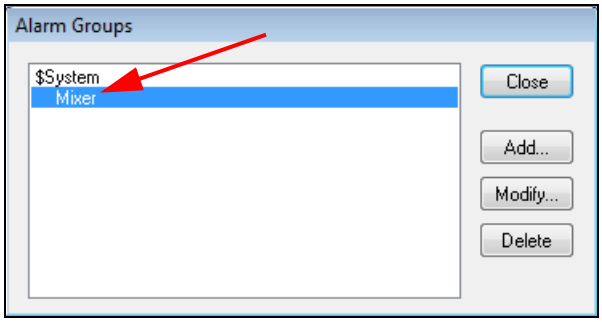


Next, you assign the **Tank000_Level** tag to the **Mixer** alarm group.

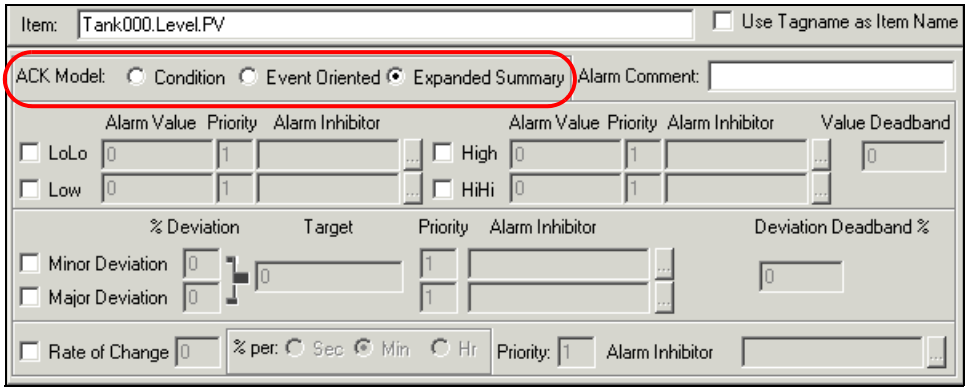
20. Click **Group**.



21. In the **Alarm Groups** dialog box, click **Mixer**.



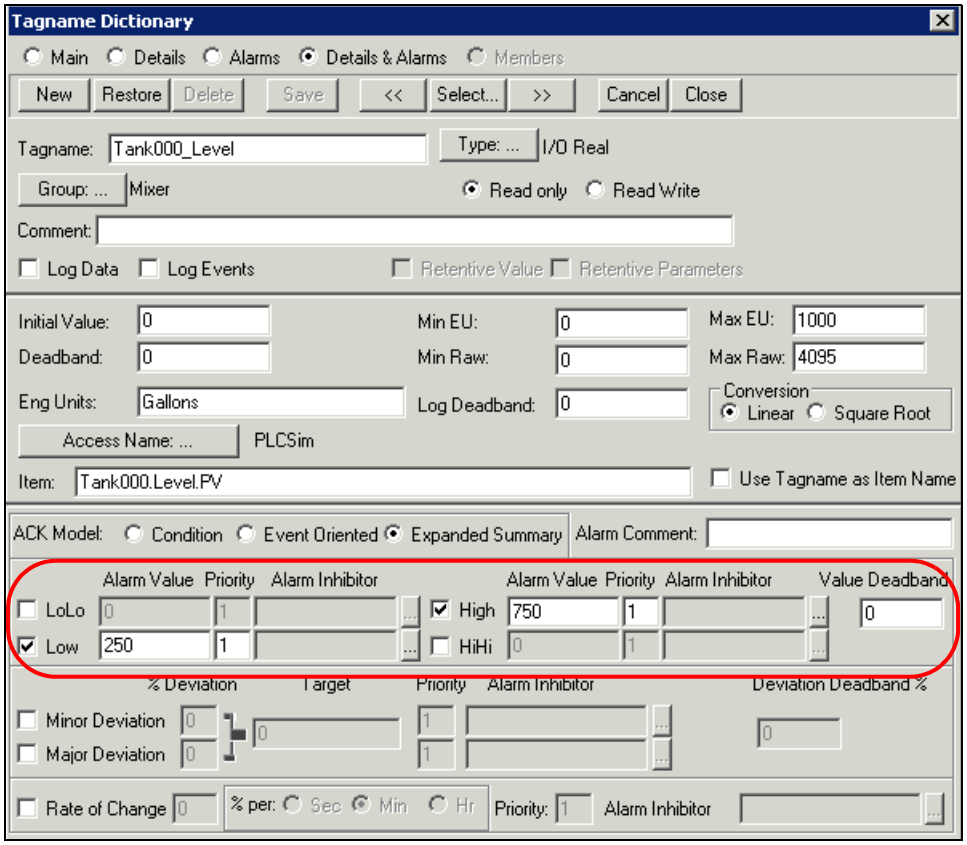
22. Click **Close** to return to the **Tagname Dictionary**.
23. From the **ACK Model** options, click **Expanded Summary**.



With this option, operators are required to acknowledge every sub-state transition of the alarm. This is a requirement of most regulated industries.

24. Enable and configure the following alarm values:

	Alarm Value	Priority
Low	250	1
High	750	1

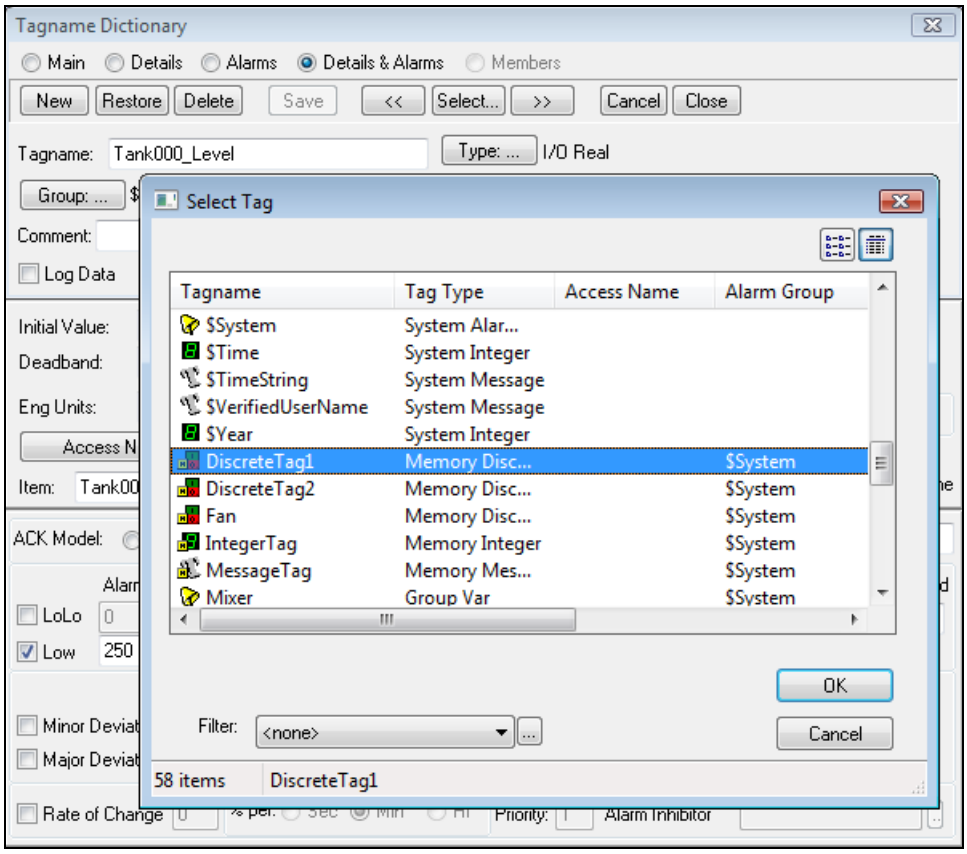


25. Click **Save**.

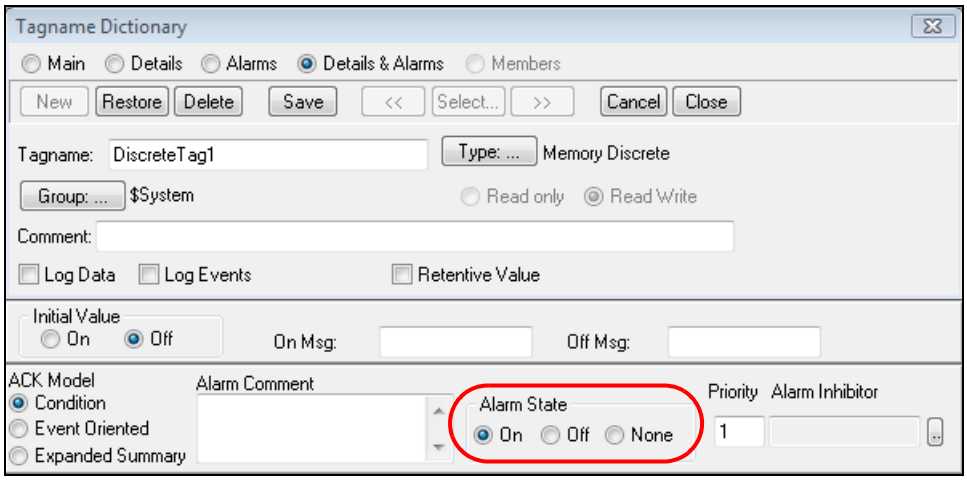
Configure a Discrete Alarm

Now configure Discrete tags to define an On state for your alarm.

26. In **Tagname Dictionary**, click **Select**.
27. Select **DiscreteTag1**.



28. Click **OK**.
29. In the **Alarm State** area, click **On**.



30. Keep the remaining defaults and click **Save**.

Configure an Inhibit Tag

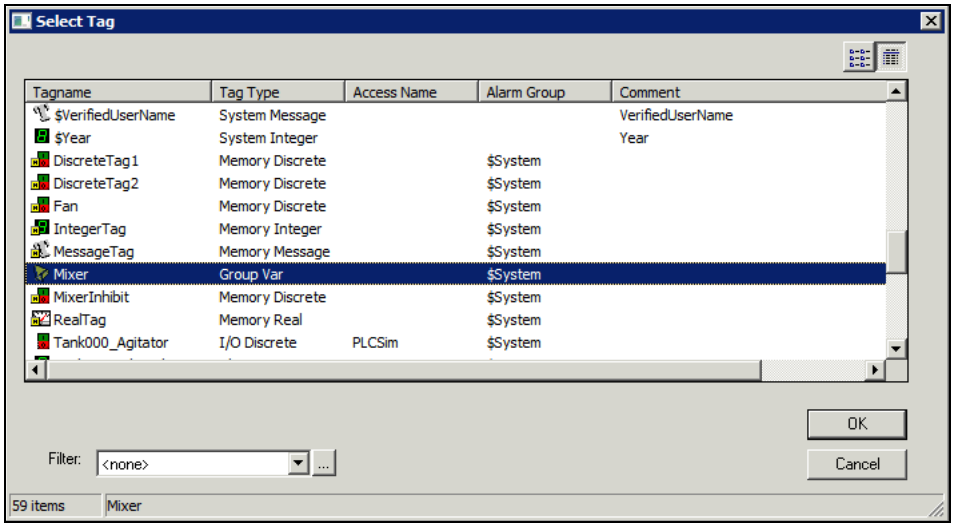
Next, you create a Discrete tag to control inhibition states for the Mixer group.

31. Click **New** and create a **Memory Discrete** tag with the following settings:

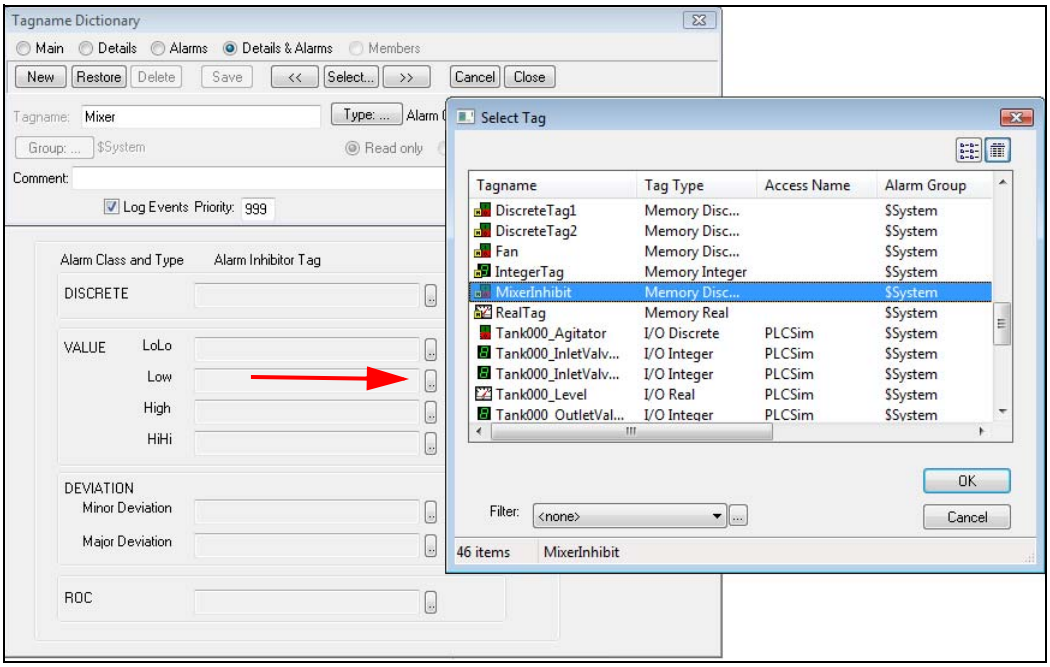
    Tagname: MixerInhibit
    Type: Memory Discrete
    Alarm State: None

Important: WindowMaker remembers the last actions performed. Review settings to ensure settings are as you intend.

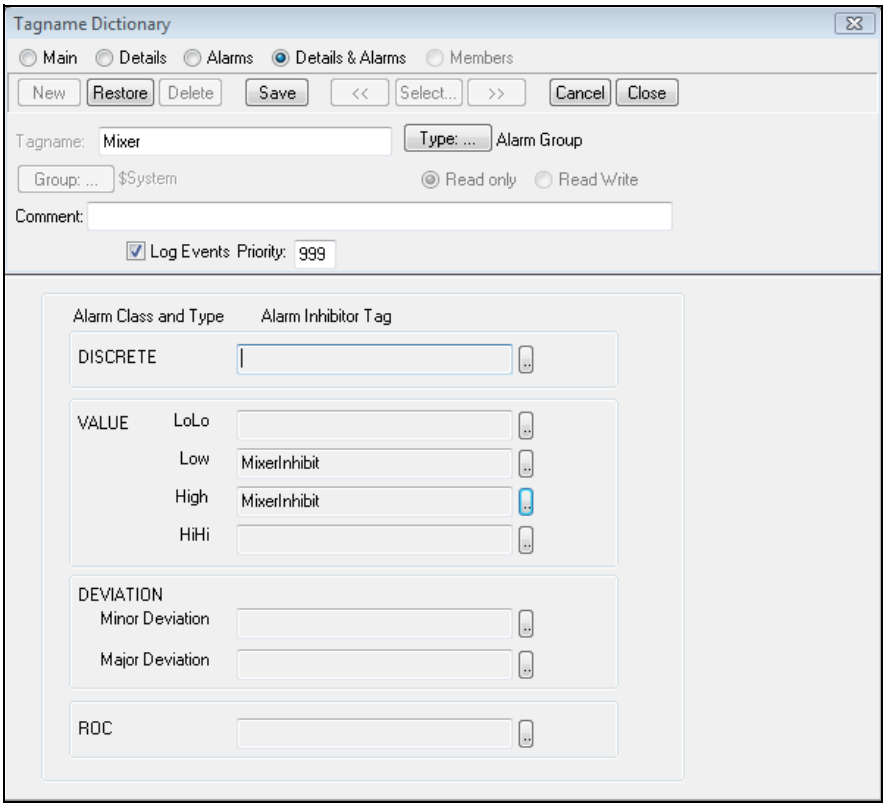
32. Click **Save**.
33. Click **Select** and double-click the **Mixer** Alarm Group.



34. In the **VALUE** area, **Low** field, click the ellipsis button to open the **Select Tag** dialog box.
35. Scroll down and double-click **MixerInhibit**.



36. Perform the above steps to select **MixerInhibit** for the **High** value.

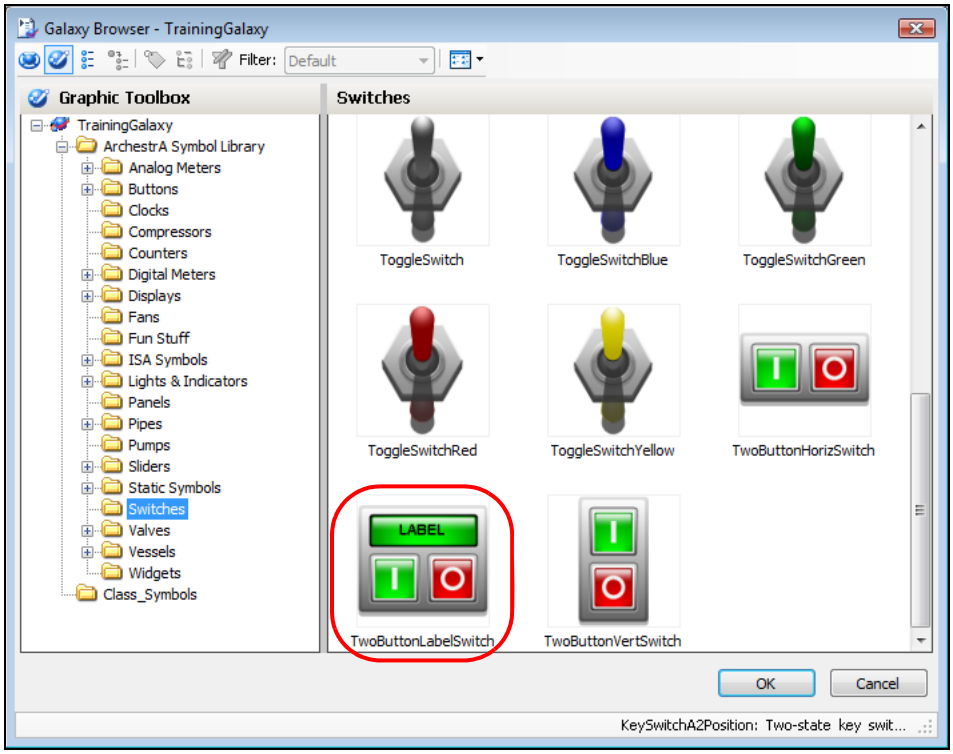


37. Click **Close** to save and close the **Tagname Dictionary**.

Display Alarms with Symbol Graphics

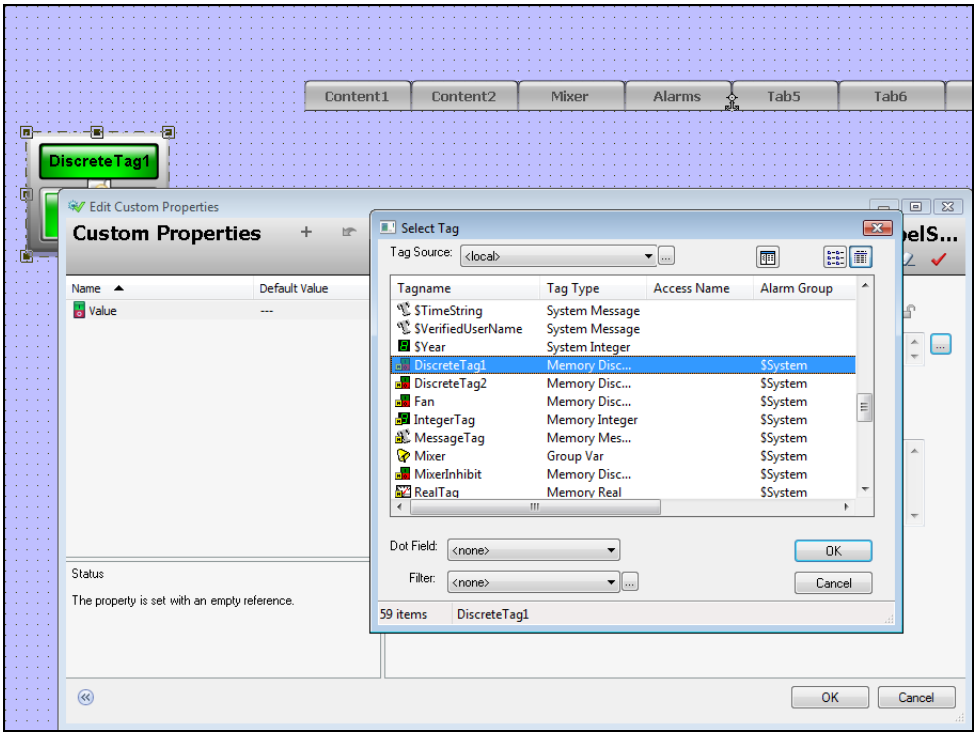
Now, you modify the tabs to include your new Alarms window.

38. In the **Menu** window, double-click the **NavigationTabs8** graphic.
39. Click **TabCount** and change the **Default Value** to **4**.
40. Click **OK**.
41. Right-click the **NavigationTabs8** graphic and select **Substitute | Substitute Strings**.
42. Change **Tab4** to **Alarms**, then click **OK**.
43. On the toolbar, click **Embed ArchestrA Graphics**.
44. In **Graphic Toolbox**, click **Switches** and select **TwoButtonLabelSwitch**.



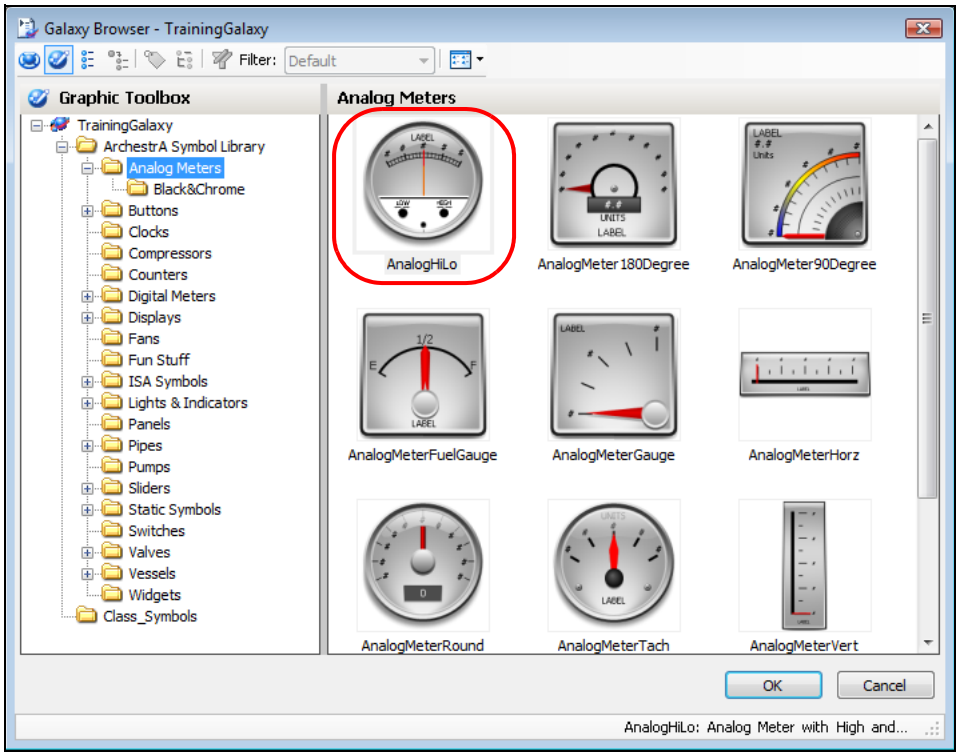
45. Click **OK**.

46. In the **Alarms** window, place the graphic in the upper-left corner.
47. Use **Substitute Strings** to change the label to **DiscreteTag1**.
48. Double-click the **DiscreteTag1** graphic and link the **Value** property to the **DiscreteTag1** tag. Make sure you remove the preceding **---,** if exist.



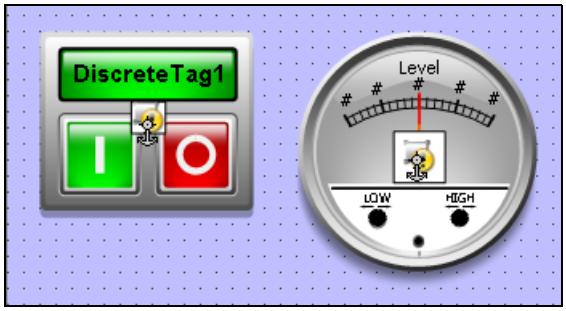
49. Click **OK**.
50. Click **OK** to close the **Custom Properties** dialog box.

51. Embed **Analog Meter\AnalogHiLo** in the **Alarms** window.



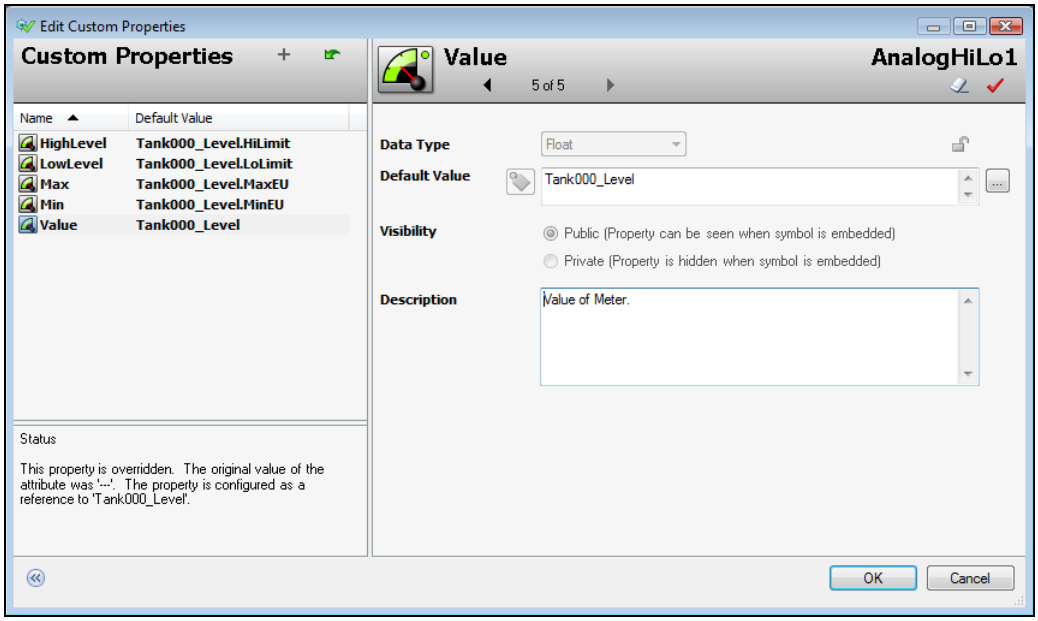
52. Position **AnalogHiLo** to the right of the **DiscreteTag1** graphic.

53. Change **LABEL** to **Level**.



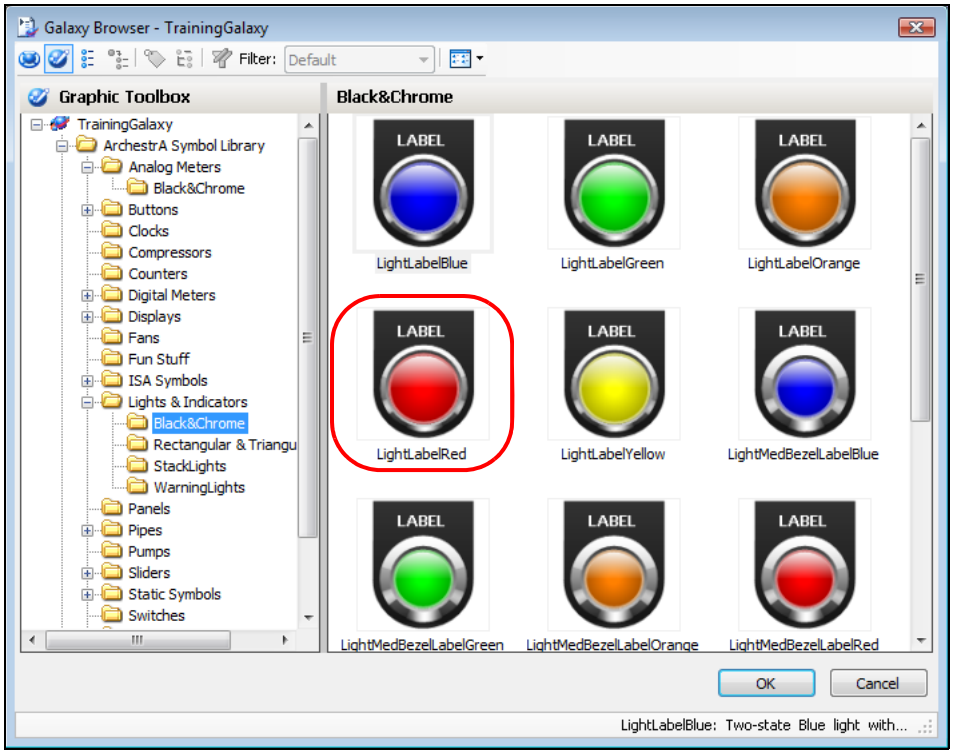
54. Link the following **Custom Properties** to the **Level** graphic:

Name	Default Value
HighLevel	Tank000_Level.HiLimit
LowLevel	Tank000_Level.LoLimit
Max	Tank000_Level.MaxEU
Min	Tank000_Level.MinEU
Value	Tank000_Level

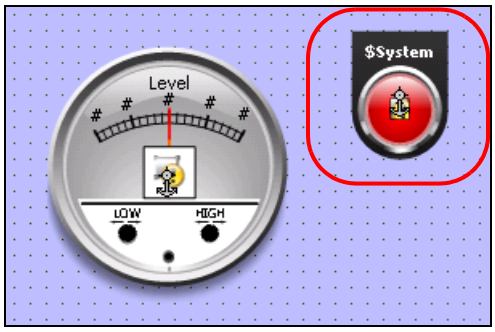


55. Click **OK**.

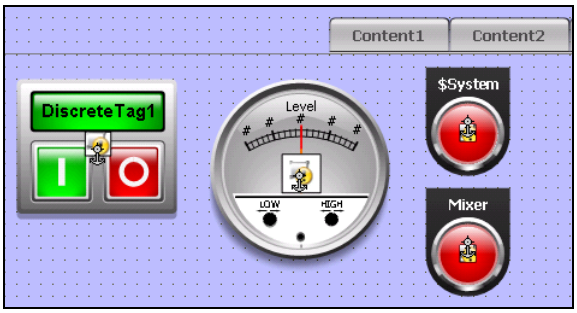
56. Embed **Lights & Indicators\Black&Chrome\LightLabelRed**.



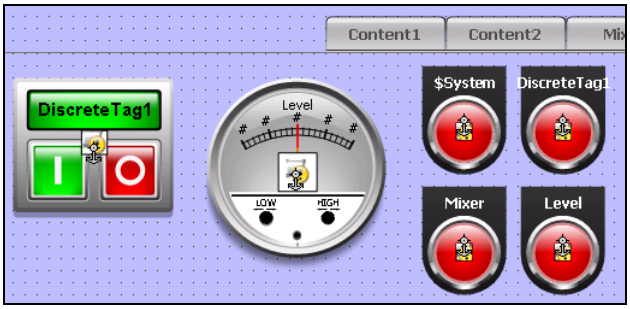
57. Place the graphic next to the **Level** graphic.
58. Change the label to **$System**.
59. Make the **$System** graphic approximately half the height of the **Level** graphic.



60. Press **Ctrl+D** to duplicate the **$System** light, then position it below the original.
61. Change the new button graphic label to **Mixer**.

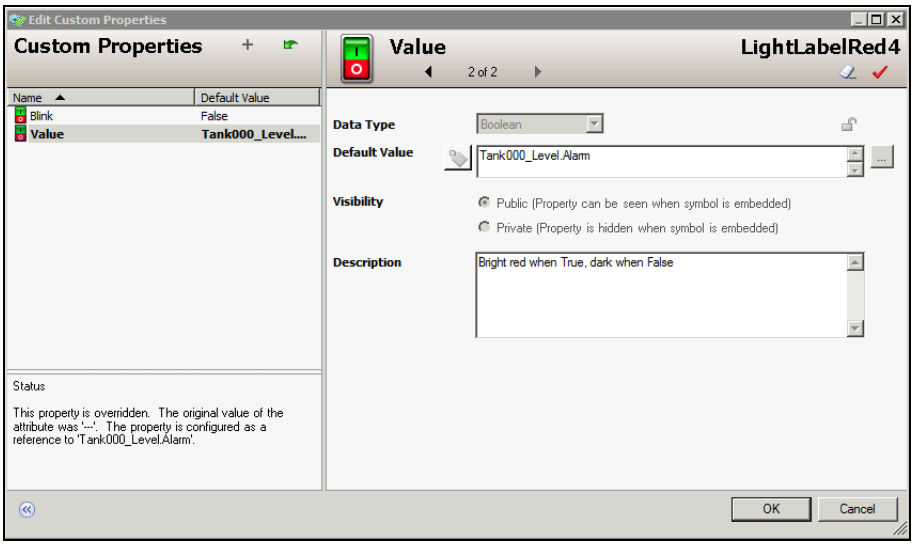


62. Duplicate both lights and place them to the right of the **$System** and **Mixer** lights.
63. Label the new lights **DiscreteTag1** and **Level**.

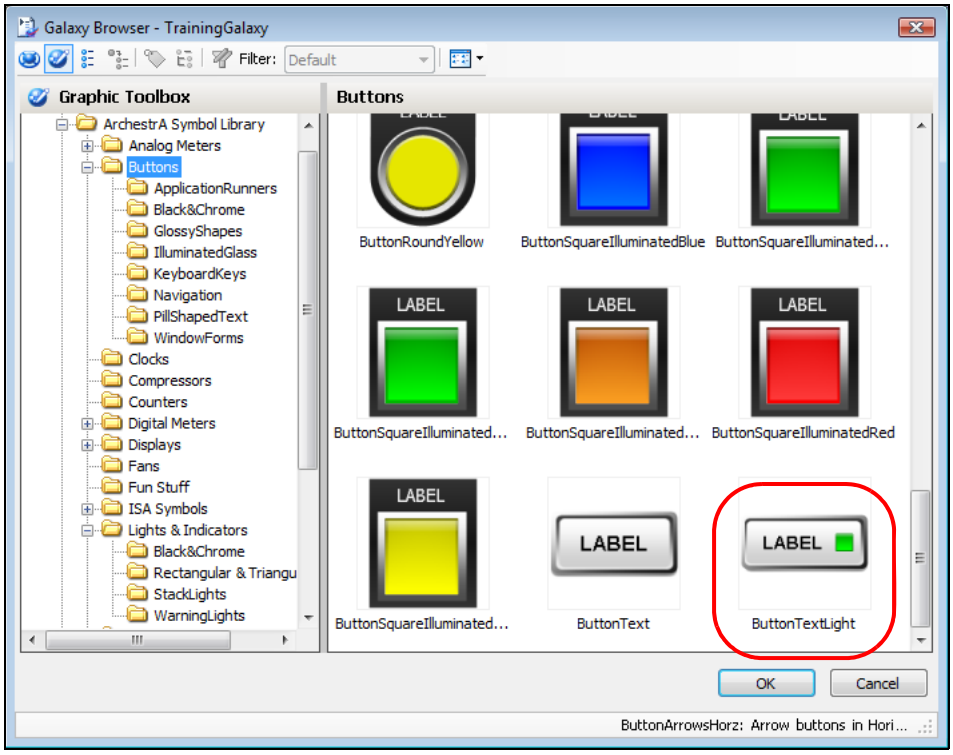


64. Configure **Value\Default Value** for each of the lights as follows

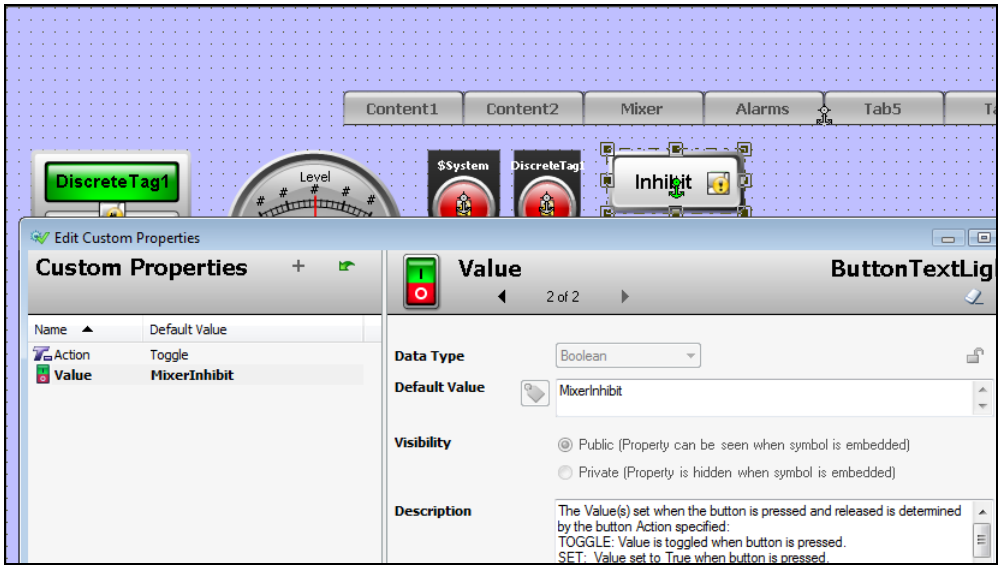
Graphic	Default Value and Dot Field
$System	$System.Alarm
Mixer	Mixer.Alarm
DiscreteTag1	DiscreteTag1.Alarm
Level	Tank000_Level.Alarm



65. Embed **Buttons\ButtonTextLight**.



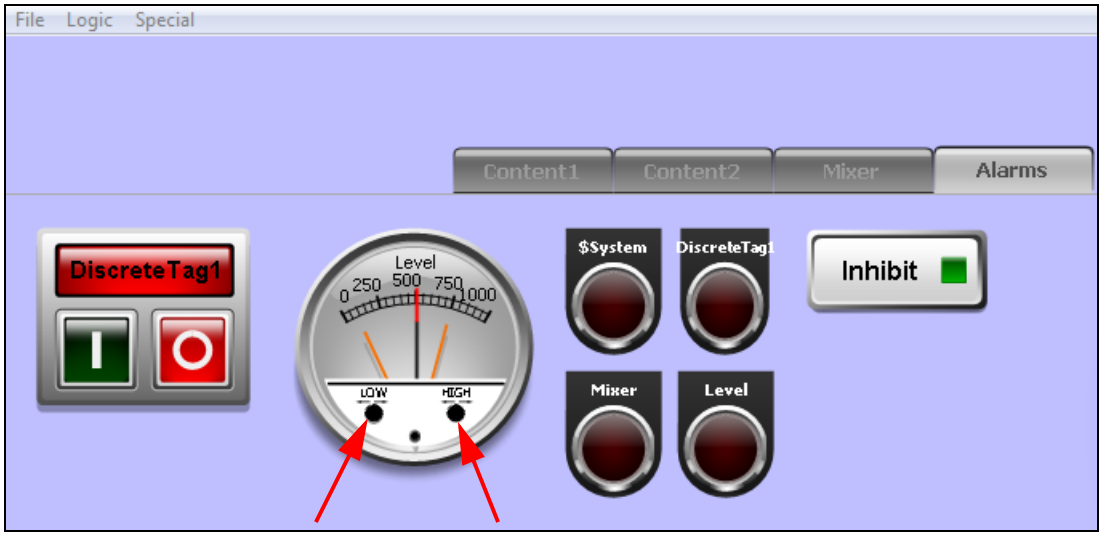
66. Place it to the right of the **LightLabelRed** graphics.
67. Change the label to **Inhibit**.
68. Double-click to open the **Custom Properties** dialog box.
69. Link **Value\Default Value** to **MixerInhibit**.



The light will illuminate when the **Mixer Alarm Group Value Limits** for **Low** and **High** alarms are inhibited. By default, this will toggle the **MixerInhibit** tag.

Test What You Have Built

70. Test in **Runtime**.
71. Click and drag the **LOW** and **HIGH** sliders to force or remove an alarm state.



Alarms display when levels are outside the **LOW** and **HIGH** settings on the **Level** gauge.

Note: The **$System** master alarm group reflects the alarm state of all subordinate alarms. This includes the alarm states of the **Mixer Alarm Group** and the individual alarms for **Tank000_Level** and **DiscreteTag1**.

---

72. Click **Inhibit** to inhibit the **Mixer Alarm Group**.

    Observe the **Mixer** and **Level** indicator lights will not illuminate because **Tank000_Level** belongs to the **Mixer Alarm** group.

73. While **Inhibit** is enabled, click **DiscreteTag1** to toggle on and off.

    Notice **DiscreteTag1** still indicates an alarm because it does not belong to the **Mixer Alarm Group**.

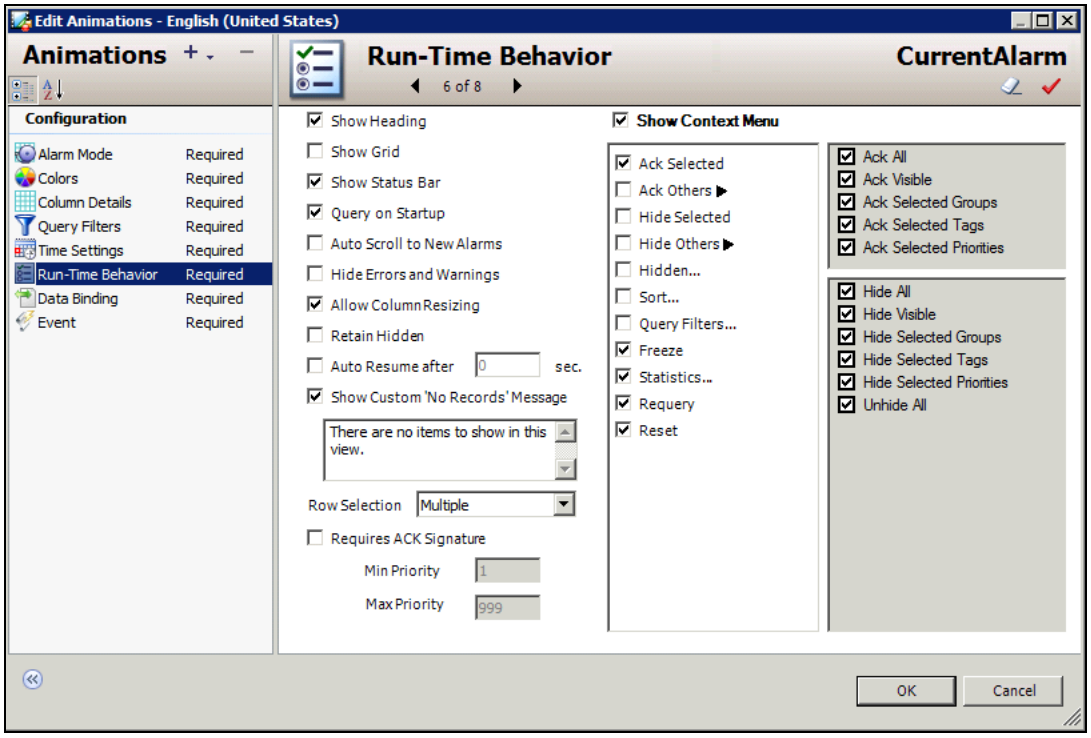
Section 2 – Alarm Client

This section describes configuration and implementation methods for the Alarm Client. It also describes Alarm Client features such as right-click and context menu capabilities.

Overview

The Alarm Client is a tool that allows the operator to see the state of alarms and events and is built with functionality for sorting, acknowledging, and handling Alarms. The Alarm Client allows users to right-click or use the context menu to interact with the Alarm Client. You can also create a reusable sort or switch between live alarms and historical analysis modes.

The Alarm Client can be configured to subscribe to alarms and events generated by Alarm Providers. In addition, the Alarm Client can be configured to subscribe to only selected Alarm Areas for the provider based on its query filters.



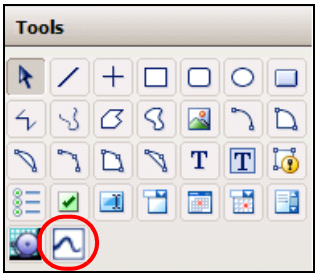
Use the Alarm Client to create an alarm display for real-time visualization and acknowledgment of alarms coming from InTouch, or to enable or disable alarming on an automation object. You also use the Alarm Client to create an alarm display for visualization of historical alarms and events logged in the Alarm Database.

Using the Alarm Client

You can create applications that generate alarms and events to provide the status of a running application. You use the Alarm Client to view alarms and for real-time alarm reporting.

- Alarms warn about process conditions that can potentially cause problems. An alarm is an abnormal condition that requires immediate attention. An operator usually acknowledges an alarm. You can set up an alarm to become active when a process value exceeds a defined limit.
- Events represent normal system, application, or user occurrences that produce status messages. A typical event occurs when an operator logs on to an application at the beginning of a work shift.

Access the Alarm Client form the ArchestrA IDE, **Tools** pane.



Lab 11 – Implementing an Alarm Client

Introduction

In this lab, you implement the Alarm Client and review alarms. You also define the look and behavior of the Alarm Client. You add controls to change the display at runtime and allow for the acknowledgment of alarms.

Objectives

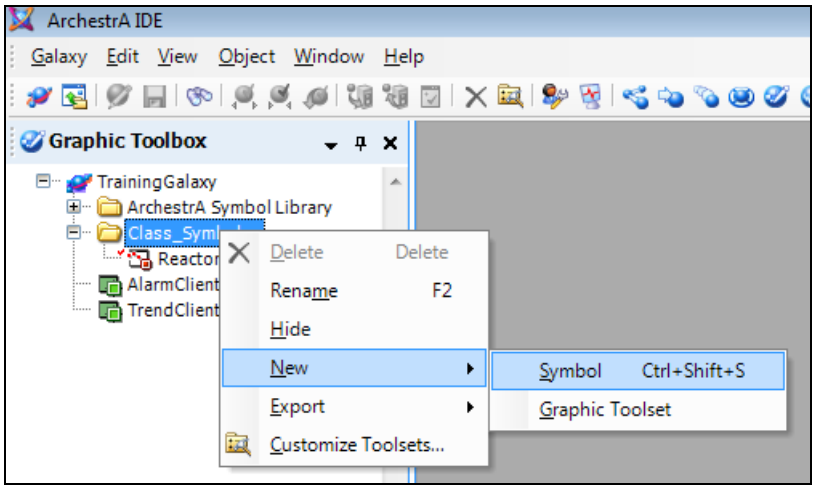
Upon completion of this lab, you will be able to:

- Configure and use the Alarm Client
- Link ArchestrA animations to InTouch tags using InTouch:Tagname syntax
- Add InTouch animations to embedded ArchestrA graphics
- Use additional Alarm Dot Fields

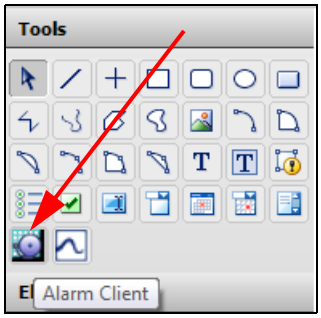
Create an Alarm Client

First, you add an Alarm Client to the Alarms window and modify it.

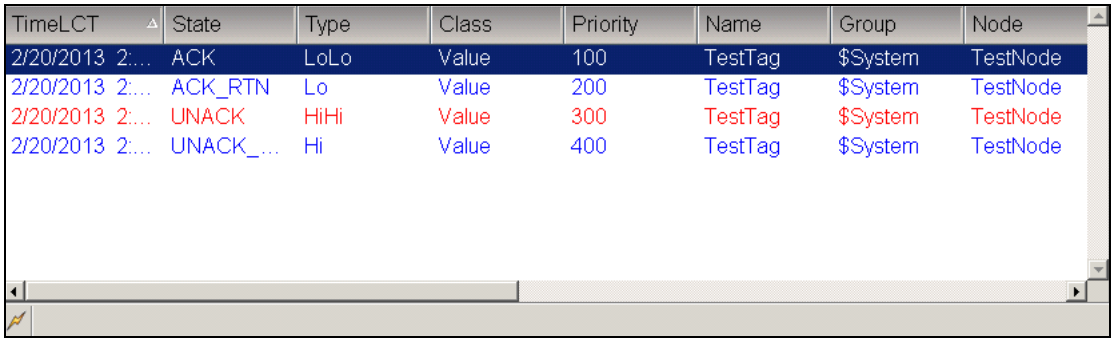
1. Close WindowViewer.
2. In ArchestrA IDE, **Graphic Toolbox**, right-click **Class_Symbols** and select **New | Symbol**.



3. Name the new symbol **Alarms**.
4. Double-click **Alarms** to open it in the **ArchestrA Symbol Editor**.
5. In **Tools** pane, click the **Alarm Client** tool.



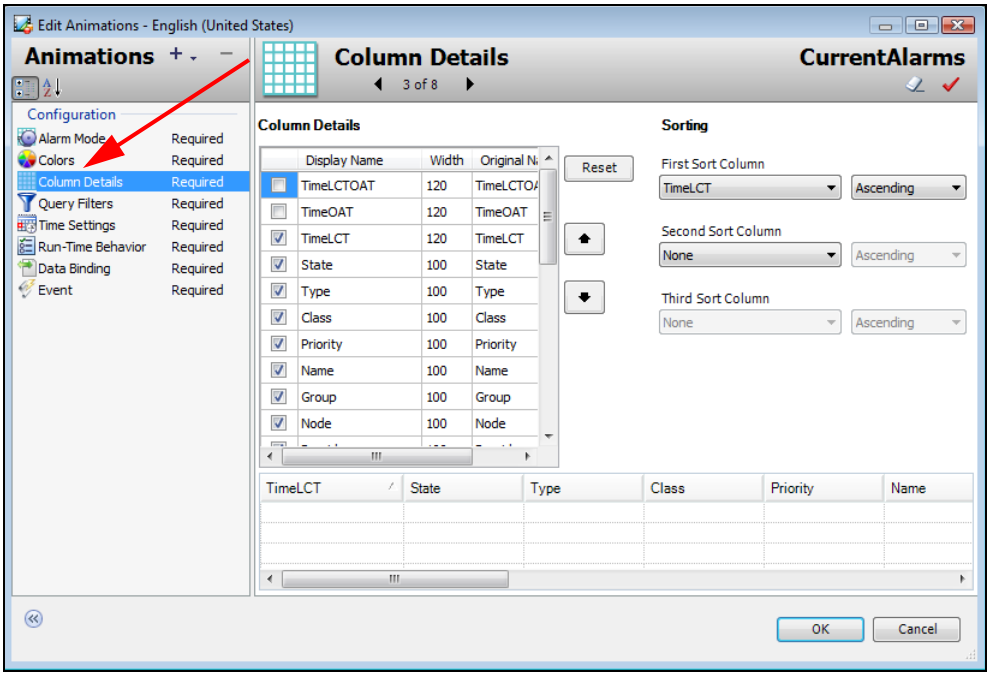
6. On the canvas, place the **Alarm Client** and stretch the width to fit the visible canvas area, making sure the height is **300** pixels.



7. Change the **Alarm Client** name to **CurrentAlarms**.

Modify Column Details

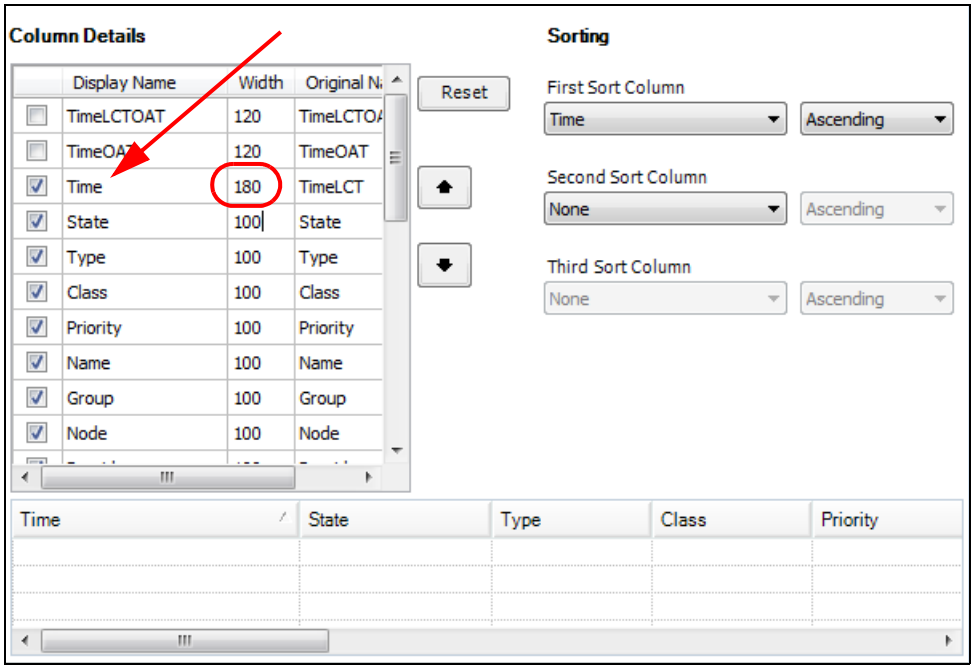
8. Double-click **CurrentAlarms**, then click **Column Details**.



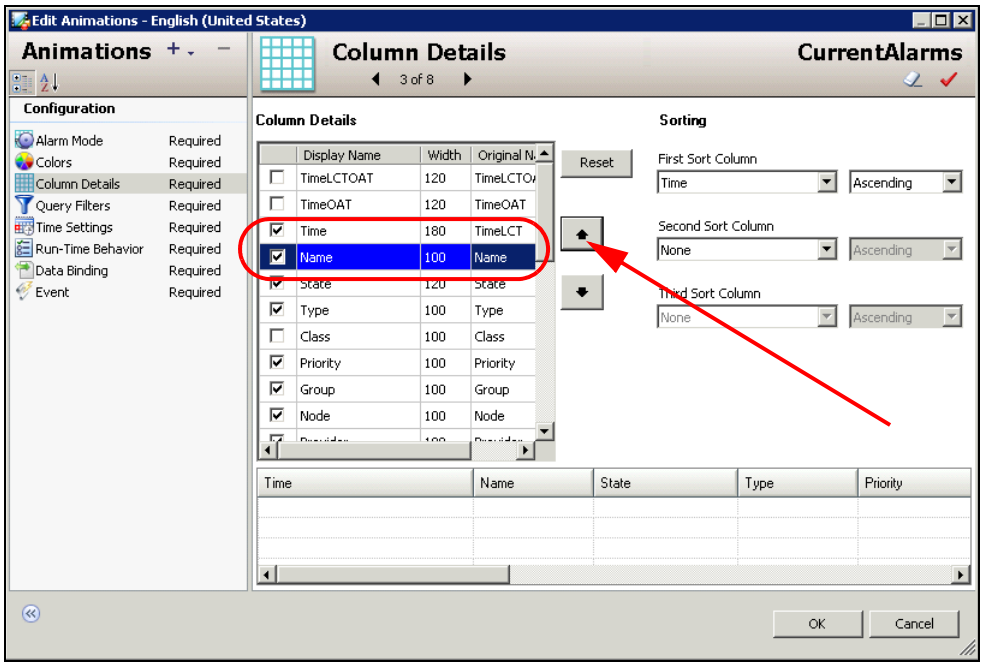
9. In the **Column Details** area, rename **TimeLCT** to **Time**.
10. In the **Time** row, increase **Width** to **180**.

    Notice that the column name changed in the bottom grid.

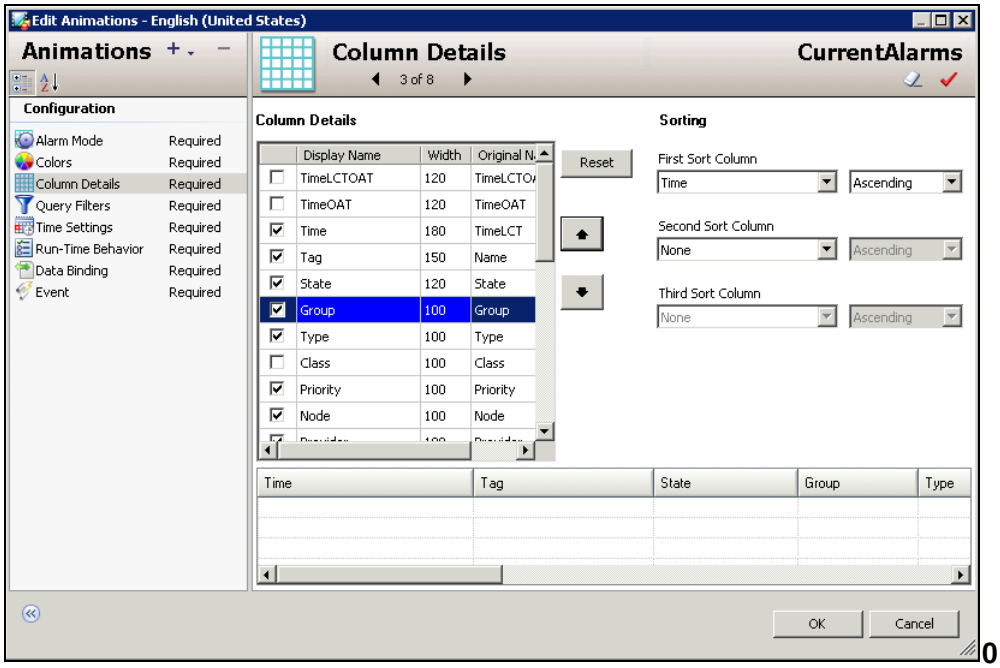
Note: **Width** measurements are measured in pixels.



11. Click **State** and change the **Width** to **120**.
12. Clear the **Class** check box to hide the column.
13. Click **Name** and use the up arrow to move it below **Time**.



14. In the **Display Name** column, change **Name** to **Tag**, and the **Width** to **150**.
15. Move **Group** directly below **State**.



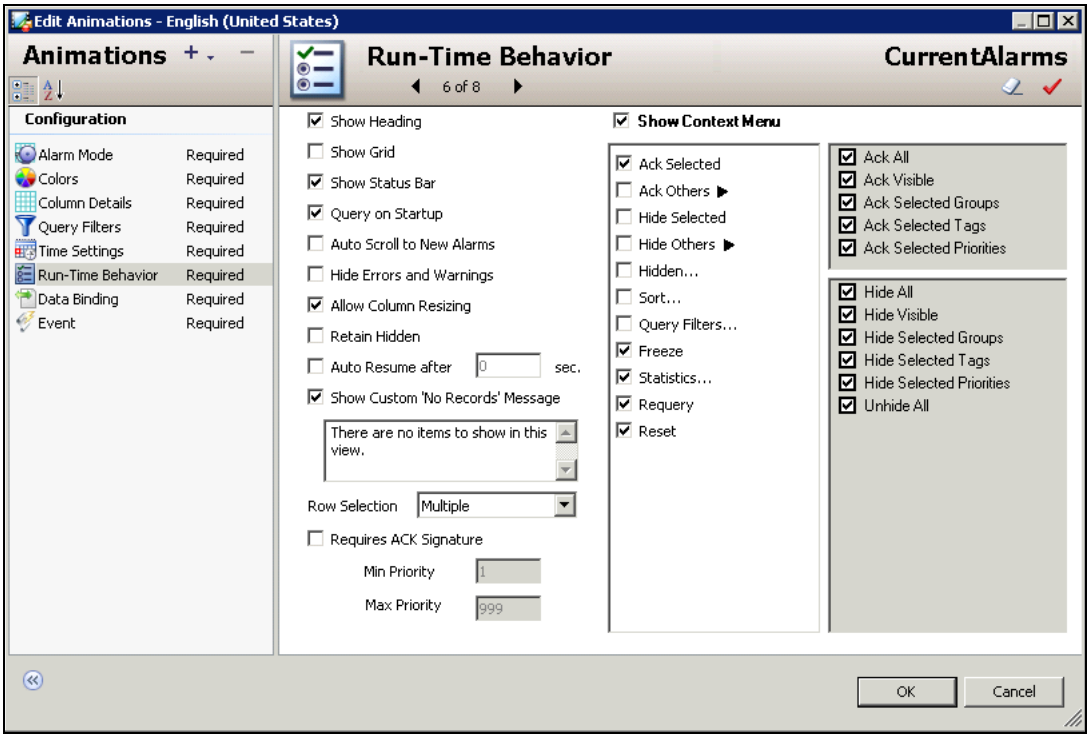
16. Scroll down and check **Value** and **Operator** to enable them.

Define Runtime Behavior

17. In **Configuration** pane, click the **Run-Time Behavior** configuration to display available configuration options.
18. Check **Show Custom 'No Records' Message**.
19. Enter the following in the message field:

    There are no items to show in this view.

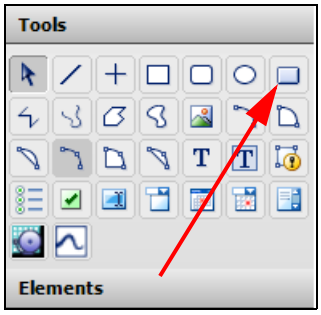
    This message appears when there are no records to display.
20. In the **Show Context Menu** list, clear the check box for the following options:
    - **Ack Others**
    - **Hide Selected**
    - **Hide Others**
    - **Hidden**
    - **Sort…**
    - **Query Filters…**



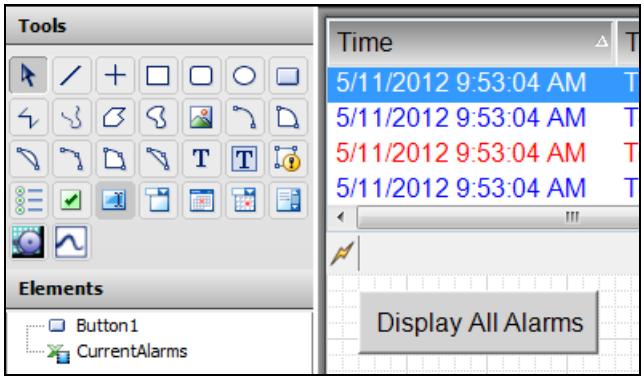
21. Click **OK**.

Create Buttons to Display and Acknowledge Alarms

22. Click the **Button** tool, then click and drag to create a button below the alarm display.



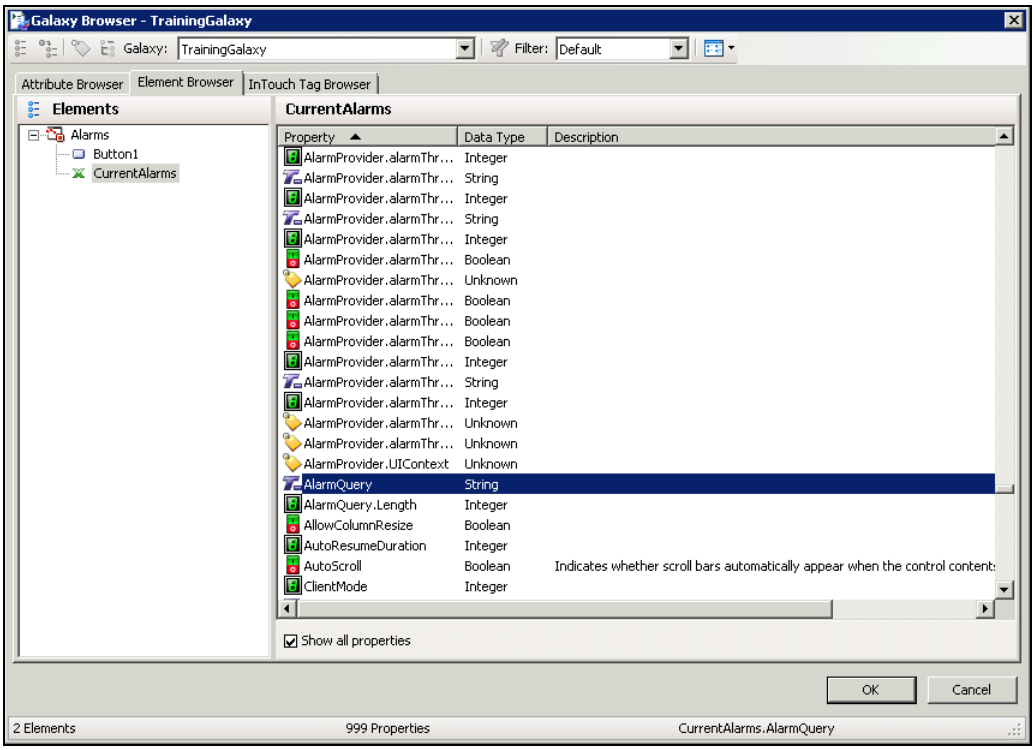
23. Label the button **Display All Alarms**.
24. Adjust the button size to fit the text.



25. Double-click the button to open the **Edit Animations** dialog box.
26. Click **+** and select **Fill Style**.
27. In the **States** area, click **Boolean**.



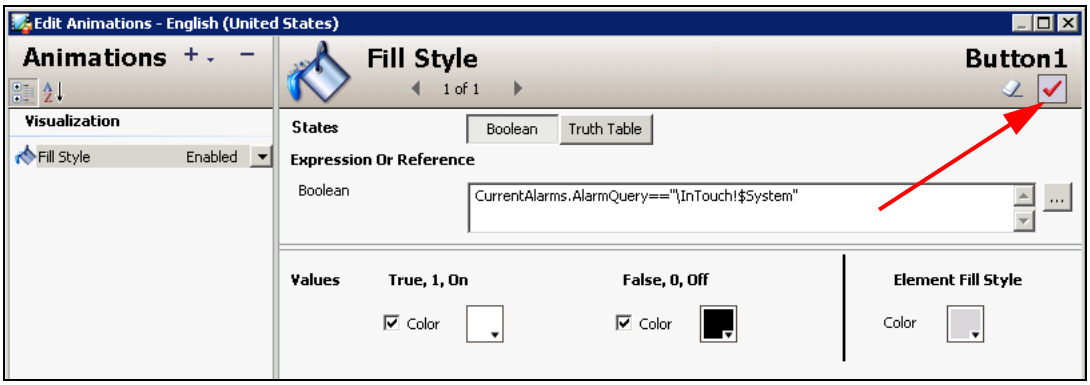
28. In the **Boolean** expression field, double-click to open the **Galaxy Browser**.
29. On the **Element Browser** tab, click the **CurrentAlarms** element and check **Show all properties**.
30. Scroll down and click **AlarmQuery**.



31. Click **OK**.
32. Add the following to the **Boolean** expression:

    == "\InTouch!$System"

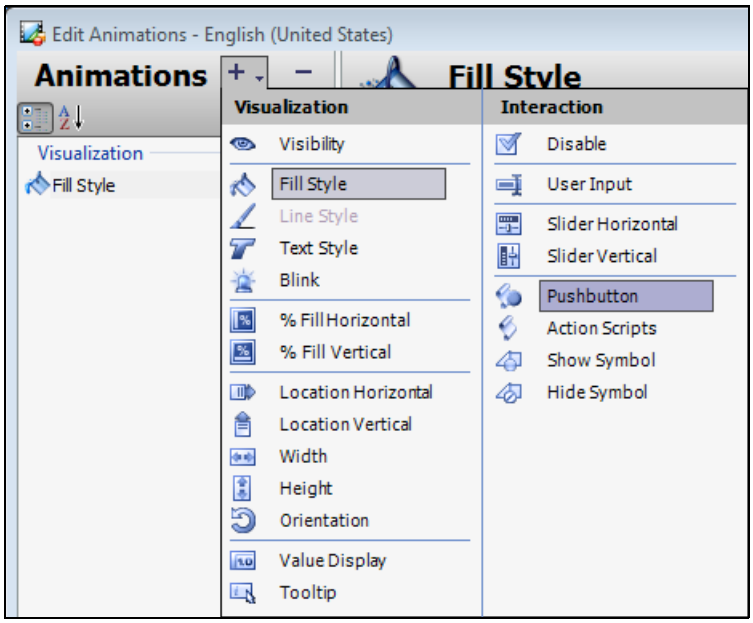
33. Click the **Validate** button to validate the expression.



34. In The **Values** area, change the default **True, 1, On** color to **Green**.
35. In the **False, 0, Off** area, uncheck the **Color** check box.

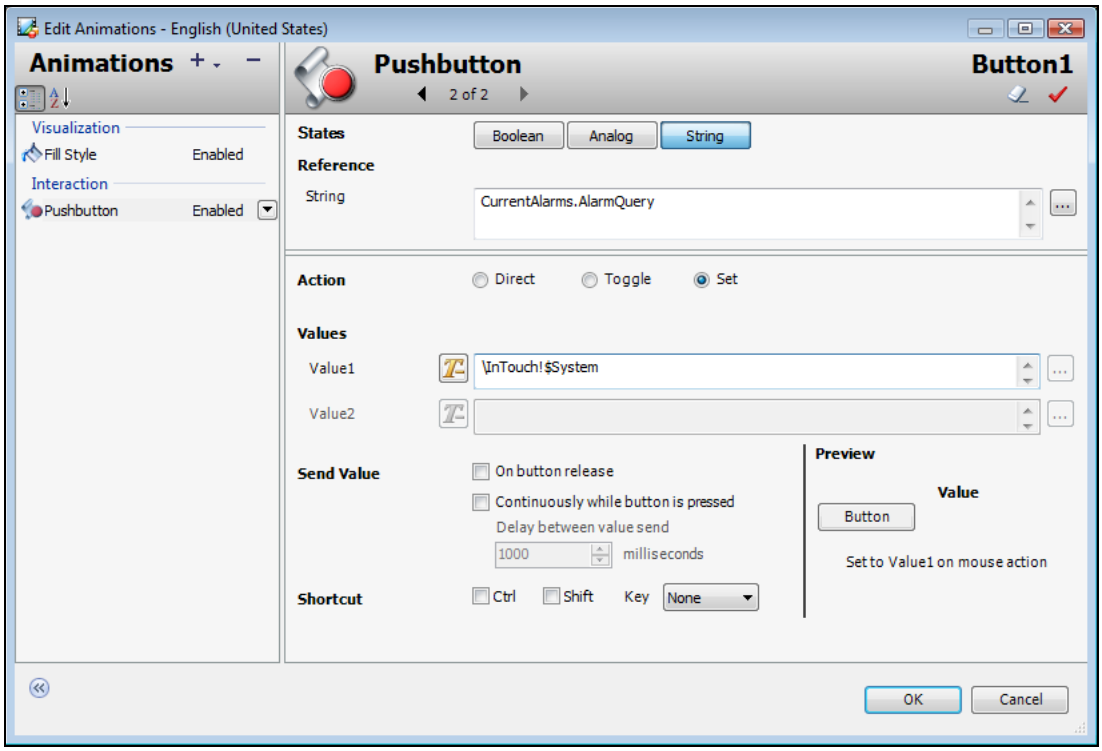


36. Click **+** and select **Pushbutton**.



37. Animate the **Pushbutton** state with the following settings:

States:	String
Reference String:	CurrentAlarms.AlarmQuery
Action:	Set
Value1:	\InTouch!$System



38. Click **OK**.

39. Duplicate the **Display All Alarms** button and change the text to **Display Mixer Alarms**.
40. Place **Display Mixer Alarms** next to the first button.
41. Double-click the **Display Mixer Alarms** button and replace **$System** with **Mixer** as follows:

    Fill Style

    Boolean: CurrentAlarms.AlarmQuery == "\InTouch!Mixer"

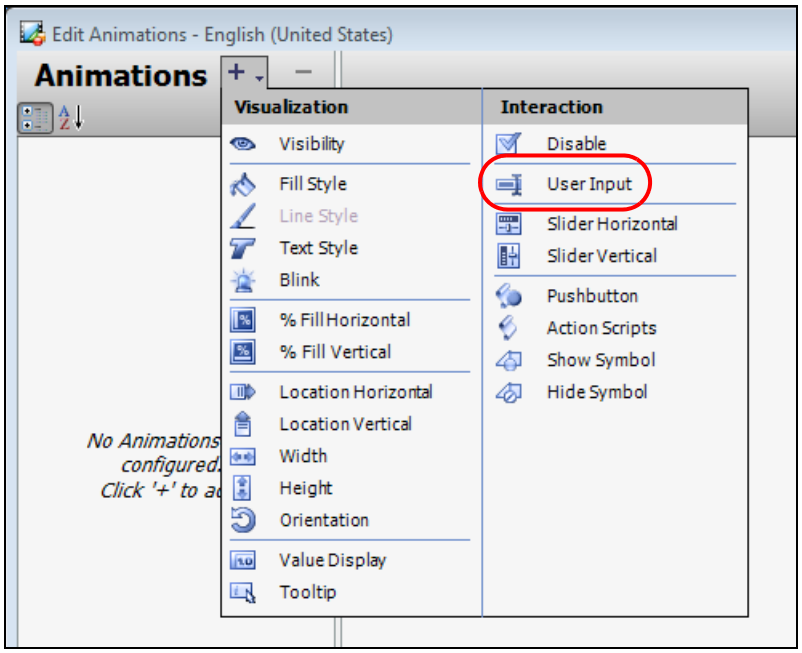
    Pushbutton

    States: String

    Value1: \InTouch!Mixer
42. Click **OK**.
43. Create a new button named **Ack All Alarms**.

Note: Do not duplicate an existing button.

44. Place it next to the **Display Mixer Alarm**.
45. Double-click the **Ack All Alarms** button to animate.
46. Click **+** and select the **User Input** animation.



47. Click **Boolean**, then in the **Reference** field enter **InTouch:$System.Ack**.

48. Animate the remaining options as follows:

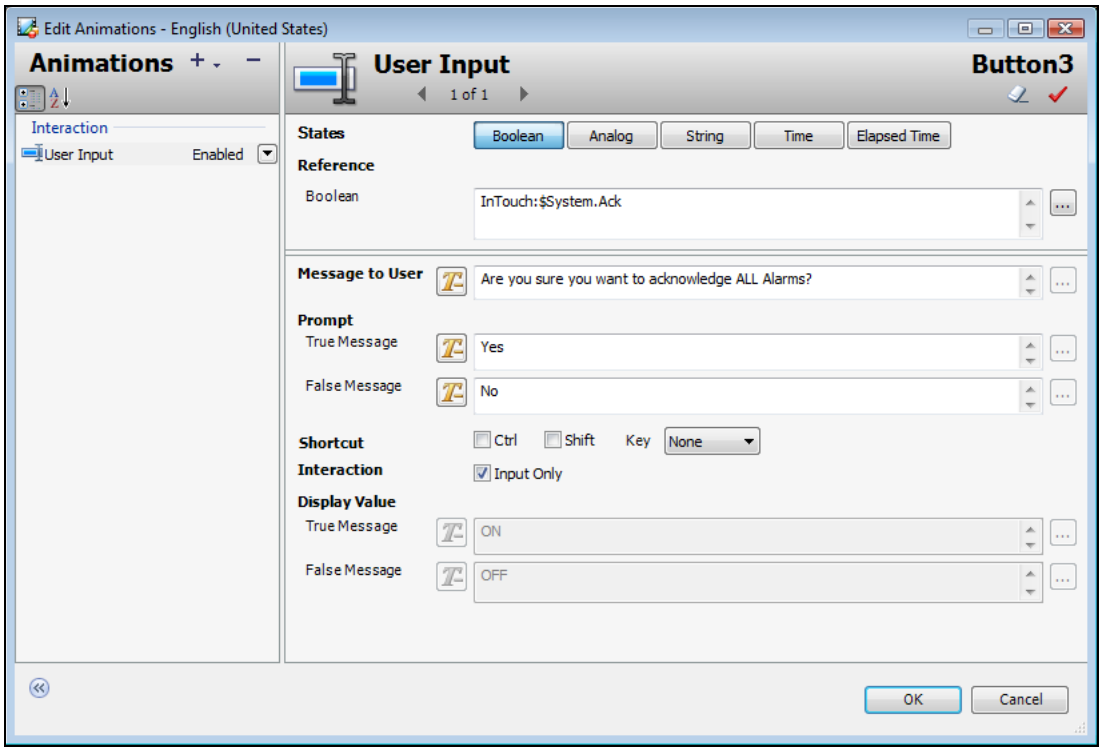
    Message to User: Are you sure you want to acknowledge ALL Alarms?

    Prompt

    True Message: Yes

    False Message: No

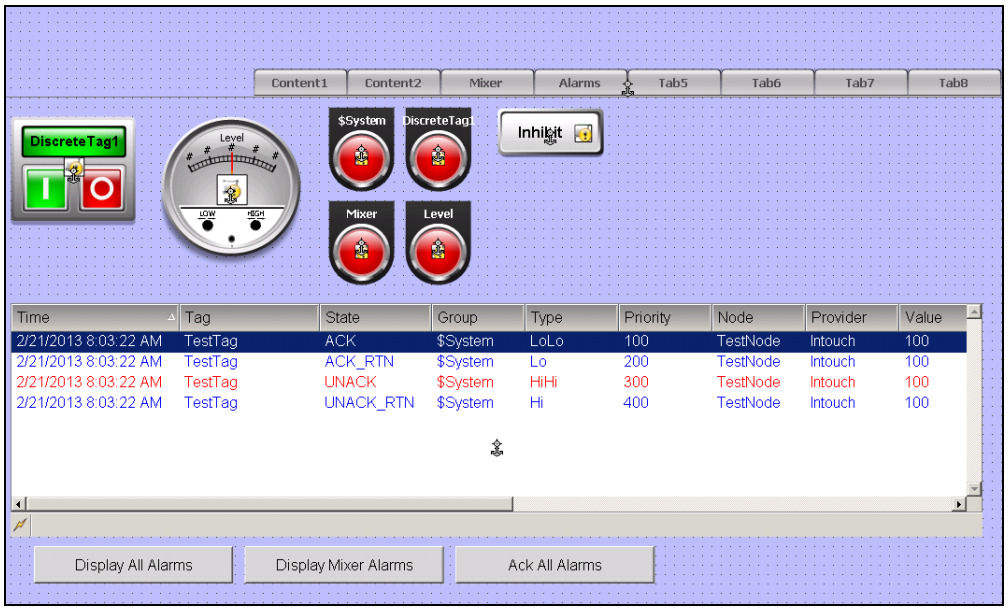
    Interaction: Input Only (checked)



49. Click **Validate** ☑.
50. Click **OK**.
51. Click **Save and Close**.
52. Enter an appropriate check in **Comment** and click **OK**.

Display Alarms Records

53. In WindowMaker, open the **Menu** and **Alarms** windows.
54. Embed your new **Alarms** symbol in the **Alarms** window.
55. Resize the symbol to fit your window, as shown.

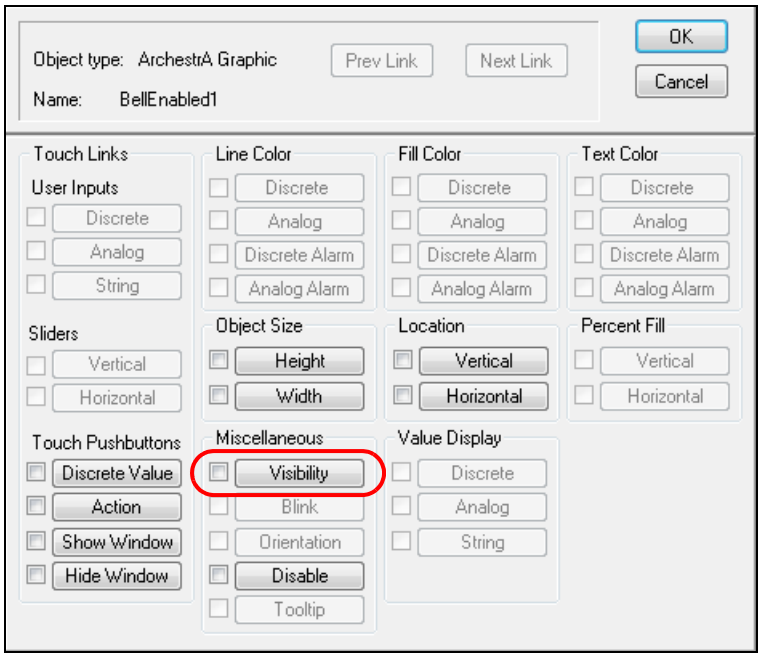


Leave room at the bottom of the window to accommodate additional graphics in a future lab.

56. In the **Menu** window, embed the **Widgets\BellEnabled** symbol above the **Alarms** tab, and resize to fit, as shown.

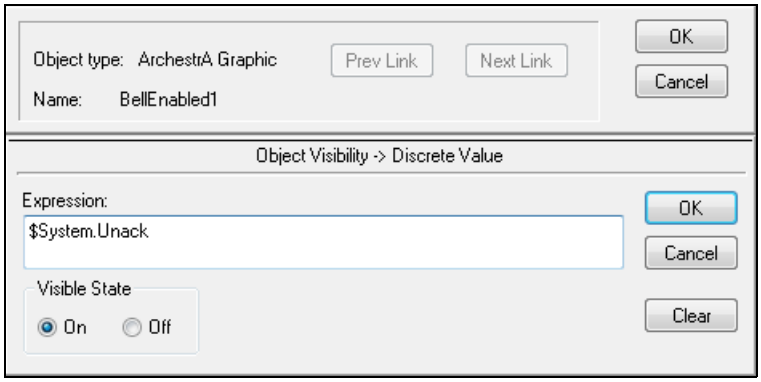


57. Right-click the bell symbol and select **Animation Links**.
58. In the **Miscellaneous** area, click **Visibility**.



59. In the **Expression** field, add the following visibility expression:

    $System.Unack

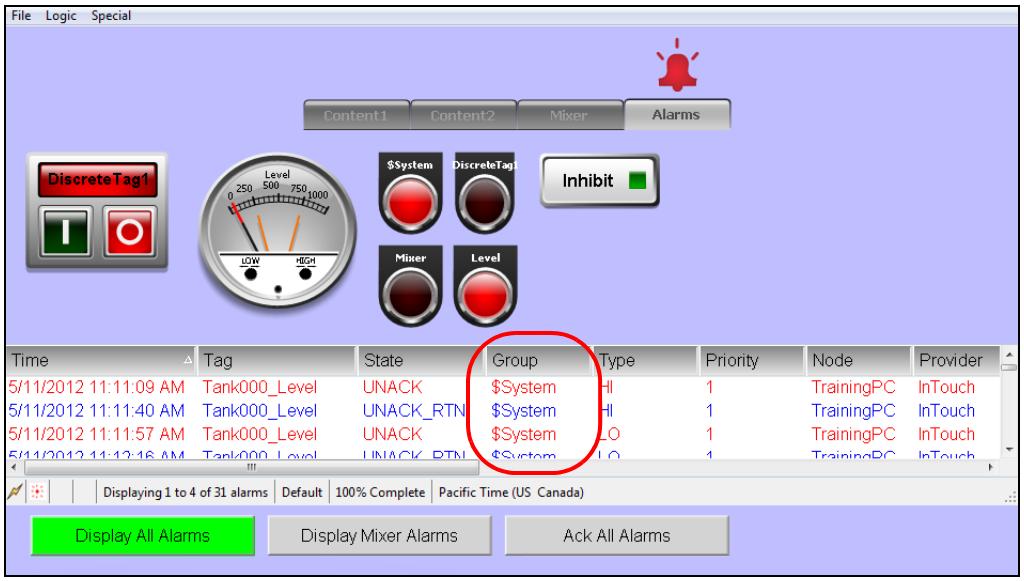


    This indicates that when there are unacknowledged alarms, the symbol is visible.

60. Keep the remaining default and click the top **OK** button.

Test What You Have Built

61. Click **Runtime**, then ensure the **Inhibit** is disabled so you can observe all alarms.
62. Click **Display Mixer Alarms** to review only alarms associated with the **Mixer** alarm group.
63. Click **Display All Alarms**, and scroll to look at your unacknowledged alarms.

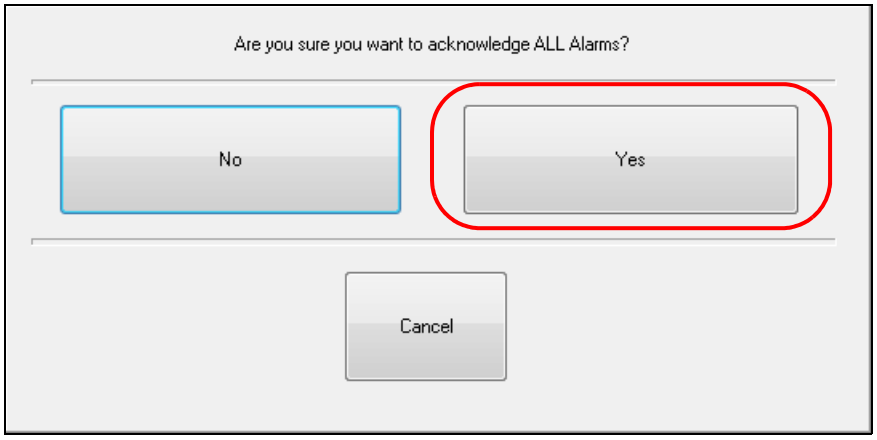


64. Trigger the **DiscreteTag1** alarm.

    Notice that the bell symbol is visible in the **Menu** window.

65. Click **Ack All Alarms**.

    The acknowledgment dialog box appears.

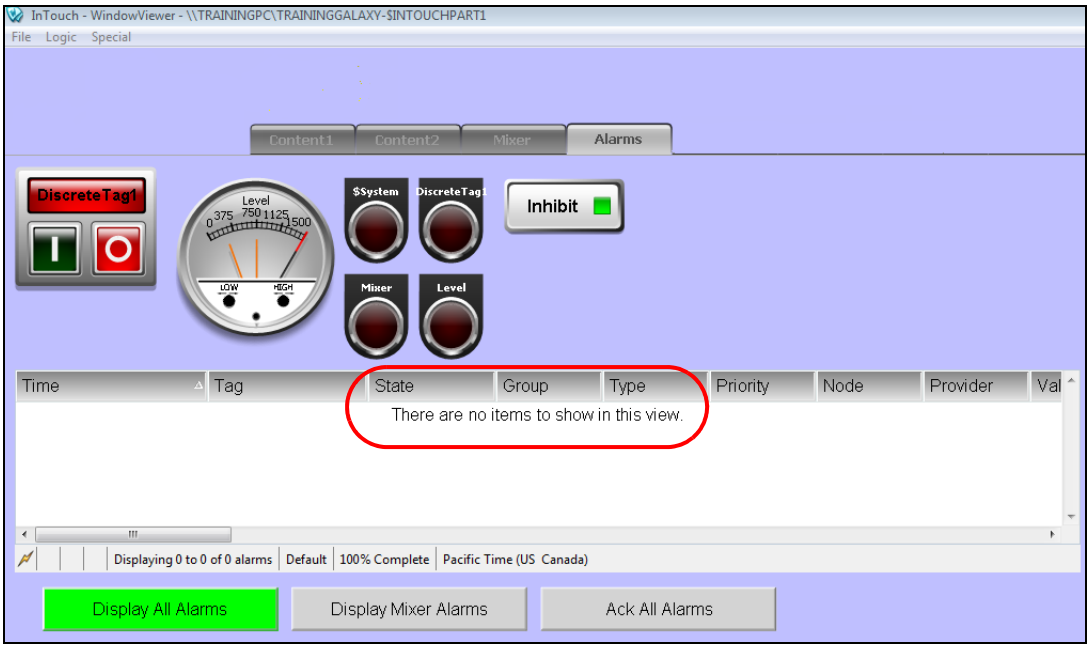


66. Click **Yes**.

    Notice that the bell symbol in the **Menu** window disappears.

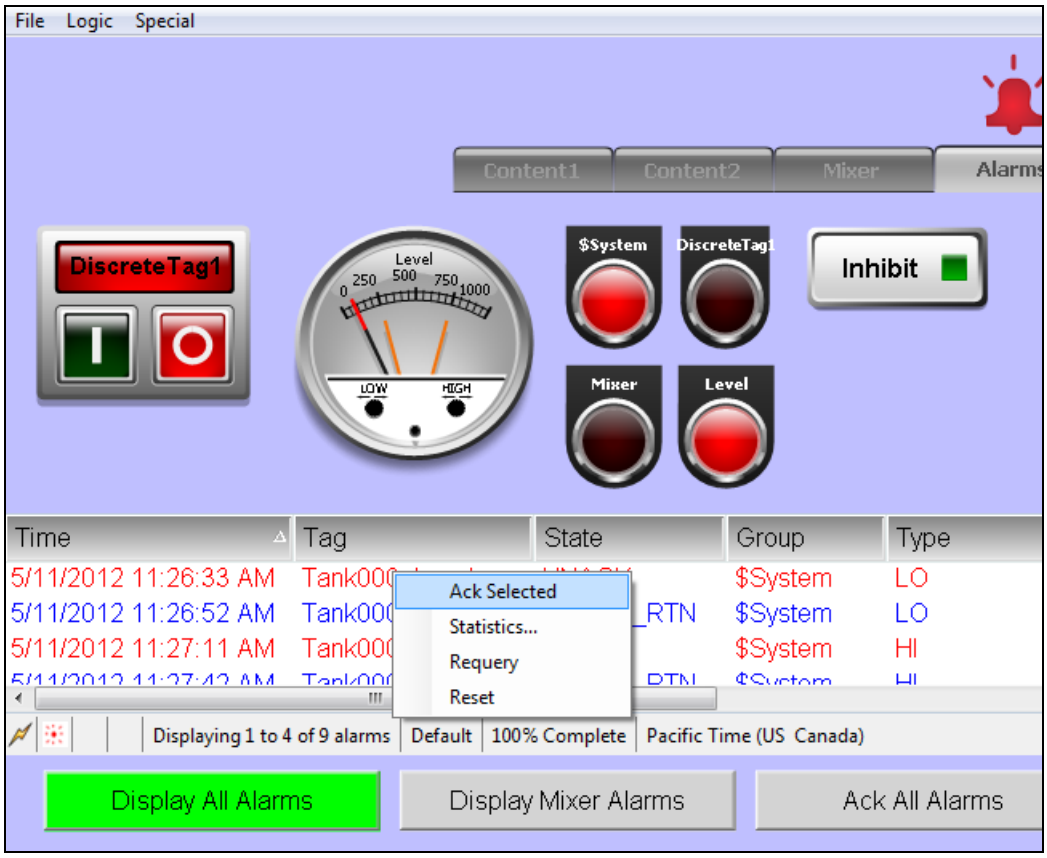
67. Turn off **DiscreteTag1**.
68. Click **Inhibit**.

    Your **Alarm Client** message appears.

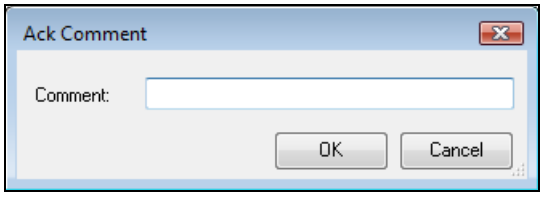


If the message does not display you may need to resize the embedded **Alarms** symbol in WindowMaker.

69. Trigger a **DiscreteTag1** alarm.
70. In **Alarm Client**, click the **DiscreteTag1** alarm record to select it.
71. Right-click the selected **DiscreteTag1** alarm record and select **Ack Selected** to acknowledge the alarm.



The **Ack Comment** dialog box appears.



72. Enter a comment in the field.
73. Click **OK**.

    The acknowledged alarm changes from red to black.
74. Click **Development!** to return to WindowMaker.

Section 3 – Alarm DB Logger

This section describes the Alarm DB Logger utility and how to configure the Alarm Client to connect to the logger utility.

Overview

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). The Alarm DB Logger is an Alarm Consumer. You configure it with an alarm query that defines which alarms you want to log. You 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.

Alarm DB Logger

Alarm DB Logger is an Alarm Consumer. You configure it with an alarm query that defines which alarms are to be logged. You 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.

Alarm DB Logger Manager takes care of user input and database configuration. The Alarm DB Logger Manager allows you to select the mode in which the Alarm DB Logger will run (either as a windows service or a normal application).

Note: The Alarm DB Logger Manager only saves the setting values into the registry. The utility is responsible for starting and stopping the Alarm DB Logger. It is also responsible for displaying the status of Smart Cache. When Alarm DB Logger Manager (almlogwiz.exe) is closed while wwalmlogger.exe is running (either by pressing the **Esc** key or by clicking the **X** button on the upper-right of the dialog box), the logging operation does not stop.

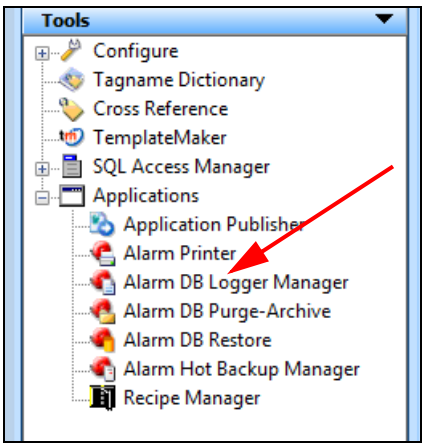
The progress control status indicates the percentage fill of the in-memory buffer with alarm records. The alarms are buffered when the SQL Server connection is down and/or when the alarms are coming at a rate faster than the logging rate of Alarm DB Logger.

The Alarm DB Logger Manager provides you with 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 (depending on the time out for a connection attempt associated with SQL Server)
- Store the setting in the registry

Alarm DB Logger Manager

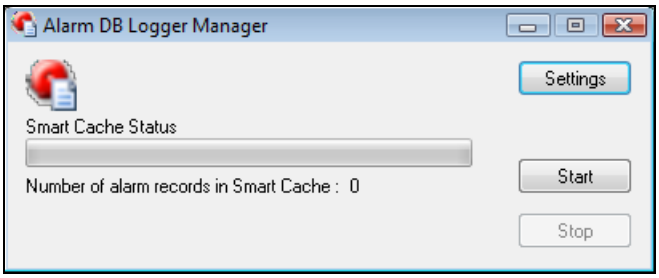
Access the **Alarm DB Logger Manager** from the **Tools** pane, **Applications** tools.



The Alarm DB Logger logs alarm data into the database. If the OLEDB Provider is SQL Server, you need to 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.

The Alarm DB Logger consists of a Manger Utility and a Configuration Utility. When enabled, the Manager Utility runs as a background task, and continues running even if WindowViewer stops.

This utility is a separate executable that solely takes care of starting and stopping the logging operations. It starts and works either as a service or a normal application (depending upon the running mode you select in the Alarm DB Logger Manager). The logging utility retrieves the setting information from the registry and performs the logging.

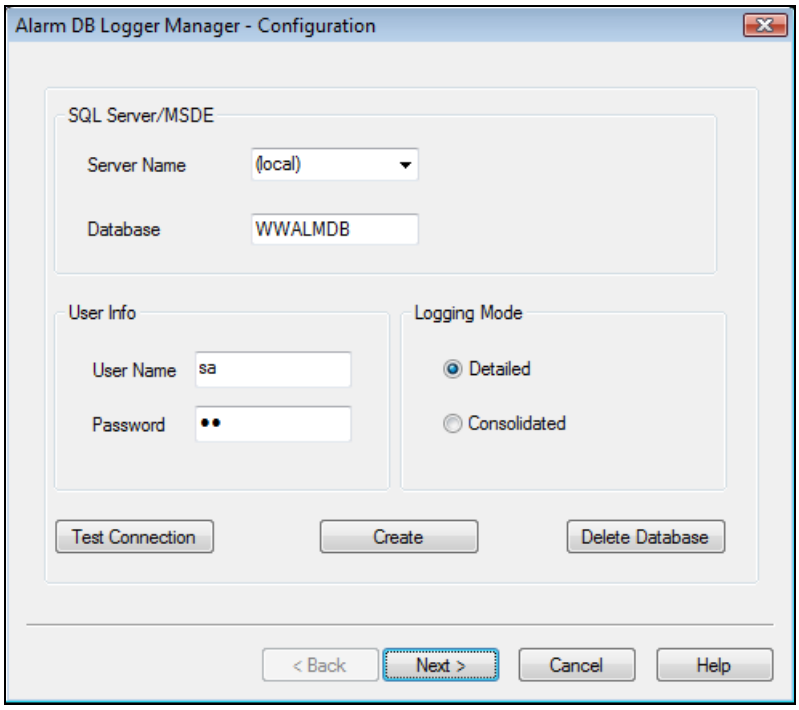


Configuration Utility

Before you begin using the Alarm DB Logger, you must configure a few items, such as your database connection, your query list, the logging mode and so on. This information is configured via the Alarm DB Logger Manager.

The first **Alarm DB Logger Configuration** dialog box deals with your database connection. You can either select SQL Server or Microsoft Data Engine (MSDE) for storage of data. The **Alarm DB Logger Configuration** utility also allows you to create necessary data structures (tables, views, and stored procedures) and to test your database connection.

Before you begin using the Alarm DB Logger, you must configure items, such as your database connection, your query list, and the logging mode. This information is configured using the **Alarm DB Logger Manager** where you define the details and location of your database connection.



The utility also allows you to create necessary data structures such as tables, views, and stored procedures, and to test your database connection.

When you have configured your settings for the Alarm DB Logger, click the **Start** button to start.

The alarm record cache will load. When the cache reaches zero, you can close the **Alarm DB Logger Manager**. The utility continues to run in the background even though the dialog box has closed.

Alarm DB Logger Features

The InTouch Alarm DB Logger stores the UNACK duration value in the alarms database. The UNACK Duration is the ACK time minus the time of the most recent alarm transition, where all dates and times are represented in GMT. Whenever an alarm record is ACKed, the Alarm DB Logger calculates the UNACK Duration as follows:

- If the alarm has been through a sub-state transition, for example, Hi to HiHi, the UNACK duration is the ACK time minus the sub-state time. Otherwise, the UNACK duration is the ACK time minus the alarm origination time.
- If the alarm record transition type is any type other than ACK, the Alarm DB Logger logs the default value for the UNACK Duration column (NULL value).

The Alarm DB View Control also displays the return duration. This is calculated as the return time minus the alarm origination time. The alarm duration is not stored in the database, but is calculated on demand and presented by the database views.

Lab 12 – Viewing Alarm History with the Alarm DB Logger

Introduction

In this lab, you create and link to an alarm database and configure a new Alarm DB Logger display and use it to review historical alarming events.

Objectives

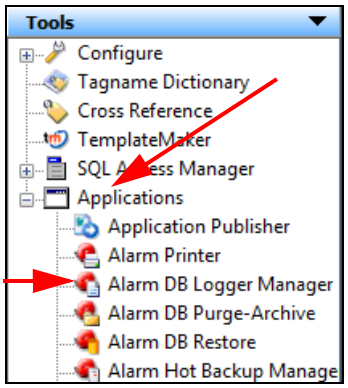
Upon completion of this lab, you will be able to:

- Enable and configure the Alarm DB Logger
- Configure an Alarm Client to connect to the database
- Review historized alarms with the Alarm Client

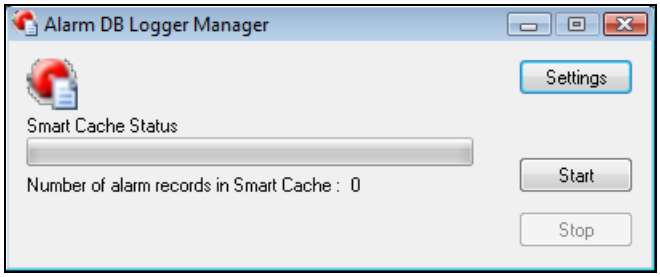
Alarm DB Logger

First, you start Alarm DB Logger. This process runs in the background, and continues running even if WindowViewer stops.

1. In the WindowMaker **Tools** pane, expand **Applications** and double-click **Alarm DB Logger Manager**.



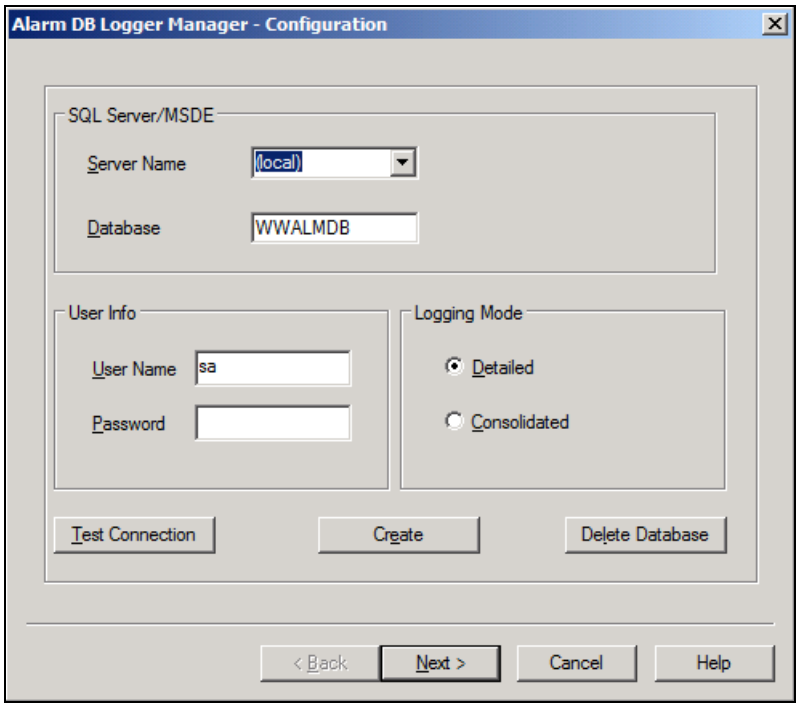
The **Alarm DB Logger Manager** dialog box appears.



2. Click **Settings** to configure the **Alarm DB Logger**.

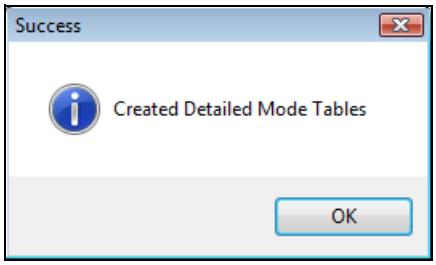
Note: There may be a short pause while the **Configuration** dialog box loads.

The **Configuration** dialog box appears.



3. In the **SQL Server/MSDE** area, **Server Name** field, enter your student node name (provided by instructor).
4. Keep the default **Database** name.
5. In the **User Info** area, **User Name** field, leave the default **sa**.
6. Enter the password provided by your instructor.
7. Click **Create**.

   When complete, a **Success** message appears.



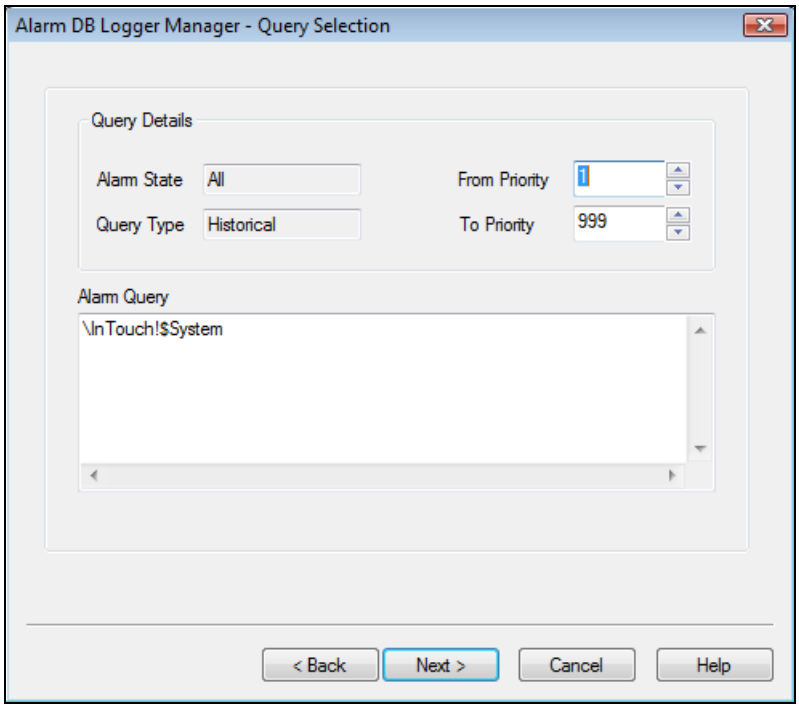
8. Click **OK**.

Log Mode Options

Next, you determine which alarms you want to record to the database.

9. Click **Next**.

   The **Query Selection** dialog box appears.

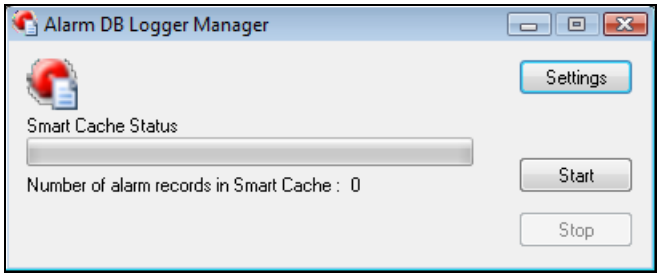


10. Keep all defaults and click **Next**.

    The **Advanced Setting** dialog appears.

11. Accept all defaults and click **Finish**.

12. Click **Start** to begin logging.



Alarm DB Logger Manager counts up all of the alarms in the queue and dumps those into the database one-by-one.

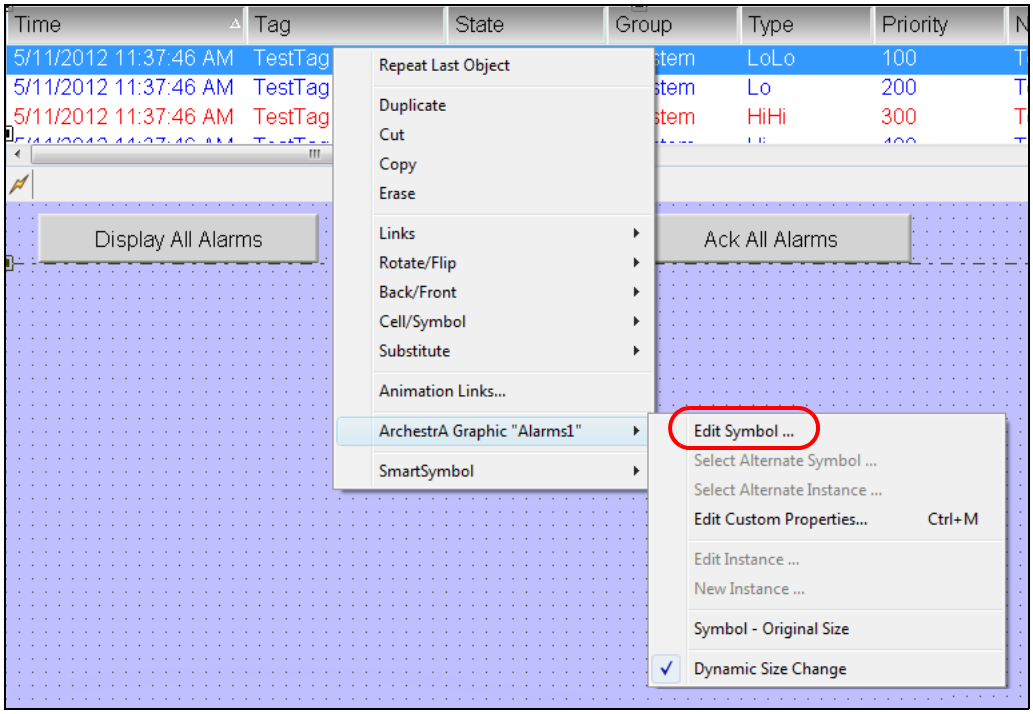
13. Click **X** to close the **Alarm DB Logger Manager** dialog box.

Note: The Alarm DB Logger (wwalmlogger.exe) continues running and recording alarms even after the configuration application is closed.

Historical Alarm Client

Now, you add another Alarm Client to the Alarms symbol.

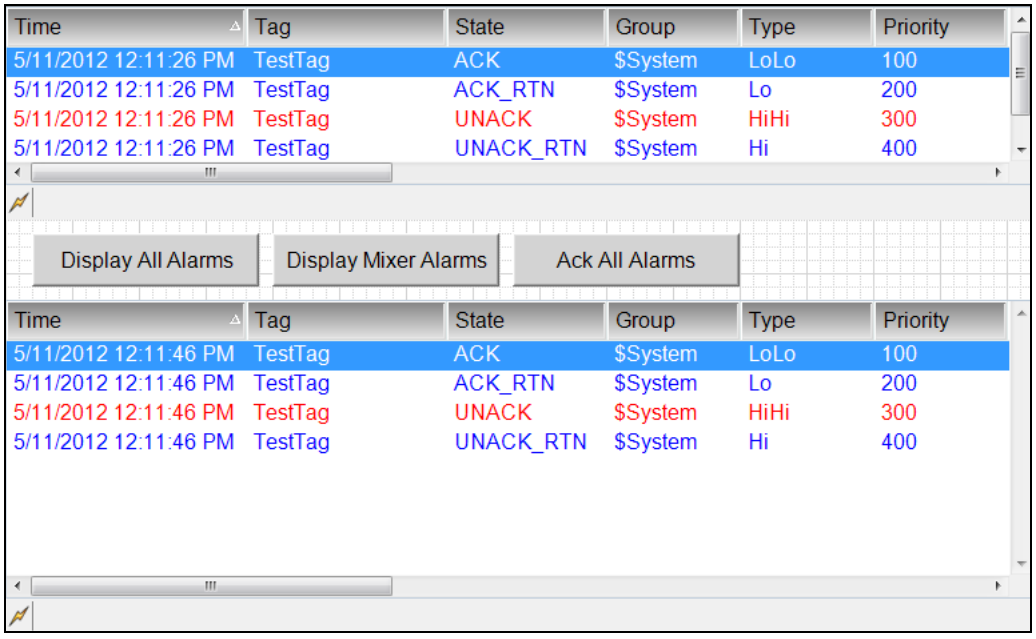
14. In WindowMaker, **Alarms** window, right-click the **Alarms** symbol and select **ArchestrA Graphic "Alarms1" | Edit Symbol**.



The **ArchestrA Symbol Editor** appears.

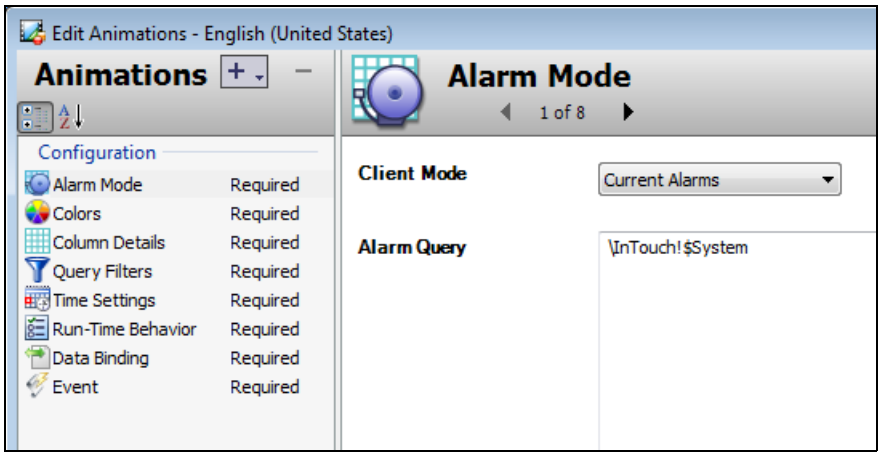
15. Duplicate the existing alarm client and name the new instance **HistoricalAlarms**.

16. Move it below the alarm buttons.



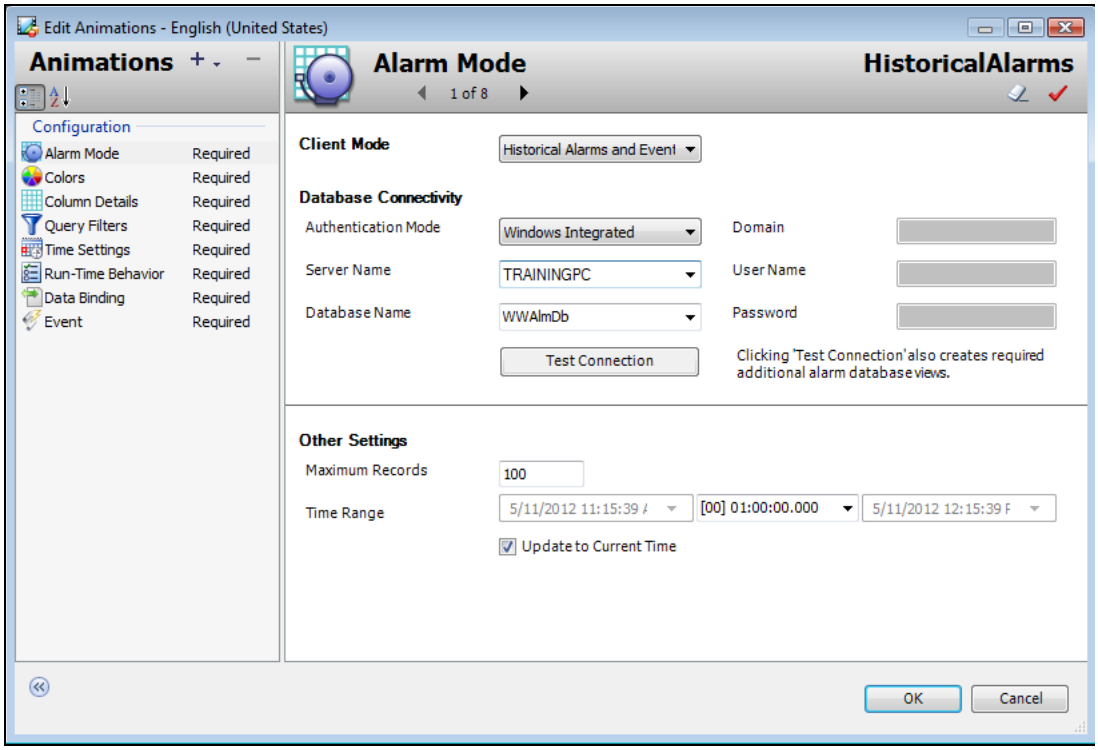
17. Double-click **HistoricalAlarms** to configure it.

    The **Edit Animations** window appears.



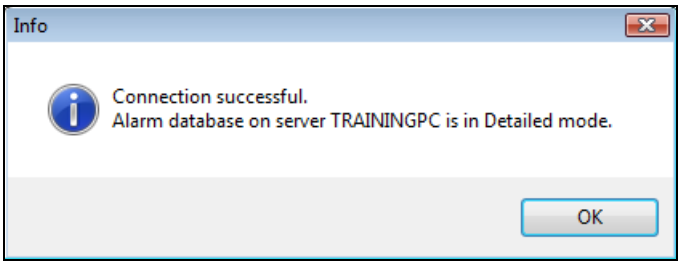
18. In **Configuration** pane, click **Alarm Mode**, then click the **Client Mode** drop-down list and select **Historical Alarms and Events**.

19. In the **Database Connectivity** area, enter the server name to your student node name (provided by instructor).



20. Click **Test Connection**.

    A successful connection message appears.

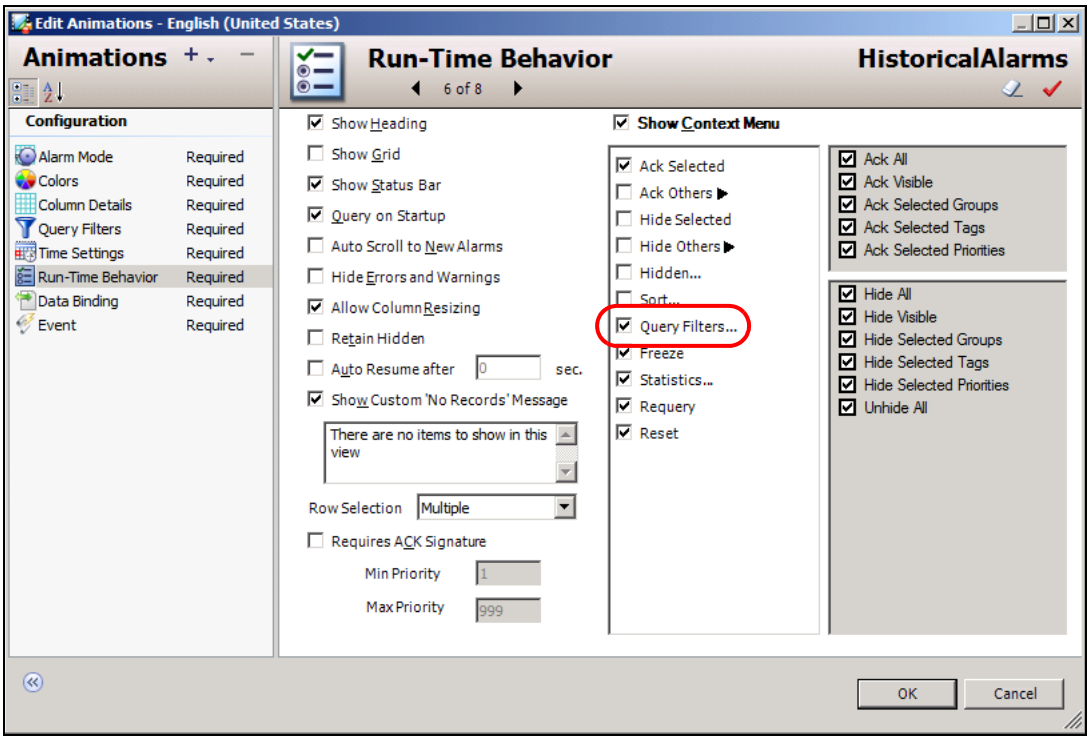


21. Click **OK**.

Add Query Filters

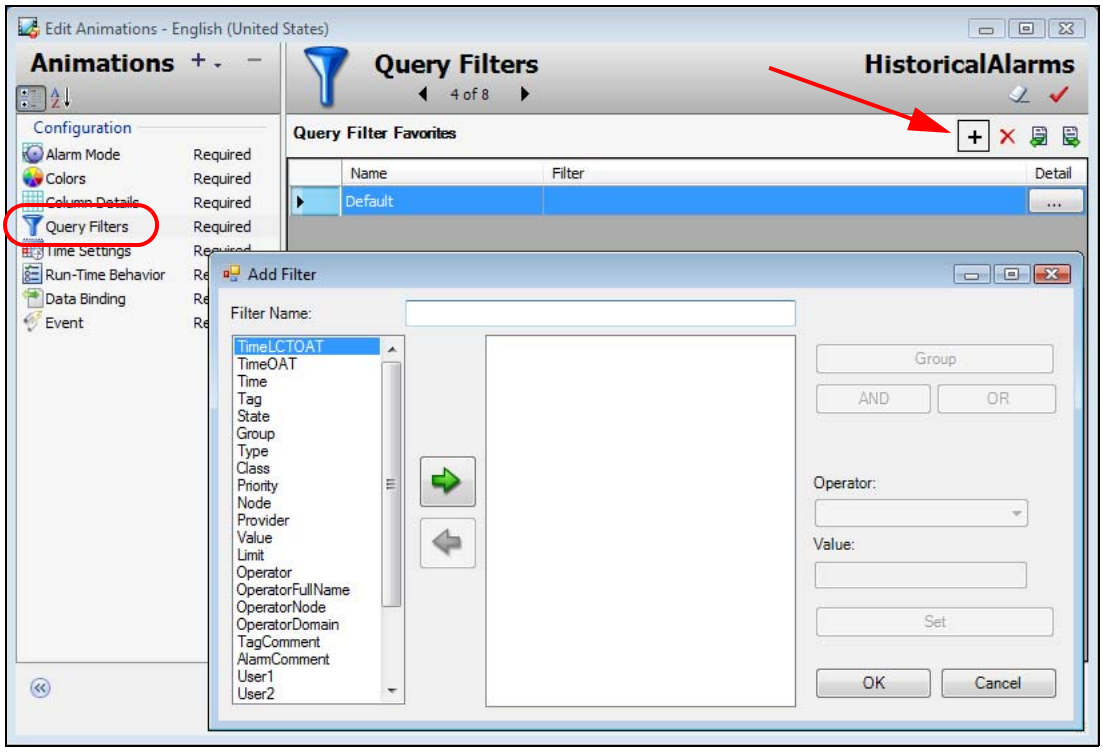
Next, you add a Query Filters option. You use query filters to do research with the Historical Alarm Client.

22. In the **Configuration** list, click **Run-Time Behavior**.
23. In the right pane, check **Query Filters**.



24. In the **Configuration** list, click **Query Filters**, then in the **Query Filter Favorites** area click **+**.

    The **Add Filter** dialog box appears.

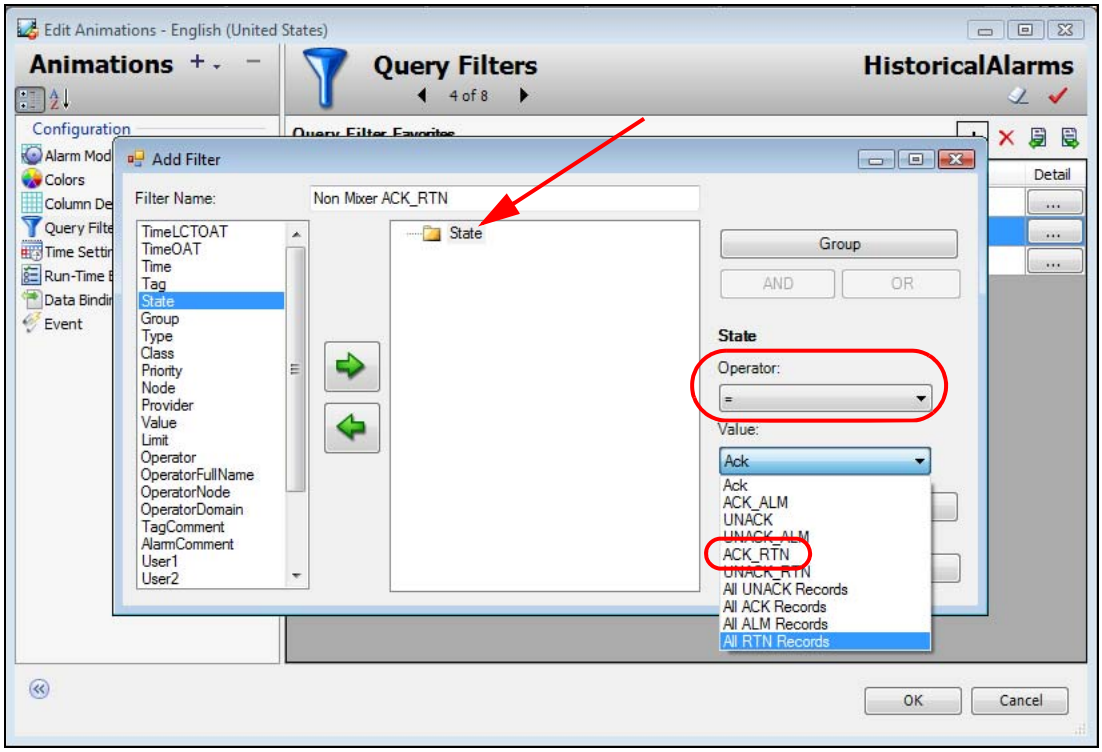


25. In the **Filter Name** field, enter **Non Mixer ACK_RTN**

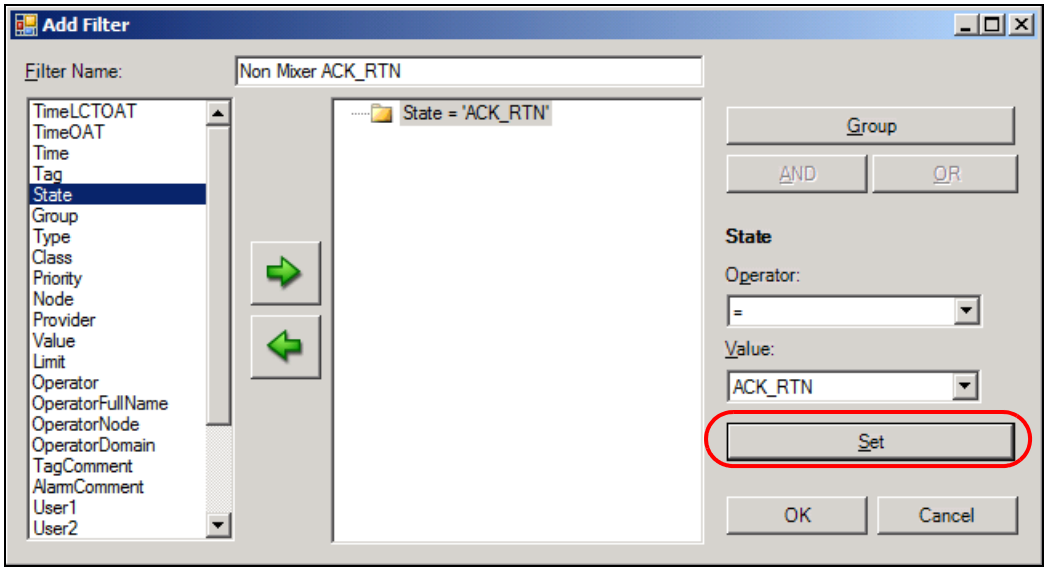
26. In the **Filter Name** list, click **State**, then click the button to move it to the pane to its right. Observe the new **State** filter has been added to the **Filter Definition** list.

27. In the **Filter Definition** list, select **State**, then confirm or modify the filter definition as follows:

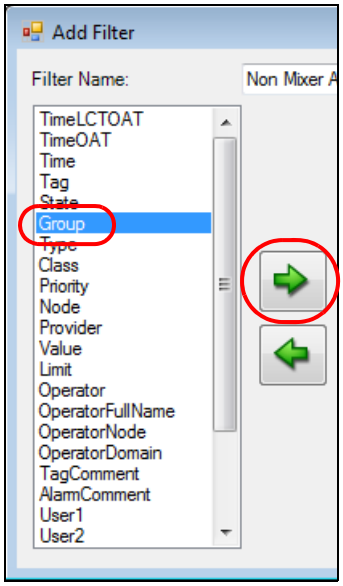
    Operator: = (default)
    Value: ACK_RTN



28. Click **Set** to configure this option.



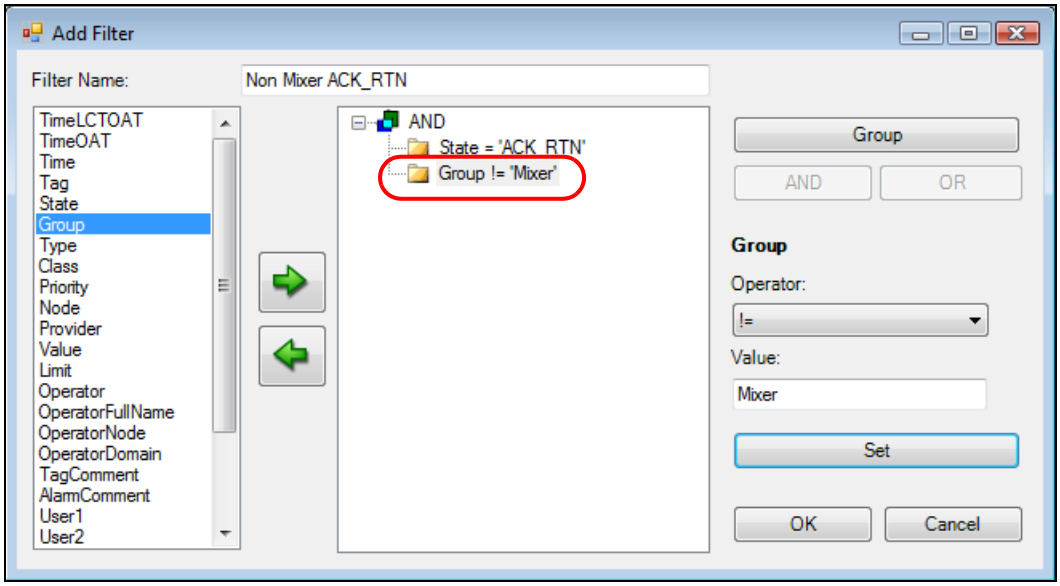
29. In the **Filter Name** list, click **Group**, then click the button to move it to the pane to its right.



30. In the **Filter Definition** list, select **Group** and modify its filter definition as follows:

    Operator: !=

    Value: Mixer

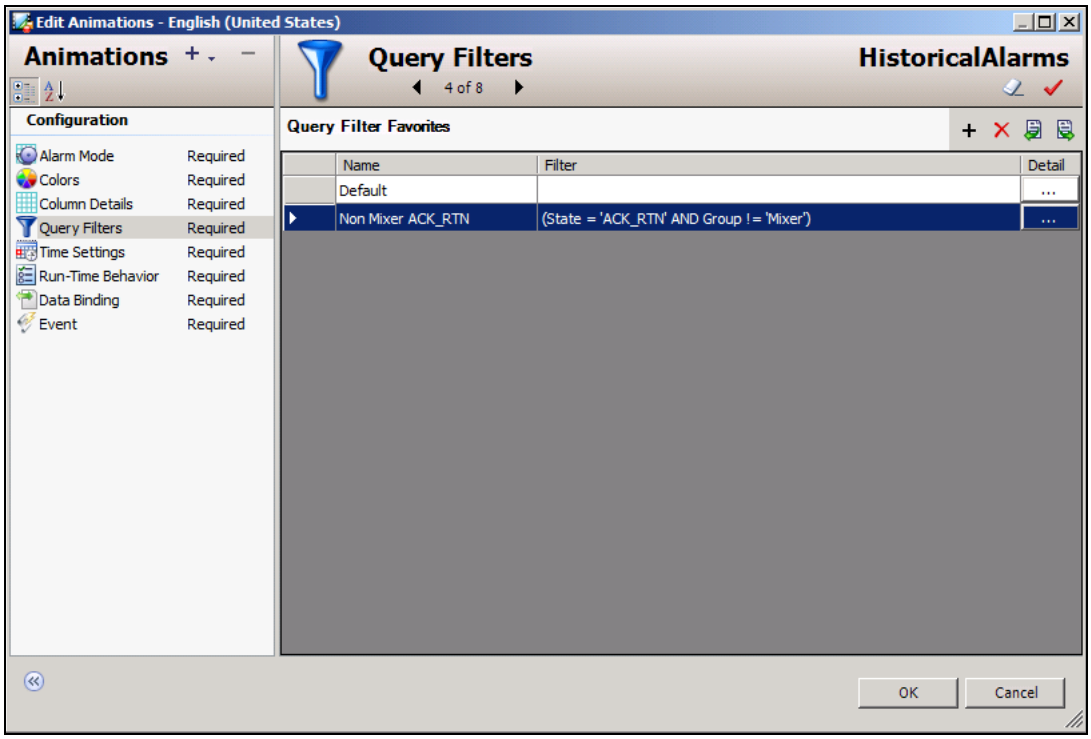


31. Click **Set**.

    The system will only look at the **Acknowledge Return To Normal** alarms for all items except items in the **Mixer** group.

32. Click **OK**.

    You return to the **Filter Animations** dialog box.



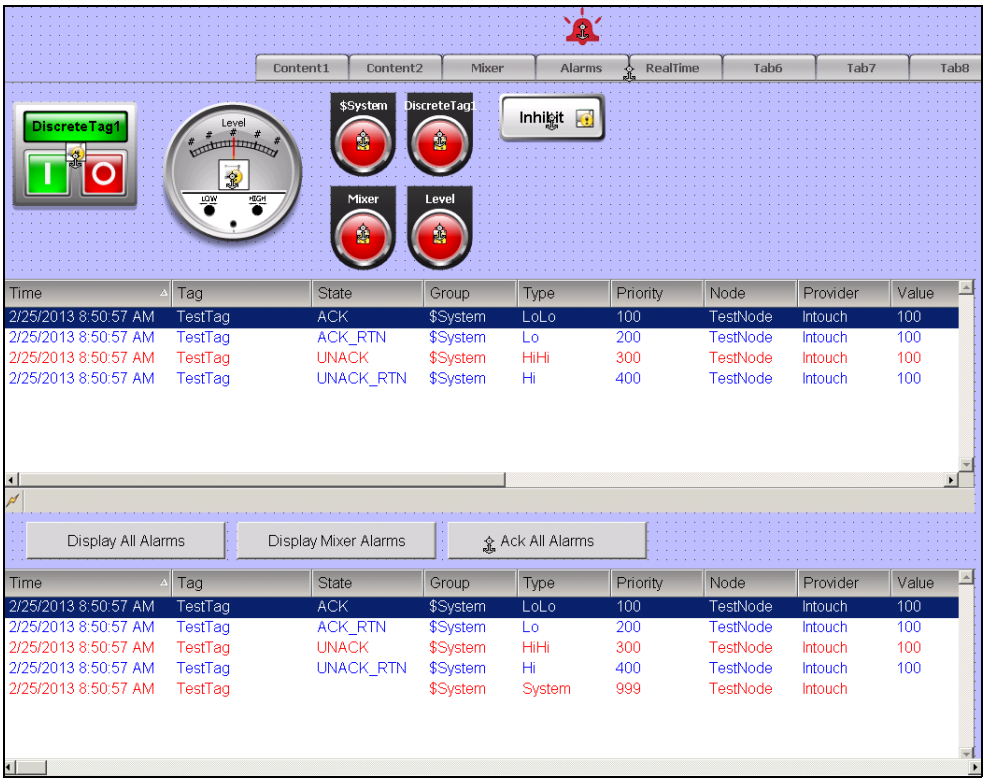
33. Click **+** to add another **Query Filter**.
34. In the **Filter Name** field, enter **Tank000_Level UNACK_RTN**.
35. Move the following items to the **Filter Definition** list, then confirm or modify the definition as follows:

    Click **Set** after each entry.

Option	Operator	Value
State	=	UNACK_RTN
Tag	=	Tank000_Level

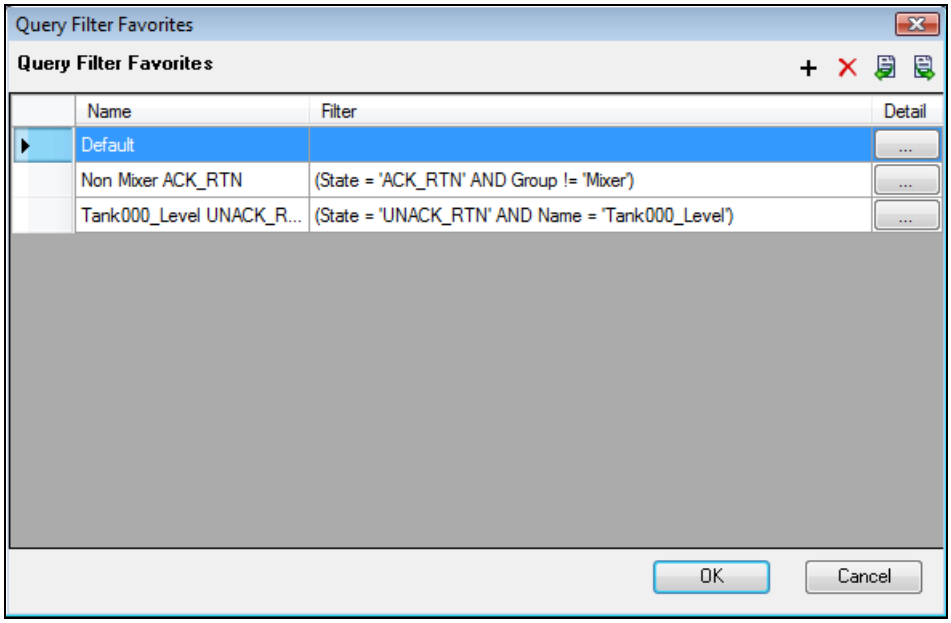
36. Click **OK**.
37. Click **OK** again to close the **Filter Animations** dialog box.
38. Click **Save and Close** to commit your change.
39. Enter an appropriate comment and click **OK**.

40. In WindowMaker, update your symbol.
41. Position and size the symbol to fit the bottom of the **Alarms** window as shown.



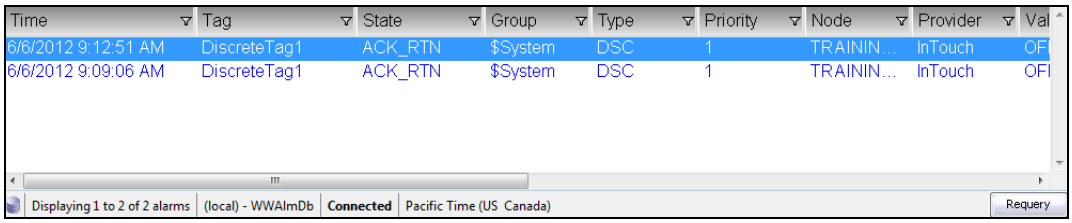
Test What You Have Built

42. Switch to **Runtime** and click the **Alarms** tab.
43. Trigger and acknowledge the alarm for **DiscreteTag1** two times.
44. Right-click **Historical Alarm Client** that you just created and select **Query Filters**.



45. In the **Query Filter Favorites** dialog box, click **Non Mixer ACK_RTN** and click **OK**.

    You can view all of the **DiscreteTag1** alarm occurrences in the display.



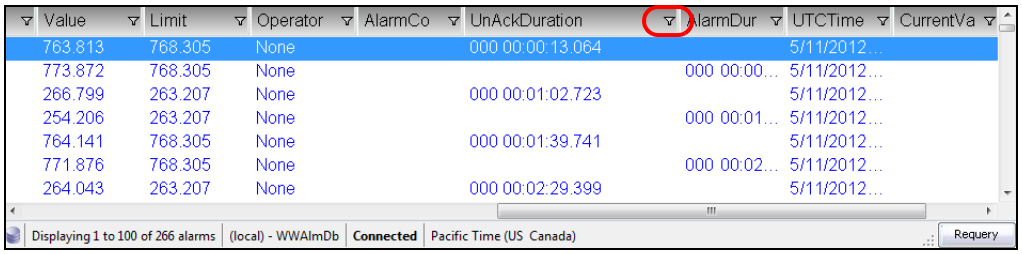
46. Verify there are several records in the **CurrentAlarms** display for the **Tank000_Level** tag.

Note: **Tank000_Level** alarms will only be displayed when the **Inhibit** button is not enabled.

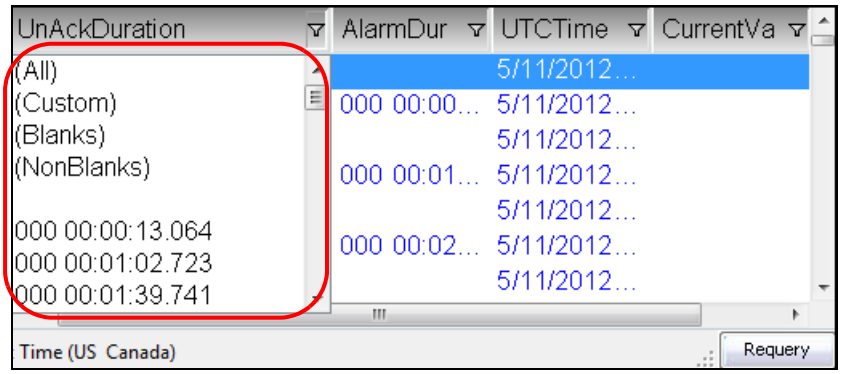
47. Click the **Ack All Alarms** button and click **Yes** to acknowledge all alarms.
48. Right-click **Historical Alarm Client** and select **Reset**.

    This restores the **Default** filter.

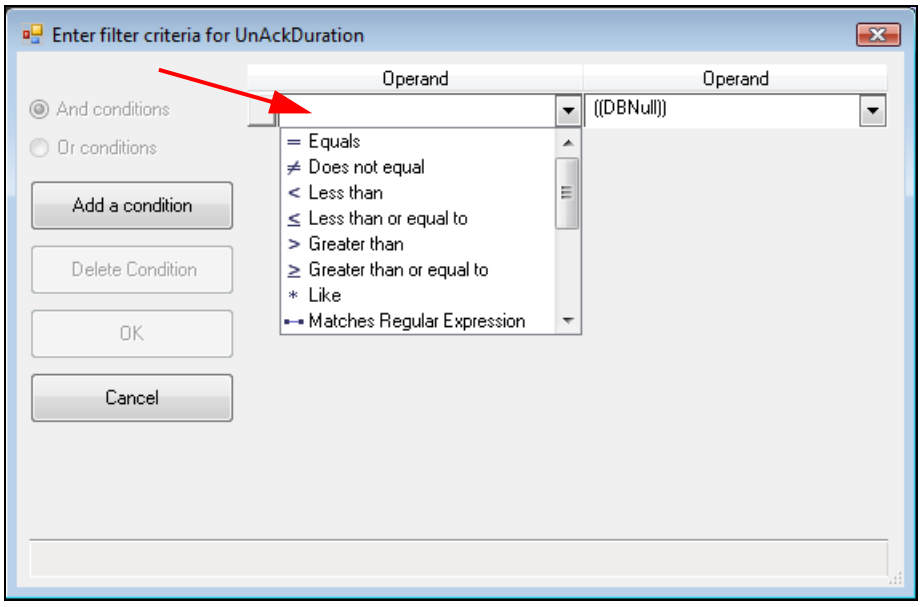
49. Scroll to the right and expand the **UnAckDuration** column.
50. Click the runtime query filter icon.



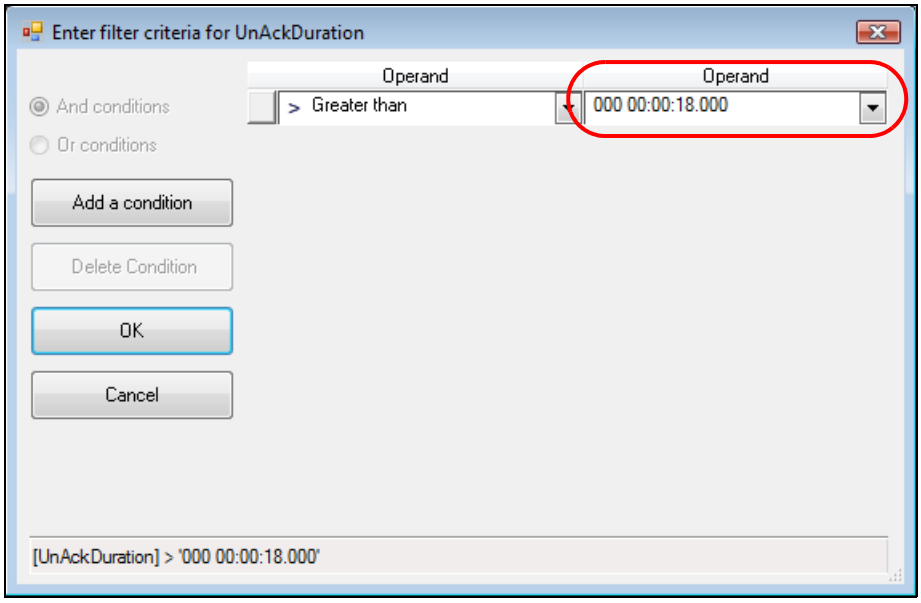
An option list displays.



51. Click **(Custom)** to open the **Enter filter criteria for UnAckDuration** dialog box.
52. Click the left **Operand** drop-down list and select **> Greater than**.



53. Click in the right **Operand** field and enter **000 00:00:18.000**.



Note: The divisions of this mask are days, hours, minutes, seconds, and milliseconds.

54. Click **OK**.

    Alarms greater than 18 seconds display.

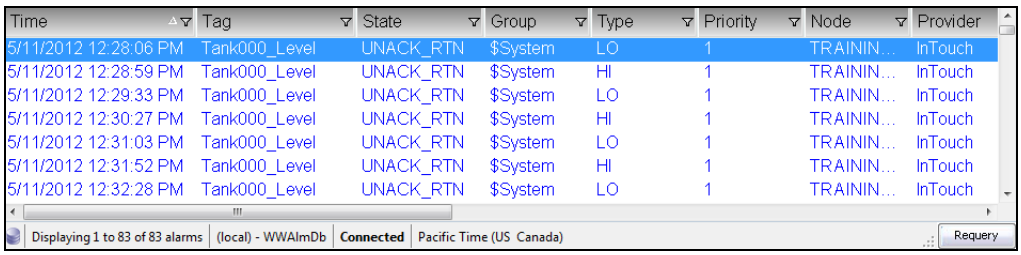
55. Return to the **Enter Filter** window and change the right **Operand** from 18 seconds (**000 00:00:18.000**) to 1 minute (**000 00:01:00.000**).

56. Click **OK**.

57. Review the results.

58. In the **Query Filter Favorites** dialog box, click **Tank000_Level UNACK_RTN** and click **OK**.

    This displays instances where an alarm for the **Tank000_Level** tag returned to a normal state before it was acknowledged.



Time	Tag	State	Group	Type	Priority	Node	Provider
5/11/2012 12:28:06 PM	Tank000_Level	UNACK_RTN	$System	LO	1	TRAININ...	InTouch
5/11/2012 12:28:59 PM	Tank000_Level	UNACK_RTN	$System	HI	1	TRAININ...	InTouch
5/11/2012 12:29:33 PM	Tank000_Level	UNACK_RTN	$System	LO	1	TRAININ...	InTouch
5/11/2012 12:30:27 PM	Tank000_Level	UNACK_RTN	$System	HI	1	TRAININ...	InTouch
5/11/2012 12:31:03 PM	Tank000_Level	UNACK_RTN	$System	LO	1	TRAININ...	InTouch
5/11/2012 12:31:52 PM	Tank000_Level	UNACK_RTN	$System	HI	1	TRAININ...	InTouch
5/11/2012 12:32:28 PM	Tank000_Level	UNACK_RTN	$System	LO	1	TRAININ...	InTouch

59. Click **Development!** to return to WindowMaker.

Section 4 – Alarm Printing

This section describes alarm printer configuration and best practices for printed alarm output.

Overview

Alarm Events represent normal system status messages, and do not require an operator response. A typical event is triggered when a certain system condition takes place, such as an operator logging into InTouch. You can configure InTouch to log an event to the alarm database and print it to a printer.

As the plant engineers create InTouch tags and alarm conditions and severity levels, each alarm can then be prioritized by severity. You can then easily display and print alarms based on severity. Alarms can also be assigned to a logical alarm group. The groups provide a way of categorizing alarms based on an organization, plant layout, or any other metric. Alarm groups are also useful for filtering alarm displays, alarm printers, and acknowledgment scripts.

The InTouch **Alarm Printer** utility is used to print alarms from multiple nodes. You can print alarm records stored in the alarm memory on an event by event basis using a dedicated line or network printer. Also, you can use the Alarm Printer to save alarm records to a file.

Configure the Distributed Alarm system to print certain events on a line printer as they occur. Typically, you print alarms immediately to record information in the event of a catastrophic failure. Generally, you use a dot matrix printer connected through a serial or parallel port directly to the computer running the InTouch application. Windows network printers and laser printers are usually inappropriate for gathering data for catastrophic events because they hold entire pages in memory before actually printing a page.

Alarm Printer Manager

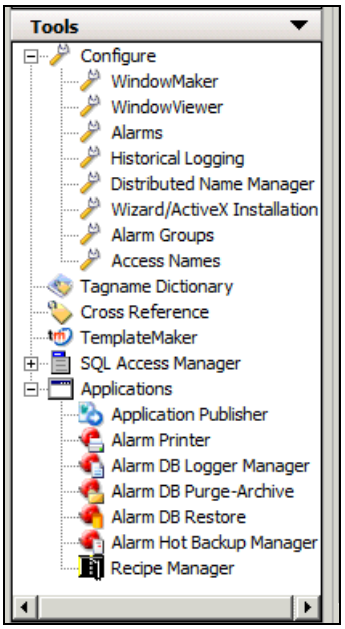
You use the InTouch **Alarm Printer** utility to print alarms from multiple nodes. You can print alarm records stored in the alarm memory on an event by event basis using a dedicated line or network printer. Also, you can use the Alarm Printer to save alarm records to a file.

Each query, logs all of the alarms specified in the Alarm Printer configuration file (.alc) that is currently open. If no file has been specified, the settings currently selected during Alarm Printer configuration are used.

You can run multiple queries with Alarm Printer. Each query uses different parameters and is associated with a separate instance of the Alarm Printer. If two instances of Alarm Printer are running the same query, the entries are duplicated.

While Alarm Printer is running, you can manually start or stop queries. Be sure that you have printing enabled.

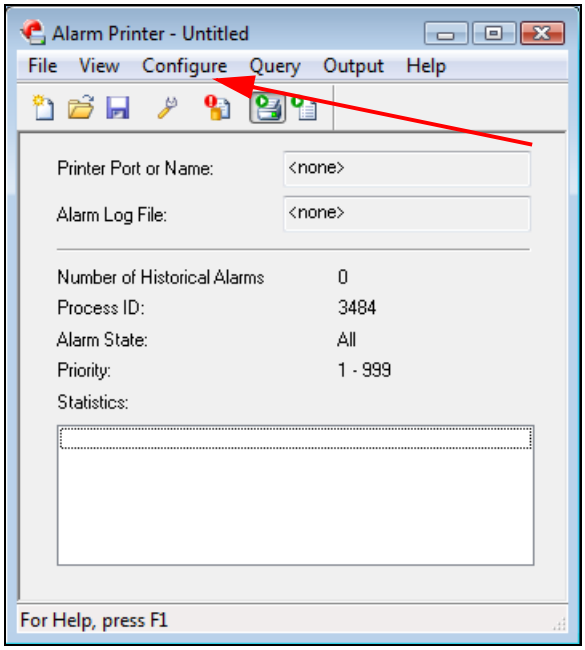
The **Alarm Printer** utility can be accessed from the WindowMaker, **Tools** pane.



Configuring Alarm Printing

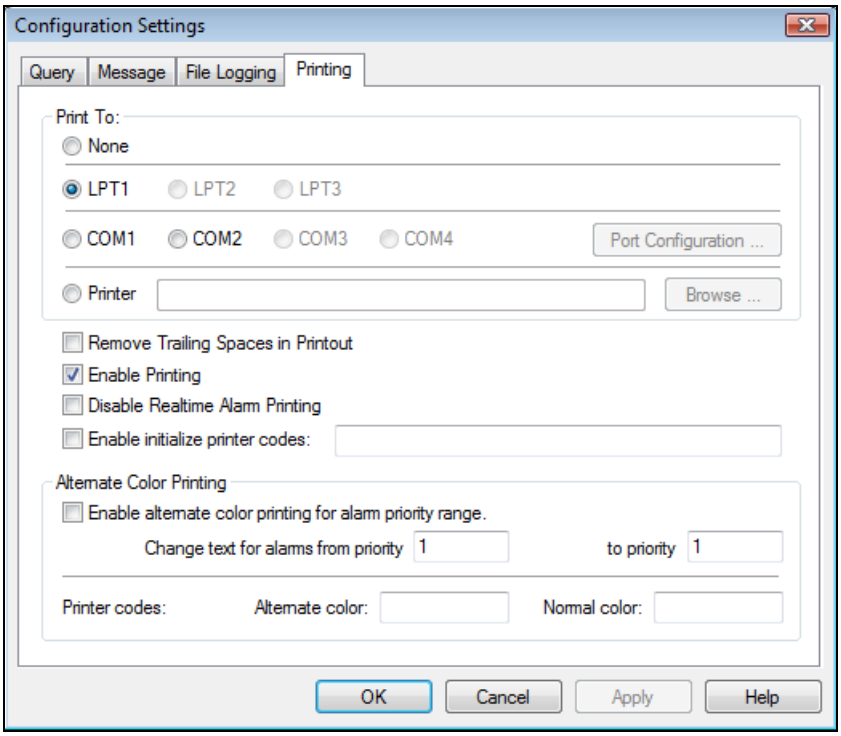
You can run multiple instances of the **Alarm Printer**. Each instance of the Alarm Printer must be configured to print to a different printer and must be configured with a separate alarm query. You can also configure separate instances of Alarm Printer to print alarms in specific priority ranges.

For example, one Alarm Printer instance can print only high priority alarms, while another instance prints only low priority alarms. Likewise, you can use one instance of the Alarm Printer to print alarms from one area of the factory while another instance prints alarms from a different area.



You can also configure the **Distributed Alarm** system to print certain events on a line printer as they occur. This requires a line type or dot matrix printer. It is not compatible with ink jet or laser printer technology as each line of an alarm or event consumes 1 page in the printer.

The line printer requires an LPT or COM type port on the local machine, this can make interfacing with the Alarm Printer more complex to implement in newer systems where these ports have been replaced by USB ports and Ethernet connections. Virtualized servers can also present complexities with hardware peripheral interfaces that require local physical connections to the machine.



Typically, you print alarms immediately to record information in the event of a catastrophic failure you have hard-copy to as backup. In some regulated industries, the paper is collected on a periodic basis. Networked printers hold entire pages in memory before actually printing a page, so this information can be lost in the event of a power failure.



invensys
Learning Services

Module 5 – Data Tracking and Trending

Section 1 – Real-Time Trend 5-3

Lab 13 – Building a Real-time Trend 5-7

Section 2 – Historical Trend 5-17

Lab 14 – Examining Data Retrieved from InTouch History 5-29

Module Objectives

- Use a real-time trend
- Enable historical logging in InTouch
- Use historical trend wizards
- Examine historical data

Section 1 – Real-Time Trend

This section describes the real-time trends used to view recent activities. It also discusses strategic placement and memory management of real-time trends.

Overview

InTouch provides two types of trend display objects: **Real-time** and **Historical**. Both trend objects can be configured to display graphical representations of multiple tags over time.

Real-time trends allow you to chart up to four pens (data values). Historical trends allow you to chart up to eight. Both types of trends are created using special tools in WindowMaker.

InTouch also provides you with complete trend configuration control. For example, you can specify the time span, value range, grid resolution, location of time stamps, location of value stamps, number of pens, and color attributes.

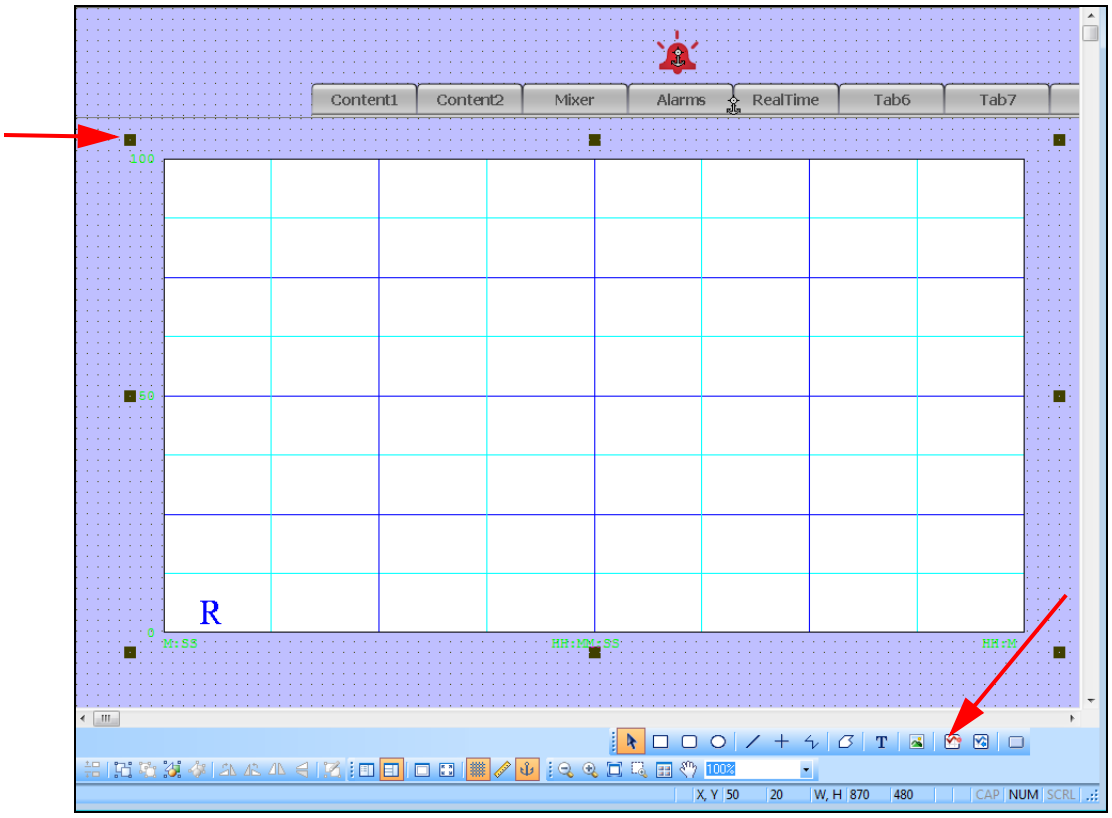
InTouch also supports a distributed history system that allows you to retrieve historical data from any InTouch historical log file, even those across a network.

Creating a Real-Time Trend

To create a real-time trending chart, you place the **Real-time Trend** object on the canvas and resize to accommodate the trend you want to create.

In runtime, data is written in the trend from right-to-left. There is no limit to the number of trends that can be placed on a window.

A trend object is like any other object drawn in WindowMaker. It can be moved by selecting it with the mouse or resized by selecting one of the object handles.

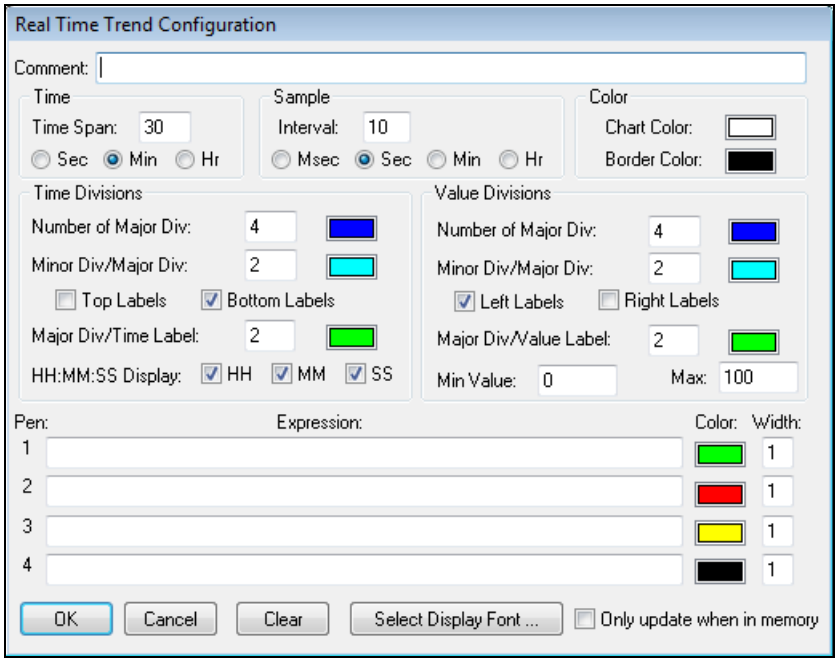


Configuring a Real-Time Trend

The first time a real-time trend object is pasted into a window, the system default configuration settings are used. Once a real-time trend has been configured, the next trend created will, by default, have the same settings.

Note: All entries made in the **Real Time Trend Configuration** dialog box are independent of the size of the trend and cannot be modified in runtime.

Once you have placed the object into your application, double-click it to open the **Real Time Trend Configuration** dialog box. This dialog box contains all the configurable options to create the trending chart where you define X- and Y- axis details. You also define chart and pen colors and other aesthetic values for your real-time trend.



The **Only update when in memory** option defines whether you update the trend when it is displayed in an active window, or constantly even if the window is not open. If this option is not selected, this may result in slightly slower system performance.

X-Axis

On the X-axis, you configure the time span, time divisions, and frequency for your trend expression.

Y-Axis

On the Y-axis, you configure value divisions and define minimum and maximum values for a trended tag. You also enter the tag or expression for each pen you want to trend. The ability to trend expressions is useful when creating custom displays to show tagnames with widely different ranges.

Up to four pens can be visible in a trend. The pens can be used to display any local tagname or an expression that contains one or more local tagnames.

Lab 13 – Building a Real-time Trend

Introduction

In this lab, you create a window with a Real-time Trend element and configure the element to trend Tank Level, Agitator, Inlet Valve 2, and Temperature.

Objectives

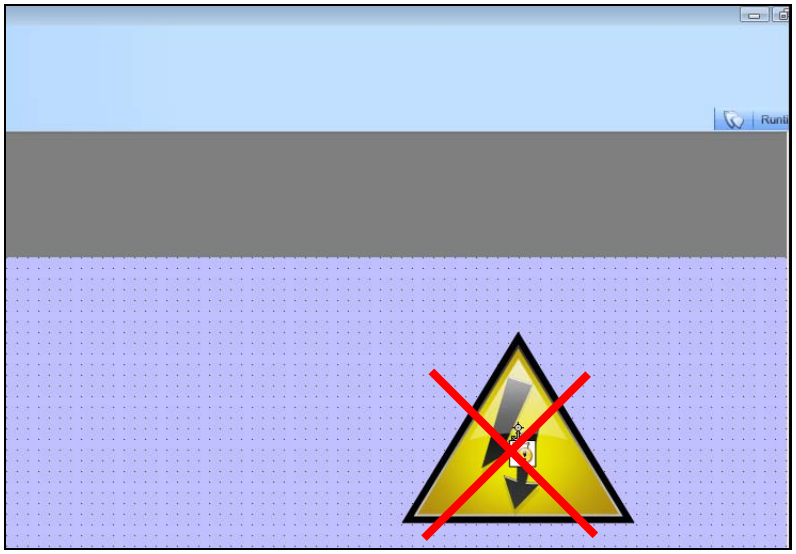
Upon completion of this lab, you will be able to:

- Configure pen expressions for the Real-time Trend
- Modify the appearance and behavior of the Real-time Trend

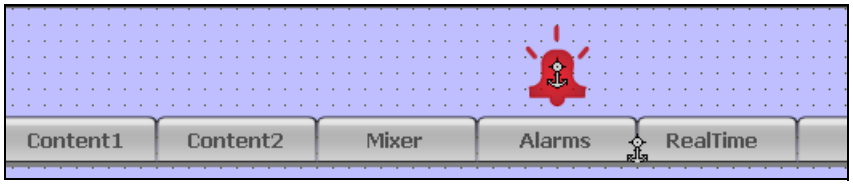
Create a Real Time Trend Window

First, you add a Real Time window.

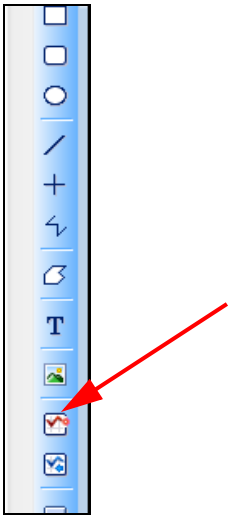
1. In WindowMaker, open the **Content2** window.
2. Right-click and select **Save As**.
3. Rename the window **RealTime**.
4. Click **OK**.
5. Delete the warning light graphic from the window.



6. In the **Menu** window, double-click the **NavigationTabs8** graphic and change the **TabCount** to **5**.
7. Using the **Substitute Strings**, rename **Tab5** to **RealTime**.

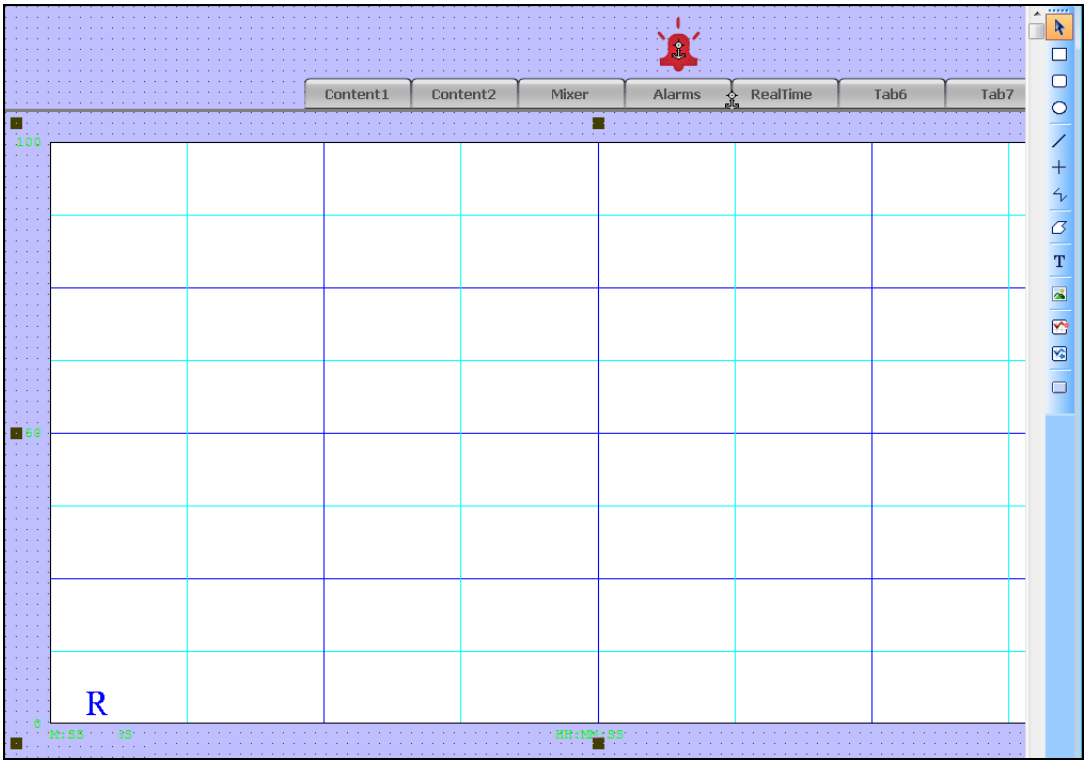


8. On the **Drawing** toolbar on the right, click **Real-time Trend**.



9. In the **RealTime** window, click and drag the **Real-time Trend** to create a large rectangle that fills at least **75%** of the window.

   Leave a little room at the bottom for a legend.



Configure Real Time Trend

10. Double-click the **Real Time Trend** and configure it as follows:

Comment: Mixer Details

Time

- **Time Span:** 1 Min

Sample

- **Interval:** 500 Msec

Color

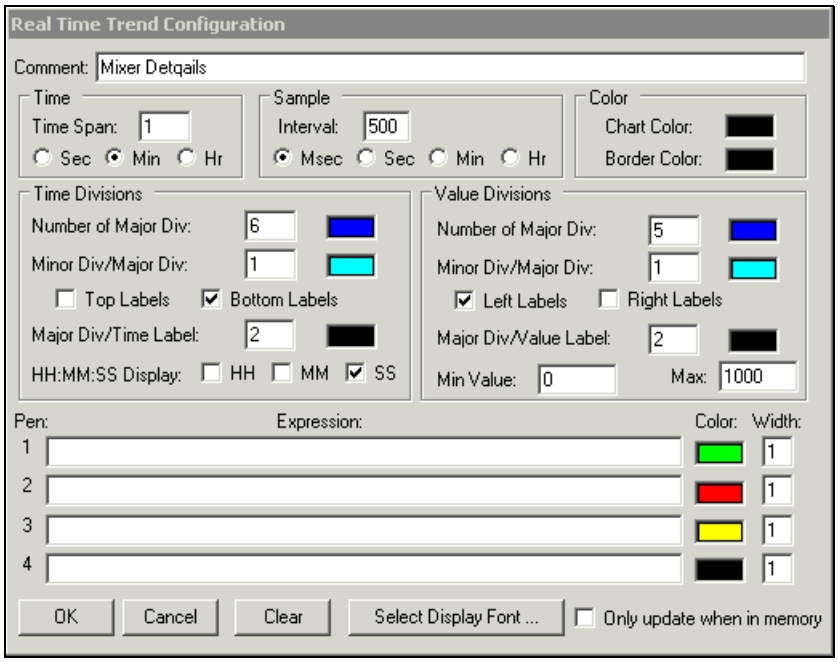
- **Chart Color:** Black

Time Divisions

- **Number of Major Div:** 6
- **Minor Div/MajorDiv:** 1
- **Major Div/Time Label Color:** Black
- **HH:** Uncheck
- **MM:** Uncheck

Value Divisions

- **Number of Major Div:** 5
- **Minor Div/Major Div:** 1
- **Minor Div/Value Label Color:** Black
- **Max:** 1000



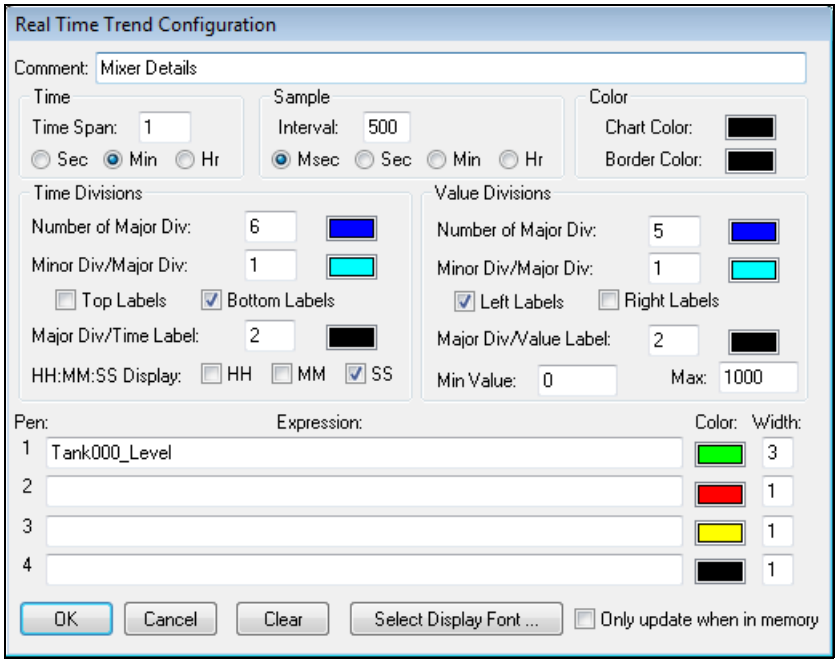
11. Double-click the **Pen 1 Expression** field.

    The **Tagname Dictionary** appears.

12. Double-click **Tank000_Level** to place the tag in the **Expression** field.

13. Ensure the **Color** is **Green**.

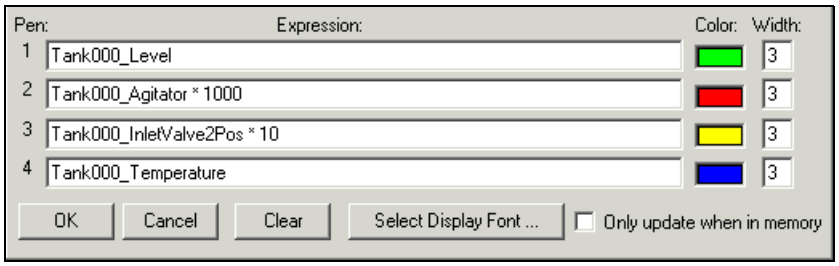
14. Change the **Width** to **3**.



Note: The width option indicates pixel width of pen line.

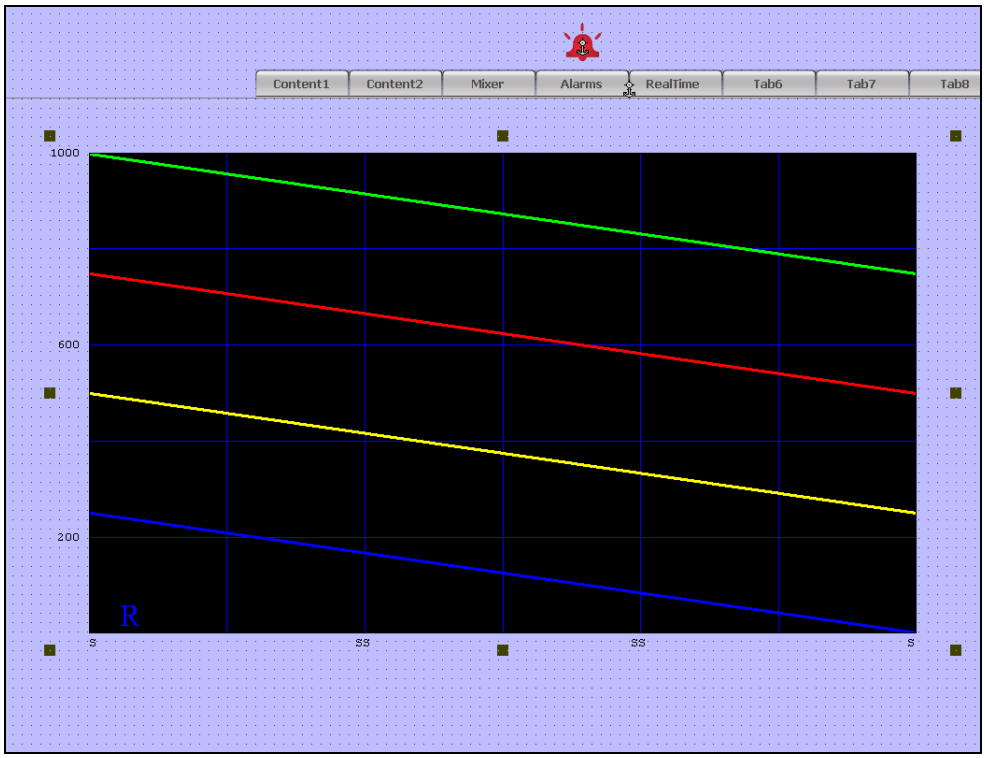
15. Configure the remaining **Pen Expression** options as follows:

Pen	Expression	Color	Width
2	Tank000_Agitator * 1000	Red (default)	3
3	Tank000_InletValve2Pos * 10	Yellow (default)	3
4	Tank000_Temperature	Blue	3



16. Click **OK** to close the configuration window.

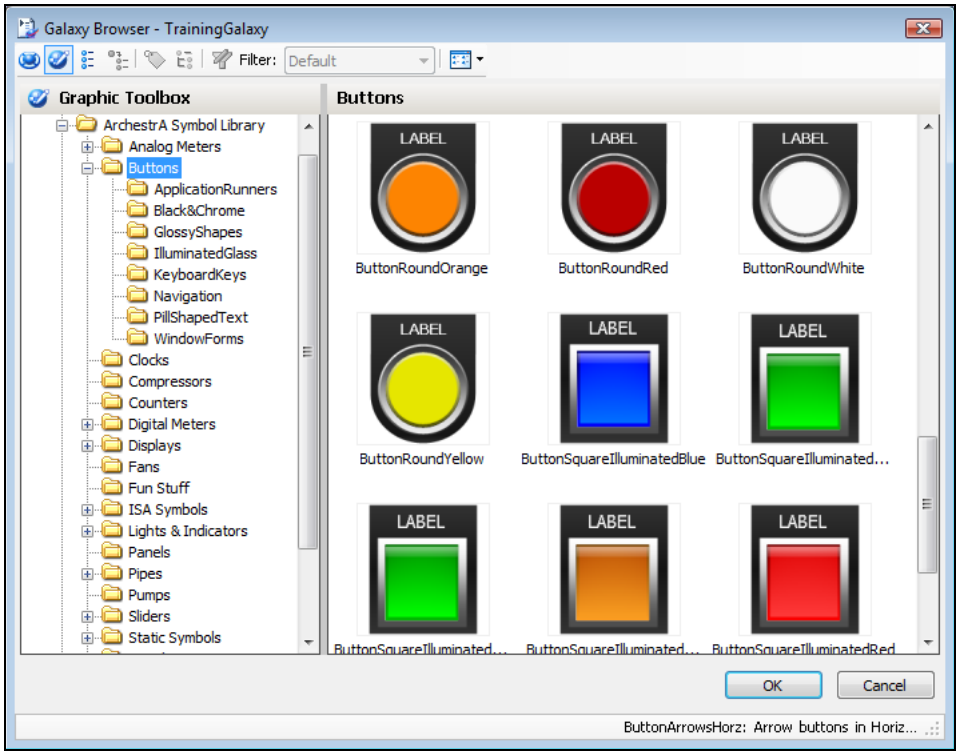
    The **Realtime** trend reflects the new configuration.



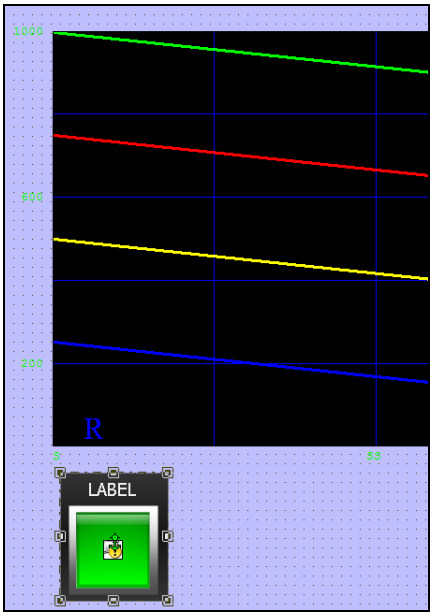
Set Up A Legend

Next, you scroll down to access the area at the bottom of your RealTime window to develop a trend legend. You add indicator lights associated with each trend pen.

17. From **Graphic Toolbox**, click **Buttons** and double-click **ButtonSquareIlluminatedGreen**.

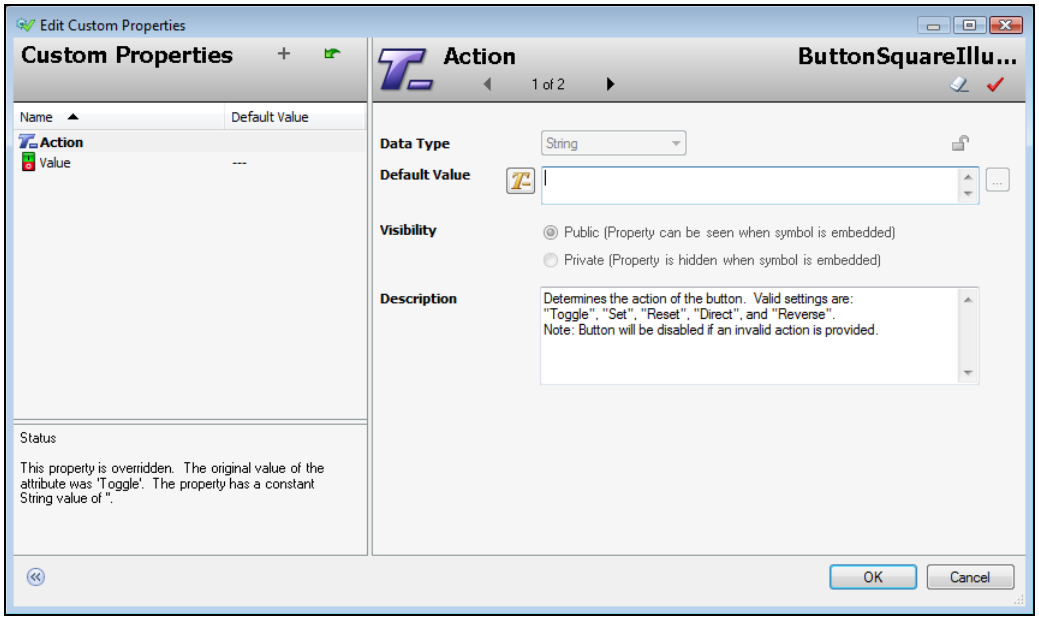


18. Place the graphic below the **Real-time Trend**.



19. Change **LABEL** to **Level**.

20. Double-click the **Level** button to open the **Edit Custom Properties** dialog box.
21. Click **Action** and clear the **Default Value** field.

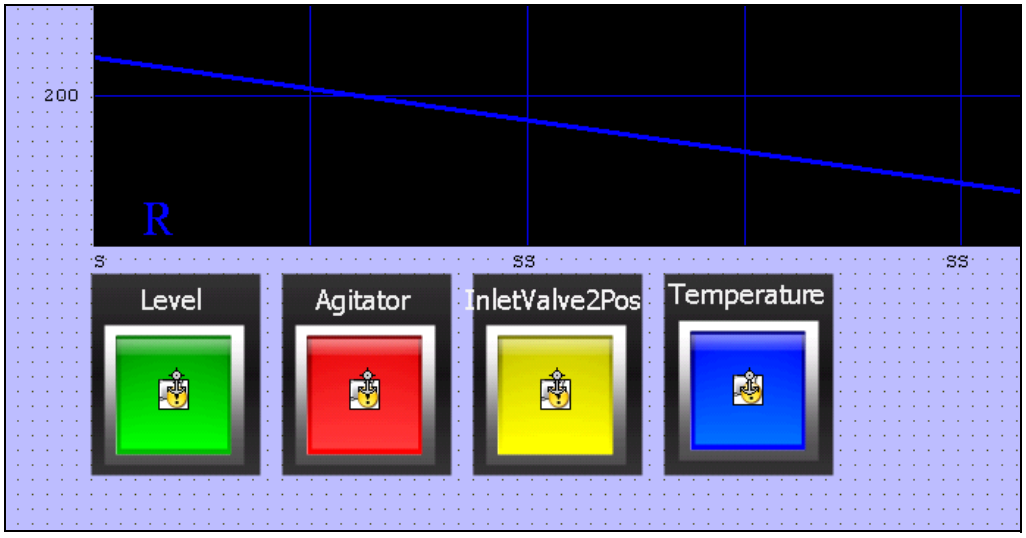


22. Click **Value** and change the **Default Value** to **1**, then click **OK**.

Embed the following button graphics and place to the right of the **Level** button.

23. Change the **LABEL** for each new button and modify the **Custom Properties** as follows:

Graphic name	Label	Action	Value
ButtonSquareIlluminatedRed	Agitator	<empty>	1
ButtonSquareIlluminatedYellow	InletValve2Pos	<empty>	1
ButtonSquareIlluminatedBlue	Temperature	<empty>	1



Test in Runtime.

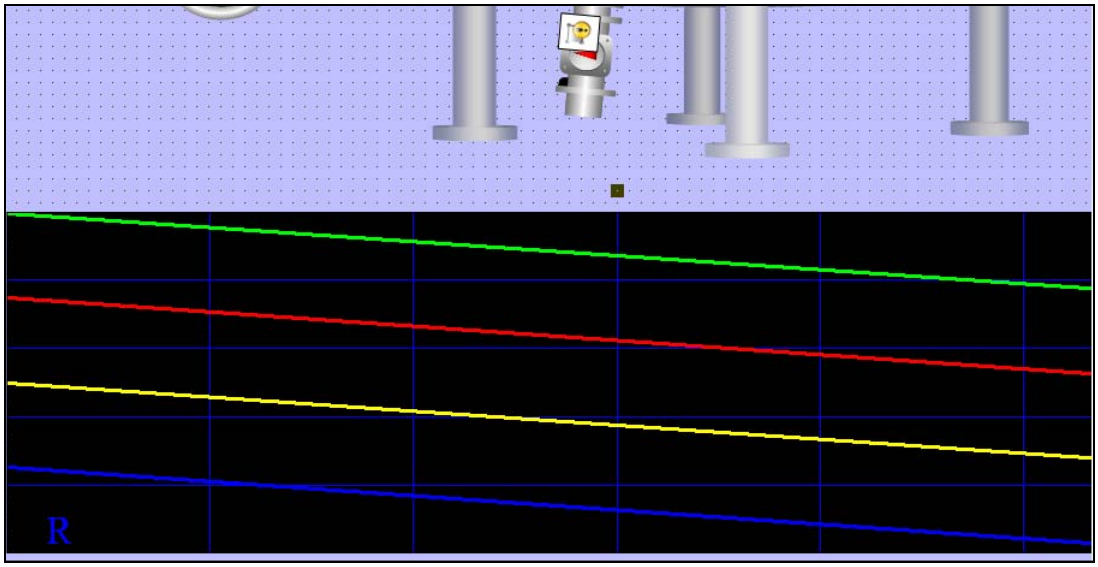
24. Verify that the **Level** flat -lines during the **InletValve2** transition.
25. Verify that the **Level** does not decrease the moment the **Agitator** stops.

    Why do you think this occurs?
26. Verify that the **Temperature** and **Level** rise and fall proportionately.

Optional Steps

Real-time trends can be a useful addition to a supervisory window.

27. Add a **Real-time Trend** to the **Mixer** window.



28. Test in **Runtime**.

    Notice that even when switching windows, **Real-time Trends** retain information as long as WindowViewer is running.

Section 2 – Historical Trend

This section introduces InTouch historical logging, and explains the historical trend configuration options.

Overview

Historical trends provide snapshots of data from a time and date in the past. Unlike real-time trends which dynamically update based on their configuration, historical trends are only updated systematically through scripts or by operator interaction. You can trend up to eight tags (pens) at one time with no limit to the number of trends you display. This provides you with complete flexibility in interface design for your historical trend.

With the InTouch Historical Subsystem enabled, each tagname is logged historically when the **Log Data** option for that tag is enabled in the **Tagname Dictionary**. After data has been archived historically, it can be retrieved through tools such as the historical trend or using the historical wizards.

One of the tools available are **Scooters**, which are controls that the operator slides over the trend to point to specific pen traces at a specific time, and can be used to set the boundaries for a zoom command. For example, when the operator positions the scooter over an area on the trend that has visible data, the time and values at that location for all data points in your trend are returned and can be displayed. You can also create buttons to zoom in and out between the scooters or to data, such as the maximum to minimum value. You can also display average and standard deviation for the entire trend.

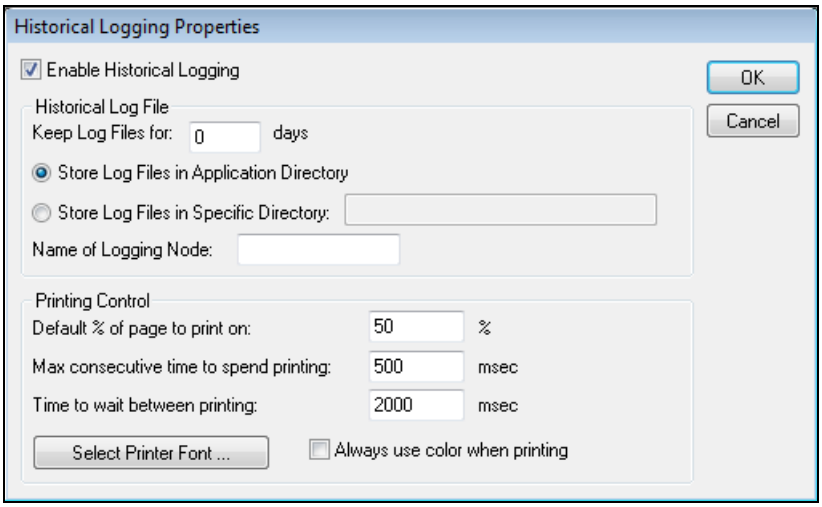
You can configure historical trends to scroll a specified amount of time. Create custom scales and link to the **.MinEU** and **.MaxEU** tagname Dot Fields to display the minimum and maximum Engineering Units. The distributed history system extends the retrieval capabilities of historical trends to include remote log databases. This system allows you to display information from multiple historical log databases in a single trend.

In addition to trending capabilities, InTouch includes the **HistData** utility, designed to work with InTouch historical log files. The HistData utility converts encrypted historical log files (.LGH) to comma separated variable files (.CSV) for use in spreadsheet or text editing environments such as Microsoft Excel.

Configuring Historical Logging

To configure tags for historical logging, you enable **Log Data** in the **Tagname Dictionary**.

To display the **Historical Logging Properties** dialog box, on the **Special** menu, select **Configure | Historical Logging**.



First, you select **Enable Historical Logging** to allow for global tag logging. Then in the **Keep Log Files for** field, enter the number of days (prior to today) for log files to keep saved to disk.

For example, if you enter 10 and today is the 12th day of the month, the log files for the 2nd through the 12th (10 days plus today) will be saved to disk. The file for the 1st will automatically be deleted. If you enter a zero, the log files will be kept indefinitely.

Store Log Files in Application Directory: Saves the historical log file in an application directory.

Store Log Files in Specific Directory: Saves files to another directory. Enter the complete path to the directory you want to use. This entry must be either a DOS or UNC path such as **C:\histlog**.

Name of Logging Node: Name of the node where the Historical Subsystem will run.

Default % of page to print on: Trend to page size percentage ratio.

Max consecutive time to spend printing: Number of milliseconds, also called processor time slice, the historical trend print module will spend consecutively printing.

Time to wait between printing: Number of milliseconds the historical trend print module is to wait between printouts.

Stopping and Starting Historical Logging at Runtime

Historical logging can be started and stopped in runtime by changing the value of the **$HistoricalLogging $System** tag or by using the **Restart Historical Logging** menu option from the WindowViewer menu.

InTouch Historical Log Files

In WindowViewer, the values of logged tags are written to the historical log file each time they change more than the specified **Log Deadband** setting, and by default once an hour regardless of change. For a tag's value to be written to the historical log file, it must be configured to be logged in the Tagname Dictionary.

For integer and real (floating point) tag types, you can set the Log Deadband in their respective details dialog boxes. The Log Deadband controls how many Engineering Units a tag's value must change before it is logged to disk.

InTouch will create and save two historical log files per day (24 hours). Therefore, disk space must be considered when this value is set. If your hard disk does not have enough free space to save a historical log file, logging will stop. You must free some disk space and then restart logging.

By default, historical log files are named as follows:

YYMMDD00.LGH and **YYMMDD00.IDX**

where: **YY** equals the year the file was created

MM equals the month the file was created (01-12)

DD equals the day the file was created (01-31)

00 represents hours; always displays zeros (regardless of when the log files were created).

For example, if the files were created on December 28, 2013, they would be named as follows:

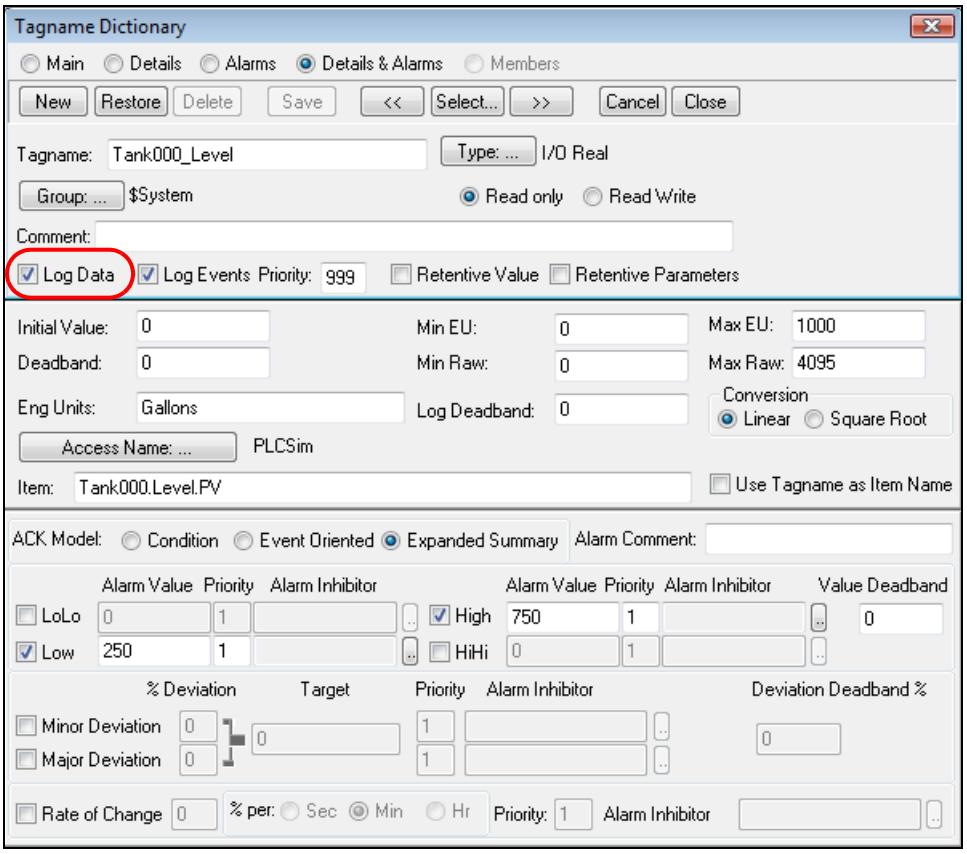
13122800.LGH and **13122800.IDX**

Logging a Tag

To log tags, **Historical** logging must be enabled. If a tag is changed from logged to not logged, the data already logged for the tag will not be accessible. However, if logging for a tag is re-enabled, the data is available but will display a gap in a trend trace.

Any changes made in WindowMaker to logging while WindowViewer is running, are ignored until WindowViewer is restarted.

To log tags, open the **Tagname Dictionary** and open the desired tag's definition. Select the **Log Data** option at the bottom-left of the dialog box, **Save** and **Close**.



Note: The **Historical Trend** displays from 0-100% of EU scale, this allows unlike data to be displayed side by side.

Creating a Historical Trend using Wizards

A Trend is like any other object drawn in WindowMaker. It is selected from the drawing toolbar, then after being placed on a window it may be moved by grabbing it with the mouse or it can be resized by grabbing one of the object handles. You can place multiple trends in a window but each must be configured to be driven by a separate Historical Tag. The primitive trend requires a lot of basic configuration before it can be used; it is therefore, strongly advised to use the Trend Wizards.

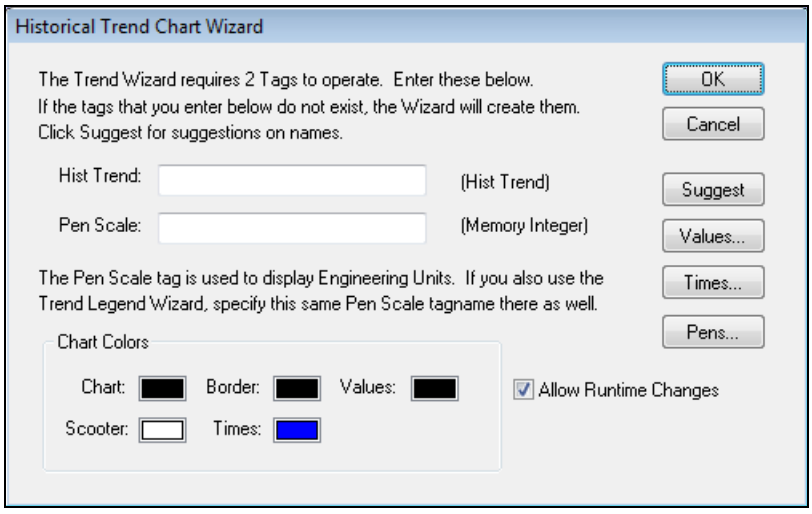
Wizards are pre-drawn individual graphical components that are embedded in an InTouch window and configured during development. They save a considerable amount of time during application development because they are easy to use and easy to configure, and bring many enhancements and features for operators to use in runtime.

The InTouch historical trend wizards are found by clicking the **Wizard** selector, then choosing the Trend Wizards library. These wizards provide a quick and easy way to create a historical trend window. Trend Wizards allow you to configure a full-featured historical trend with features such as scooters and zooming with just a few mouse clicks. Configuration of the Hist Trend w/Scooters and Scale wizard is similar to the object on the Draw Objects toolbar.

Inserting and Configuring the Historical Trend Chart Wizard

To insert a historical trend, click the **Wizard** tool to open the **Wizard Selection** dialog box, and then select **Trends** to display its contents. Next, select the **Hist Trend with Scooters and Scale** wizard and click and drag to place and size it on the InTouch window.

To configure your **Historical Trend**, open the **Historical Trend Chart Wizard**, accessible by double-clicking the **Trend** wizard placed in your window.



Configuration options for the **Historical Trend Chart** include the following:

Suggest: The wizard automatically suggests the controlling HistTrend tag and a tag to control the scaling.

Values: Allows you to customize the vertical axis divisions, format, and range of the Historical Trend Chart.

Times: Allows you to customize the horizontal axis divisions, time span, and time format.

Pens: Allows you to select values and colors to be used to plot each individual pen

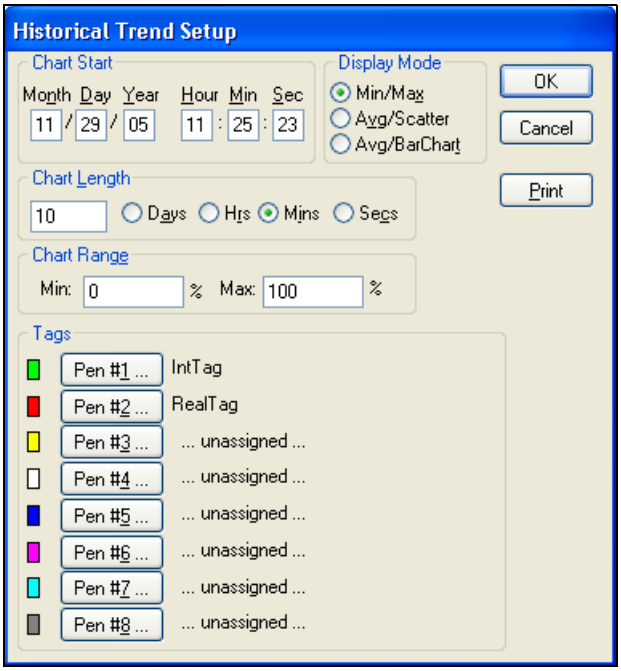
Configuring the Trend Pens

The Historical Trend Wizard has eight individual pens that can be configured to plot the value of specific tags. To configure any pens in the wizard, you must first ensure WindowViewer is closed, otherwise all pen fields will be grayed out and disabled. To select a tag for any pen, double-click the pen field. Only tags which have Log Data enabled in the Tagname Dictionary will appear in the list. Select a tag and repeat for any additional pens.

Configuring the Historical Trend Setup from WindowViewer

Select the **Allow Runtime Changes** option to allow the Trend to be clicked in runtime, automatically bringing up its basic configuration window, which includes setting the date and time span, wither it's a scatter or plot type chart, and to modify tags that are being plotted.

To see these options displayed, click **Runtime** to switch, then click the trend in WindowViewer.



The configuration options for a Historical Trend are available via the **Historical Trend Setup** dialog box, and they are as follows:

Option	Purpose
Chart Start	Enter the starting date and time to begin charting.
Display Mode	Select one of the three available modes: ● The **Min/Max Historical Trend** mode displays the trends or changes in the percentage of Engineering Units scale as a vertical line over the time span, with emphasis on time flow and rate of change and not on the amount of change. ● The **Average/Scatter Historical Trend** mode displays the average value of the point during the time intervals. ● The **Average/Bar Chart Historical Trend** mode displays the average value of the point during the time intervals in bar form.
Chart Length	Use to enter the horizontal (x-axis) length of time to be displayed on the chart, as well as selecting the time increments for the length (i.e., if you enter a 1 and select Hrs, the trend will be one hour long).
Chart Range	Use to enter the percentage of Engineering Units scale that the trend is to zoom in/out (vertical y-axis) range to be displayed on the trend. The units for the range are a percentage of the Engineering Units scale. Set the values at from 0 to 100.
Pen#	Click within the blank field to select open the **Tag Browser** in filtered selection mode. Enter the tag that the selected pen will plot on the trend chart.

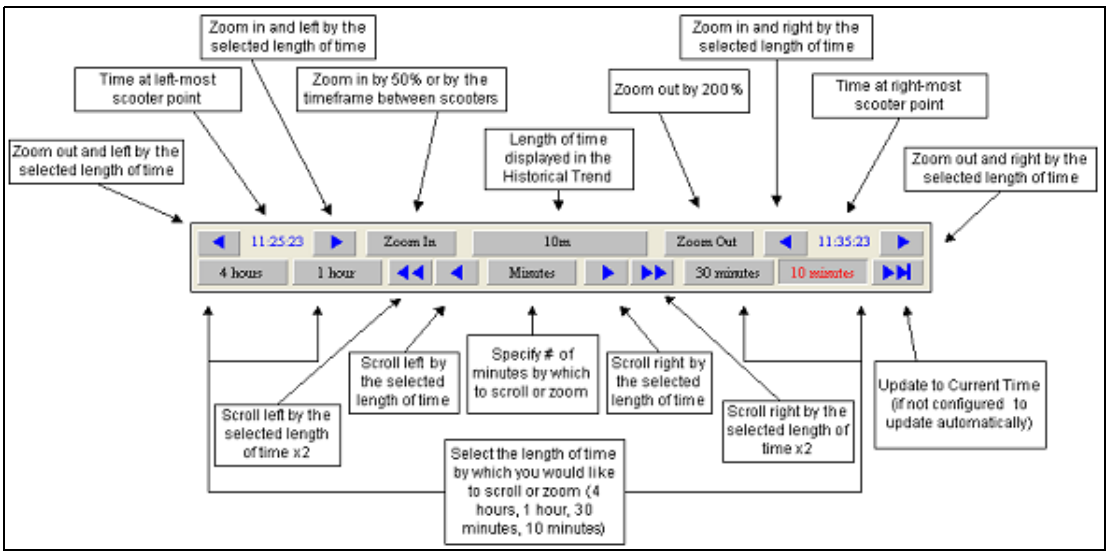
Additional Historical Trend Wizards

There are additional Historical Trend wizards to enhance movement function and pen control. For these components to all work together, the same **Hist Trend** tagname must be used by all wizards, as follows:

Movement Function / Pen Control	Purpose
Trend Zoom / Pan Panel / Trend Pen Legend	Add zoom and movement functions or pen controls to your historical trend using one of these three controls.
Trend Zoom / Pan Panel	Place and configure the Trend Zoom / Pan Panel using the same steps outlined in the Inserting and Configuring the Historical Trend Chart Wizard section.

Inserting a Trend Zoom / Pan Panel Wizard

The **Trend Zoom / Pan Panel** wizard is one of the components of historical trending. These components are configurable and they are illustrated below:



To configure the **Trend Zoom / Pan Panel** wizard, start by using the **Suggest** button. Use the scooters at the bottom of the **Hist Trend with Scooters and Scale** wizard to navigate forward and backward in time, or to **Zoom In** with less data and more detail, or **Zoom Out** for more data and less detail. Performance is impacted by the amount of data being requested to display, for example requesting an hour's worth of data might be instant, but requesting a year's worth of data could take some time.

The configurable components are as follows:

Component	Purpose
Trend Pen Legend	Displays tagname and values at Scooters for one pen. By default, the value of the pen at the left-most point on the Historical Trend is displayed on the left, while the value of the pen at the right-most point is displayed on the right side of the Trend Pen Legend. However, you can use the Scooters at the bottom of the Historical Trend wizard to display a pen's value at a specific time. Upon selection, you are prompted to place one or more Trend Pen Legend(s). Confirm the default options (tagnames, pen colors, pen number, etc.)
HistData Wizard	Saves historical data to a .CSV file. After placing a HistData wizard into your InTouch window, configure it using the Suggest option.

Historical Trend Dot Fields

For any given historical trend tag, there are many Dot Fields that apply to historical trending tags:

Dot Field	Description
.ChartLength	Read/write integer tagname .field used to control the length of time displayed in a Historical Trend graph. .ChartLength displays the length of the chart in seconds.
.ChartStart	Read/write integer tagname .field used to control the starting time and/or to scroll the corresponding Historical Trend. .ChartStart displays the number of elapsed seconds since 12:00 a.m., 1/1/70.
.DisplayMode	Read/write analog tagname .field used to determine the method to be used in displaying values on the trend.
.MaxRange, .MinRange	Read/write real tagname .fields used to represent the percentage of the tagname's Engineering Unit range that should be displayed for each tagname being trended. The limits for .MaxRange and .MinRange are from 0 to 100 and .MinRange should always be less than .MaxRange. If a value less than 0 or greater than 100 is assigned to either of these fields, the value will be clamped at 0 or 100. If .MinRange is greater than or equal to .MaxRange, the trend will not display any data.
.Pen1 - .Pen8	Read/write TagID type tagname .fields used to control the tagname being historically trended by each pen. A TagID type tagname can only be equated to another TagID tagname. It cannot be mixed with any other tagname type unless the .TagID extension is added to the other tagname. .TagID cannot be used for remote history provider tagnames.
.ScooterLockLeft	Read/write discrete field. When the value of this field is TRUE, the RIGHT scooter cannot move to the left of the left scooter's position. (0=FALSE, 1=TRUE).
.ScooterLockRight	Read/write discrete field. When the value of this field is TRUE, the LEFT scooter cannot move to the right of the right scooter's position. (0=FALSE, 1=TRUE).
.ScooterPosLeft	Read/write real field which represents the position of the left scooter (range 0.0 to 1.0).
.ScooterPosRight	Read/write real field which represents the position of the right scooter (range 0.0 to 1.0).
.TagID	Read/write TagID tagname .field used in conjunction with the Historical Trend .Pen1 - .Pen8. TagID tagnames monitor and/or control the tagname being trended by a pen.
.UpdateCount	Read-only integer .field that is incremented when retrieval is complete for the trend.
.UpdateInProgress	Read-only discrete .field that shows historical data retrieval status (0=no retrieval in progress, 1=retrieval in progress).
.UpdateTrend	Read/write discrete tagname .field that can be set to 1 to cause a Historical trend to update using all current values.

HistData Wizard

The HistData wizard allows the extraction of historical data into a CSV file. The CSV or Comma-Separated Value file format is common and many tools such as Microsoft Excel support viewing this type of data. The wizard can create the necessary tagnames and scripts automatically using the **Suggest** button. The HistData wizard is shown below:



The following tags are created automatically by the HistData wizard along with an Access Name for HISTDATA.EXE:

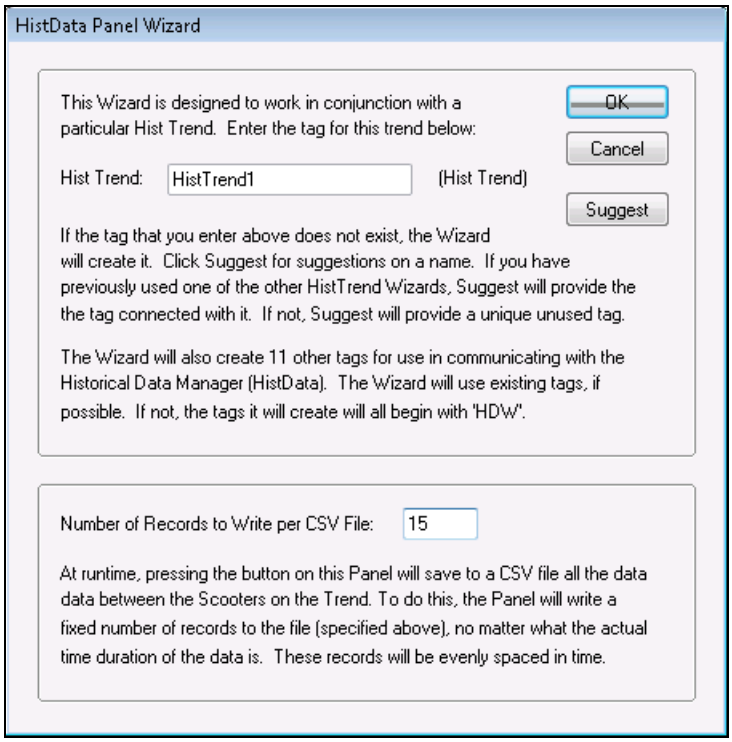
Tag	I/O Data Type and ITEM
HDWDATADIR	Message DataDir
HDWDBDIR	Message DbDir
HDWDURATION	Message Duration
HDWDBDIR	Message DbDir
HDWERROR	Message Error
HDWFILENAME	Message FileName
HDWINTERVAL	Message Interval
HDWSTARTDATE	Message StartDate
HDWSTARTTIME	Message StartTime
HDWSTATUS	Message Status
HDWTAGS, HDWTAGS1, HDWTAGS2	Message Tags, Tags1, Tags2. **Note:** If additional tags are needed continue the sequence with HDWTAGS3 Item: Tags3 etc.
HDWWRITEFILE	Integer WriteFile

For details of each tag and its use with examples, see the *InTouch HMI Data Management Guide*.

The HistData wizard will create these tags for you when you use the suggest button in the wizard. There are additional tags available to interface with HISTDATA.EXE not shown and not created automatically by the wizard, including HDWSendData, HDWData, and PRINTTAGNAME.

To configure the wizard, double-click the trend and click **Suggest**, making sure the Hist Trend tagname matches the same one used in the Historical Trend Chart. Then the HistData wizard will extract only the pens for the time shown on the chart.

Specify the number of records up to 500. The actual number of records extracted to the CSV file is N+1. Additionally, the wizard automatically adds a header row with the tagnames included. Values for each tag appear in columns with a time stamp accompanying each entry.



As with any I/O Server, the HISTDATA.EXE application must be running in order to extract the data. The next step is to provide in the blue text link a legitimate path and filename, followed by the **Enter** key to lock in the entry. Finally, when the operator clicks the **Send To File** button, the indication of success is in two parts; first the path and filename are validated by changing the blue text to all uppercase characters, second if there are any errors, the most critical error encountered is displayed in red flashing text on the right side of the wizard.

Since the HistData wizard relies on the configuration and settings of the current historical trend chart to automatically fill out its tagnames for extraction, this will limit output to 8 pens and restrict the data to only what is shown on the current trend. For larger extractions, more tags, or unique time intervals, the application designer will have to build a window with user input links for each of the tags, and create HDWTAGS3 … for the additional tags.

Lab 14 – Examining Data Retrieved from InTouch History

Introduction

In this lab, you modify existing tags to enable Historical Logging. You also configure historical trends and then view historical data.

Objectives

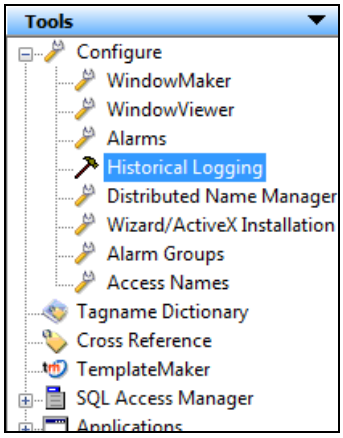
Upon completion of this lab, you will be able to:

- Enable Historical Logging
- Configure tags for historical logging
- Configure historical trend wizards
- Retrieve historical data

Create a Historical Trend Window

First, you modify existing tags to enable data logging and create a new window to display historical trends for Tank000.

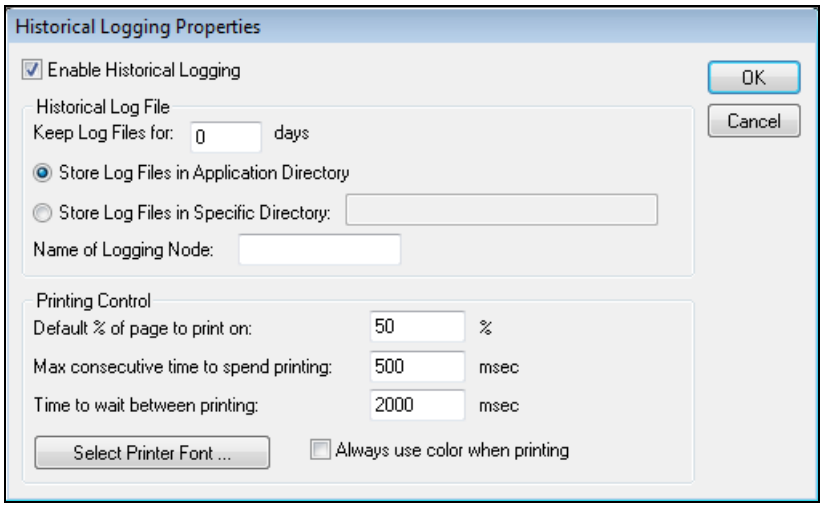
1. Close WindowViewer.
2. In WindowMaker, in **Tools** pane, expand **Configure**.



3. Double-click **Historical Logging**.

   The **Historical Logging Properties** dialog box appears.

4. Check **Enable Historical Logging**.

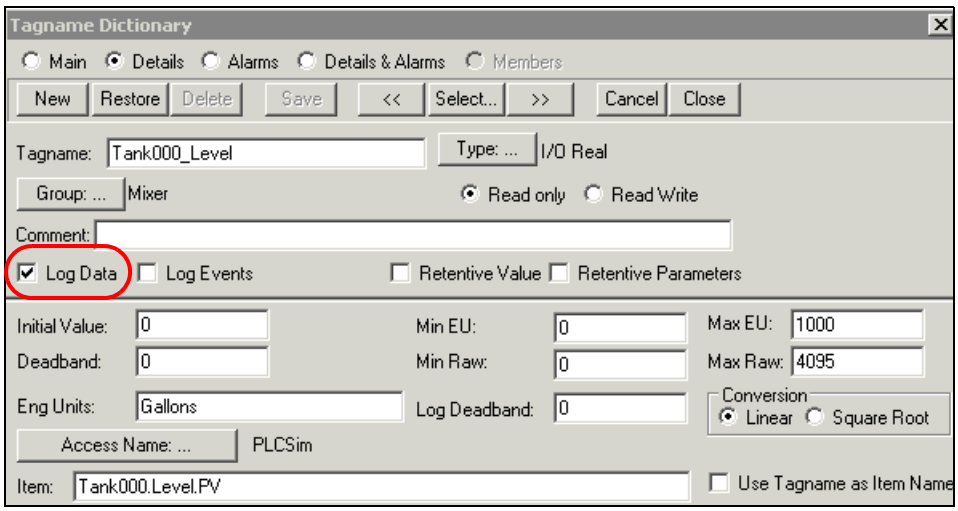


5. Click **OK**.

Configure Tags for Historical Logging

In Tagname Dictionary you must enable Log Data for each tag you historize.

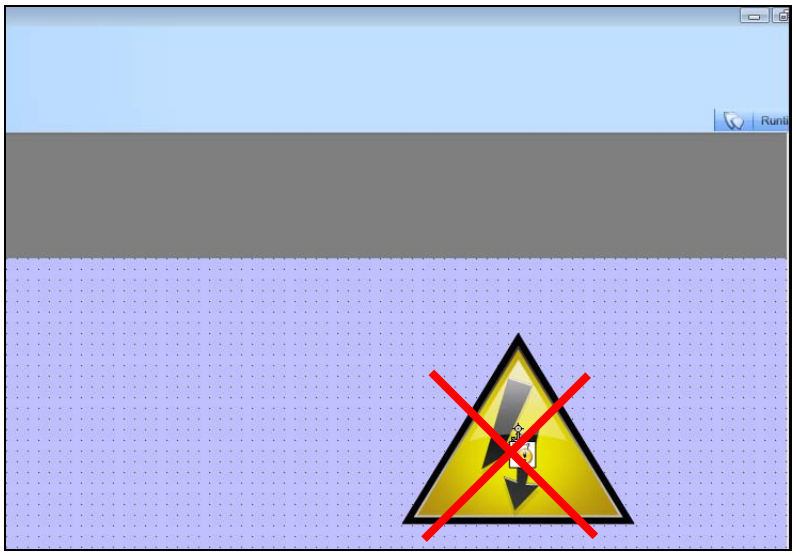
6. In **Tagname Dictionary**, enable **Log Data** for the following tags, clicking **Save** after each:
   - **Tank000_Level**
   - **Tank000_Agitator**
   - **Tank000_InletValve1Pos**
   - **Tank000_InletValve2Pos**



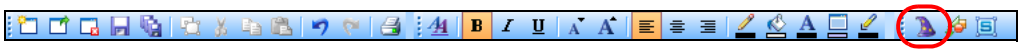
7. Click **Close**.

Next, you create a new Historical window.

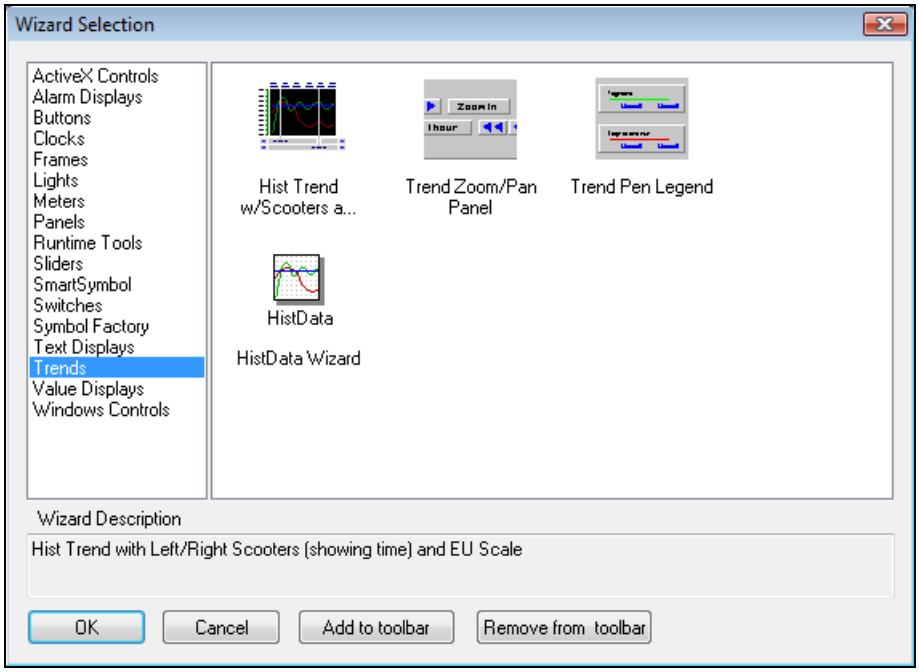
8. Open the **Content2** window.
9. Right-click and select **Save As**.
10. Name the new window **History**.
11. Click **OK**.
12. Remove the warning light from the window.



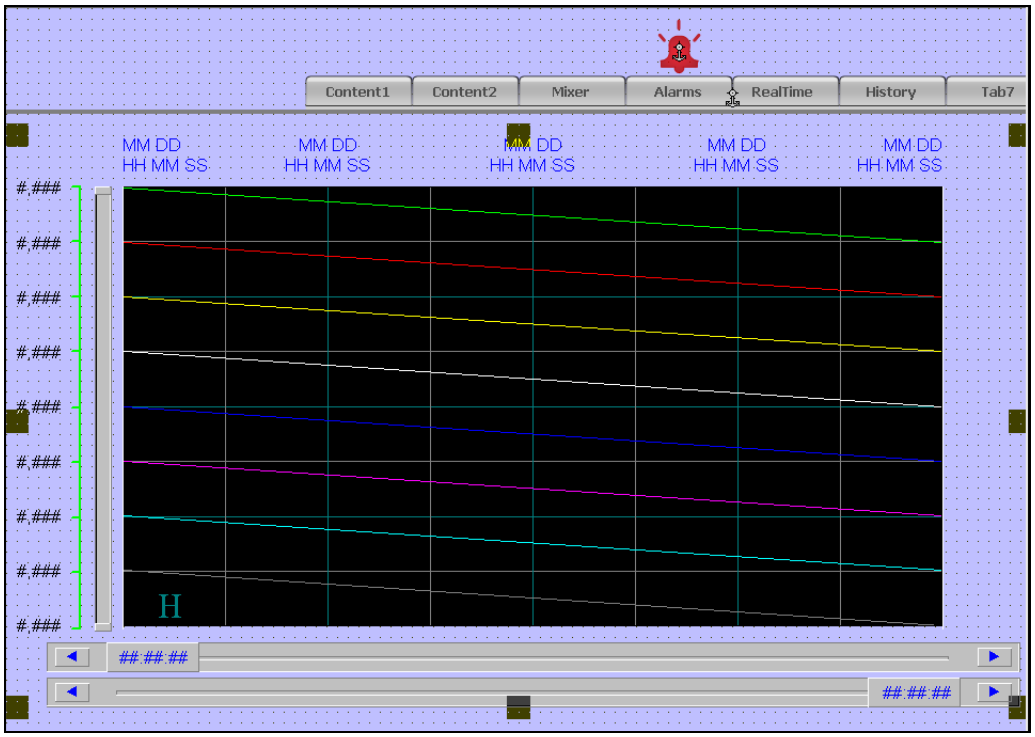
13. Open the **Menu** window and make the following changes:
    - Change the **TabCount\Default Value** to **6**
    - Change the **Label** for **Tab6** to **History**
14. On the toolbar, click the **Wizards** button to open the **Wizard Selection** dialog box.



15. Click **Trends** to view the available trend wizards.



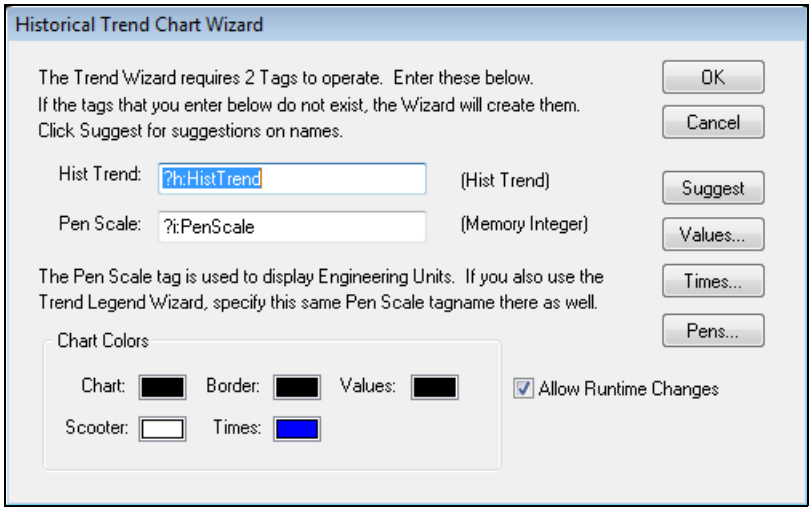
16. Click **Hist Trend w/Scooters and Scale**, then click **OK**.
17. Click on the **History** window to place the **Wizard**.
18. Resize the **Wizard** to fit the window.



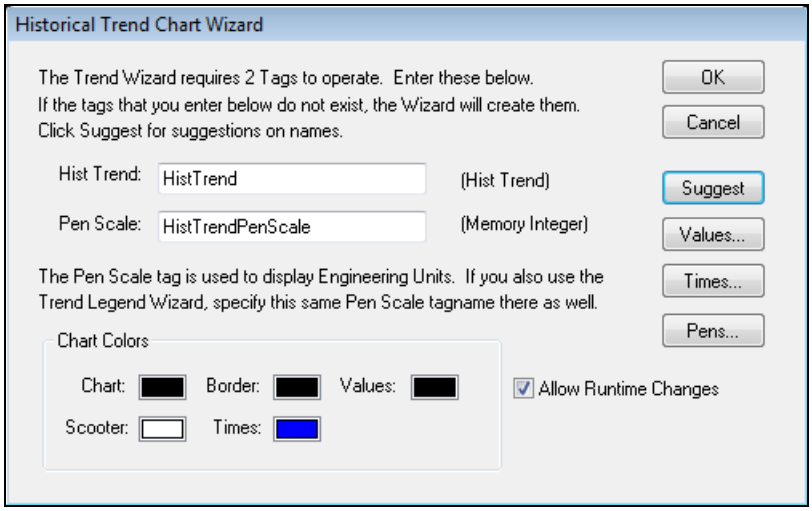
Configure Historical Trend Wizards

19. Double-click the **Hist Trend w/Scooters and Scale** wizard to configure the trend.

    The **Historical Trend Chart Wizard** dialog box appears.



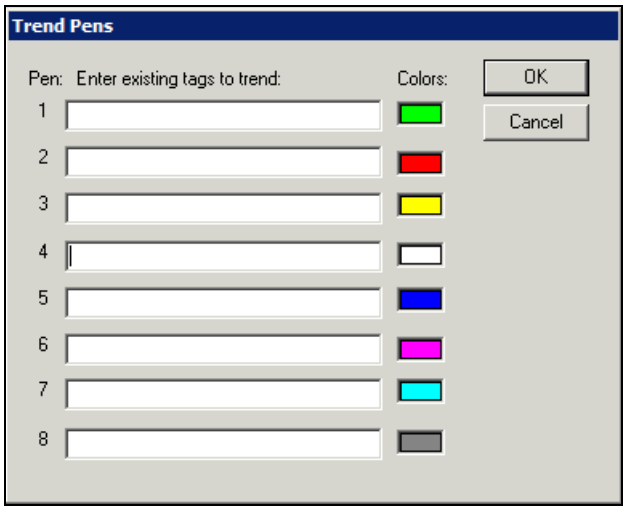
20. Click **Suggest**.



21. Keep all suggestions and keep the **Historical Trend Chart Wizard** open.

22. Click **Pens** to configure individual pen options.

    The **Trend Pens** dialog box appears.

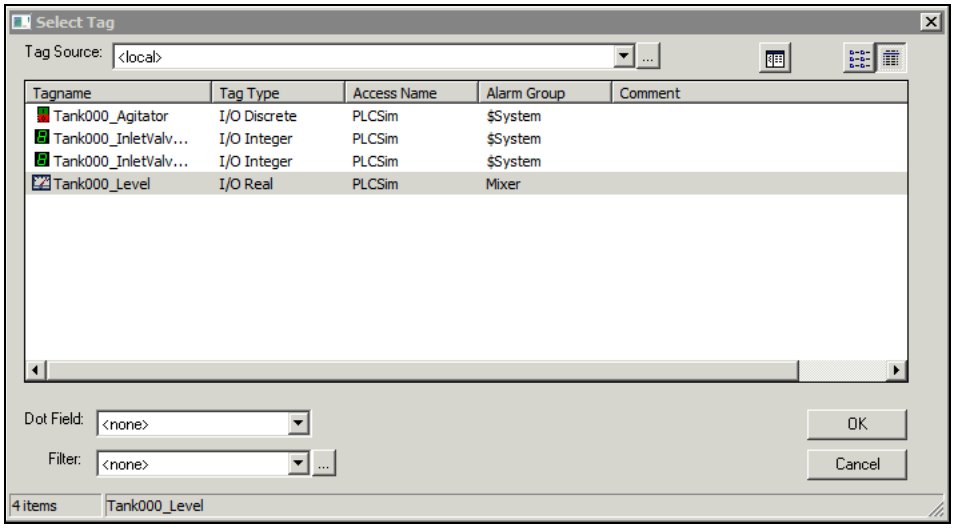


23. Double-click the **Pen 1** field.

    The **Select Tag** dialog box appears.

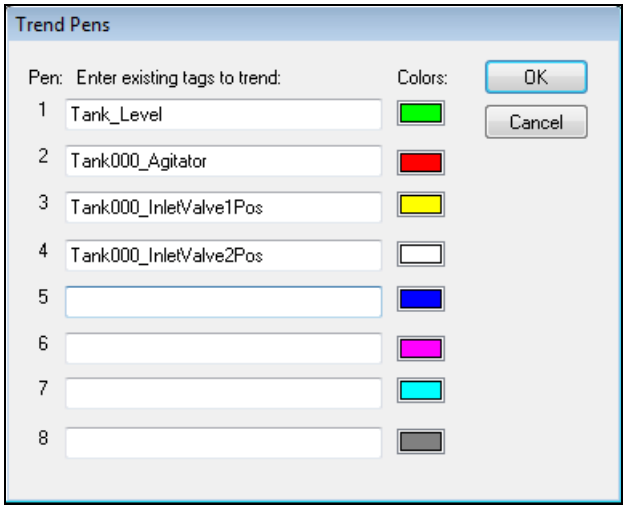
    Notice the list is exclusive to the tags that have **Log Data** checked in the **Tagname Dictionary**.

24. Click **Tank000_Level**, then click **OK**.



25. Repeat these steps for each of the following pens:

Pen	Enter existing tags to trend
2	Tank000_Agitator
3	Tank000_InletValve1Pos
4	Tank000_InletValve2Pos

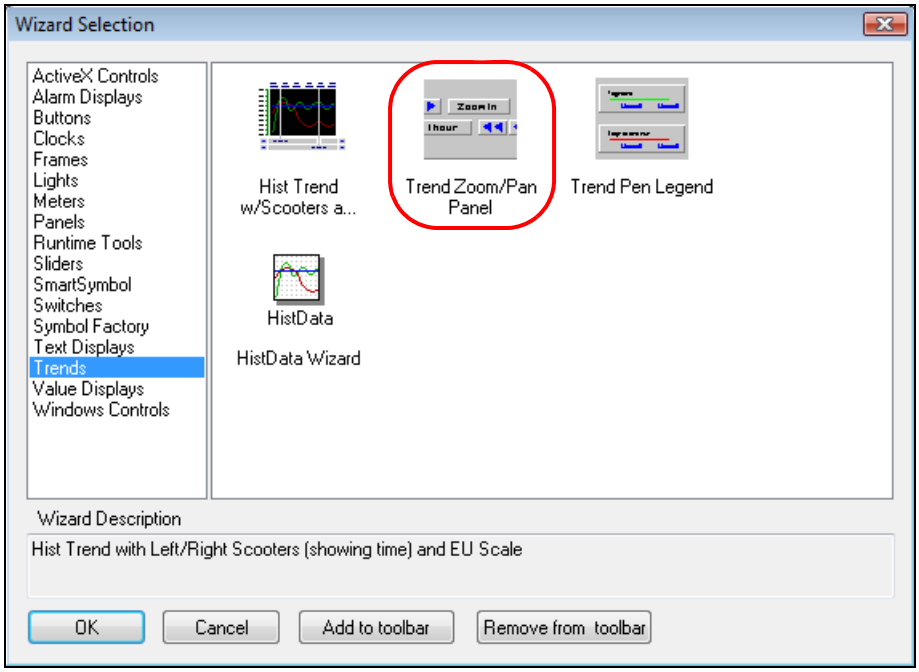


26. On the **Trend Pens** dialog box, click **OK**.
27. On the **Historical Trend Chart Wizard** dialog box, click **OK**.

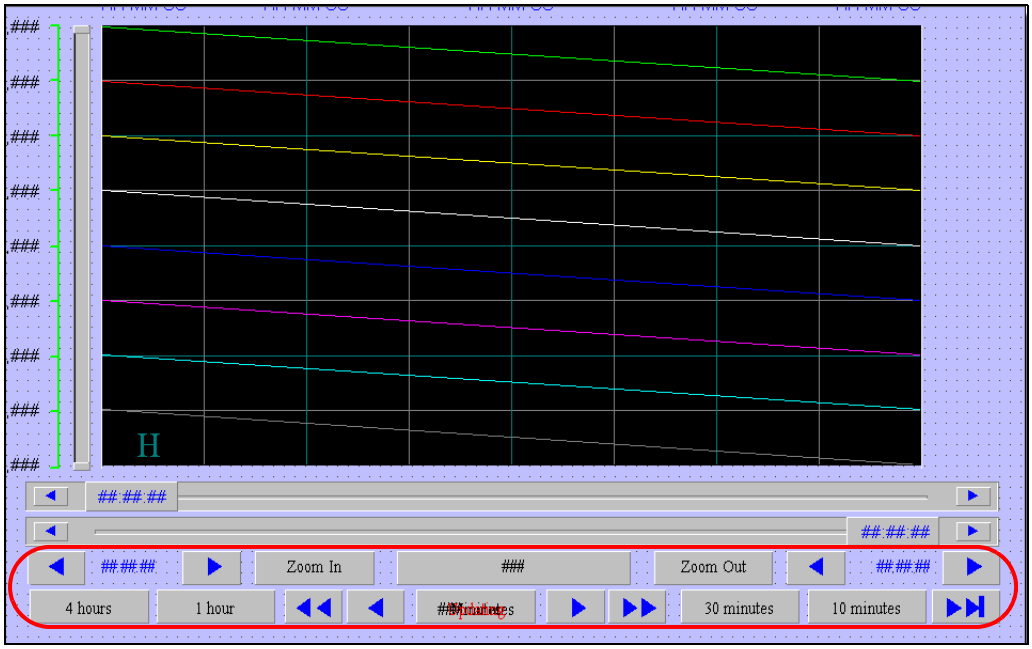
    This option automatically creates **HistTrend** and **HistTrendPenScale** tags in the **Tagname Dictionary**.

Configure Trend Panel Legend

28. Click the **Wizard** button again, then click **Trends**.
29. Double-click **Trend Zoom/Pan Panel**.

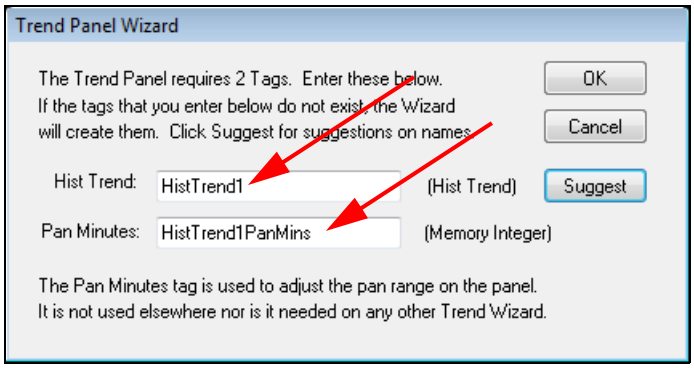


30. Place **Trend Zoom/Pan Panel** just below the **Hist Trend w/Scooters and Scale** wizard.
31. Resize **Trend Zoom/Pan Panel** to fit the window width.

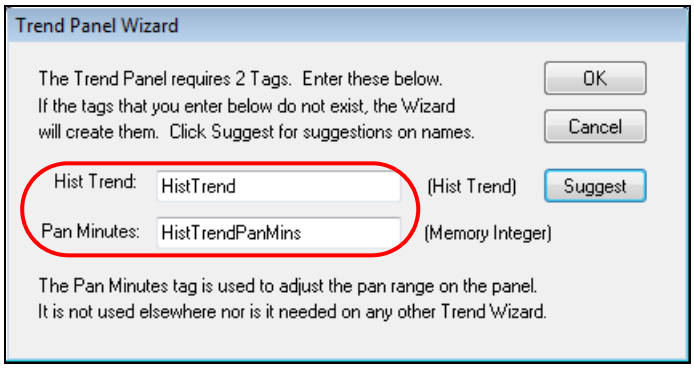


32. Double-click **Trend Zoom/Pan Panel**, then click **Suggest**.

    Notice the panel automatically increments your tagnames.

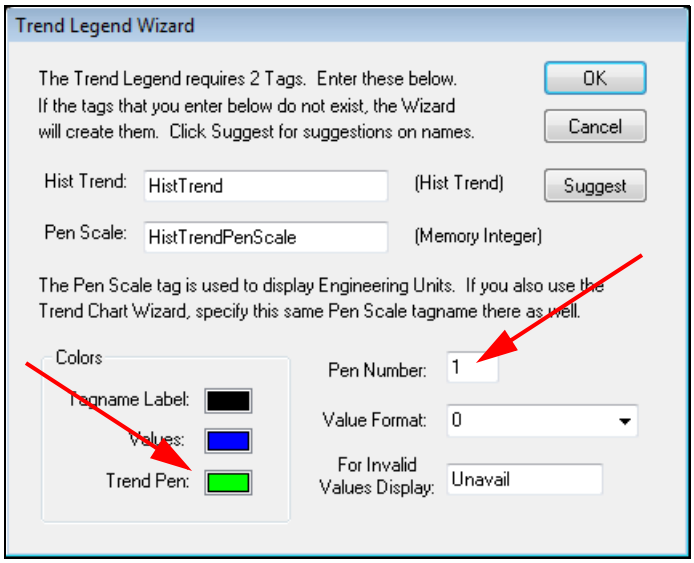


33. Name the tags **HistTrend** and **HistTrendPanMins** to match the same tags used to configure the **Historical Trend w/Scooters Scale** wizard (remove the **1** from the tagnames).



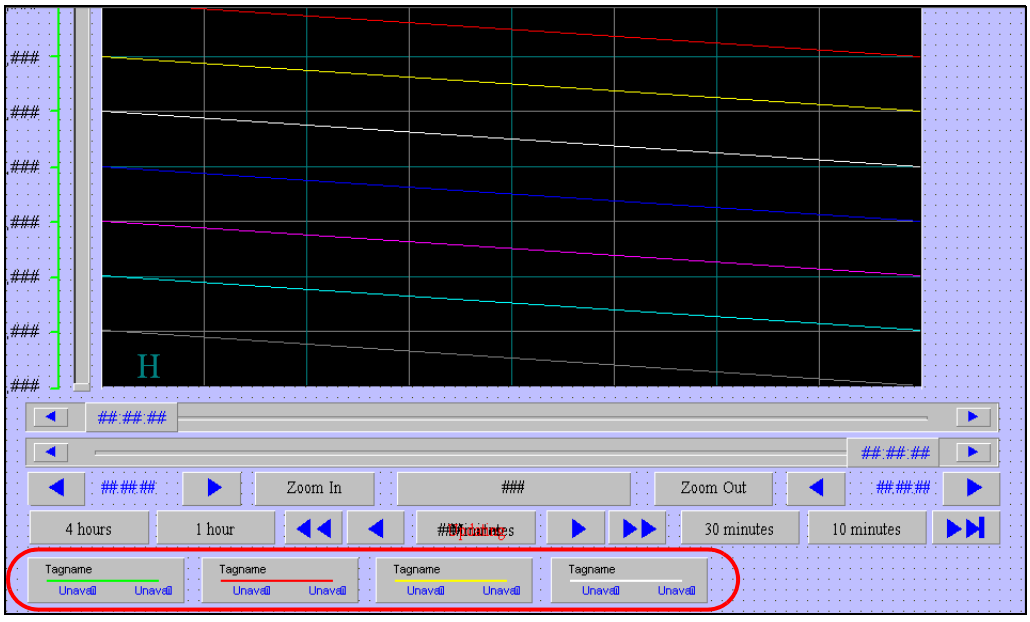
34. Click **OK**.

35. In the **Wizards Selection** dialog box, select **Trends\Trend Pen Legend** and place it on the window below **Trend Zoom/Pan Panel**.
36. Double-click the wizard, then click **Suggest**.
37. Remove the **1** from the tags in **Hist Trend** and **Pen Scale**.
38. Verify **Trend Pen** is green and **Pen Number** is **1**.



39. Click **OK**.
40. Duplicate the **Legend** wizards for **Pens 2**, **3**, and **4**, and configure as follows:

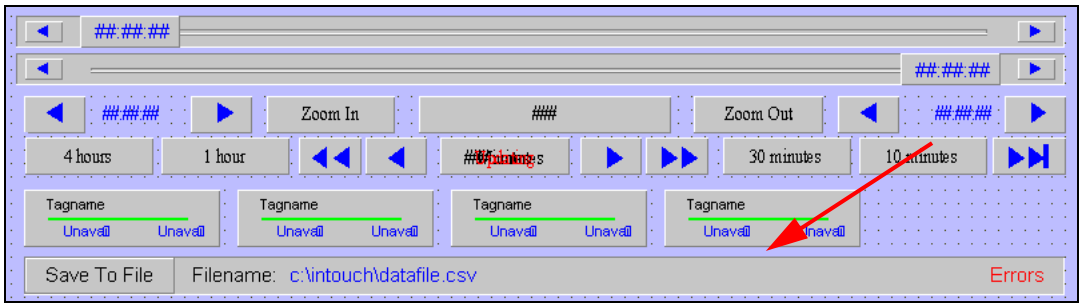
Colors: Trend Pen	Pen Number
Red	2
Yellow	3
White	4



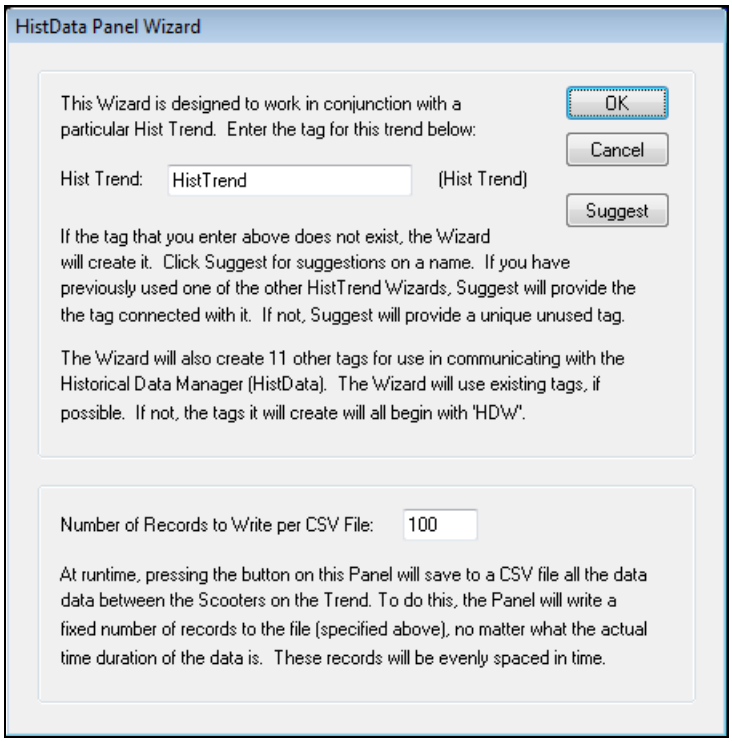
Create a HistData Wizard

Next, you add a HistData wizard to extract historical values and create a CSV file. You can use this file to produce a text-based version of the information from the trend chart.

41. On the window, place **Trends\HistData Wizard** below the pen legends and resize to fit the window width.



42. Double-click the wizard, then click **Suggest**.
43. In the **Hist Trend** field, change to **HistTrend**.
44. Change **Number of Records to Write per CSV File** to **100**.



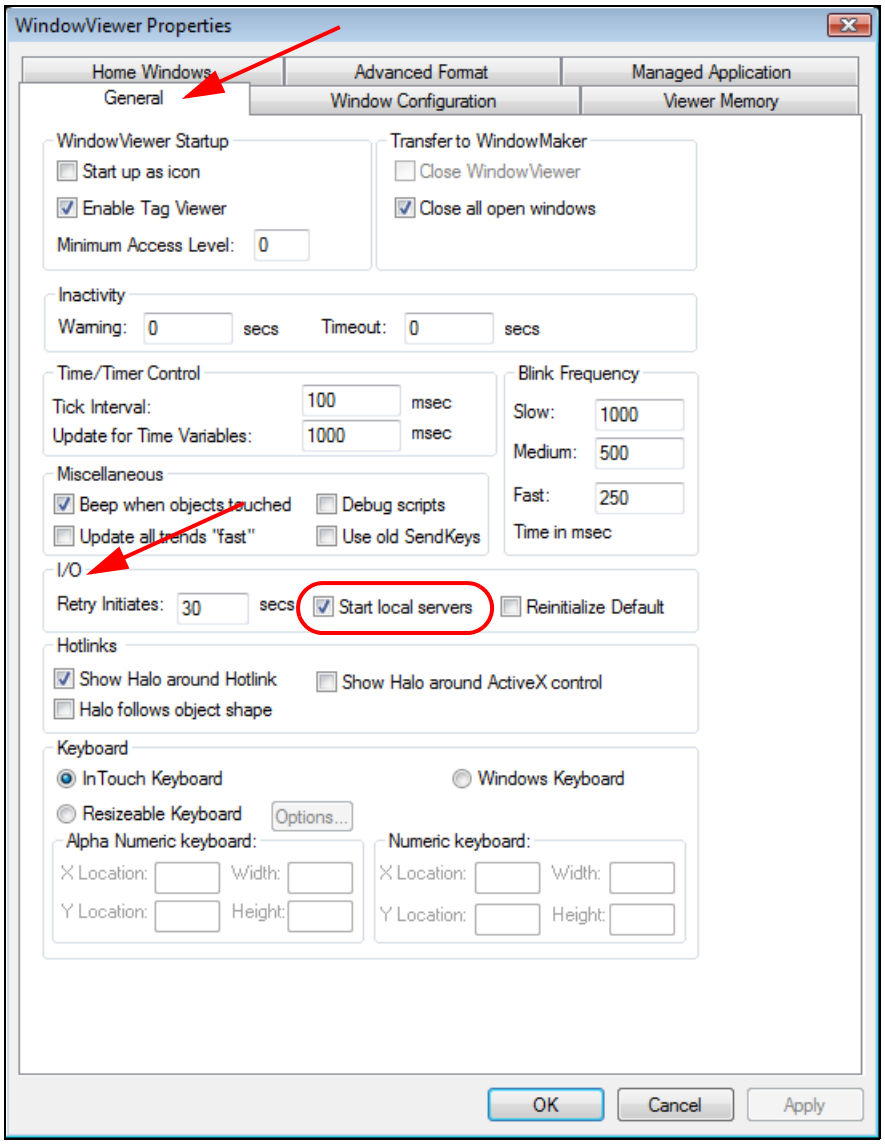
45. Click **OK**.

Note: This wizard creates an **Access Name** called “HistdataViewstr” which will be used to communicate with the Wonderware HistData software.

To ensure WindowViewer verifies HistData is running, you must verify the WindowViewer I/O configuration is set to automatically Start Local Servers.

46. In **Tools** pane, double-click **Configure\WindowViewer**.

47. On the **General** tab, verity that the **I/O** option **Start local servers** is checked.

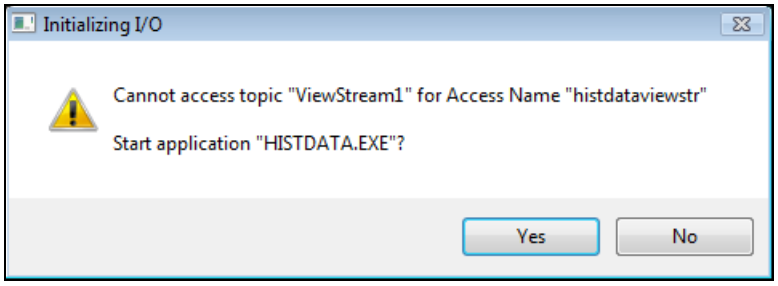


48. Click **OK**.

Test Trend History

49. Test in **Runtime**.

    The following prompt displays:



50. Click **Yes**.

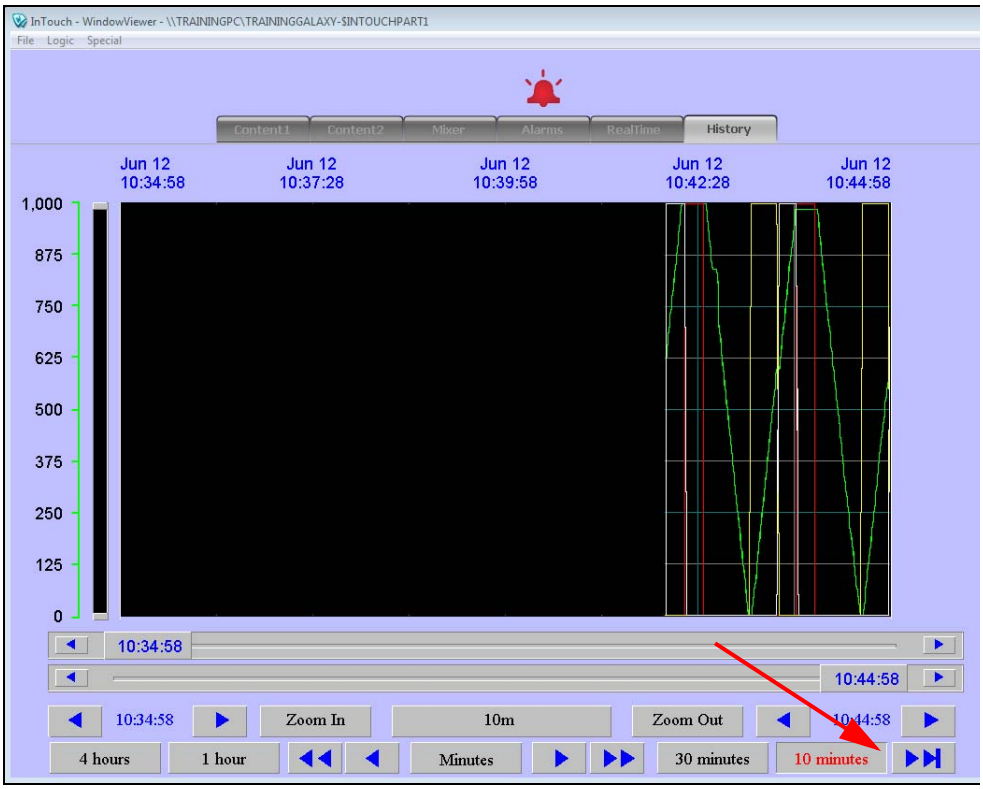
Note: If you click **Yes** and **HistData.exe** is in the system environment variable path, it will automatically start the program.

51. Confirm the **Historical Data Manager** task is running on the Windows taskbar.

Note: To manually start the **HistData** server, click the **Start** menu and select **All Programs | Wonderware | InTouch | HistData**.

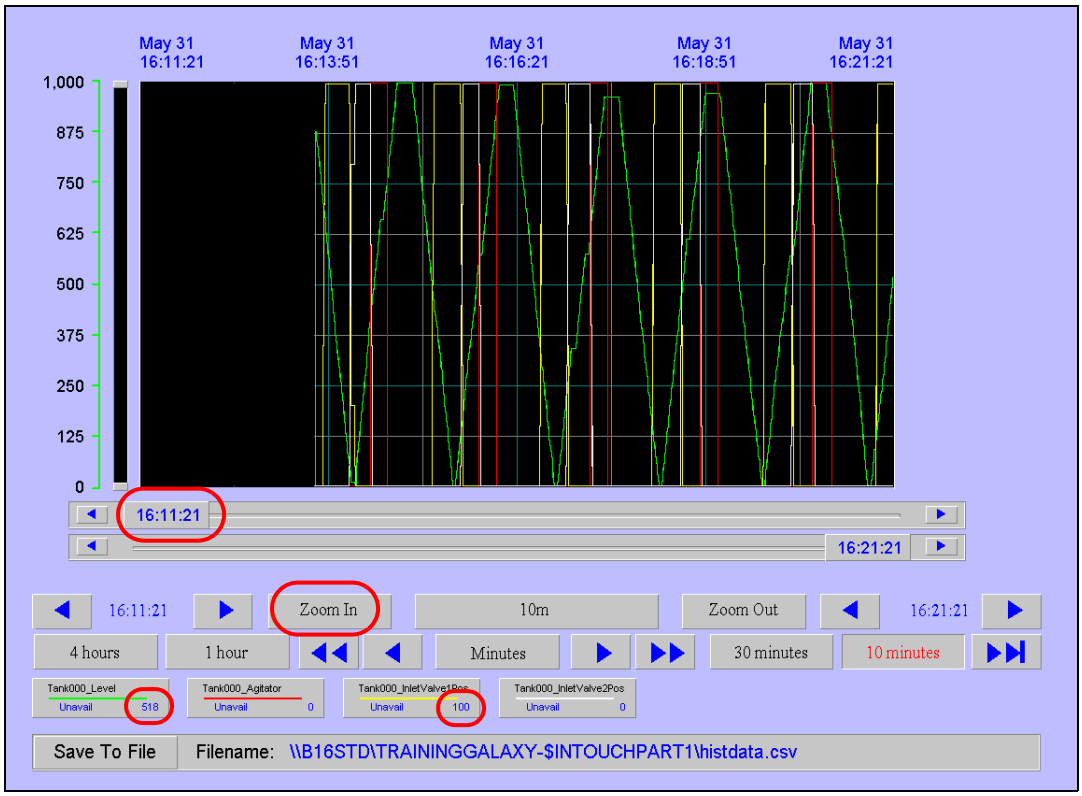
52. Click the **History** tab.

53. Click the current time button ⏭ to go to the current time.



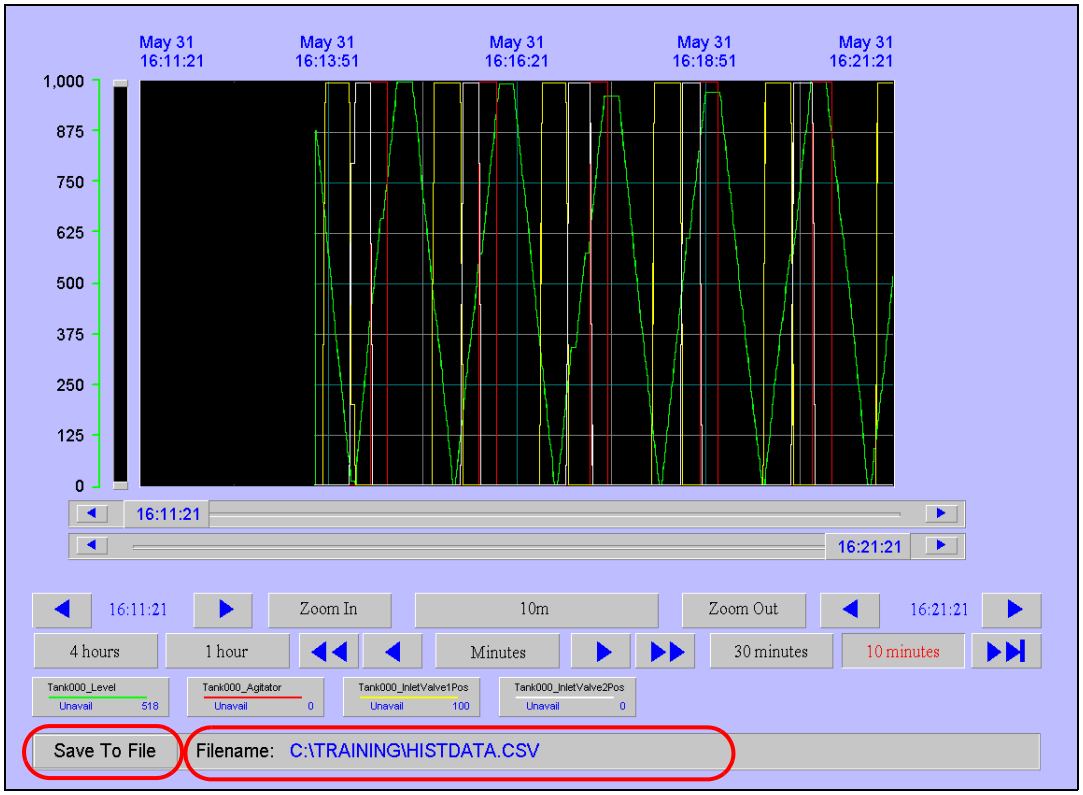
54. Slide the left scooter until it touches the edge of the visible data in the trend.
55. In **Trend Zoom\Pan Panel**, click the **Zoom In** button.

    Notice the legend for each **Pen** data as you drag the slider across the view. The values in each **Trend Pend Legend** changes based on the slider location. The left value is based on the location of the left slider line. The right value is based on the location of the right slider line.



56. Click **Zoom In** or **Zoom Out** to view different levels of detail.
57. In **HistData Wizard**, click the blue text and enter **C:\Training\HistData.csv**, then press **Enter** to save the change.

58. Finally, click **Save To File** to save your historical data.



Note: To view the historical data file, navigate to **C:\Training** and double-click **HISTDATA.CSV**.



invensys
Learning Services

Module 6 – Application Security

Section 1 – InTouch Security 6-3

Lab 15 – Securing Your Application 6-9

Lab 16 – Managing Automatic Logout 6-29

Module Objectives

- Define basic security models
- Explain the InTouch security model
- Explain the OS security model
- Implement InTouch security

Section 1 – InTouch Security

This section provides an overview of ArchestrA, InTouch, and Operating System security models. The InTouch-based security model is then described in greater detail. Additional security options through runtime environment customizations have also been explained here.

Overview

InTouch security provides three security models for the InTouch developer. The three security models are as follows:

- **InTouch Security** – Control specific functions a user is allowed to perform by linking those functions to internal tags. Create audit trails to tie the user to all alarms and events that occur while the user is logged on to the system.
- **ArchestrA Security** – ArchestrA security is centralized industrial security used for Application Server, and is centralized whether Network Application Development (NAD) is used or not.
- **Operating System Security** – System based security authentication using user names and windows domain to control access to certain application control or functionality. The operating system security model allows InTouch to meet security requirements in certain regulated industries.

This course and labs focus on using InTouch Security with light lecture coverage of OS Security.

Securing an InTouch Application

After the basics of what it means to secure and application, the need for security is varied. You will take this from the perspective of InTouch Application Developers, who primarily want to secure unauthorized access to the operating system, audit user actions, and secure parts or all of the InTouch application, such as navigation to restricted controls. To accommodate the different needs, the InTouch security scheme includes various options for a variety of applications.

In some cases, the top security concern is to limit the user access to other Windows applications or the operating system on the local node.

Script functions are available to lock the use of keys like <Alt> or <Tab>. There are ways to configure WindowViewer (View) to always run it maximized, or to hide its menu so that users cannot close it.

In other instances, security can include auditing the users' actions (keeping track of what was done and when it was done). InTouch events and inactivity system tags can track these actions.

In other cases, security would mean restricting features of the application based on privileges set forth for the current user (like a User ID), or the group the user belongs to (like Roles). This functionality is accomplished by setting a **visibility** or **disable** link, for example, a push button which navigates to another window containing sensitive financial or safety information.

InTouch retains the traditional InTouch security based on a binary file containing a list of users of the application with passwords and access levels. InTouch includes additional security script functions and system tags.

ArchestrA Security

ArchestrA-based security for Application Server is centralized regardless of whether NAD is used or not. Application Server Security activities are configured externally in the Integrated Development Environment (IDE). For more information on the IDE, see the *Wonderware ArchestrA Integrated Development Environment (IDE) Guide*.

The details for features available with the Application Server are covered in a separate course.

InTouch Security

By applying security to your application, you can control specific functions that a user is allowed to perform by linking those functions to internal tags. Audit trails can be created that tie the user to all alarms/events that occur during the time he/she is logged on to the system.

Security is based on the concept of the user logging on to the application and typing an operator name and password. The administrator configures a user name, password, and access level for each user.

There is no association between Microsoft Operating System Security and InTouch Security; this is done using Operating System Security.

Operating System Security

The security authentication model is operating system-based. User names are the Windows Domain Name/User name pairs (logon name and full name). Each required user name is assigned to a particular group in the domain. The group is assigned an access level that can be used to control access to certain application control / functionality. Operating system-based security uses the InTouch script function **AddPermission** to assign and maintain a list of users and their corresponding access levels. This list, created after the execution of the AddPermission() call, is written to disk. The Operating System Security model allows InTouch to meet security requirements in certain regulated industries. One of these specifications established by the Food and Drug Administration is FDA 21 CFR Part 11 electronic signatures regulation.

Implementing Security

To implement security in an InTouch application, you must first create users. Next you create a way to log in to the application window. When a new application is created, the user name is set to **None** with an access level of **0** and has no password (blank). This access level does not allow access to security comments. This level also prevents access to the **Configure Users** command in both WindowMaker and WindowViewer.

It is important to configure a user name for the System Administrator with an access level equal to or greater than 9000 in order to access Configure Users.

Once a user logs on to the application, access to any protected function will be granted upon verification of the users' password and access level against the value specified for the internal security tagname linked to the function. For example, you can control access to a window, the visibility of an object, and so on by specifying that the logged on users' Access Level must be greater than 2000.

The commands used to establish security on an application are located in the **Security** command under the **Special** menu in both WindowMaker and WindowViewer. The security commands are used to log on and off the application, change passwords, and configure the list of valid user names, passwords and access levels. The user can log on to the application on the **Special** menu by selecting **Security | Logon** in WindowViewer, or by creating a custom log on window with user

input animated buttons linked to internal security tags such as $OperatorEntered and $PasswordEntered. As soon as a legitimate entry has been made to both of these $System tags, the system will automatically login.

Securing the Operating System

Many InTouch applications require a user to constantly monitor processes running around the clock. In these cases, it is often desirable to have InTouch as the only Windows application that users have access to. The operating system task bar, system files, and other Windows applications are inaccessible if View is configured properly. It is recommended after logging in as administrator, that the administrator password be changed.

Securing the Application

InTouch provides a number of System tags and script functions that allow you to configure security for the individual application. By applying security to your application, you can control specific functions a user is allowed to perform by linking those functions to internal tagnames. Once a user is validated and logged on to the InTouch application, the **$Operator** and **$AccessLevel** system tags contain information about the user and their access level.

To log on to an InTouch application, on **Special** menu, users select **Security | Logon**, call a script function to display a Logon dialog box, or log on via a custom logon window.

Note: See the InTouch user guides for complete documentation on System Tags and script functions.

$AccessLevel and $Operator are the best tags to use for monitoring security and to determine access for areas and features of your application. These are $System tags used to control what a user has access to during Runtime. $AccessLevel is a best practice choice primarily because over time users are promoted, demoted, fired, and hired, and any function tied to a specific user name will be disabled for all others. On the other hand, tying a navigation button to an access level of 4000, then giving all trained operators level 4000 will grant access to the right group of people.

For example, to disable a pushbutton that displays Environmental or Safety Information, assign a Disable animation link to the navigation button using an expression of **$AccessLevel < 9000** with animation of **Disabled**.

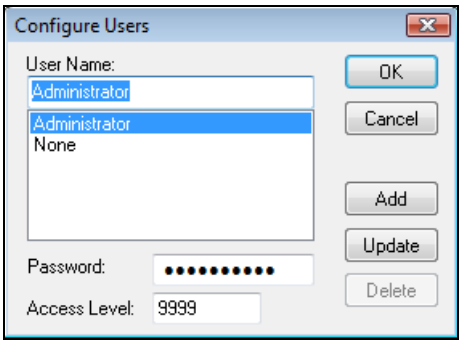
Configuring Security Levels

To configure InTouch Security, on the **Special** menu, select **Security | Select Security Type | InTouch**.

The first time that you use security, you must log on as:

- Name: **Administrator**
- Password: **wonderware**

Use the **Configure Users** dialog box to change the Administrator password and to add users to InTouch.



Note: Use caution when updating the Administrator password. An error could prevent you from logging in again as administrator.

To create a new user, you must enter the following information and **Add** the following information:

- **User Name:** Assign a name to the user
- **Password:** Enter a password (up to 32 characters)
- **Access Level:** Enter a value representing access level (lowest = 0 to highest = 9999)

To modify an existing user name, display the **Configure Users** dialog box to make the changes. When complete, use Update to save the changes.

Creating a Custom Security Logon Window

$OperatorDomainEntered, **$OperatorEntered**, and **$PasswordEntered** are the three tags to be used to create a custom logon feature. To do this, create a popup window and set user touch links (user input string) to these tags. If the provided credentials are validated, **$Operator**, **$AccessLevel**, **$OperatorName**, $**OperatorDomain** will update based on the user input.

Note: The **$OperatorDomainEntered** tag is for OS Security only.

Configuring Log In and Log Off

Logging on to and logging off from an InTouch application varies by the type of security used to protect an application. You can configure the application to automatically log off the user after specified inactivity duration.

Log On

In addition to configuring user log on, you can configure log on options using scripts and with System tags.

- Use the **PostLogonDialog()** script to show the InTouch Logon dialog box and returns TRUE.
- Use the **AttemptInvisibleLogin()** to attempt logging on to InTouch using the supplied credentials.
- Use **System Tags** $OperatorEntered, $PasswordEntered and $OperatorDomainEntered in a script to set the user name, password, and domain name. These tags are internal message type tags that are intended for write operation only.

If the log on is successful, the $AccessLevel system tag is set to the predefined value associated with the user in the InTouch security user list.

Log Off

Use the following scripting and System tag options to configure user log off options.

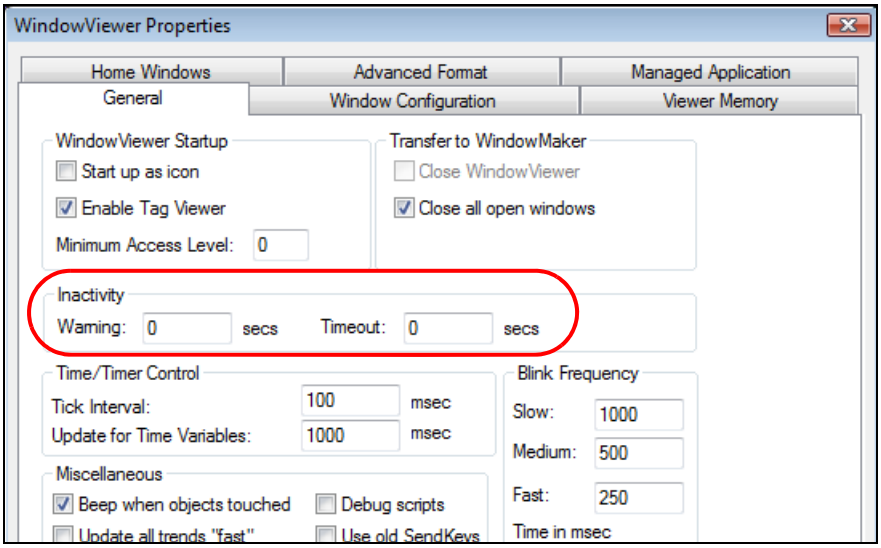
There are four topics pertinent to logging out of the InTouch security system.

- Use the **LogOff()** script to log, logs off the currently logged on user and sets the current user status to the default None user.
- Use the **$OperatorEntered = “None”** System Tag when you create a custom log on window. You can link touch-sensitive input objects and QuickScripts to this tag to set the user name for the logon.

When you log off your InTouch application using the Log Off option from **Special | Security**, the User Name is reset to **None** with an Access Level of **0**.

Automatic Log Off

The warning and timeout settings can be used to configure your application so that it automatically logs the user off if there has been no activity for a specified period of time. Configure the warning and timeout settings in the **WindowViewer Properties** dialog box accessed from **Configure** options.



Enter the number of seconds that can elapse with no user activity (mouse clicks or keystrokes) before the system discrete tag **$InactivityWarning** is set to **1** (True). The **Timeout** feature can be used independently of the **Warning** feature. However, the Timeout value must be greater than the Warning value for proper use of both system tags. For example, Warning becomes true after 30 seconds of inactivity, and Timeout becomes true after an additional 15 seconds (for a total of 45 seconds) of inactivity.

Additional security implementation methods and techniques are discussed further in the next Module “Enhancing Your Application with QuickScript”.

Lab 15 – Securing Your Application

Introduction

In this lab, you enable InTouch security, configure users and passwords, and use InTouch animations to secure parts of the InTouch application.

Objectives

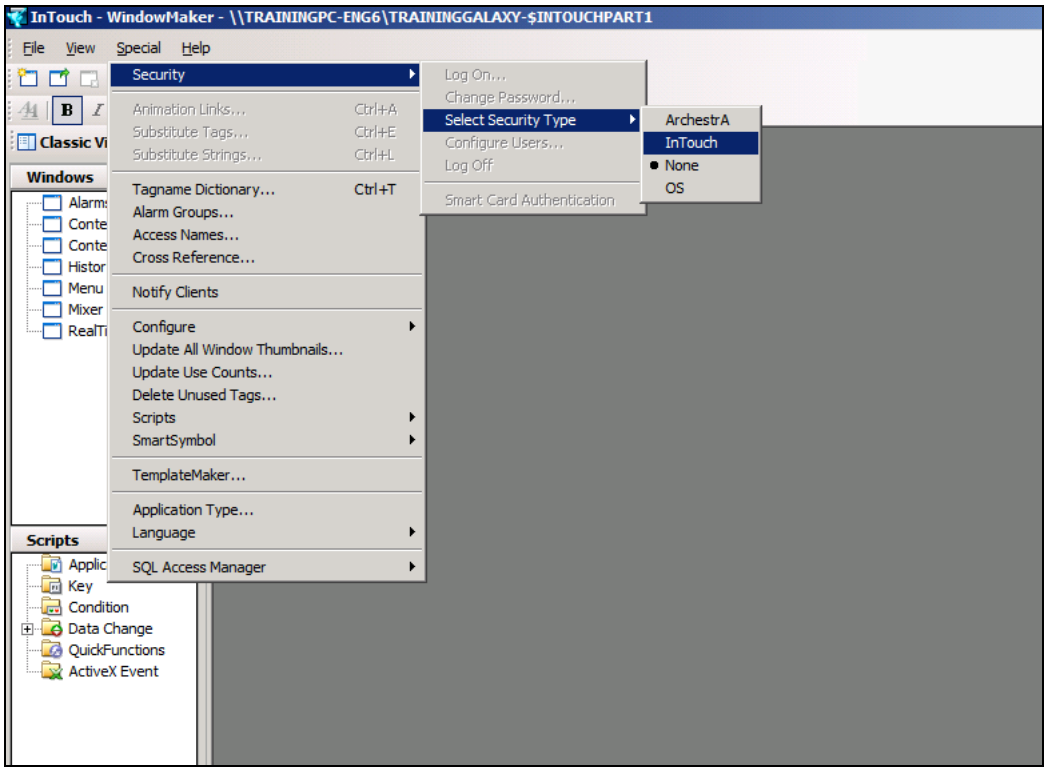
Upon completion of this lab, you will be able to:

- Enable InTouch security
- In InTouch, create users and passwords
- Use system tags to create an operator login
- Use InTouch Animations to secure your application
- Track user with alarm acknowledgment

Enable Security

First, you enable security for your InTouch application.

1. Close WindowViewer.
2. In WindowMaker, on the **Special** menu, click **Security | Select Security Type | InTouch**.



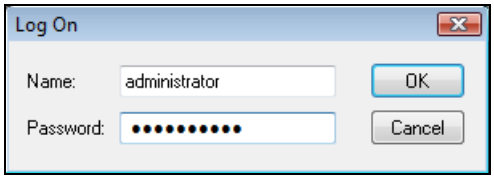
3. On the **Special** menu, click **Security | Log On**.

   The **Log On** dialog box appears.

4. Enter the following credentials:

   Name: administrator

   Password: wonderware

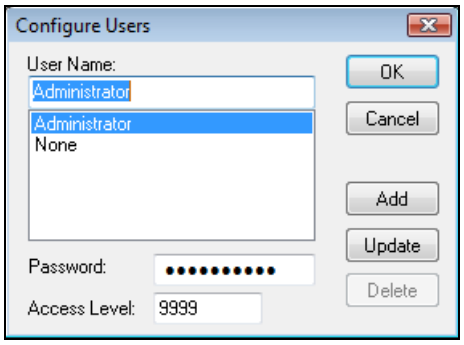


5. Click **OK**.

Next, you configure users.

6. On the **Special** menu, click **Security | Configure Users**.

   The **Configure Users** dialog box appears.



7. Click **Administrator** and change the password to **iom**.

Important: Be careful at this step, any mistake in spelling could prevent you from logging in again as **Administrator**.

Note: The security level that permits configuring users, is set internally in the software to >=9000. This cannot be altered by the user.

8. Click **Update**.
9. In the **User Name** field, enter the following information to add a supervisor user named **Clipboard**:

   User Name: Clipboard
   Password: iom
   Access Level: 8000

10. Click **Add**.
11. Enter an operator user named **Wrench** as follows:

    User Name: Wrench
    Password: iom
    Access Level: 5000

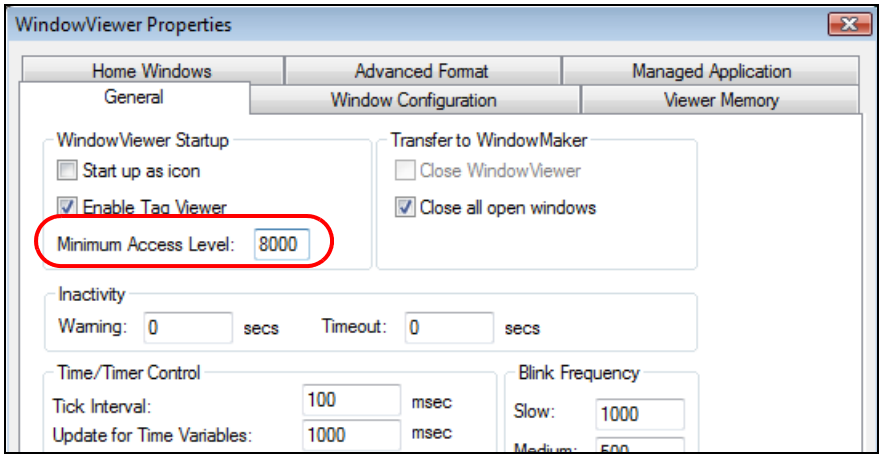
12. Click **Add**.
13. Click **OK**.

Now, you secure the Tag Viewer from unauthorized users.

14. In **Tools** pane, double-click **Configure\WindowViewer**.

    The **WindowViewer Properties** dialog box appears.

15. On the **General** tab, **WindowViewer Startup** area, change **Minimum Access Level** to **8000**.



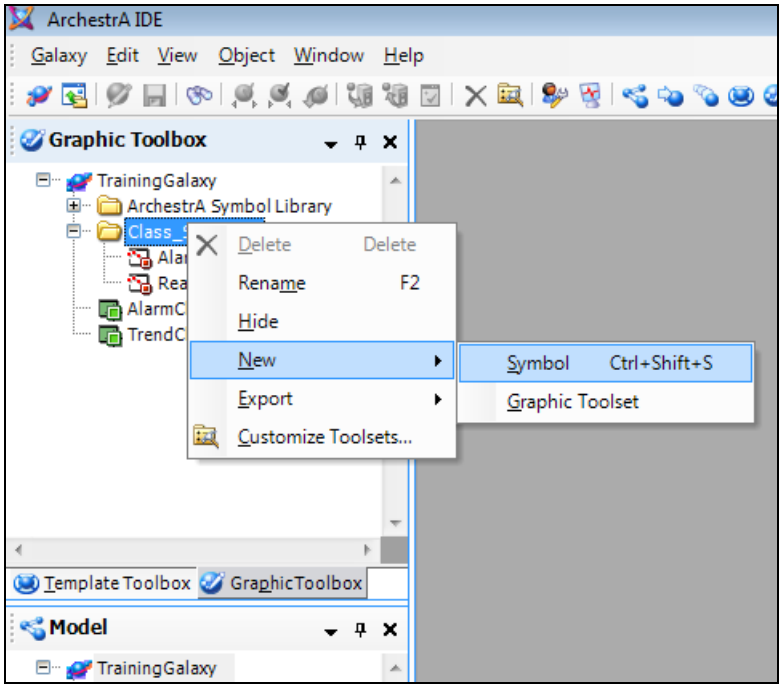
    This restricts **Tag Viewer** access to the users **Clipboard** and **Administrator**.

16. Click **OK**.

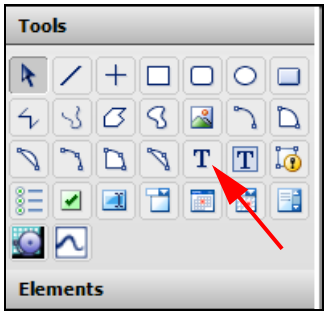
Add Logon Functionality

Next, you add a security login panel to your Menu window.

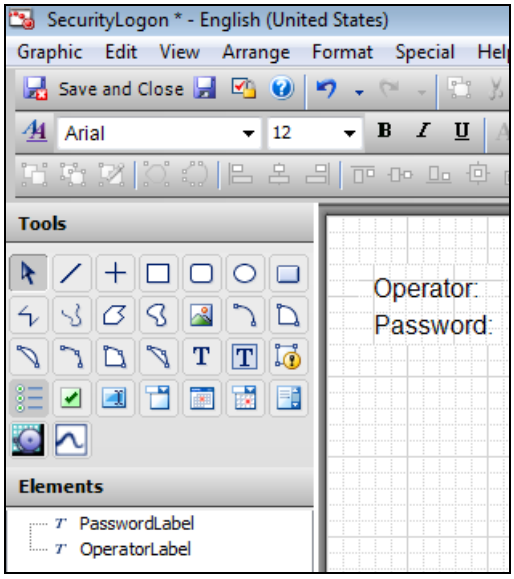
17. In ArchestrA IDE, in the **Class_Symbols** toolset, create a new symbol.



18. Name the new symbol **SecurityLogon**.
19. Double-click **SecurityLogon** to open the **ArchestrA Symbol Editor**.
20. In **Tools** pane, click the **Text** tool, then click on the canvas to enter a label.



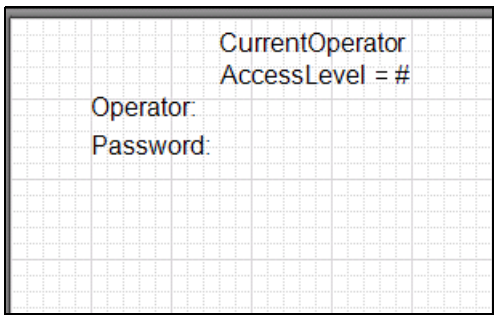
21. Enter **Operator:**, then press **Enter** to start a new text element.
22. Enter **Password:**, then press **Enter**.
23. Rename the **Text1** element as **OperatorLabel**.
24. Rename the **Text2** element as **PasswordLabel**.



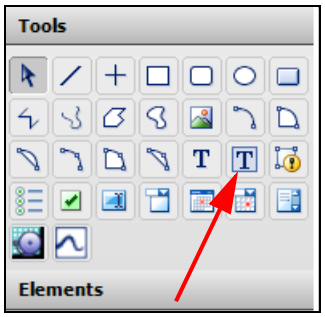
25. Add two more text labels and rename as follows:

Text	Name
CurrentOperator	OperatorDisplay
AccessLevel = #	AccessLevelDisplay

26. On the canvas, place the text as shown below.

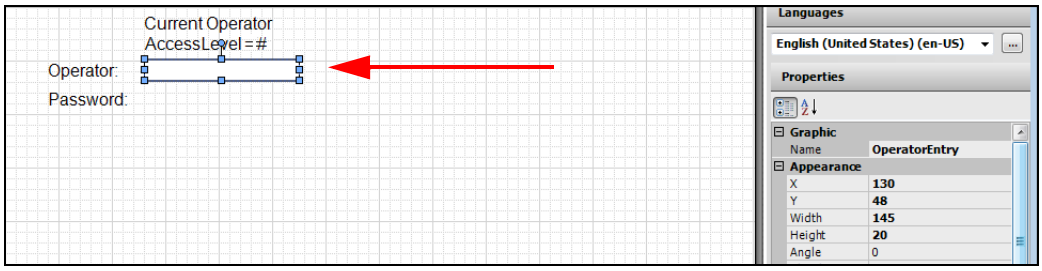


27. Click the **Text Box** tool, then click and drag a rectangle on the canvas.



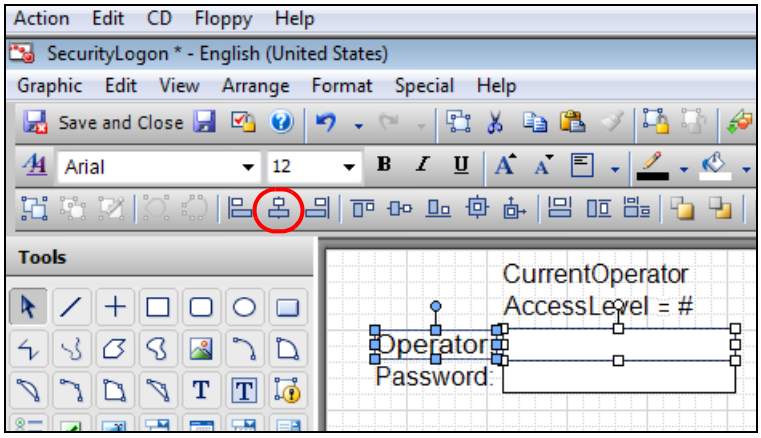
28. Press **Delete** key to remove the text, then press **Enter** to complete the **Text Box** element.
29. Adjust the **Width** to **145** and the **Height** to **20**.

    Position as shown:



30. Name the text box **OperatorEntry**.
31. Duplicate the text box and place it next to the **Password:** label.
32. Name the text box **PasswordEntry**.
33. Click **OperatorLabel**, then hold down the **Shift** key and click **OperatorEntry** to select both elements.

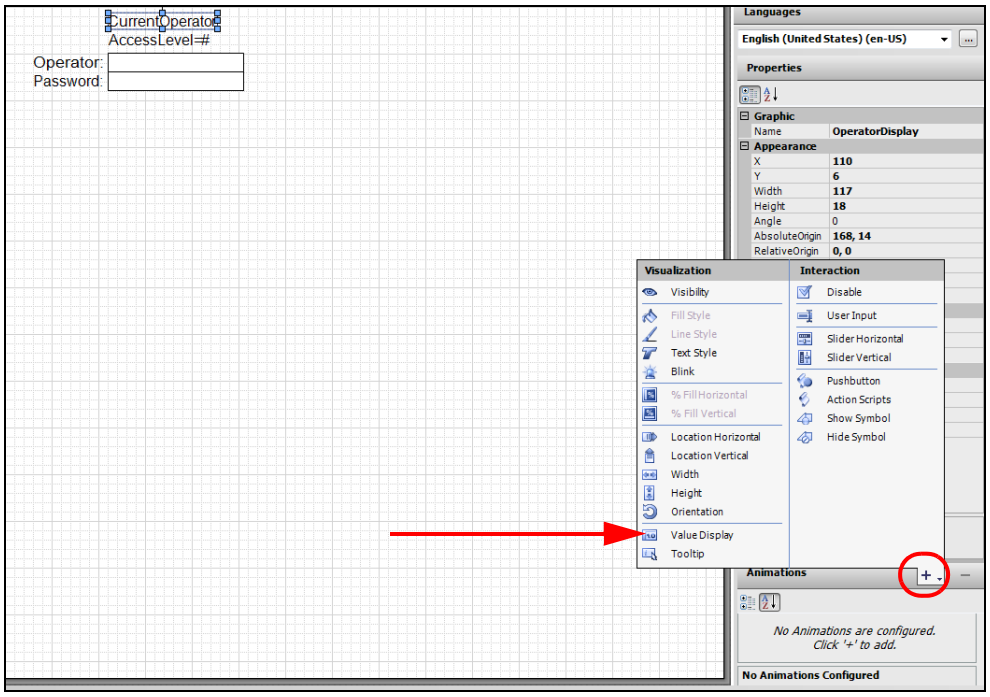
34. On the toolbar, click **Align Middle**.



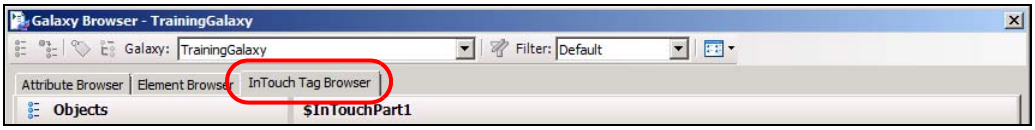
35. Repeat the previous steps to align **PasswordLabel** and **PasswordEntry**.

36. In **Elements** pane, click **OperatorDisplay**, then in the **Animations** pane click **+** to add an animation.

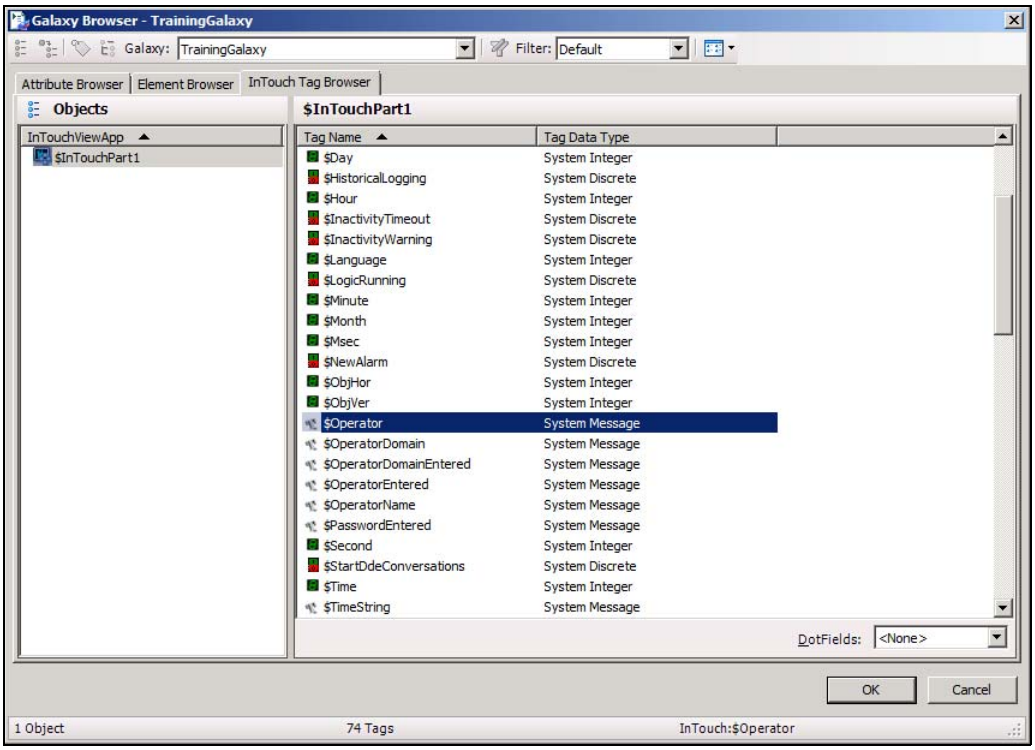
37. Click **Value Display**.



38. Click the **String** button, then click **Browse Galaxy** ... at the end of the **String** field.
39. In **Galaxy Browser**, click the **InTouch Tag Browser** tab.

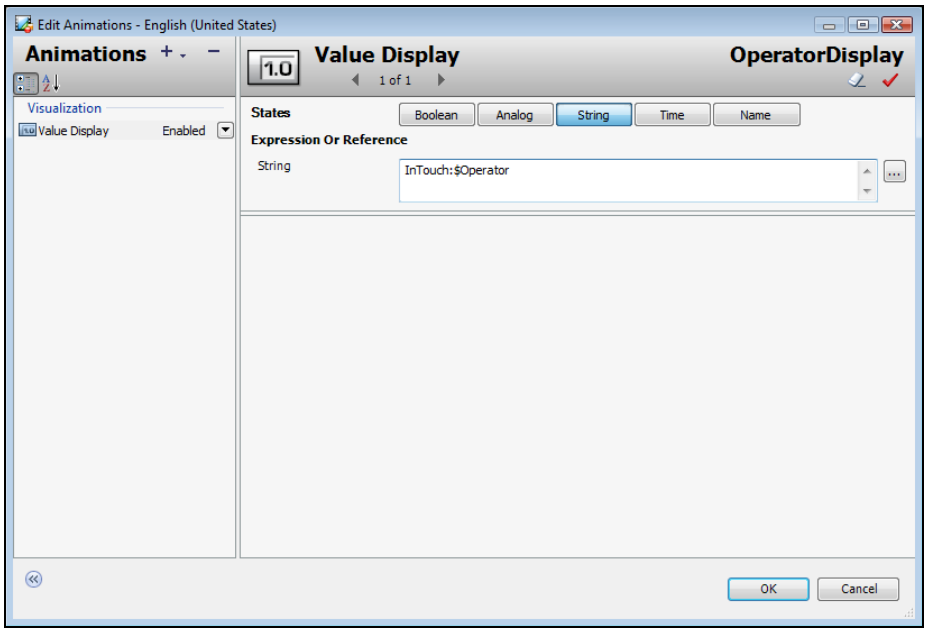


40. In **Objects** pane, under **InTouchViewApp**, click **$InTouchPart1**.
41. In the list of tags for **$InTouchPart1**, click **$Operator**.



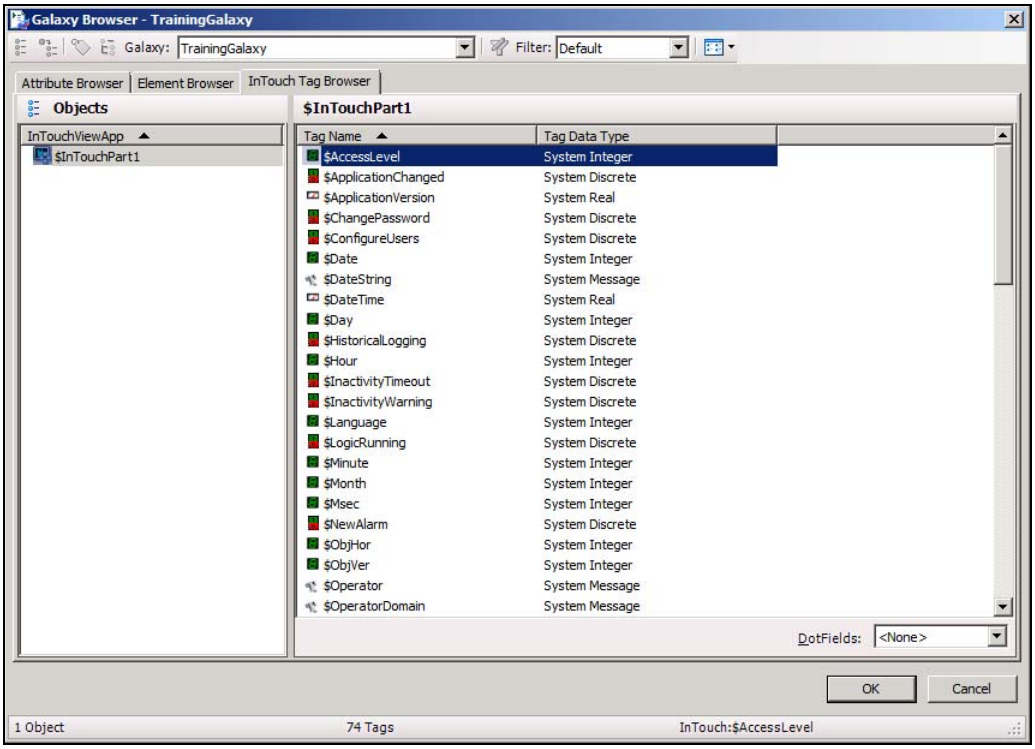
42. In **Galaxy Browser**, click **OK** to return to the **Edit Animation** dialog box.

    The selected tag appears in the **String** field as **InTouch:$Operator**.

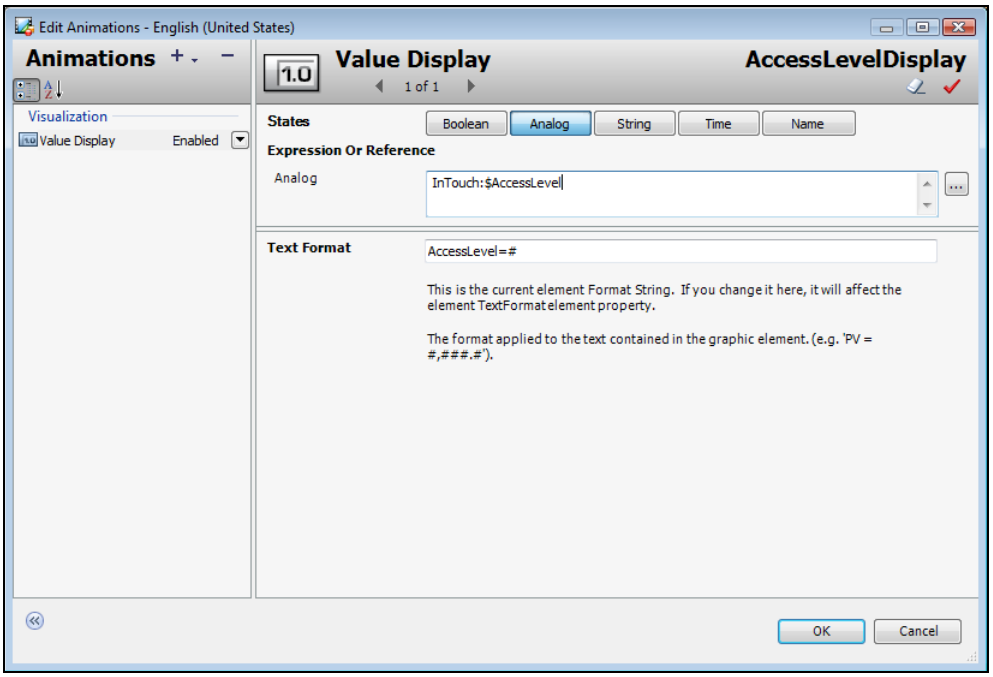


43. Click **OK**.
44. In **Elements** pane, click **AccessLevelDisplay**, then in the **Animations** pane click **+** to add an animation.
45. Click **Value Display**.
46. Click the **Analog** button, then click **Browse Galaxy** ... at the end of the **Analog** field.

47. In **Galaxy Browser**, in the **InTouch Tag Browser** tab**,** double-click **$AccessLevel**.



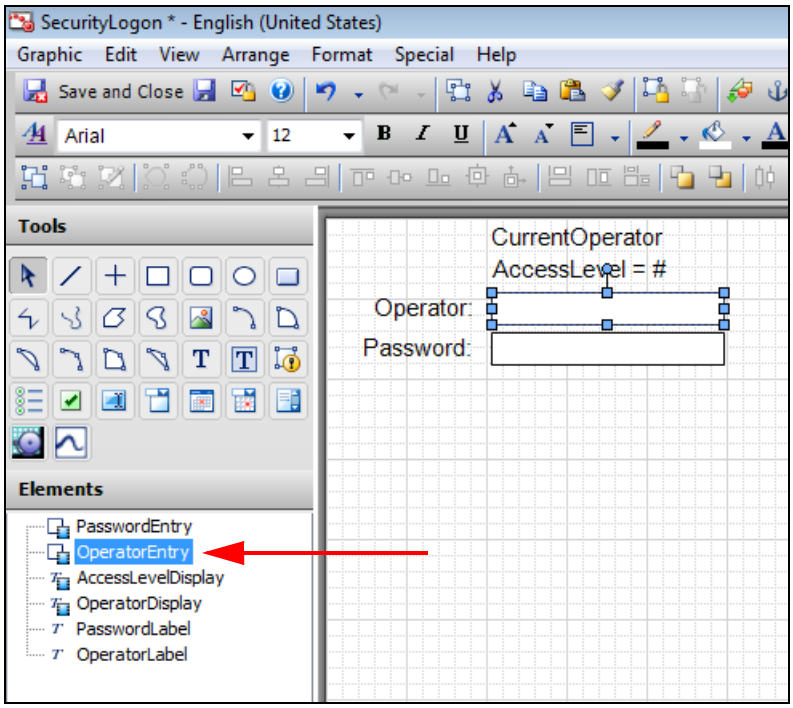
The selected tag appears in the **Analog** field as **InTouch:$AccessLevel**.



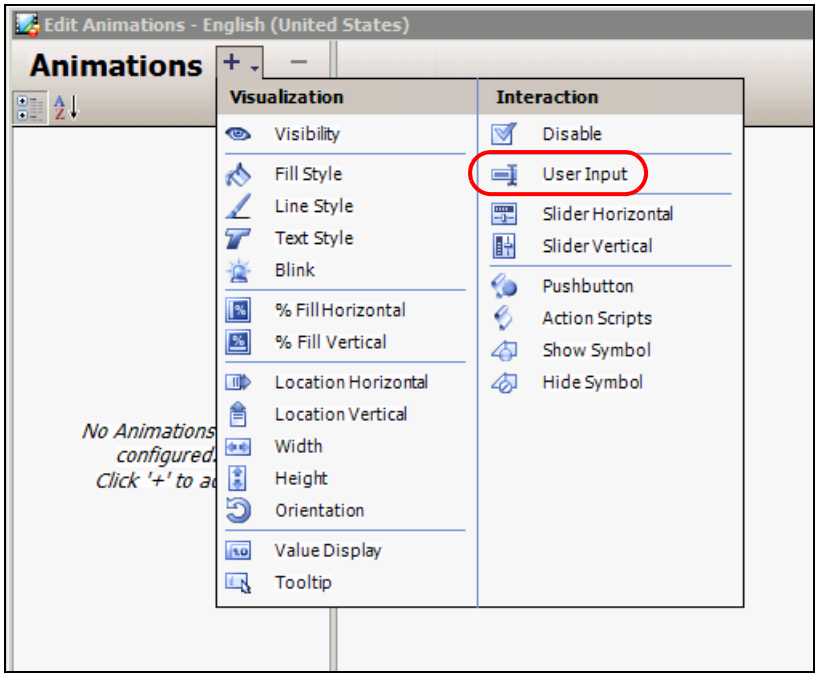
48. Click **OK** to close the **Edit Animations** dialog box.

Next, you animate the operator entry and password entry fields to allow user input.

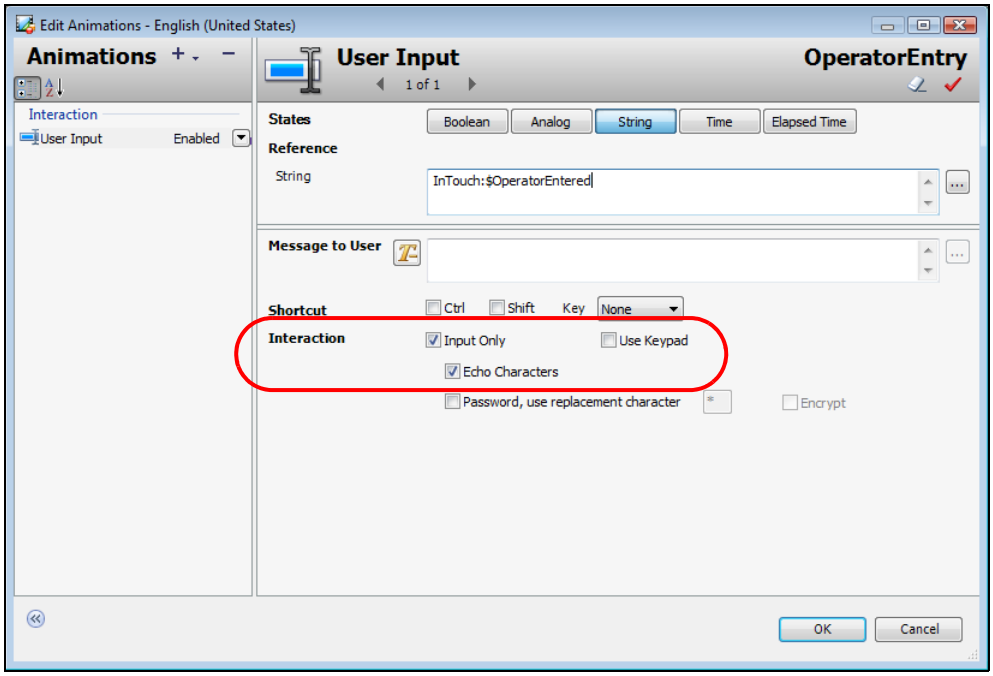
49. In **Elements** pane, double-click **OperatorEntry**.



50. Click **+** and select **User Input**.

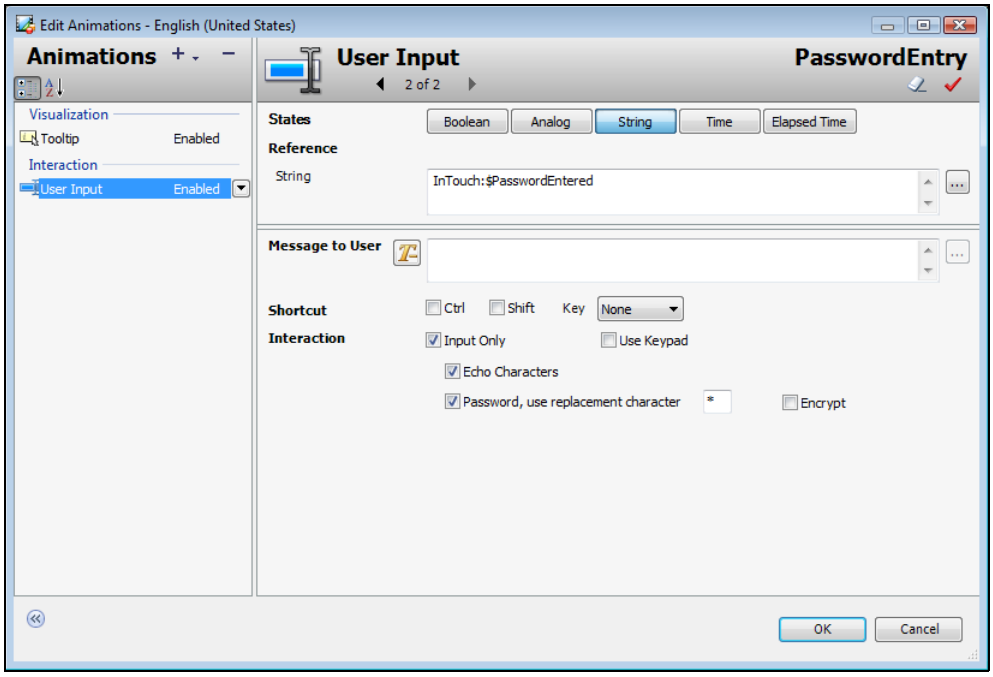


51. Select the **String** state, then click **Browse Galaxy**.
52. From **InTouch Tag Browser**, select **$OperatorEntered**.
53. In the right pane, in the **Interaction** area, check **Input Only**, and uncheck **Use Keypad**.
54. Ensure **Echo Characters** is checked.



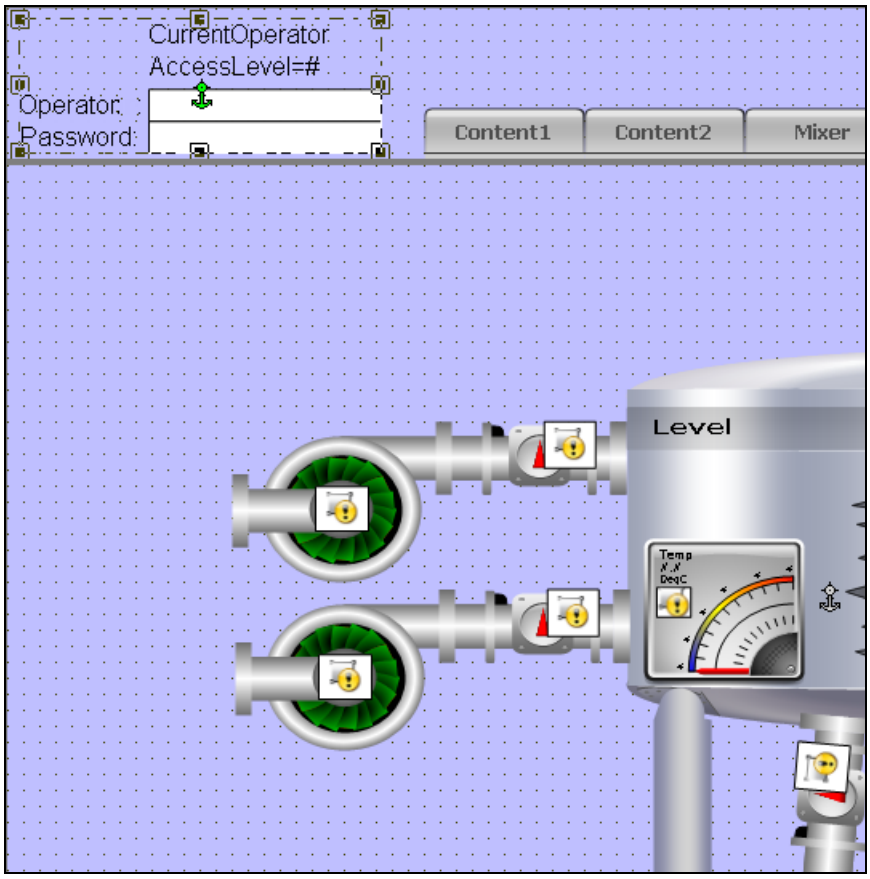
55. Keep the remaining default options and click **OK**.

56. In **Elements** pane, double-click **PasswordEntry**, then click **+** and select **User Input**.
57. Select the **String** state, then click **Browse Galaxy**.
58. From **InTouch Tag Browser**, select **$PasswordEntered**.
59. In the right pane, in the **Interaction** area, check **Input Only**, and uncheck **Use Keypad**.
60. Check **Password, use replacement character**.



61. Keep remaining default options and click **OK**.
62. Click **Save and Close**, then enter an appropriate comment.

63. In WindowMaker, open the **Menu** and **Mixer** windows.
64. In the **Menu** window, embed the **SecurityLogon** symbol.

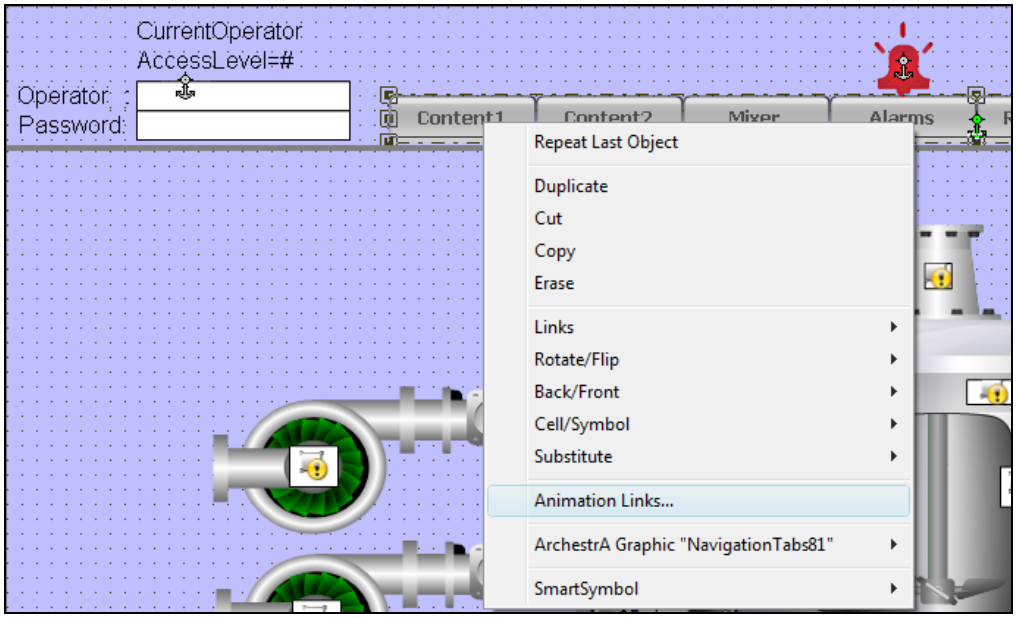


Adjust the graphic to fit, as needed.

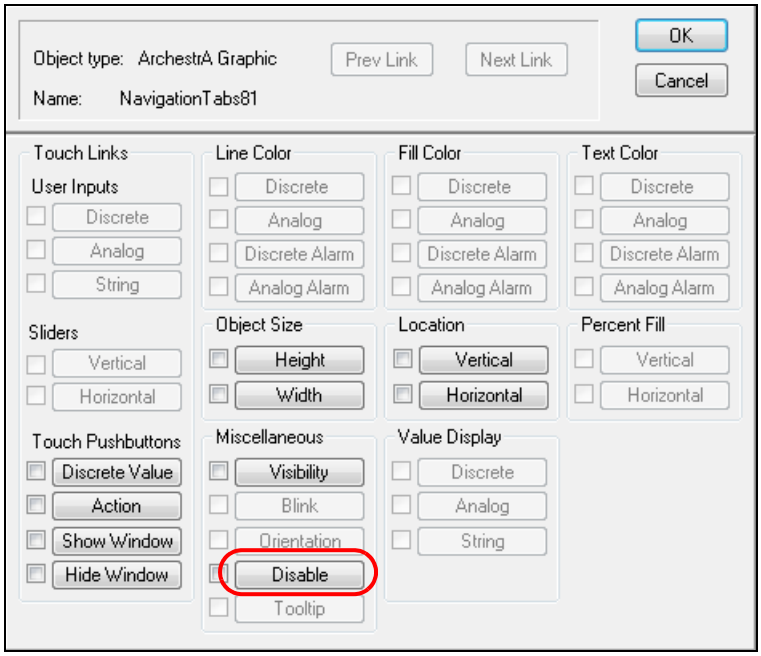
Secure the Application

Lastly, you secure your ability to navigate from window-to-window in Runtime.

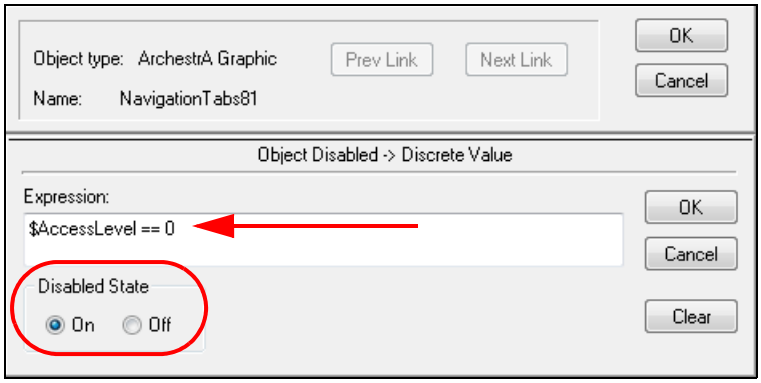
65. Right-click **NavigationTabs8** and select **Animation Links**.



66. In the **Miscellaneous** area, click **Disable**.



67. In the **Expression** field, enter **$AccessLevel == 0**, then change the **Disabled State** to **On**.



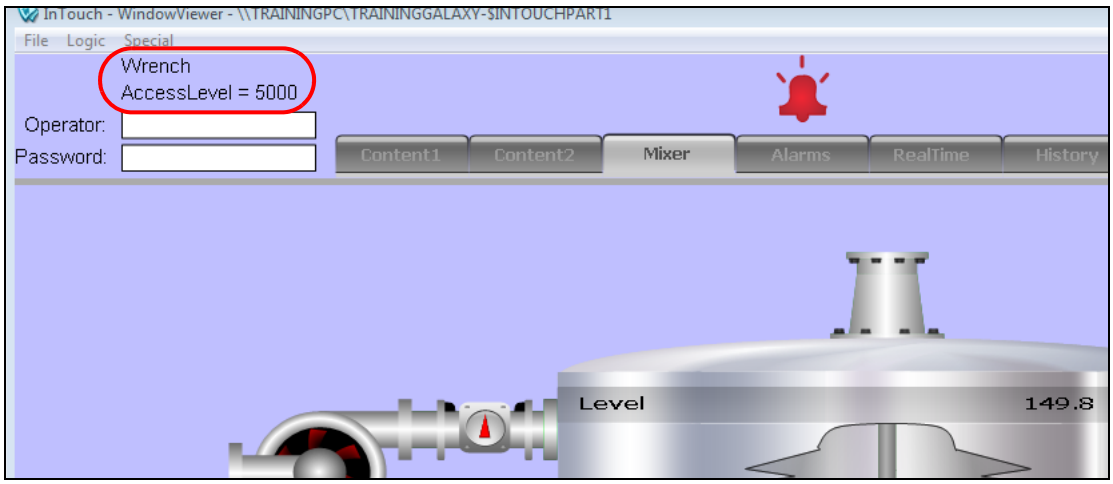
68. Click the top **OK** button.

Test the Application Security

69. Switch to **Runtime**.
70. Verify that while no one is logged in, the navigation tabs are disabled.
71. Click the **Operator** field, enter **Wrench**, then press **Enter**.

    The field appears empty.
72. Click the **Password** field, enter **iom**, then press **Enter**.

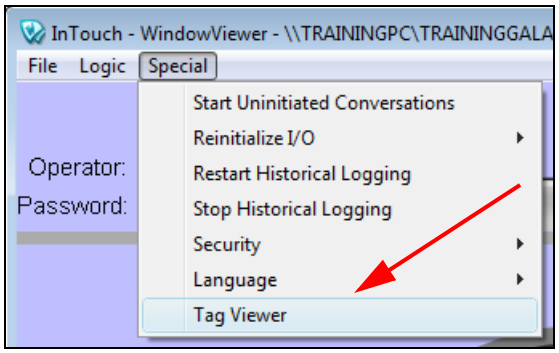
    Wrench and **AccessLevel = 5000** appears above the entry fields.



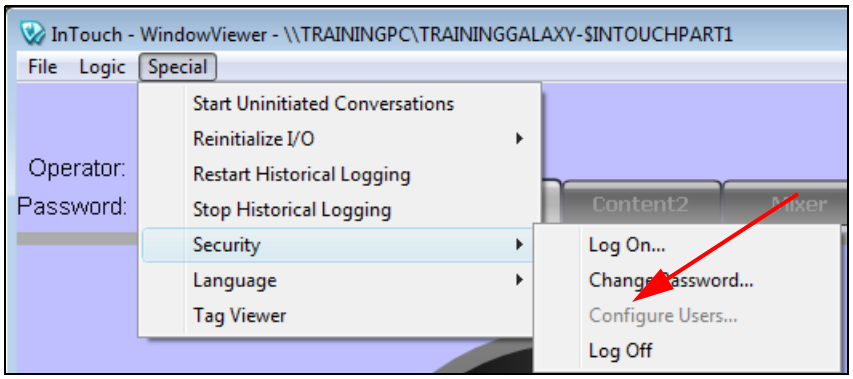
73. Verify the tabs work.

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

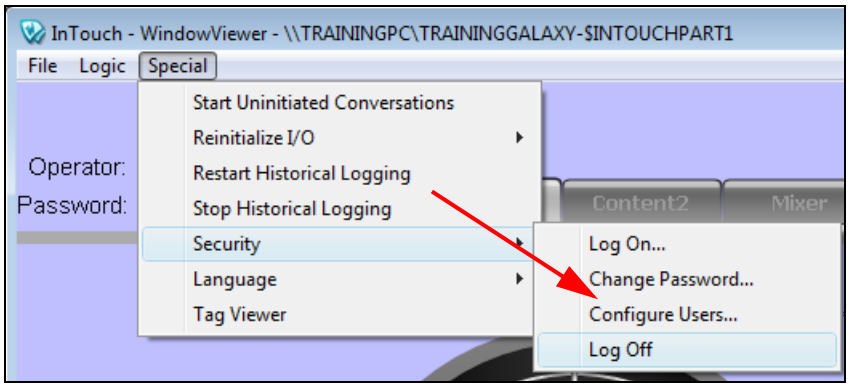
    Because **Wrench** does not have sufficient access rights, **Tag Viewer** does not open.



75. Verify that the **Special | Security | Configure Users** command is disabled.



76. Log in as **Administrator** with the password **iom**.
77. Verify that the menu comment **Special | Security | Configure User** is enabled.



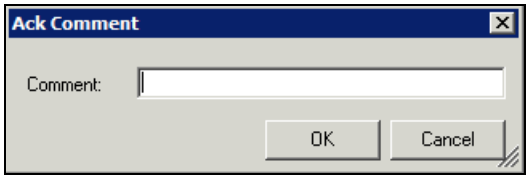
78. In the **Operator** field, enter **Clipboard**, then press **Enter**.
79. In the **Password** field, enter **iom**, then press **Enter**.
80. Click **Special | Tag Viewer**.

    The **Tag Viewer** appears.

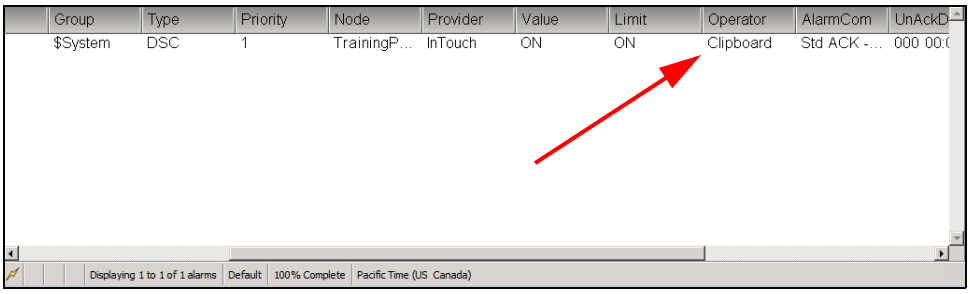
81. Close **Tag Viewer**.
82. Verify that the **Special | Security | Configure Users** command is still disabled.

Acknowledge an Alarm

83. Click the **Alarms** tab.
84. Click the **Inhibit** button, then trigger a **Discrete1Tag** alarm.
85. Click the **DiscreteTag1** alarm record in the top **Alarm Client**.
86. Right-click and select **Ack Selected**.
87. Enter an appropriate acknowledgment comment and click **OK**.



88. Scroll to the right and notice under the **Operator** column that **Clipboard** has been recorded as the operator who acknowledged the alarm.



Lab 16 – Managing Automatic Logout

Introduction

In this lab, you configure WindowViewer to automatically warn and then log out a user after a period of inactivity.

Objectives

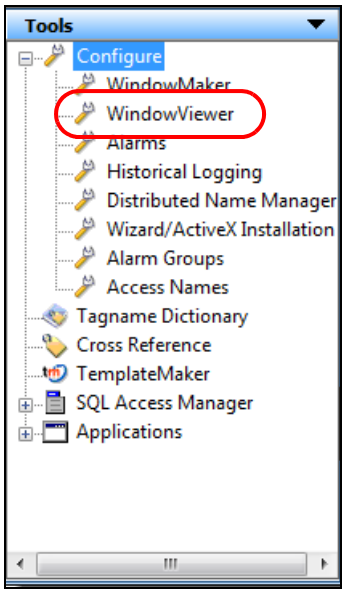
Upon completion of this lab, you will be able to:

- Create an operator inactivity warning indicator
- Use logic operators in animation

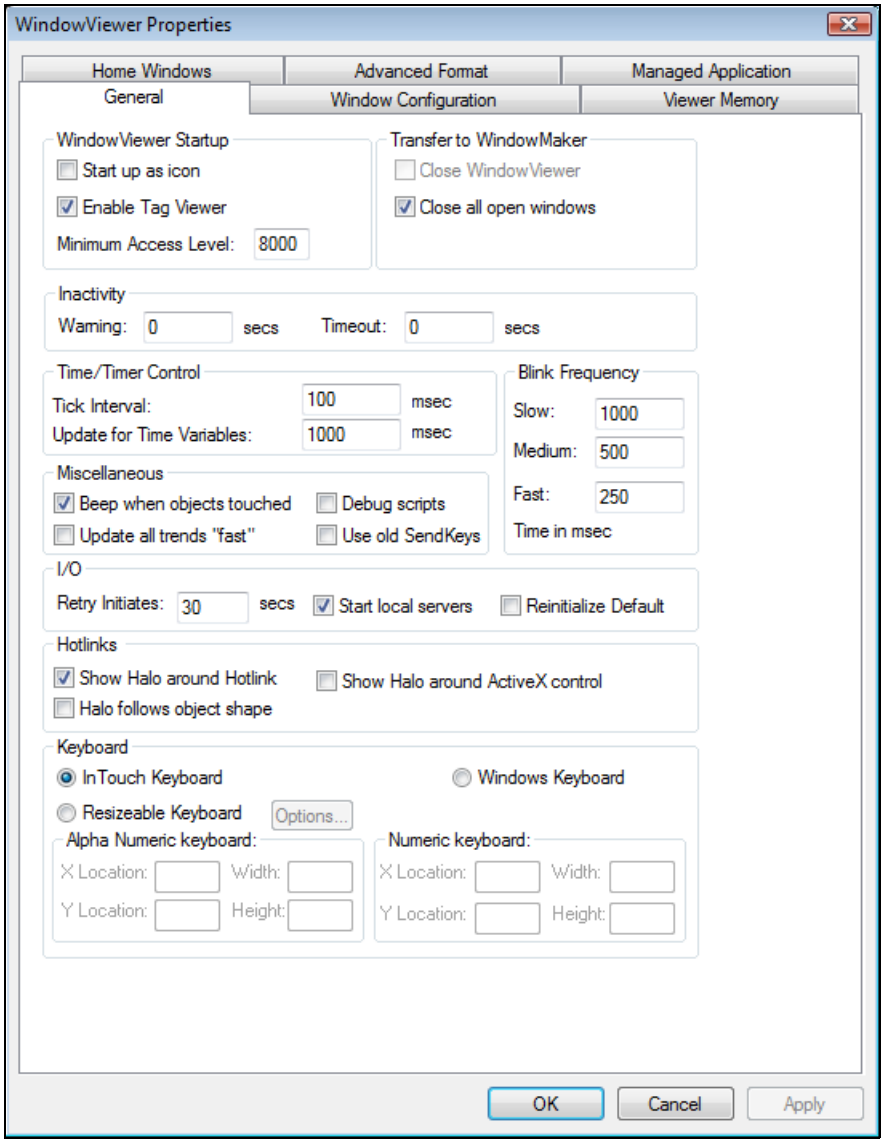
Configure WindowViewer Warning Options

First, you configure WindowViewer security options.

1. Close WindowViewer.
2. In the WindowMaker **Tools** pane, expand **Configure** and double-click **WindowViewer**.

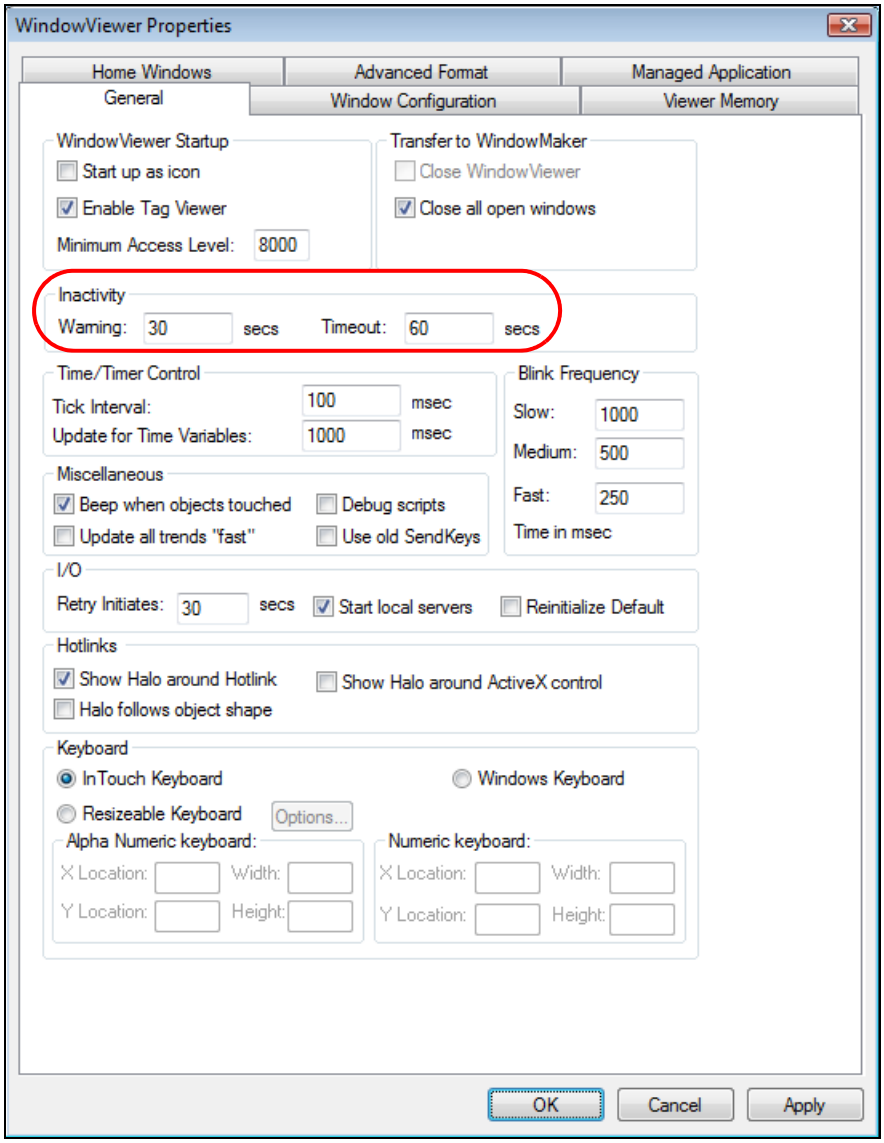


The **WindowViewer Properties** dialog box appears.



3. In the **Inactivity** area, enter the following:

   Warning: 30
   Timeout: 60

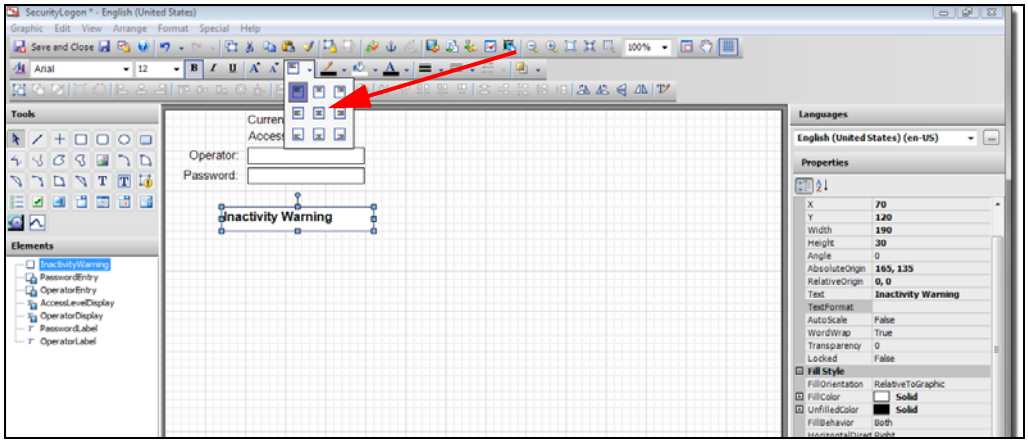


4. Click **OK**.

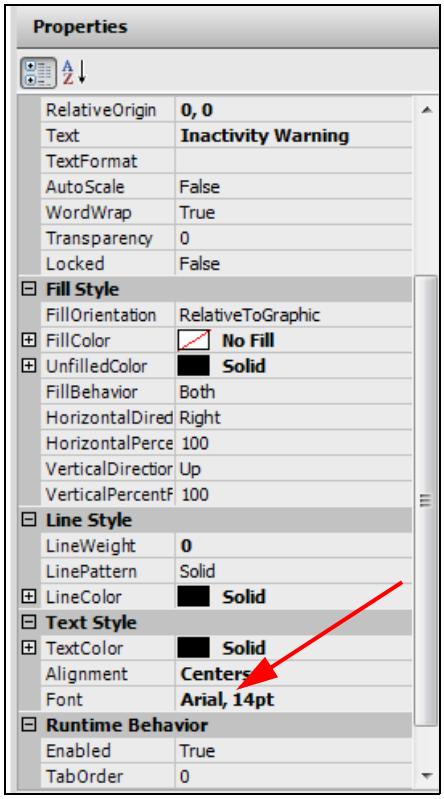
Create a LogOut Warning

Next, you create an inactivity warning to appear after a period of user inactivity.

5. In ArchestrA IDE, open the **SecurityLogon** symbol.
6. Create a text box and replace the text with **Inactivity Warning**.
7. Name the text box **InactivityWarning**.
8. On the toolbar, click **Bold**.
9. On the toolbar, click **Align Text** and select **Centers**.

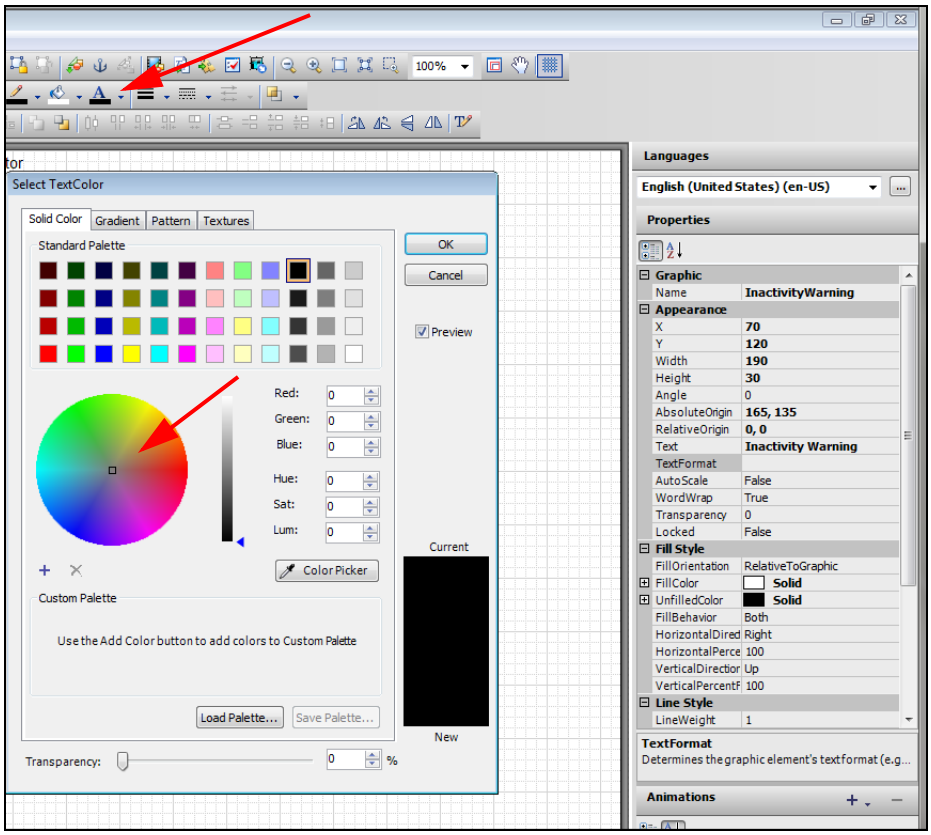


10. In **Properties** pane, change the font size to **14pt**.

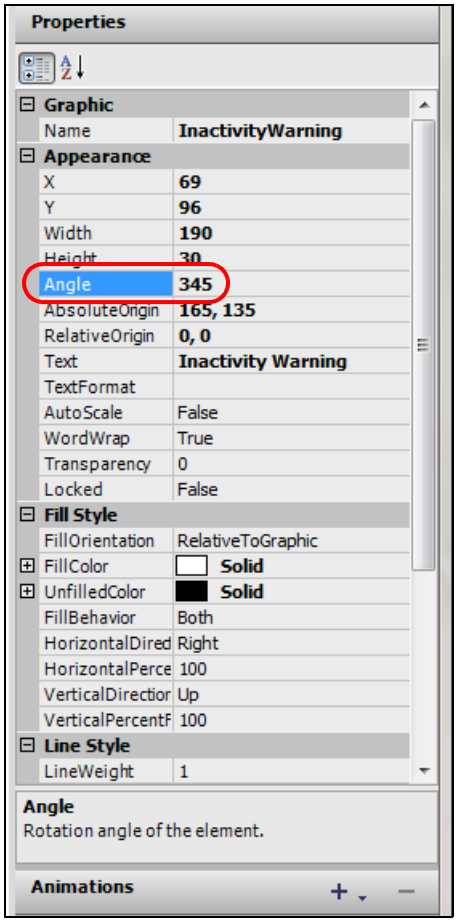


11. On the toolbar, click the **Text Color** arrow.
12. Change the **Text Color** to orange.

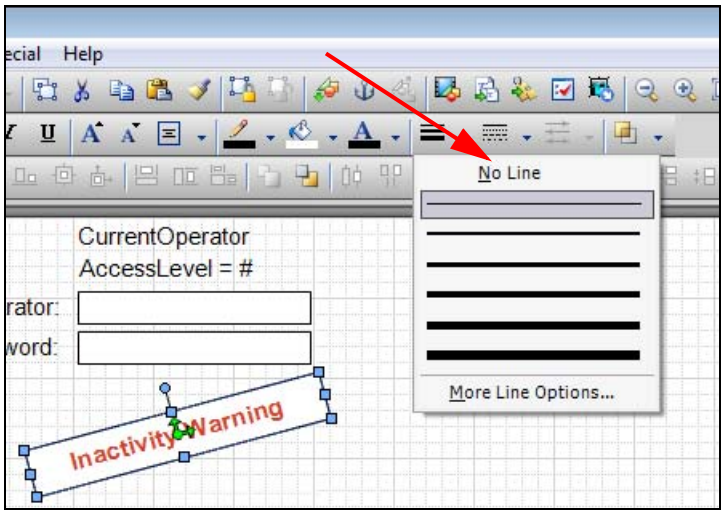
    You can use the **Text Color** dialog box to select a color from **More Solid Colors\Solid Color**.



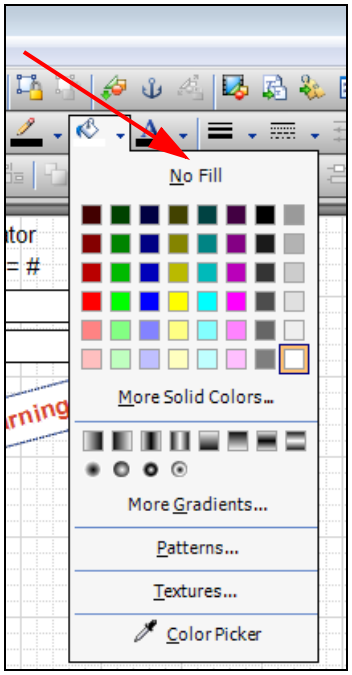
13. In **Properties** pane, **Appearance** properties, change **Angle** to **345**.



14. On the toolbar, click **Line Weight** and select **No Line**.

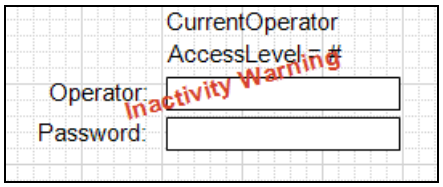


15. On the toolbar, click **Fill Color** and select **No Fill**.



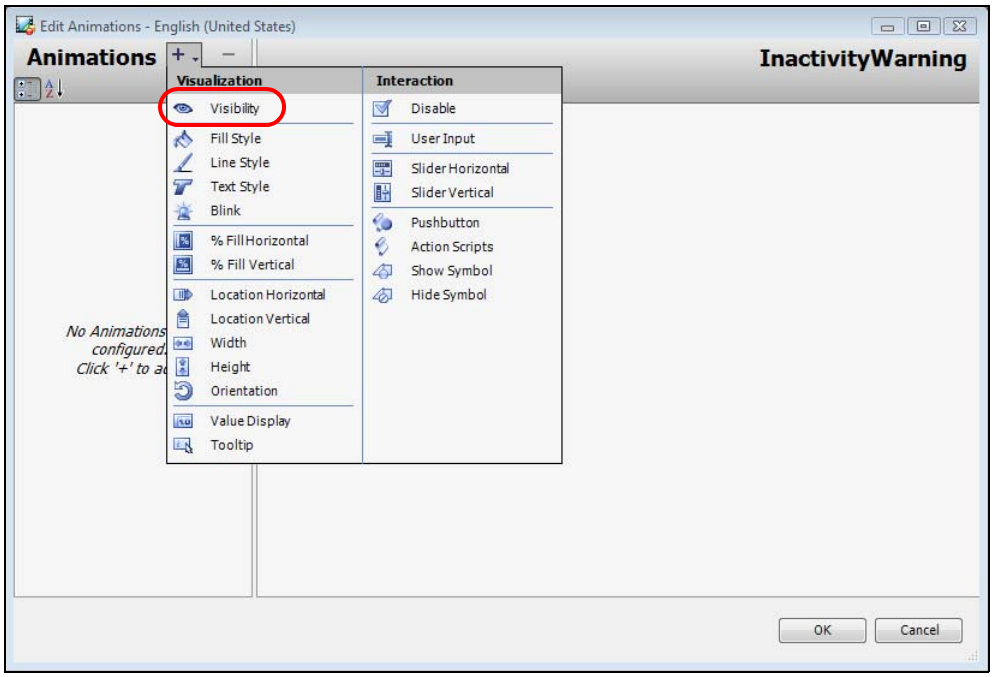
16. Move **InactivityWarning** over the other security elements
17. Size the text box to fit diagonally from corner to corner across the **SecurityLogon** as shown.

Note: Make sure the text box does not extend beyond the boundary of the **SecurityLogon** area.



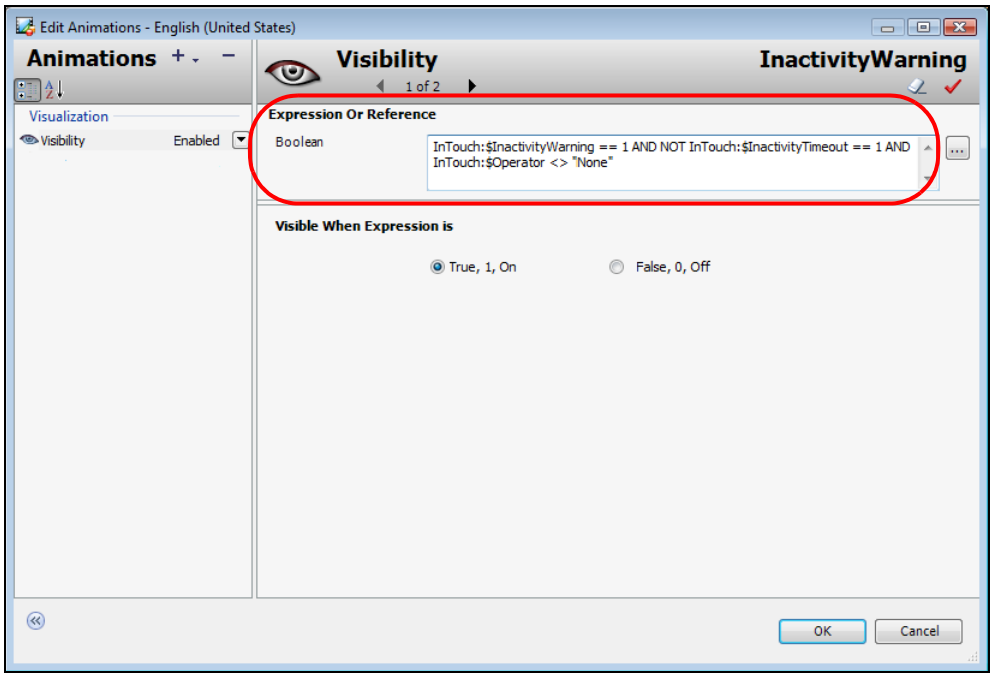
Now, you animate the security warning.

18. In **Elements** pane, right-click **InactivityWarning** and select **Edit Animations**.
19. Click **+** and select **Visibility**.



20. From **InTouch Tag Browser**, select tags to build the following expression and values:

    InTouch:$InactivityWarning == 1 AND NOT InTouch:$InactivityTimeout == 1 AND InTouch:$Operator <> “None”



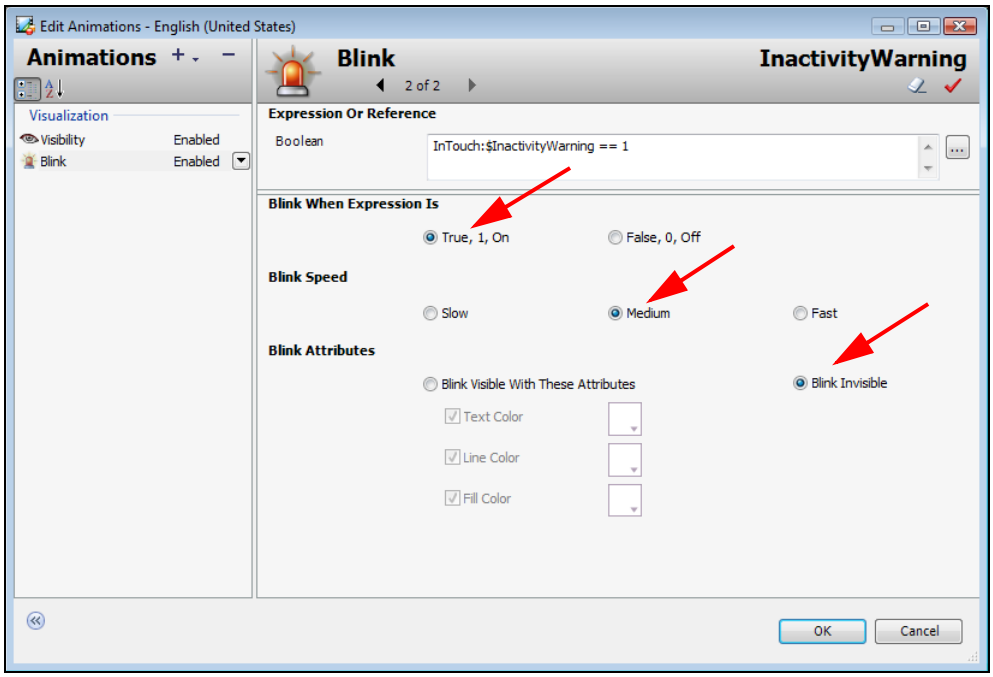
Keep the remaining defaults.

21. Click **+** and select **Blink**.
22. Enter the following expression:

    InTouch:$InactivityWarning == 1

23. Configure as follows:

Blink When Expression is:	True, 1. On
Blink Speed:	Medium
Blink Attributes:	Blink Invisible



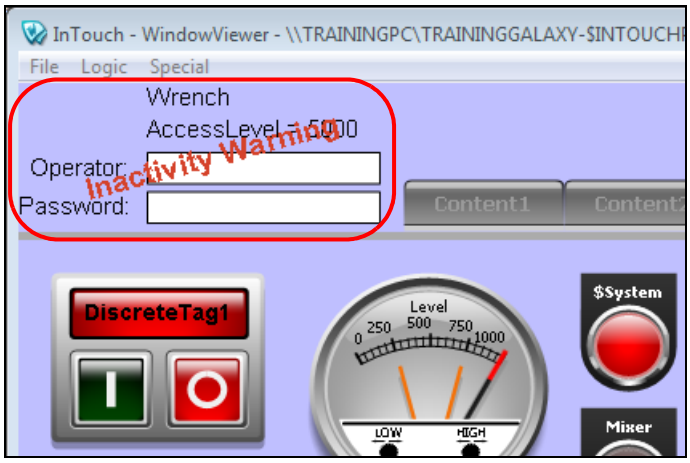
24. Keep the remaining defaults and click **OK**.
25. **Save and Close** the symbol.
26. Enter an appropriate check in comment.

Test Auto Logout

27. In WindowMaker, open the **Menu** window and click the refresh button.
28. Verify that your **SecurityLogon** symbol fits appropriately and that **InactivityWarning** does not extend beyond the other fields.
29. Open the **Alarms** window.
30. Click **Runtime**.
31. Log on as **Wrench** with the password **iom**.
32. Do not click anything for 30 seconds.

Note: Moving your mouse does not interrupt inactivity countdown.

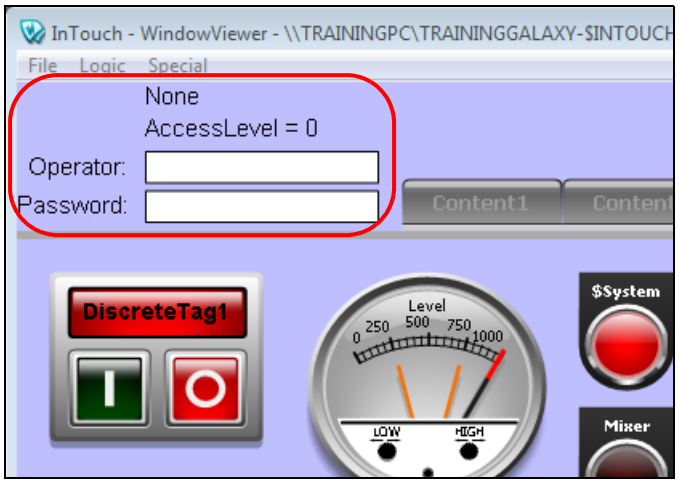
Inactivity Warning appears.



33. Do not click anything for another 30 seconds.

    Notice **Operator** changes to **None**.

    Operator is now logged out of the system.



34. Return to **Development**.

Disable Inactivity Warning Configuration

Finally, you disable the Inactivity Warning configuration so it will not impact your work in the remaining labs.

35. Close WindowViewer.
36. In WindowMaker, **Tools** pane, expand **Configure** and double-click **WindowViewer**.
37. In the **Inactivity** area, enter the following:

    Warning: 0
    Timeout: 0

38. Click **OK**.



invensys
Learning Services

Module 7 – Enhancing Your Application with QuickScript

Section 1 – Introduction to QuickScript 7-3
Section 2 – Enhancing Security with QuickScript 7-11
Lab 17 – Creating a Log Off Button 7-23
Lab 18 – Synchronizing the Navigation Tabs with Current Window 7-29
Lab 19 – Restricting User Access to Windows 7-37
Lab 20 – Defining Special User Access 7-53
Section 3 – Enhancing History with QuickScript 7-59
Lab 21 – Creating a Live Mode for the Historical Trend 7-67
Lab 22 – Changing Historical Pens at Runtime 7-75
Section 4 – Enhancing Alarms with ArchestrA Symbol Scripting 7-93
Section 5 – Enhancing Symbols with ArchestrA Symbol Scripting 7-95
Section 6 – Troubleshooting QuickScript 7-99
Lab 23 – Enhancing Alarm Client with Scripting 7-103
Lab 24 – Managing Limited Screen Real Estate 7-117

Module Objectives

- Introduce QuickScript
- Define QuickScript types
- Use the scripting editor to create QuickScripts
- Create customized and automated system functions using QuickScripts
- Introduce ArchestrA symbol scripting

Section 1 – Introduction to QuickScript

This section explains and defines the various InTouch QuickScript types, and features and functions that are unique to a particular QuickScript type. It also describes the features and functions of the QuickScript Editor.

QuickScript

QuickScript is the name of the script language, and functionality used in the InTouch environment. The actual script language is very similar to PLC languages like IEC611-31 and Structured Text.

InTouch QuickScript capabilities allow you to execute commands and logical operations based on specified criteria being met. All InTouch QuickScripts are event driven, such as a data change, a condition, a mouse click, or a timer. You can add conditional statements, loops, and local variables to the basic scripts to create complex effects in your application.

QuickScript types include the following:

- **Application –** Scripts linked to the entire application
- **Window –** Scripts linked to a specific window
- **Key –** Scripts linked to a specific key or key combination on the keyboard
- **Action –** Scripts associated with an object linked to a T**ouch Link - Touch Pushbutton - Action** animation link
- **Data Change –** Scripts linked to a tagname and/or tagname.field only
- **Condition –** Scripts linked to a discrete tagname or expression
- **ActiveX Event –** Scripts that execute ActiveX control events in runtime
- **QuickFunctions –** Scripts you create that can be called from other InTouch QuickScripts using **CALL FunctionName();** or inside animation link expressions. QuickFunctions can be either synchronous or asynchronous, while all other script types are synchronous only

Specific functions already configured to the proper syntax and format, are available from the script dialog box and can be pasted with a single mouse click. Additional information about scripting concepts and about creating and editing scripts is available in the application **Help** and in the *InTouch HMI Scripting and Logic Guide*, available within your local installation of InTouch.

You can configure most script types to execute based on multiple conditions using script triggers. For example, you can define a script to run when WindowViewer is started, and then periodically at fixed intervals while WindowViewer is running. Each script type has one or more triggers.

The order in which scripts are processed is based on a complex set of variables as the InTouch WindowViewer application's QuickScript engine is not scan based, therefore, the order of execution cannot be guaranteed. When using QuickScript, avoid building script dependencies based on any perceived sequence, the sequence can change as well based on criteria beyond the control of application designers.

For all InTouch **While** scripts (repeating at specified interval), the script will not execute the first time until the period has elapsed. Additionally, InTouch is not based on a scan engine with guaranteed interval, so the only guarantee is that the script will NOT execute faster than the interval specified. Finally, WindowViewer will make every attempt possible to run scripts as fast as the specified time, however, no script can run faster than the **Tick Interval** setting specified in WindowViewer properties. For normally behaved Windows applications, this interval should never be set faster than 20 ms (50x per second).

Global Scripts

The scope of execution (where the script lives) determines when or even if data from those scripts tags or functions is available throughout the application. Global scope script types are limited to Application Scripts, Data Change Scripts, Condition Scripts, Key Scripts, and QuickFunctions. Global scope means no matter where an operator has navigated within the application, these scripts continue executing in the background as needed and make their results continuously available.

Application Scripts

Application Scripts are linked to the entire InTouch application. These scripts execute one time when the application is started, shut down, or is running.

There are three types of Application Scripts:

- **On Startup:** Executes once when the application is initially started up. Executes before any window opens or any runtime initialization occurs. You cannot refer to ActiveX methods, properties, or events in an On Startup script. I/O communications are initialized after the application On Startup script executes. I/O tags or remote tagname references cannot be referred to in an On Startup script.
- **While Running:** Executes repeatedly at the specified frequency while the application is running. When a While Running script is selected, the Every Msec field becomes active. The field contains the number of milliseconds that must elapse before the script executes. As long as the condition or event for the While Running script is met, the script will repeatedly execute no sooner than the specified frequency, however, the script does not become backlogged if the system is busy and additional time elapses. If the functionality of a While script needs to be executed immediately, create an identical On Startup script.
- **On Shutdown:** Executes once when the application is closed. I/O tags and remote tagname references will not update in an On Shutdown script. An On Shutdown script cannot be used to startup other applications.

Condition Scripts

Condition Scripts execute when a logical condition occurs. Condition Scripts are linked to a discrete tag or expression that equates to TRUE or FALSE. This type of script is the most widely used script type. Discrete expressions containing analog tagnames can also be used. The condition can contain 1024 characters that equate to TRUE or FALSE.

Since Condition Scripts are executed based on a condition being met, you must specify the condition (a discrete tag or Boolean expression) in the **Condition** field. The expression must equate to a 1 (TRUE) or 0 (FALSE) result. For example, $Hour == 23 AND $Minute == 59, this expression remains true for 1 minute out of every day, just prior to midnight.

There are four types of scripts that can be applied to a condition:

- **On False:** Executes once when the condition transitions to false
- **While False:** Executes repeatedly while the condition is false. Frequency of execution is based upon the value of the **Every Msec** field
- **On True:** Executes once when the condition transitions to true
- **While True:** Executes repeatedly while the condition is true. Frequency of execution is based upon the value of the **Every Msec** field

Both **While True** and **While False** scripts begin executing after the specified number of milliseconds has elapsed. To cause immediate execution, create duplicate **On True** and/or **On False** scripts. All four script types can be applied to the same condition.

Data Change Scripts

Data Change Scripts execute each time their specified tagname or tagname.field in the Tagname[.field] field changes. A tagname must change by an amount greater than the deadband specified in its tagname dictionary definition. The default for the tagname deadband is 0, this means any change no matter how small will cause a Data Change Script to trigger and execute. This highlights an important concept, a Data Change Script tied to an I/O real for example a temperature, may cause the Data Change Script to execute hundreds or thousands of times a second, even for unintended noise coming from the I/O source.

Data Change Scripts are best used for monitoring events or less frequently changing data. For example, a Data Change Script tied to an I/O Discrete that monitors the start cycle on a blast freezer, where the script might log some critical setup or startup information about the process to log file. Or a Data Change Script monitors an operator initiated emergency shutdown switch and the script clears the operational state of several key process tags or changes some animation or data presentation behaviors based on the change to the switch.

Key Scripts

Key Scripts execute on a specific key or key combinations on the keyboard. They can be used to create global keys for the application, such as returning to a main menu window, logging off the operator, and so on.

There are three types of Key Scripts:

- **On Key Down:** Executes once when the key is initially pressed
- **While Down:** Executes repeatedly at the specified frequency while the key is held down. When While Down is selected, the Every Msec field is active. The field contains the number of milliseconds that must elapse before the script executes
- To execute the script immediately, create an identical **On Key Down** script. As long as the condition or event for the While Down script is met, the script will repeatedly execute at the specified frequency
- **On Key Up:** Executes once when the key is released

The key equivalents used in the local active windows for Touch Pushbutton - Action Scripts override any global Key Scripts with the same key equivalents.

Local Scripts

The scope of execution for all other script types including Window Scripts, Touch Action Scripts (also called Action Scripts), ActiveX Scripts, and ArchestrA Symbol Scripts are local execution only. Local Scripts only run from an active window. These scripts run with the operator takes a specific action on the window. If the window which hosts these scripts is hidden (closed), the scripts stop executing. Local functionality includes Action, Window, ArchestrA, and ActiveX scripts.

Note: ActiveX Scripts are covered in the InTouch 2012 R2 – Part 2 course.

Action Scripts

Action Scripts are similar to Key Scripts, except they are associated with an object linked to a **Touch Link – Touch Pushbutton – Action**. They are executed when the operator clicks or presses the object or button assigned to the link.

Note: You can access the script editor through the **Animation Link Selection** dialog box.

There are thirteen types of Action Scripts. But they can be broken down into 4 simple categories based on a simple pointing device such as a 3-button Mouse. For each of the 3 buttons, there are 4 possible states: click, double-click, held-down, and released. This represents 12 of the Touch Action Script options. The last script type executes when the mouse merely hovers over the object without clicking.

The following scripts execute once when the key or button is initially pushed down, as follows:

- **On Left Click/Key Down**
- **On Right Click**
- **On Center Click**

The following Action Scripts execute repeatedly at the specified frequency while the key or button is held down. When this is selected, the Every Msec field is active. The field contains the number of milliseconds that must elapse before the script executes:

- **While Left/Key Down**
- **While Right Down**
- **While Center Down**

The following Action Scripts execute once when the key or button is released:

- **On Left/Key Up**
- **On Right Up**
- **On Center Up**

The following Action Scripts execute when the mouse button is double-clicked. The key equivalent feature disables when these are selected, since there is no keyboard equivalent for a mouse button double-click action:

- **On Left Double-Click**
- **On Right Double-Click**
- **On Center Double-Click**

The following Action Scripts execute one time when the cursor rests over an object. The script does not trigger until the **After X msec** field value is exceeded, this helps avoid unintentional triggering as the mouse is being moved around the screen, only deliberately coming to rest over the object will trigger the script. After X msec is the length of time the cursor must be over the

object before the script executes. The key equivalent feature is disabled when this condition type is selected, since there is no keyboard equivalent for moving the mouse cursor over an object.

- **On Mouse Over**

Window Scripts

Window Scripts function within a specific window.

There are three types of Window Scripts:

- **On Show:** Executes once when the window is initially opened
- **While Showing:** Executes repeatedly at the specified frequency while the window is showing. When While Showing is selected, the Every Msec field becomes active. The field contains the number of milliseconds that must elapse before the script executes.
  To execute the script immediately, create an identical On Show script. As long as the condition or event for the While Showing script is met, the script will repeatedly execute at the specified frequency
- **On Hide:** Executes once when the window is closed. A window that is covered or hidden by other windows or whose x, y location is off the display is still active and running in memory, even if the operator cannot see its contents

ActiveX Scripts

Many ActiveX controls have events for handling things like click, double-click, mouse down, and key press. An InTouch ActiveX Event script tied to a specific event in an ActiveX control placed on an InTouch Window will execute when that event occurs. You may associate one ActiveX Event script to one event in an ActiveX control. If an ActiveX Script is written in a generic way so as to be reusable, it may be used and called from more than one ActiveX control event.

Note: Implementation of ActiveX functionality is detailed in the InTouch 2012 R2 – Part 2 course.

ArchestrA Symbol Scripts

ArchestrA scripts can run code to animate ArchestrA symbols or its elements, or perform many other tasks. The script language syntax and format is very close to that in InTouch scripts. The main difference is that ArchestrA script supports .Net script functionality, such as the ability to instantiate objects. For example, here is a script that will allow you to use the Microsoft Voice Synthesizer to convert text to speech. For a script like this to work, the client .dll must be imported through the ArchestrA IDE (see the comments section of the script and the ArchestrA documentation for details).

```
{ Prior to using this script, import System.Speech.dll from
C:\Program Files\Reference Assemblies\Microsoft\Framework\v3.0
wordstospeak is a string custom property and assigned a text string }

DIM speaker AS System.Speech.Synthesis.SpeechSynthesizer;
speaker = NEW System.Speech.Synthesis.SpeechSynthesizer;
speaker.SpeakAsync( wordstospeak );
```

ArchestrA scripts always run locally, that is an ArchestrA symbol script is only enabled when the window containing the ArchestrA symbol is active in memory (open).

Some examples of what you might do with ArchestrA Symbol scripts:

- Configure the predefined scripts of a symbol
- Add named scripts to a symbol
- Edit existing named or predefined scripts in a symbol
- Rename named scripts in a symbol
- Remove named scripts from a symbol
- Substitute references in named or predefined scripts
- Use Element Methods in named or predefined scripts

Configuring Symbols Using Scripting

You can configure your symbol with scripts that can be executed in runtime.

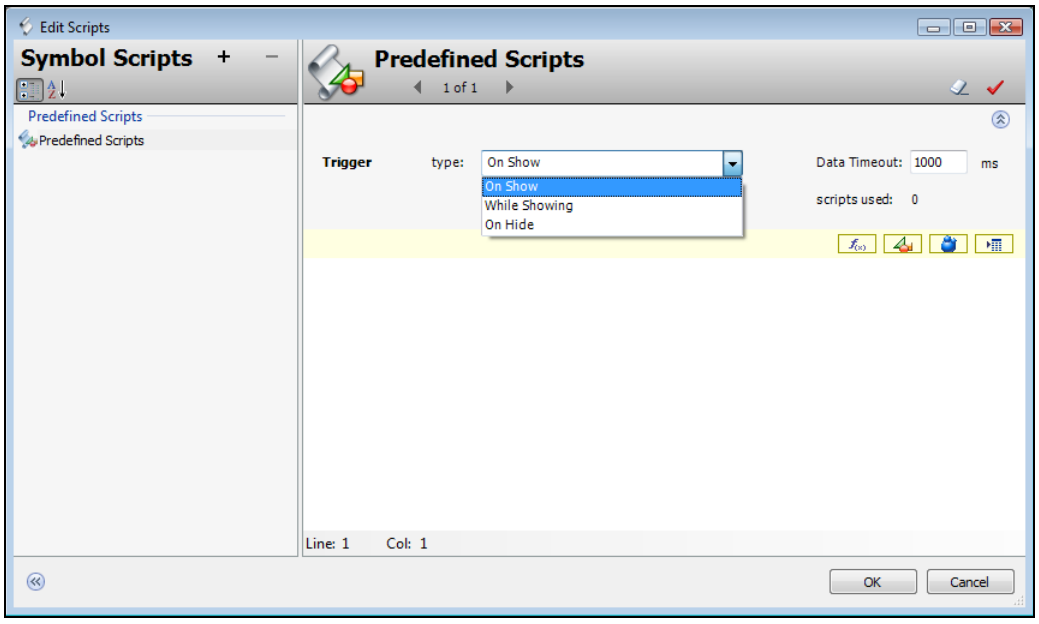
There are two types of Symbol Scripts:

- **Predefined**
- **Named**

Predefined Scripts

Predefined Scripts act much the same as Window Scripts. This is primarily because ArchestrA symbols have a Windows-like paradigm. Symbols can actually be shown in their own small window using a show symbol animation. Any ArchestrA symbol embedded in an InTouch window can process its built-in or Predefined Scripts in the following specific ways:

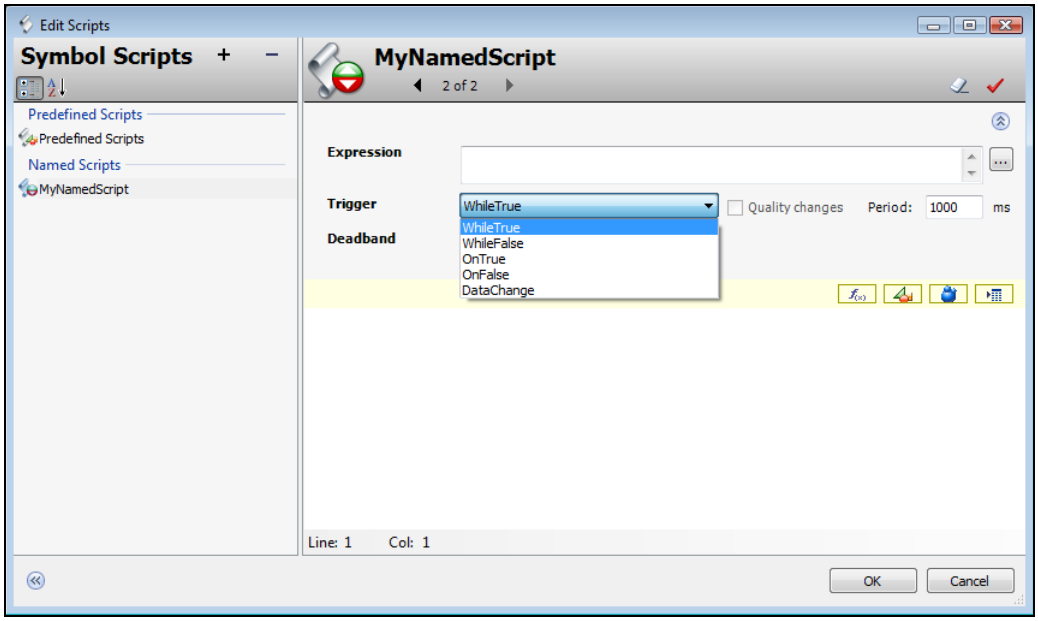
- A script that is run once when the symbol opens (when the Window opens).
- A script that is run periodically as long as the symbol is open (while the Window is open).
- A script that is run once when the symbol closes (when the Window closes).
- Any combination of the above. Any or all 3 types may have a script associated with it.



Adding Named Scripts to a Symbol

Every Named Script can contain only one trigger type, a separate Named Script must be created for additional trigger types of the same expression. Named Scripts let you execute any number of scripts in runtime that are triggered by values, custom properties, element properties, element methods, or expressions as follows:

- The value or expression is true: **WhileTrue**
- The value or expression is false: **WhileFalse**
- Transitioning from false to true: **OnTrue**
- Transitioning from true to false: **OnFalse**
- Changing in value or quality: **DataChange**



Note: Any **Named Script** that is triggered by the **DataChange** trigger type, is executed the first time the symbol is shown.

Modifying Scripts

You can edit Predefined and Named Scripts that are already associated with a symbol. You can also rename Named Scripts in a symbol. When you rename the Named Script, the functionality of the associated script does not change. You can also remove Predefined or Named Scripts.

Substituting Attribute References

You can substitute attribute references in scripts in the same way as you would with attribute references in elements. For example, you can change element properties using scripts. This enables you to configure additional runtime animation to your elements on top of the design-time animation of those elements.

When you write your script for the symbol or for one of its elements, you can use the **Galaxy Browser** to show and select either a:

- Property of an element
- Custom property of the symbol

I/O Connectivity

Managed InTouch applications have the advantage of being able to connect to remote data sources across a network. In InTouch WindowMaker, you use the **Tagname Dictionary** to define variables that hold values. In **ArchestrA Symbol Editor,** data sources can be custom properties and direct links to InTouch tags using **InTouch:Tagname** syntax. After a symbol is embedded in a window in WindowMaker, its custom properties may be linked to InTouch tagnames. If a tagname is misspelled or does not exist, you will be prompted to define the tag.

Section 2 – Enhancing Security with QuickScript

This section describes the InTouch Script Editor functionality. It also describes the Animation Links dialog box and Action Script methods you can use to secure your application.

Overview

Enhancing Security may begin with basic configuration options available on **Special** | **Configure** options, including the ability to disable the menu bar or individual menu items, the **Tag Viewer**, or even make WindowViewer impossible to close.

WindowViewer can be configured to automatically logout users based on inactivity. The **Inactivity** timeout option is accessible on **Special | Configure | WindowViewer**. Additionally, Home windows may be defined on **Special | Configure | Home Windows** so that the initial window or windows, which appear when WindowViewer is first started do not give access to any sensitive areas.

But the most significant power in customizing and enhancing security is done through InTouch scripts to build sophisticated and robust applications. You may use InTouch scripts to create a completely regulation compliant and secure application.

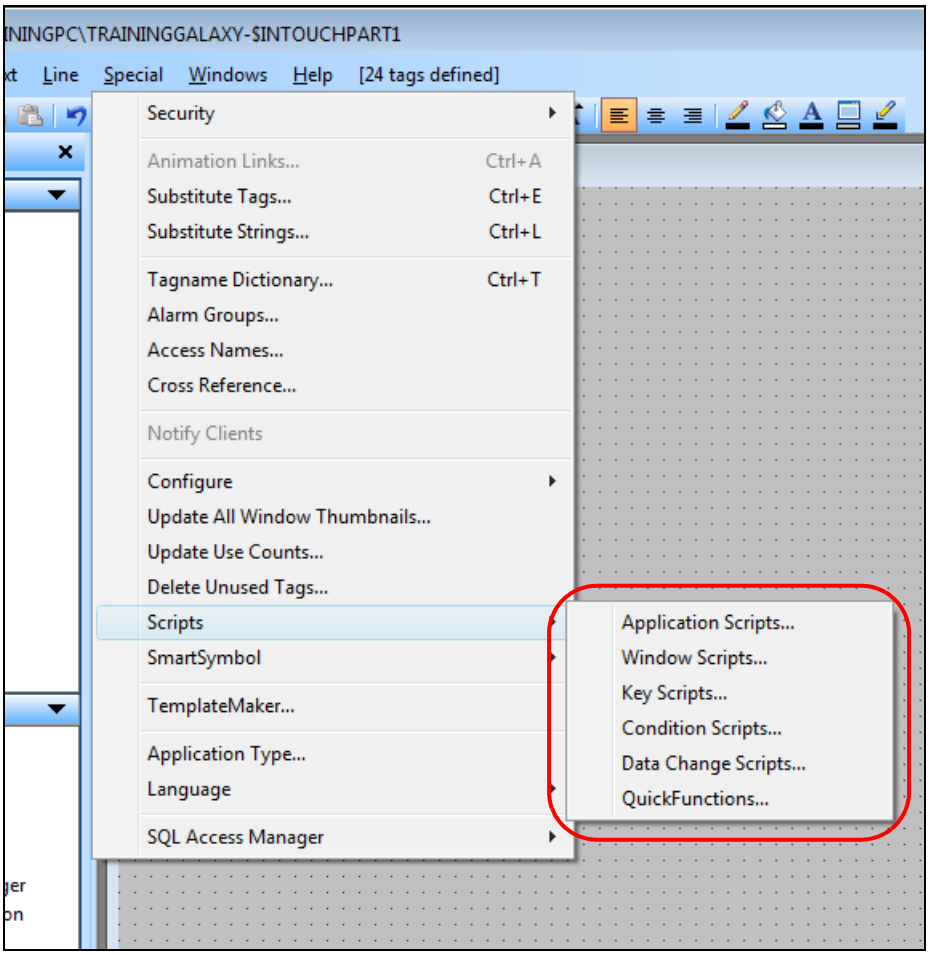
In terms of security, scripts can generally be run in two different ways:

- **Event-based** scripts run once when an event occurs allowing you to take immediate, unassisted action. For example, an event-based script can run after a condition is met or the value of a key process I/O point changes. Using I/O tagnames tied to external security systems, such as door locks and proximity alarms, or other external security stimuli, InTouch scripts can be used then to handle a wide array of standard operating procedures, work flow, or processes insuring your people, facility, products, services, and investments remain secure
- **Time-based** scripts run periodically to continue to keep systems running within operational norms or where predictive measures can be taken to avert failure or downtime

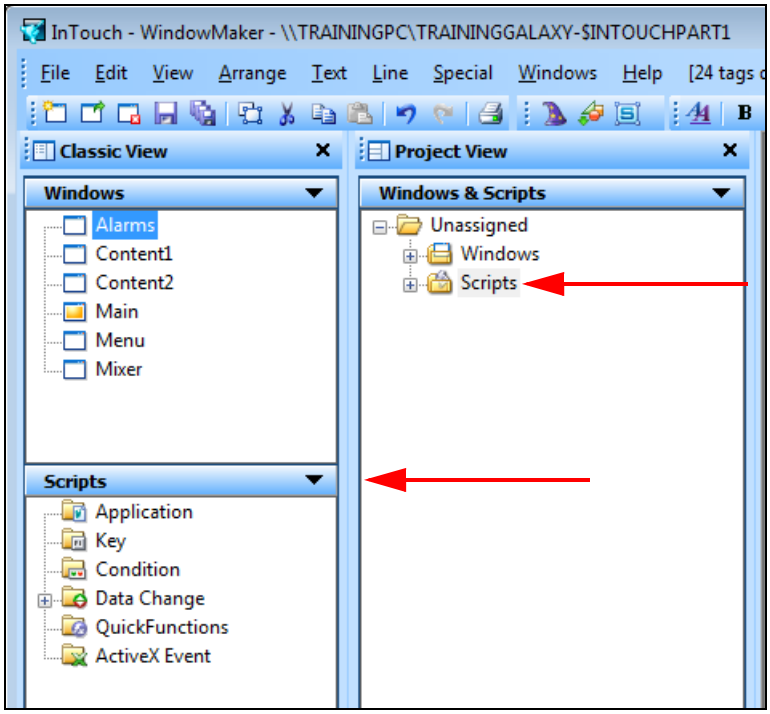
InTouch provides a number of System tags and script functions that allow you to configure security for the individual application. By combining scripts with system tags related to security, you can add a number of powerful features to your application, monitoring special conditions, and dealing with critical events for example environmental or safety issues. You can also use Touch scripts for sophisticated operator input into the system, and apply levels of security to various functions.

Working with the Script Editor

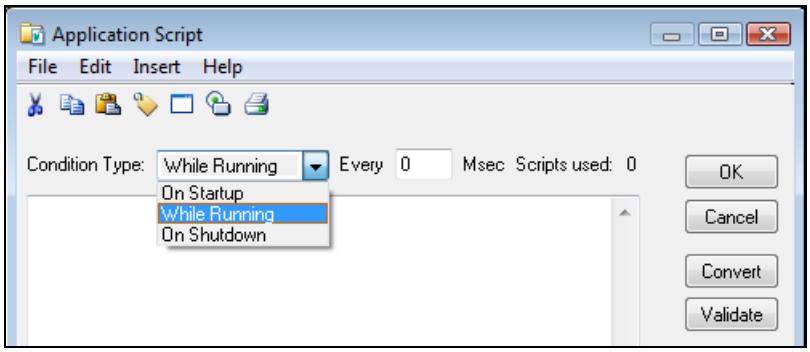
Create a new script using the **Script Editor** based on script type. The option is available from the main toolbar **Special** menu, **Scripts** options. You select a script option based on the functionality type.



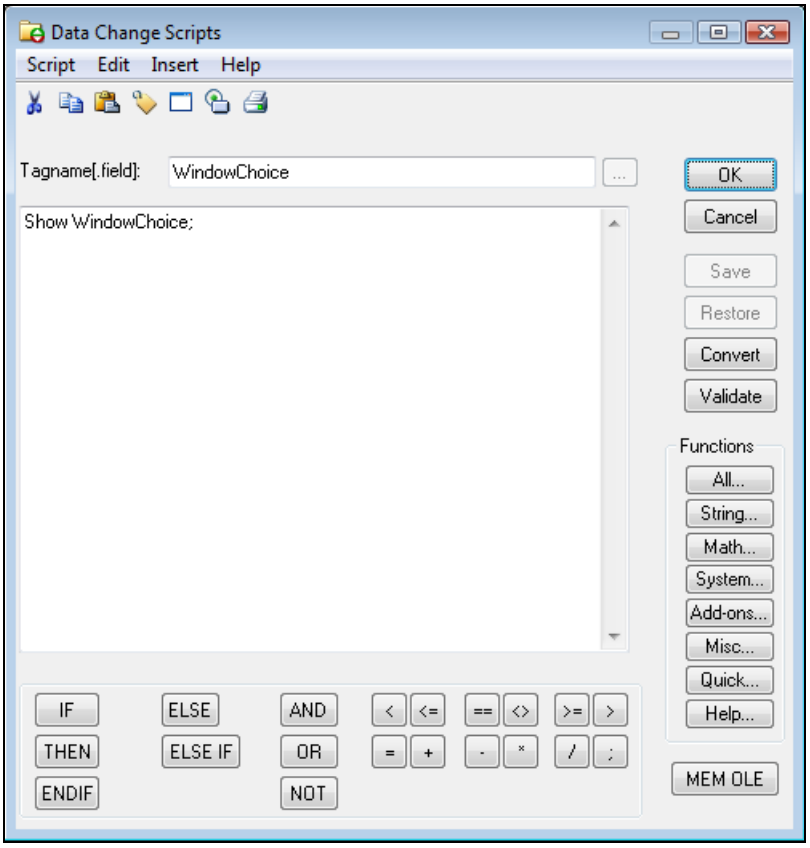
Existing QuickScripts are listed in the **Scripts** pane of **Classic View**. In **Project View**, they are listed in the **Windows & Scripts** pane.



Header options such as **Condition Type** vary based on the type of script you are creating. The following shows the Condition Type options for an Action Script.



The graphic below shows a **Data Change Scripts** dialog box.

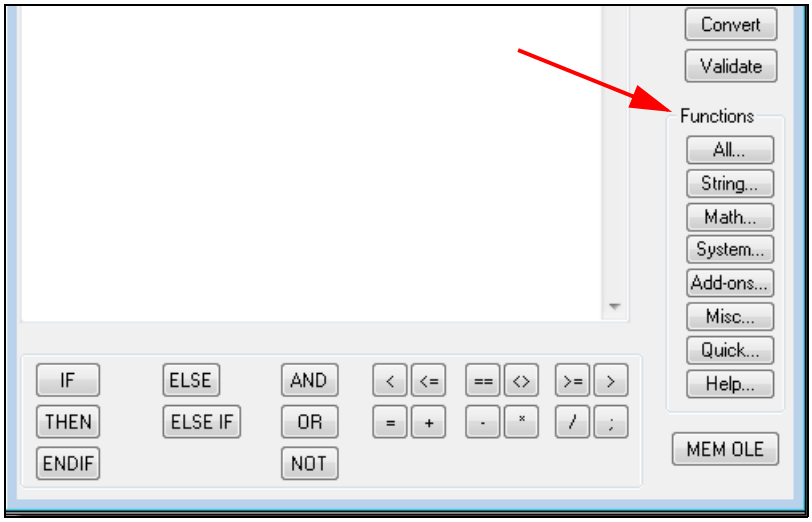


Built-In Script Functions

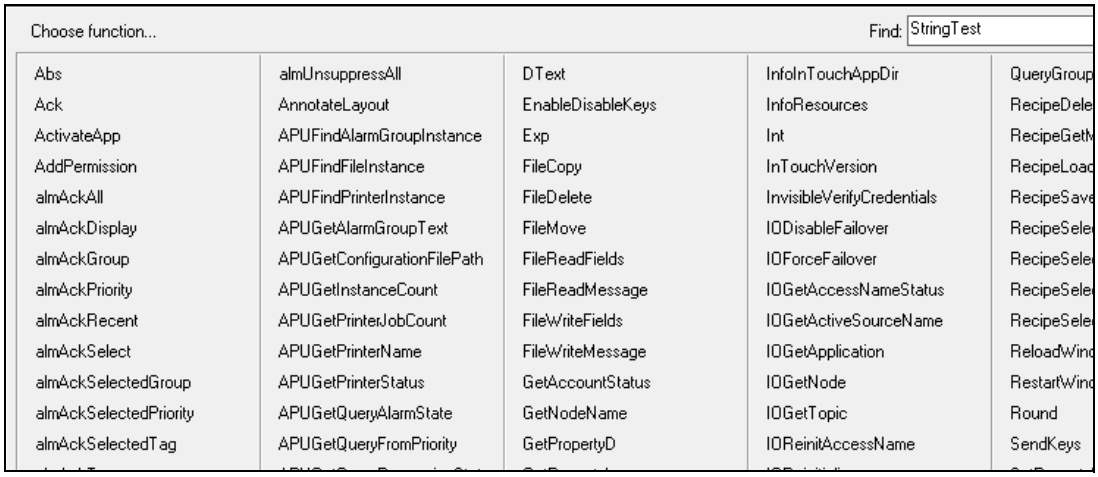
InTouch provides numerous predefined functions that can be linked to objects or pushbuttons, or used in scripts to perform a multitude of tasks.

Using InTouch scripts, for example allows an application developer to hide or show windows such as popping up an alert window when a security measure has been disabled, securing navigation, and locking down specific application features, to name a few. These functions are accessible through the **Insert | Functions** main menu or by clicking the various buttons in the **Functions** section on the right side of the **Script Editor**.

Once you select a function in its respective dialog box, the function and its required arguments are automatically pasted into your script at the cursor location. After the function is pasted into your script, highlight the argument to modify and enter the new value.



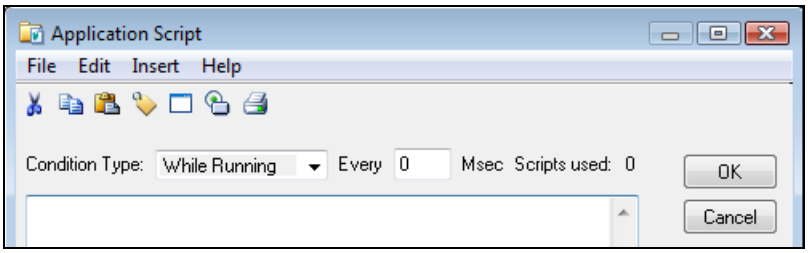
The **All** function on the script dialog box displays a list of QuickScript options you can use in your script such as **Show**, **FileCopy**, and **HideSelf**. You can also select the **Help** button for quick access to the online help system for a specific function.



Equivalency, mathematical operator, and script options are available at the bottom of the dialog box. You can use these options to quickly insert the information displayed on the button into your script. This can save time and help reduce errors.

Inserting Tags into Scripts

You can double-click in a blank area of the script area to open the **Tagname Dictionary**. You can also access the tag browser from the **Insert Tag** option on the toolbar. This opens the **Select Tag** dialog box.



You can set a Dot Field for InTouch tags. Dot Fields can access, monitor, and modify tag properties. When a tag or a tag and Dot Field is selected, it is automatically entered into the script area. The tagnames defined in a local or remote tag source can be displayed and selected in this mode. The **.Value** field is the hidden default in all cases where no Dot Field is selected. The .Value field does not have to be explicitly indicated.

Checking Scripting Syntax

Validate checks your script syntax for accuracy. You can validate at any time. Validation is automatically performed when you click **OK** or **Save**. If the system encounters errors when validating a script, a corresponding error message box will open. If no message appears, the script is accepted.

IF THEN ENDIF Statement

Branching or decision making in InTouch Scripting is handled via conditions (also called condition expressions) using a series of keywords, namely: IF, THEN, and ENDIF. Optionally, an ELSE keyword may be used between THEN and ENDIF. These keywords are used to conditionally execute a choice between two outcomes (TRUE or FALSE). The choice is made based on the evaluation of a condition or expression placed directly between the keyword IF and the keyword THEN in the sequence.

For example, IF MyTagname > 8.9 THEN

The portion of this example between the keywords, MyTagname > 8.9 is evaluated by checking if the part on the left side of the greater than symbol (>) is a larger quantity than the part on the right side. Let's say MyTagname contained a value of 10, the conditional expression would be TRUE. This results in the next part of the script to execute.

These keywords must be spelled as shown in separate words; however, they are not case sensitive. Any of the following are the same to the InTouch Script Editor: IF, iF, If, and if. However, because these are reserved keywords used for scripting, an InTouch application designer is restricted from creating an InTouch tagname using any of these keywords. Each IF must have a matching THEN and ENDIF; (followed by a semicolon). Any statements placed after the THEN also require a semicolon at the end of each statement.

The simplest form of this conditional branching functionality is illustrated by the following example:

```
IF UnauthorizedAccess == 1 THEN
   Show “Security Alert”;
ENDIF;
```

In the above script, a tagname called **UnauthorizedAccess** is evaluated and compared to a numeric value of 1, and if it is found to be TRUE, the script will proceed to the statement **Show “Security Alert”;** which will open a window called Security Alert.

If the tagname **UnauthorizedAccess** contained any value other than 1, even a value such as 1.0001 (which is not equal to 1), then the system skips over the statement and ends after the ENDIF; portion of the script. IF THEN ENDIF; is the most basic form of a decision making tool.

The following comparison operators are also valid for conditional expressions:

< Less than

> Greater than

<= Less than or Equal to

>= Greater than or Equal to

== Equivalency ("is equivalent to")

<> Not Equal to

Below are some examples of complex scripts using comparison operators.

IF THEN statement:

```
IF EmergencyShutdownSwitch_Reactor04 <> 0 THEN
   SafetyShutdownProcedure_Reactor04 = 1;
ENDIF;
```

In the above example, if EmergencyShutdownSwitch_Reactor04 was an I/O Discrete tag which is set to 1 whenever an operator or system in the field triggers a shutdown of Reactor 04, this script will take action by setting a flag called SafetyShutdownProcedure_Reactor04 to a value of 1.

IF THEN statement with ELSE clause:

```
IF Temperature > (TemperatureSetpoint + TemperatureDeadband) THEN
   Cooling = 1;
ELSE
   Cooling = 0;
ENDIF;
```

In the above example, Cooling is shutdown when the Temperature is below the setpoint plus a deadband.

You can also define a script execution frequency for some script types.

In the **While *Running/Showing/Down* Every 0 Milliseconds** text boxes, enter the number of milliseconds that must elapse before the script executes. When creating an Application **While Running** script, Window **While Showing** scripts, Condition **While On True/On False** scripts, or Key and Touch Pushbutton Action **While Down** scripts, you must specify the frequency (in milliseconds) that they will be executed.

Animation Links

There are two basic types of animation links: **Display** and **Touch** links.

- **Display** links show information to operators. Examples of display links are changing colors, hanging fill levels, horizontal or vertical movements, and blinking objects
- **Touch** links allow operator input into the system. Examples of touch links are sliders or push buttons that respond to operator input

You can define multiple links for objects or symbols. By combining various links, you can create almost any screen animation effect.

Additional Script Functions

Additional script functions include Math, System, and WW I/O Functions. For more information about these functions, please see the *InTouch HMI Scripting and Logic Guide*.

Securing the Application

The following section explores securing the application in greater detail. As discussed in the previous module, the three major aspects of security are as follows:

- Securing the Operating System
- Auditing
- Securing the Application

Customizing the Runtime Environment

Like WindowMaker, WindowViewer includes many options to customize your runtime environment. For example, you can set the blinking speed for blinking objects, the system inactivity timeout and warning values, and the windows that are automatically opened when WindowViewer is started from its icon or its menu command.

Securing any InTouch application involves identifying ways the application can be accessed by users in the Windows environment, but also includes remote clients attempting to retrieve data from WindowViewer. InTouch allows you to define various options to increase the robustness and to secure your application by turning off undesired menu bar entries, certain Alt, Esc, and Win key keystrokes and key combinations, as well as operating system lockdown. You can use these options to prevent users from performing potentially dangerous actions. In combination, these features ensure that users do not have access to change the application environment, close the application, or escape from the application to start unauthorized programs.

Secure the operating system by limiting user access to applications outside of InTouch.

Operating system security is configured from the **WindowViewer Properties** dialog box, **Windows Configuration** tab. This is accessed on the **Special** menu by selecting **Configure | WindowViewer**.

Here you can turn off undesired menu bar entries, certain keystrokes, and other settings to ensure that users do not change the application’s environment, close the application, or escape from the application to other programs.

Use **Menus** settings to show or hide different menu options within WindowViewer, hide the menu bar, prevent the operator from switching to WindowMaker, and prevent the operator from starting and stopping QuickScripts during runtime.

Use **Title Bar** options to hide the application's title bar in runtime.

Use **Miscellaneous** options to disable certain keystrokes such as Ctrl, Alt, Esc, and Win. You can also set WindowViewer window as maximized at all times.

For more information please see the *System Platform* documentation.

Restricting User Input

InTouch includes the script function **EnableDisableKeys()** that enables/disables the Alt, Tab, or Win keys in runtime. Disabling these keys prohibits users from switching between windows applications or accessing the task bar. This function, combined with WindowViewer settings to Always Maximize and hide menu bar and Windows control options, will effectively secure the operating system.

Another common practice to hide/show the menu bar in View is to create a popup window with its X,Y coordinates set to 0,-38 with a height of 38 pixels and a width equal to the display resolution (e.g. 1280). This example uses the Microsoft Windows Server 2003 Operating System, with a screen resolution of 1280 x 1024. Values will differ depending on the Operating System and resolution used. Values will be -48 and +48 if XP look and feel is applied.

Auditing the Application

When a security model is applied to an InTouch application, the Alarm and Event sub-system provides an audit trail associating users to all alarms/events that occur during the time that they are logged onto the system.

Both the Distributed Alarm Object and the Alarm Display ActiveX control provide for displaying the logged on Operator name.

The Alarm Viewer Control contains links to OS/ArchestrA security tags.

Note: Alarm Viewer Control is discussed in the InTouch 2012 R2– Part 2 course.

Scripting and System Tags

InTouch provides a number of System tags and script functions that allow you to configure security for the individual application. By applying security to your application, you can control specific functions a user is allowed to perform by linking those functions to internal tagnames. Once a user is validated and logged on to the InTouch application, the **$Operator** and **$AccessLevel** system tags contain information about the user and their access level.

To log on to an InTouch application, users can select on the **Special** menu, **Security | Logon**, call a script function to display a logon dialog box, or log on via a custom logon window.

$Operator and $AccessLevel are tags used to secure InTouch applications. These are system tags used to control what a user has access to during runtime. For example, to disable a pushbutton, assign a **Disable** or **Visibility** animation link to it with an expression, such as $AccessLevel < 9000 or $Operator == xxx.

InTouch Authentication

InTouch authentication uses a binary password file that contains user identification, passwords, and access levels. In addition, you can use Operating System authentication, ArchestrA authentication, or no authentication at all.

In WindowMaker, access security commands are available from the **Special** menu, **Security** options. These commands are used to log on and off the application, change passwords and to configure the list of valid user names, passwords, and access levels.

You can link a **User Input - Discrete** button to the **$ConfigureUsers** tag to allow an authorized user with an access level of equal to or greater than 9000 to display the **Configure Users** dialog box. The dialog box is used to edit the security user name list.

When the user clicks the button, the value of the **$ConfigureUsers** tagname is set to 1 and the **Configure Users** dialog box appears. When the user closes the dialog box, the system resets the value to 0 (This is a system discrete tag intended for write-operation only).

If the **Special** menu does not display in WindowViewer, you can create a custom logon window to allow users to log on to the application.

You can also link a **User Input - Discrete** button to the **$ChangePassword** tagname to show the **Change Password** dialog box and allow the user to change his/her password. When the user clicks the button, the value of the **$ChangePassword** tagname is set to 1 and the **Change Password** dialog box appears.

When the user closes the dialog box, the system resets the value to 0 (This is a system discrete tagname intended for write operation only.)

The **None** and **Administrator** names are reserved. Only the password of the Administrator may be changed. Once you have configured user names for your application, you should change the Administrator name's password since it will more than likely become known to most users of the system. The Administrator default access level (9999) is the highest and allows access to everything including the **Configure Users** menu command.

Creating a Custom Security Logon Window

$OperatorDomainEntered, **$OperatorEntered**, and **$PasswordEntered** are the three tags to be used to create a custom logon feature. To do this, create a popup window and set user touch links (user input string) to these tags. If the provided credentials are validated, **$Operator**, **$AccessLevel**, **$OperatorName**, and **$OperatorDomain** will update based on the user input.

Note: The **$OperatorDomainEntered** tag is for OS Security only.

If the **Special** menu is not displayed in WindowViewer, a custom logon window can be created for the user to log on to the application.

Link the **$OperatorEntered**, **$PasswordEntered** and **$OperatorDomainEntered** system tagnames to user input objects or use them in a QuickScript to set the **User Name**, **Password**, and **Domain Name**. These are internal message tagnames that are intended for write operation only.

The **$OperatorDomainEntered** is required only if the security mode is operating system-based. Also, if the security mode is operating system-based and the **$OperatorDomainEntered** value is null, access is considered a local machine.

When a user logs on to the application, InTouch verifies the password and access level against the value specified for the internal security tag linked to a function. When the verification is successful, the user is granted access the appropriate protected functions.

Defining Security System Tags

Three internal security tags are used in buttons in animation link expressions or QuickScripts, and so on. They are **$Operator**, **$AccessLevel**, and **$OperatorName**. These tags control whether or not the logged-on user is allowed to perform specific functions.

You can also use the following additional security-related system tags:

- **$AccessLevel**
- **$ChangePassword**
- **$ConfigureUsers**
- **$InactivityTimeout**
- **$InactivityWarning**
- **$Operator**
- **$OperatorName**
- **$OperatorDomain**
- **$OperatorDomainEntered**
- **$OperatorEntered**
- **$PasswordEntered**
- **$VerifiedUserName**

For example, to make an object become visible based on the logged on users' access level, the following statement could be used in a Visibility animation link expression:

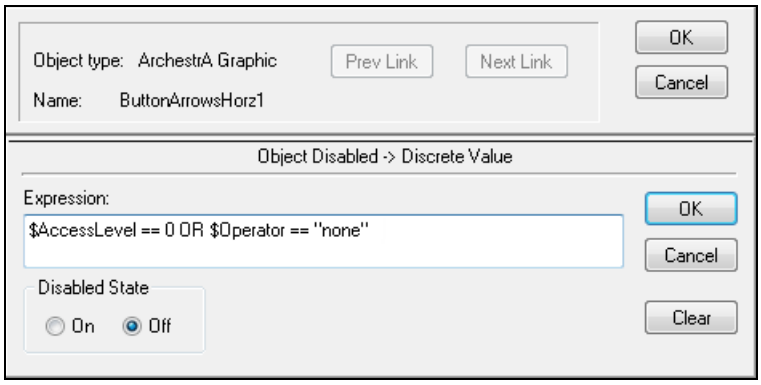
```
$AccessLevel >= 2000
```

Or, a QuickScript can be bounded by an IF statement:

```
IF $Operator == "DayShift" THEN
   Show "Control Panel Window";
   {and other lines that only execute for the DayShift Operator}
ENDIF;
```

You can also control an object's touch functionality based on the value of an internal security tagname by using the **Disable** animation link.

For example:



In this example, when no one is logged on, the object or button is secured from tampering.

Lab 17 – Creating a Log Off Button

Introduction

In this lab, you create a Log Off button to allow users to log off. You also define animation links to hide the Log Off button when no one is logged on to the system.

Objectives

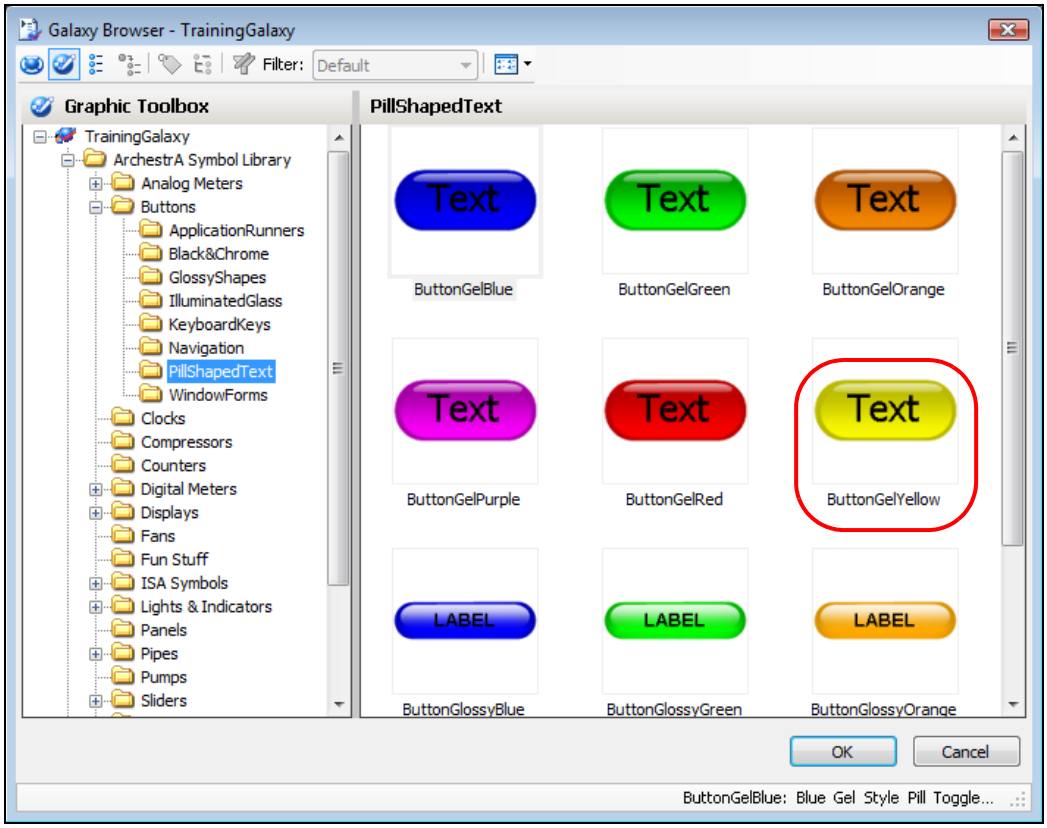
Upon completion of this lab, you will be able to:

- Create a Log Off button for users to log out manually
- Use an InTouch Action Script with embedded symbols

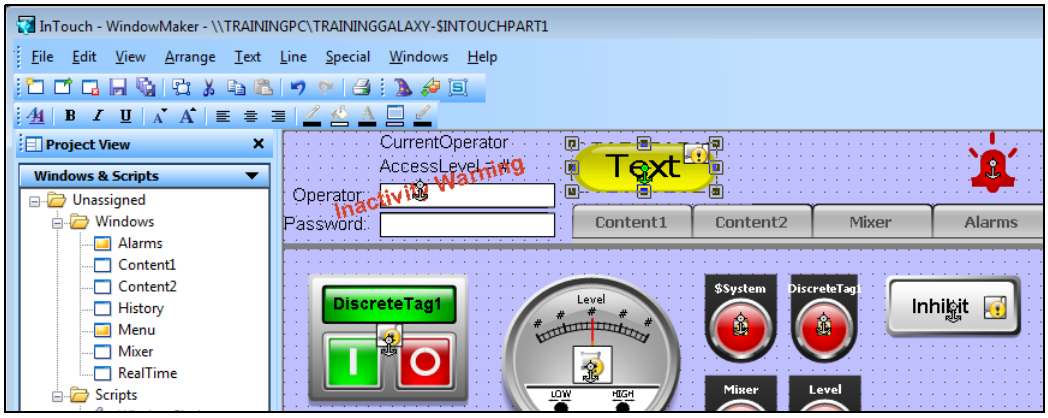
Create a Log Off Button

First, you add a button to allow users to log off the system manually.

1. In WindowMaker, from **ArchestrA Symbol Library**, expand **Buttons\PillShapedText**, and double-click **ButtonGelYellow**.



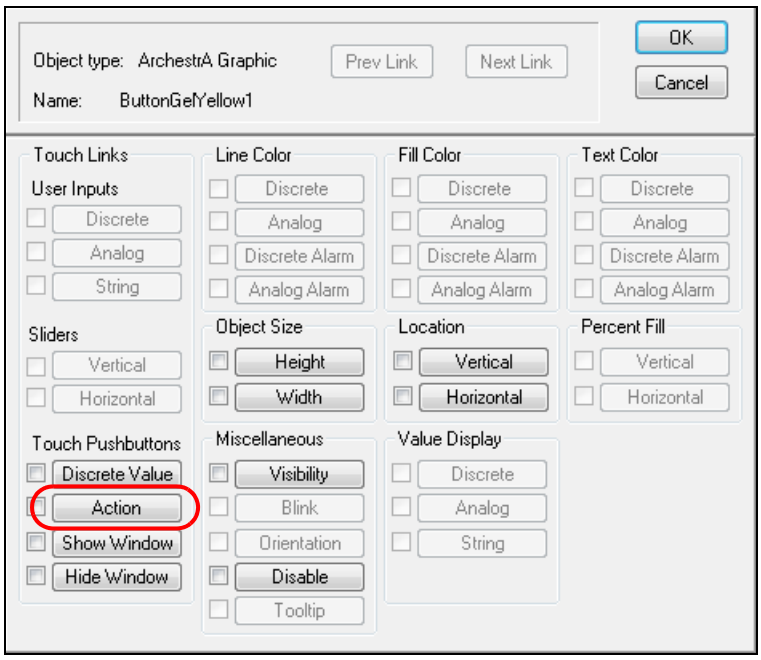
2. In the **Menu** window, place the button just above the navigation tabs, and resize if necessary.



3. Change the button label to **Log Off**.
4. Right-click the button and select **Animation Links**.

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

Note: You should click the button and not the check box. Manually checking the box, disables the animation.



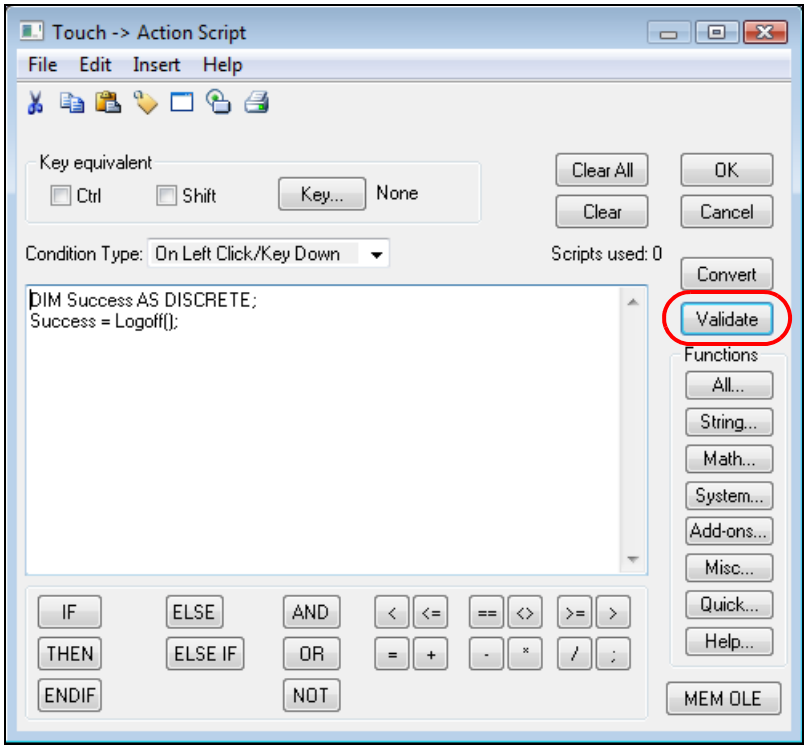
The **Action Script** dialog box appears.

Next, you dimension a tag to capture the results of the Logoff() script function.

6. Verify the default **Condition Type** is set to **On Left Click/Key Down**.
7. Enter the following script:

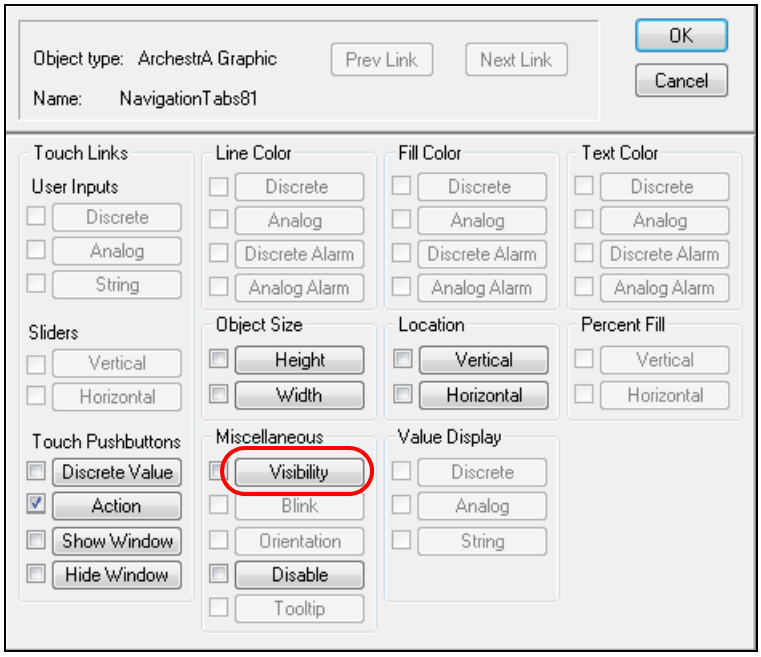
   DIM Success AS DISCRETE;

   Success = Logoff();
8. Click **Validate** to validate the script.

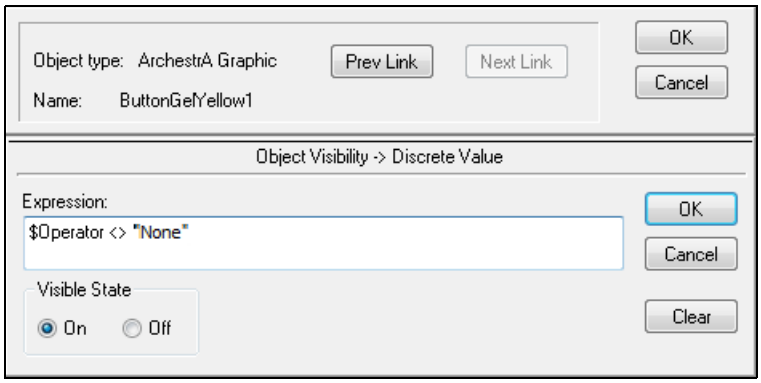


9. Click **OK**.

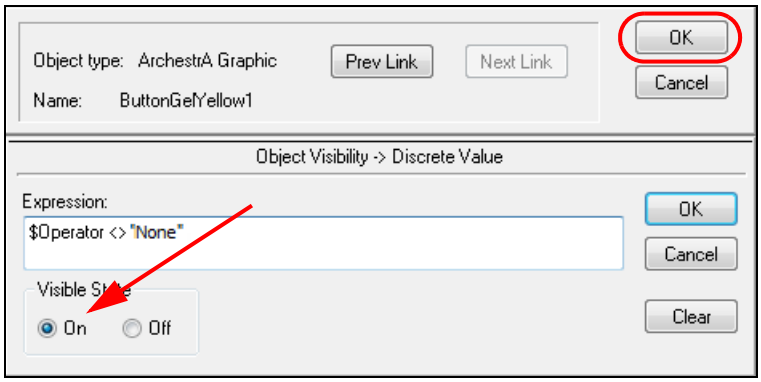
10. In the **Miscellaneous** area, click **Visibility**.



11. In the **Expression** field, enter **$Operator <> "None"**.



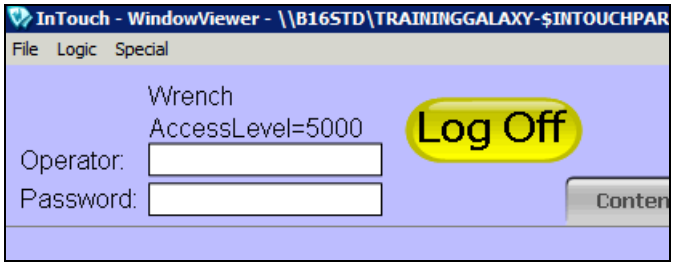
12. Verify the **Visible State** is **On**, then click the top **OK** button.



Test the Log Off Button

13. Switch to **Runtime**.
14. Sign in as **Wrench**, password **iom**.

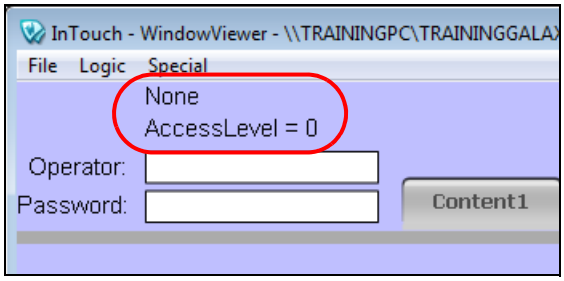
    The **Log Off** button is available



15. Click **Log Off**.

    The **Log On** status is **None**.

    Notice that the **Log Off** button disappears.



Lab 18 – Synchronizing the Navigation Tabs with Current Window

Introduction

In this lab, you synchronize the tab displayed in the menu window with the active window displayed at the bottom of the view. You modify an existing script to switch the highlighted tab in the NavigationTabs8 graphic.

Objectives

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

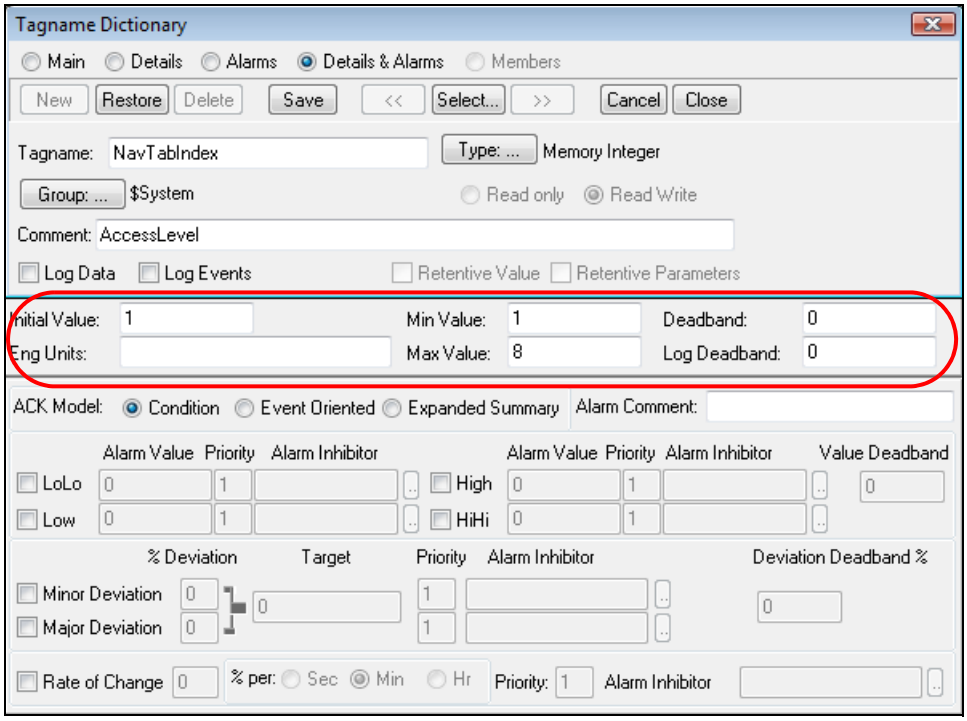
- Create an OnShow Window script

Define the NavTabIndex Memory Tag

First, you create a memory tag that will be used to synchronize the active tab on the navigation bar with the open window in the main panel.

1. Close WindowViewer.
2. In WindowMaker, open the **Tagname Dictionary**, then click **New**.
3. In the **Tagname** field, enter **NavTabIndex**
4. Click **Type**, then check **Memory Integer**.
5. Click **OK**.
6. Set the following values:

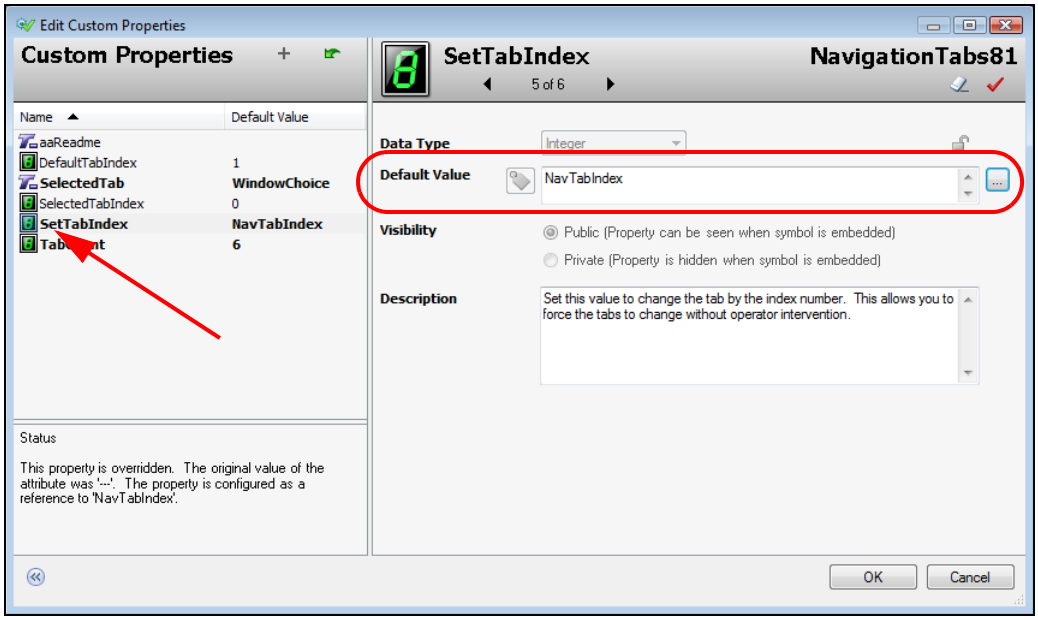
   Initial Value: 1
   Min Value: 1
   Max Value: 8



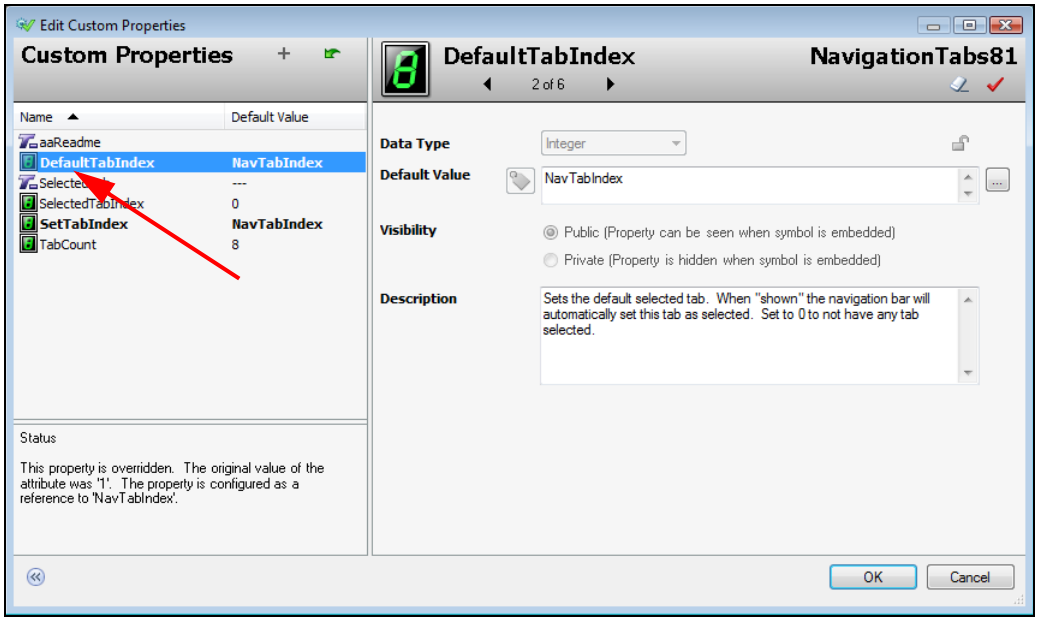
7. Click **Close**.

Modify Custom Properties

8. In the **Menu** window, double-click the **NavigationTabs8** to edit **Custom Properties**.
9. Click **SetTabIndex**, then link the **Default Value** to **NavTabIndex**.



10. Click **DefaultTabIndex**, then link the **Default Value** to **NavTabIndex**.

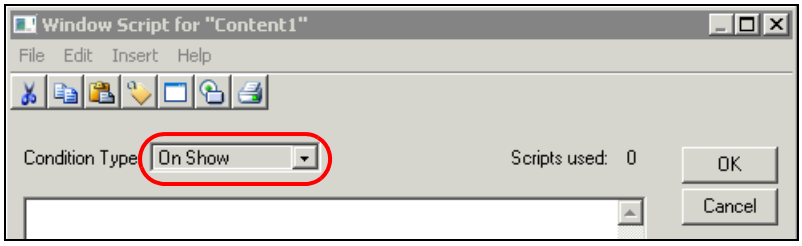


11. Click **OK**.

Define Window Scripts

Next, you define scripts for each window.

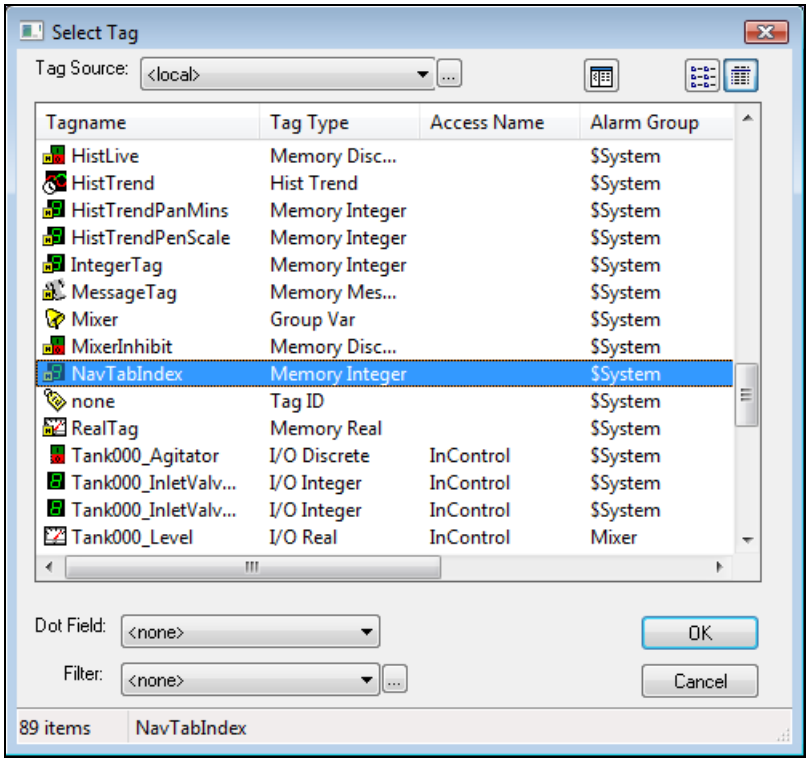
12. Right-click **Content1** and select **Window Scripts**.
13. In the **Condition Type** drop-down, select **On Show**.



14. On the top toolbar, click **Insert Tag** .

    The **Select Tag** dialog box opens.

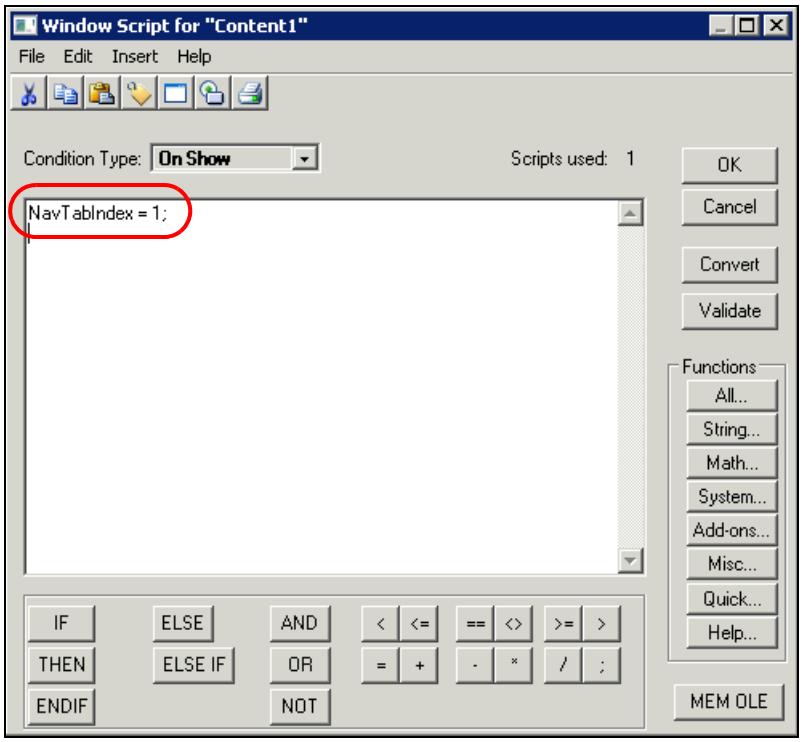
15. Double-click **NavTabIndex**.



16. At the end of the script line, enter **= 1;**

    The completed script is as follows:

    NavTabIndex = 1;

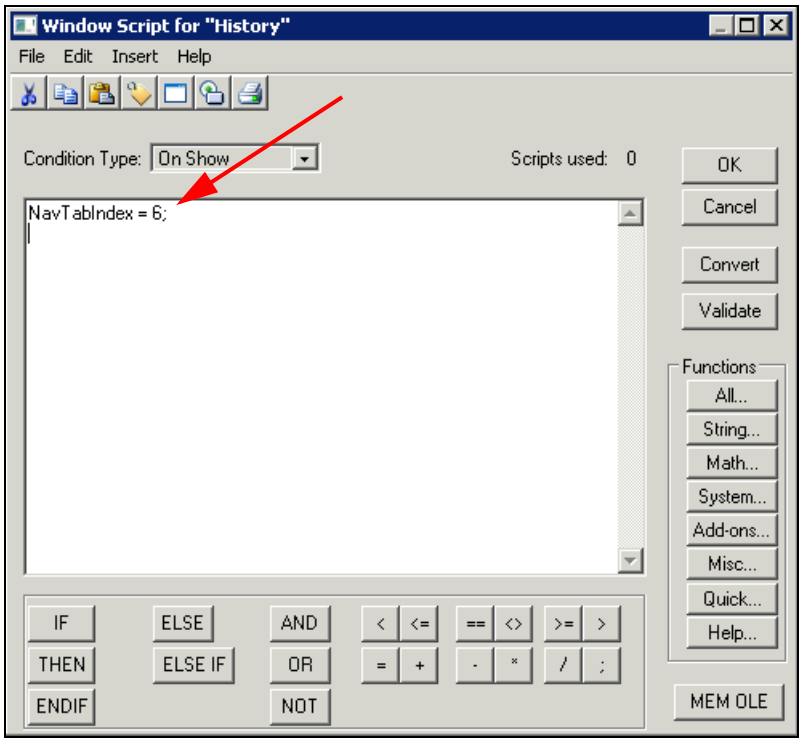


17. Click **Validate**.
18. Click **OK**.

19. Repeat the **Window Script** steps for each of the remaining windows as follows:

Window	NavTabIndex Number
Content2	2
Mixer	3
Alarms	4
RealTime	5
History	6

Important: Pay special attention to the **NavTabIndex** number. It is unique for each window to represent the tab order in the **NavigationTabs8** embedded graphic.

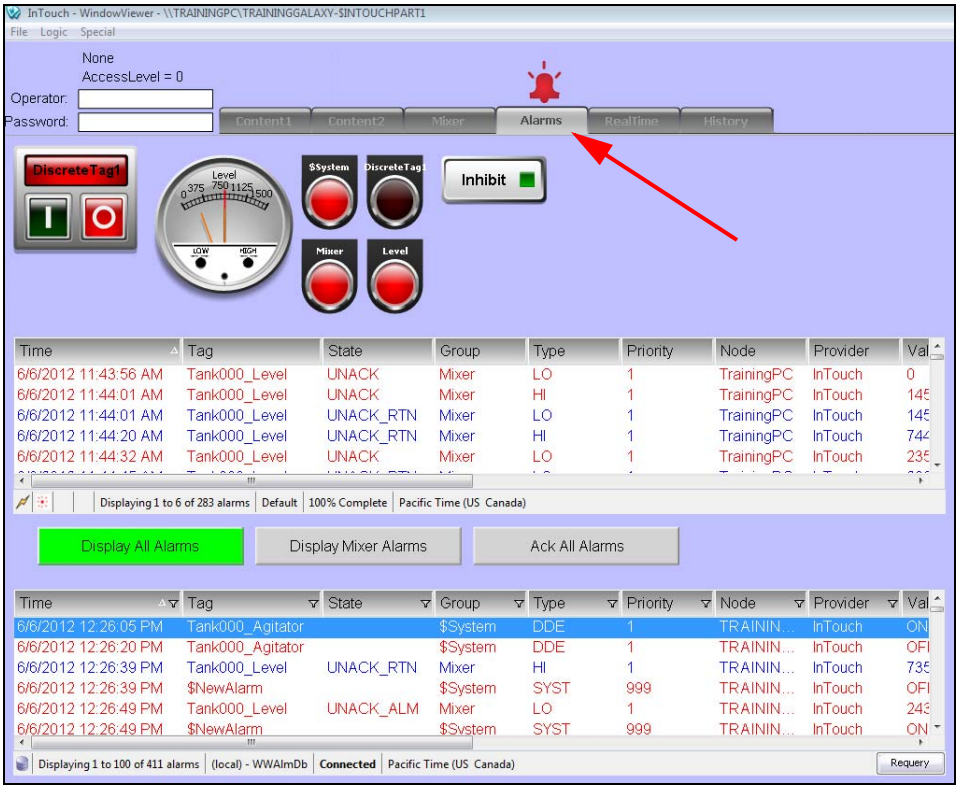


20. In WindowMaker, click **Save All Windows**, then close all windows except **Menu** and **Alarms**.



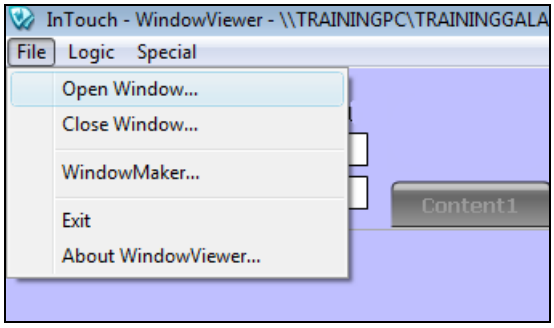
Test Navigation Tab Synchronization

21. Test in **Runtime**.

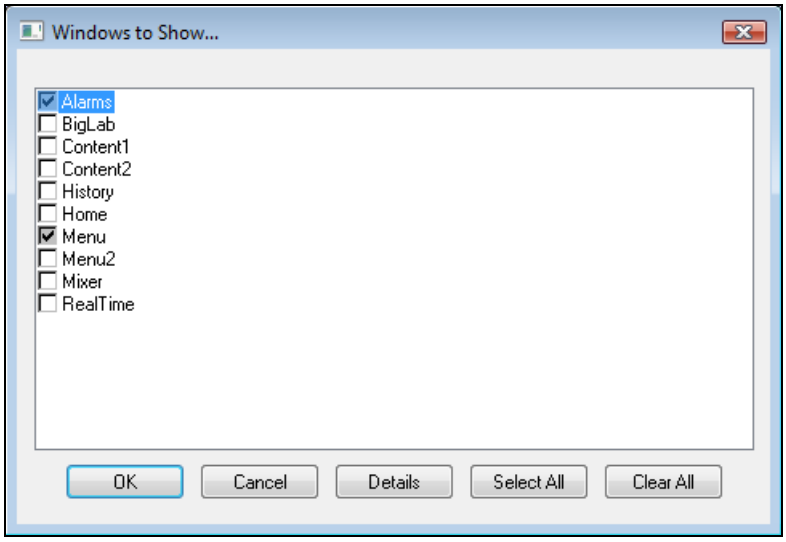


Notice the navigation bar now displays the current window.

22. On the WindowViewer **File** menu, click **Open Window**.

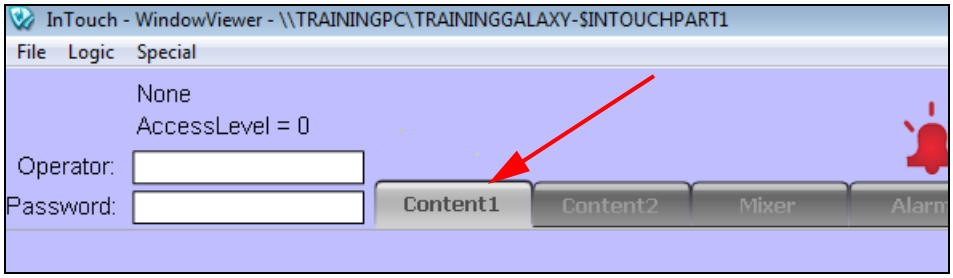


The **Windows to Show** dialog box opens.



23. Click **Content1**, then click **OK**.

Notice the navigation bar updates to the current window.



Note: The current user is **None**, yet the **File** menu allowed you to bypass the security you added to disable the **NavTabs** when no one is logged on. In the next lab, you increase security to prevent unauthorized access.

Lab 19 – Restricting User Access to Windows

Introduction

In this lab, you create a home window for unauthorized users, and restrict unauthorized access to the application.

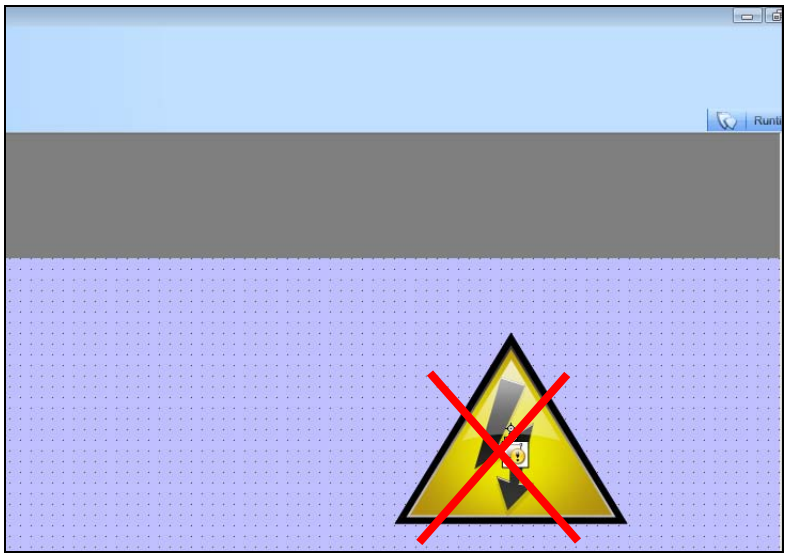
Objectives

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

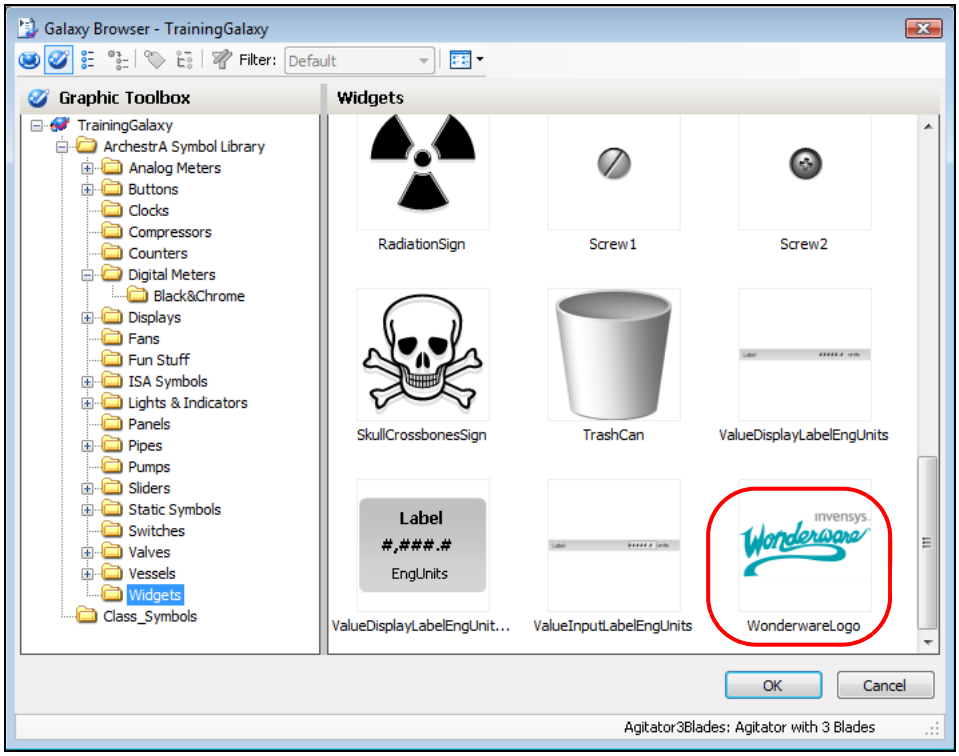
- Permit window navigation for authorized users
- Use a Data Change Script
- Use a Condition Script
- Insert comments into a script
- Clear the contents of a script

Create a Home Window

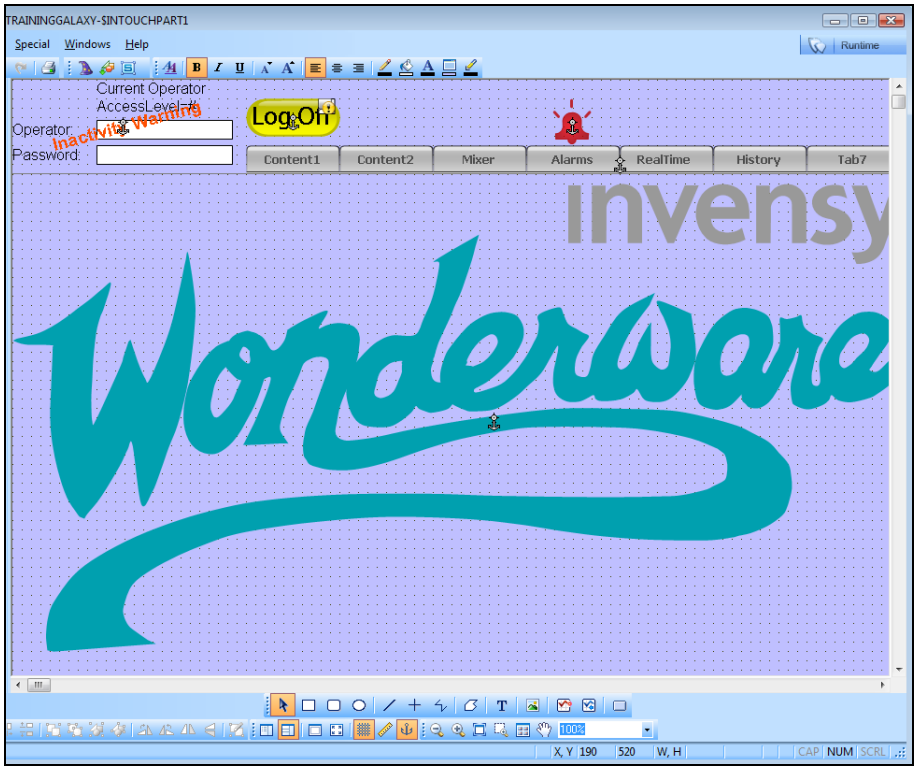
1. In WindowMaker, open the **Content2** and **Menu** windows.
2. Make a copy of the **Content2** window and name it **Home**.
3. Delete the warning light from the new **Home** window.



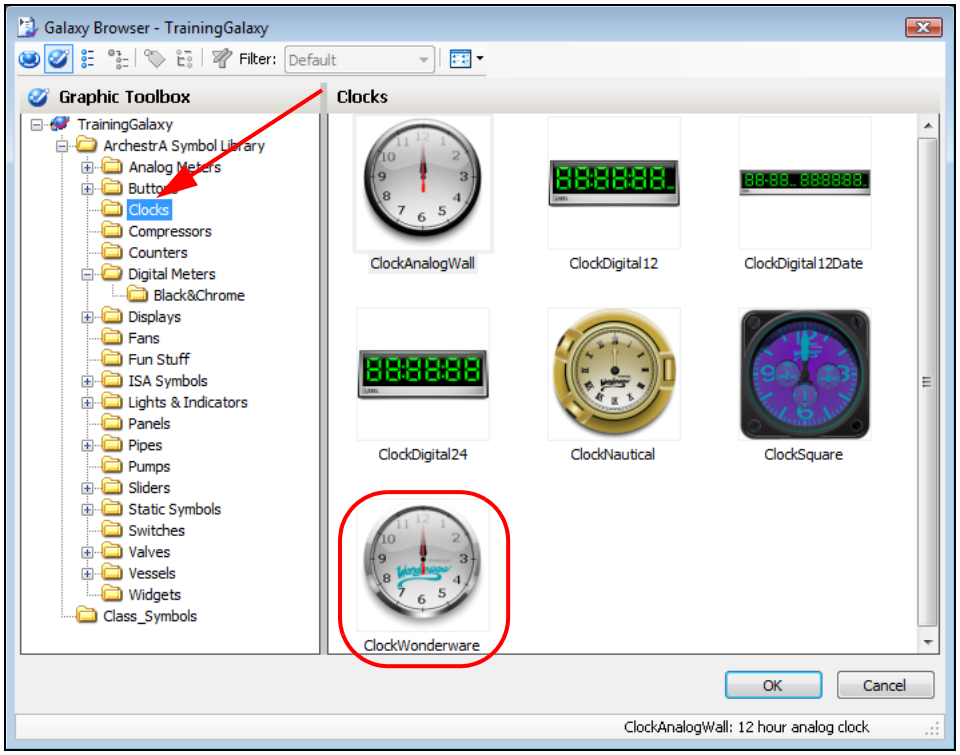
4. Embed **Widgets\WonderwareLogo** in the **Home** window.



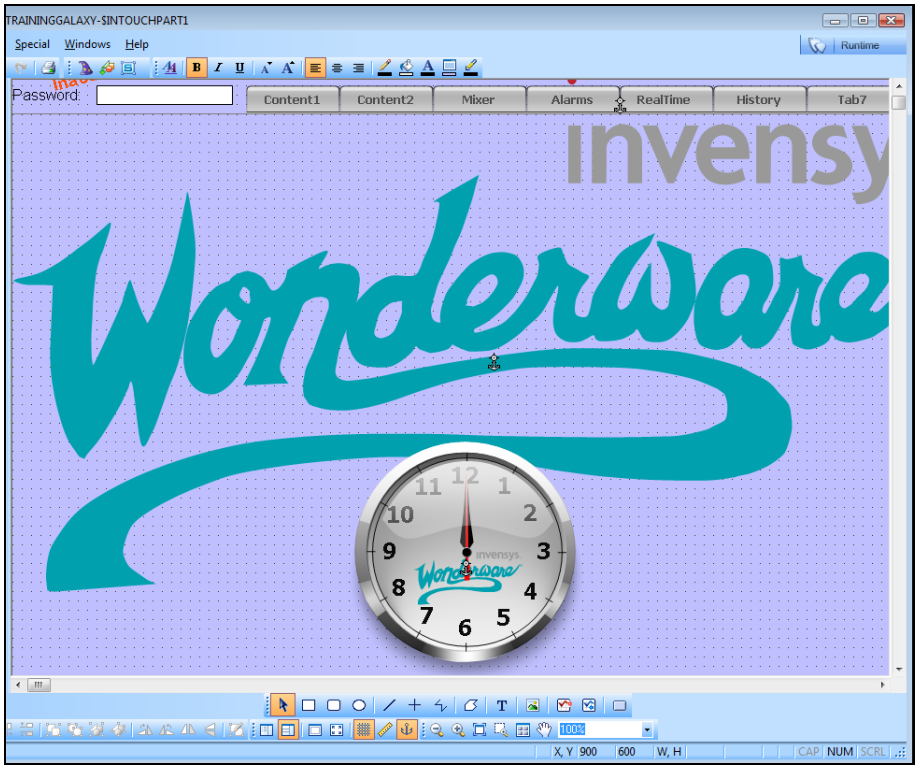
5. Resize the graphic to fill the window.



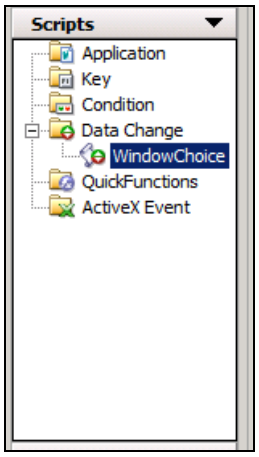
6. Embed **Clocks\ClockWonderware** in the **Home** window.



7. Place the graphic below the logo and resize.



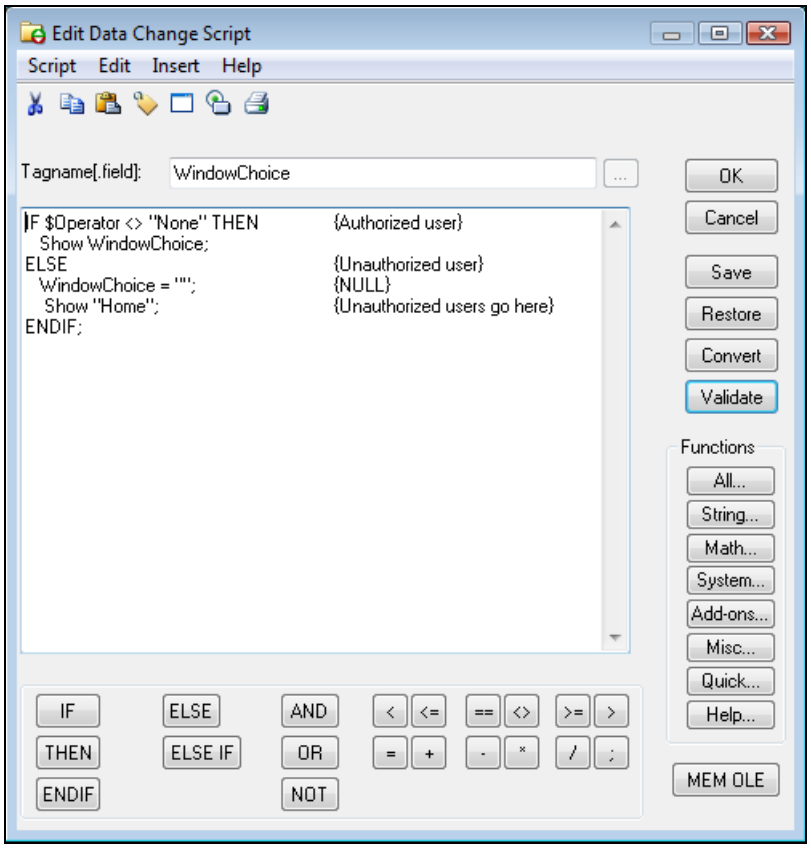
8. In **Scripts** pane, expand **Data Change** and double-click **WindowChoice** to edit the script.



The **Edit Data Change Script** dialog box appears.

9. Replace **Show WindowChoice;** with the following:

```
IF $Operator <> "None" THEN      {Authorized user}
    Show WindowChoice;
ELSE                             {Unauthorized user}
    WindowChoice = "";           {NULL}
    Show "Home";                 {Unauthorized users go here}
ENDIF;
```



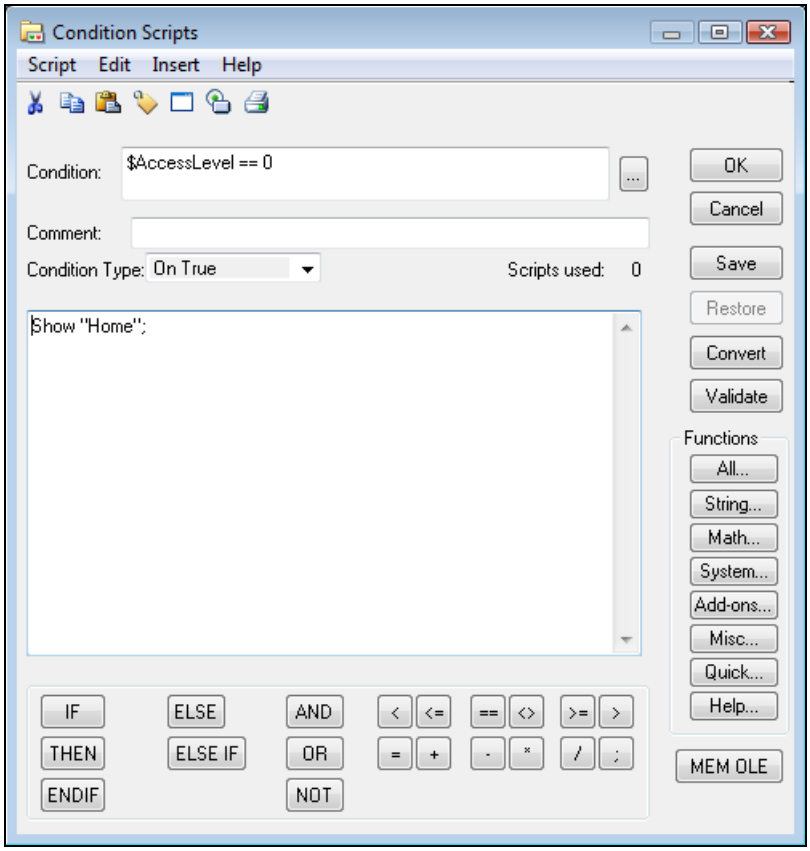
10. Validate the script and click **OK**.

Next, you create a condition script that will display the Home window when no one is logged in.

11. On the **Special** menu, click **Scripts | Condition Scripts**.
12. In the **Condition** field, enter the following:

    $AccessLevel == 0
13. From the **Condition Type** drop-down, select **On True**, then enter the following script:

    Show "Home";

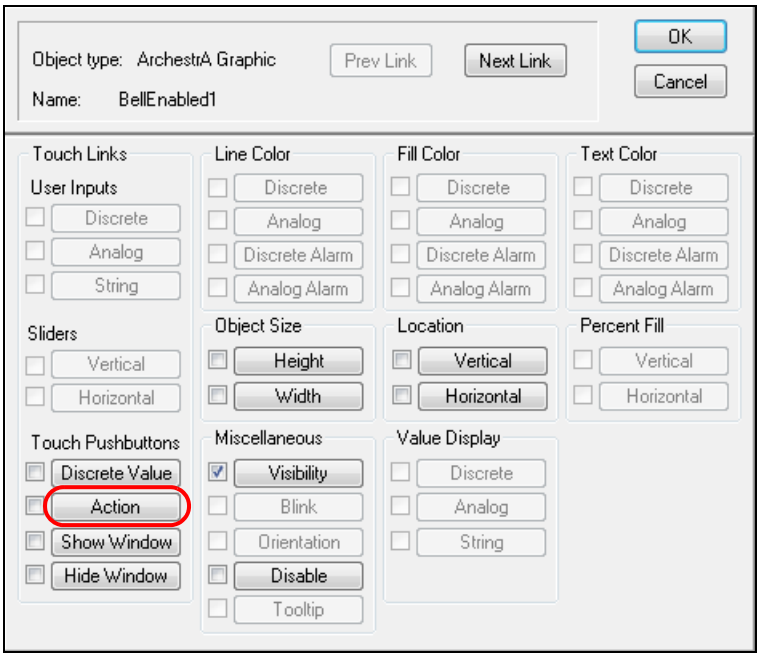


14. **Validate** and click **OK**.

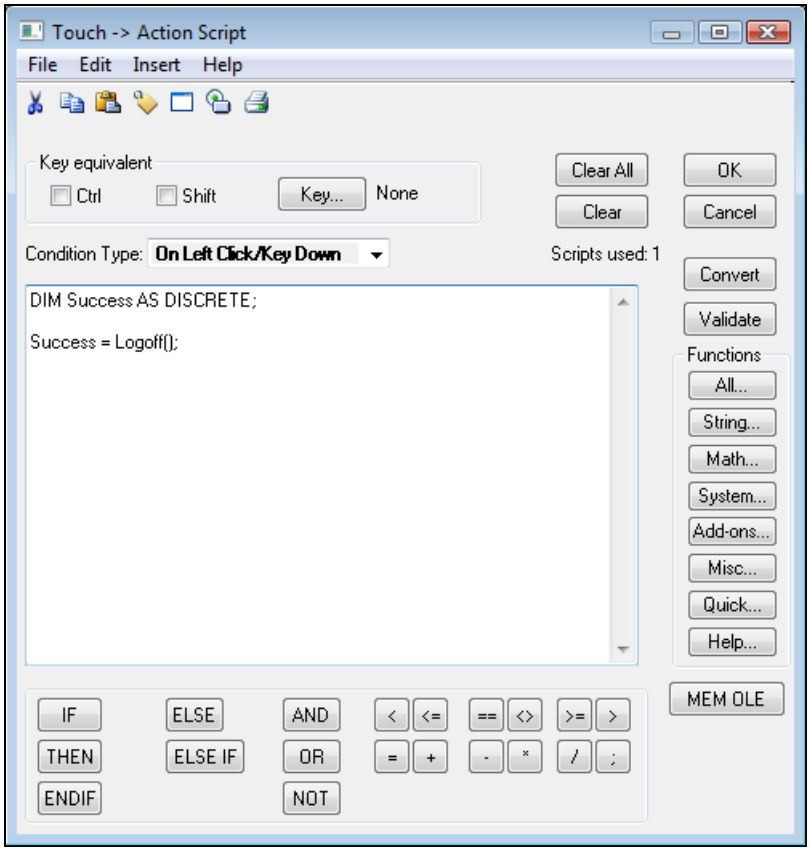
Add Window Navigation to the Alarm Bell

Next, you define an Action Script to navigate to the Alarms window using the Alarm Bell.

15. On the **Menu** window, right-click the alarm indicator bell and select **Animation Links**.
16. In the **Touch Pushbuttons** area, click **Action**.

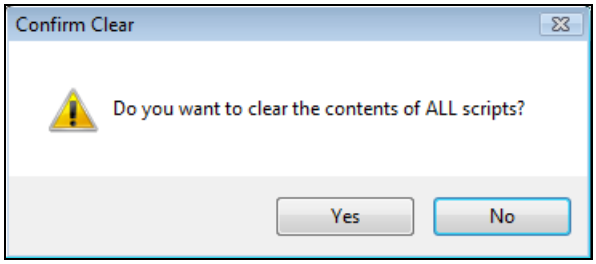


The **Action Script** dialog box appears.



17. Click **Clear All** to clear any existing content.

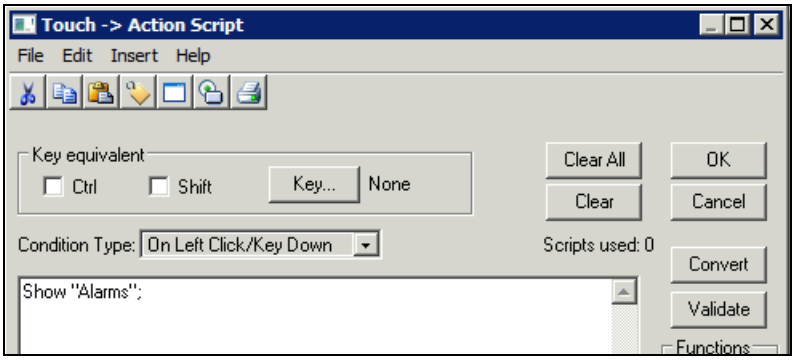
    The **Confirm Clear** message appears.



18. Click **Yes**.

19. In the script area, enter the following script:

    Show "Alarms";

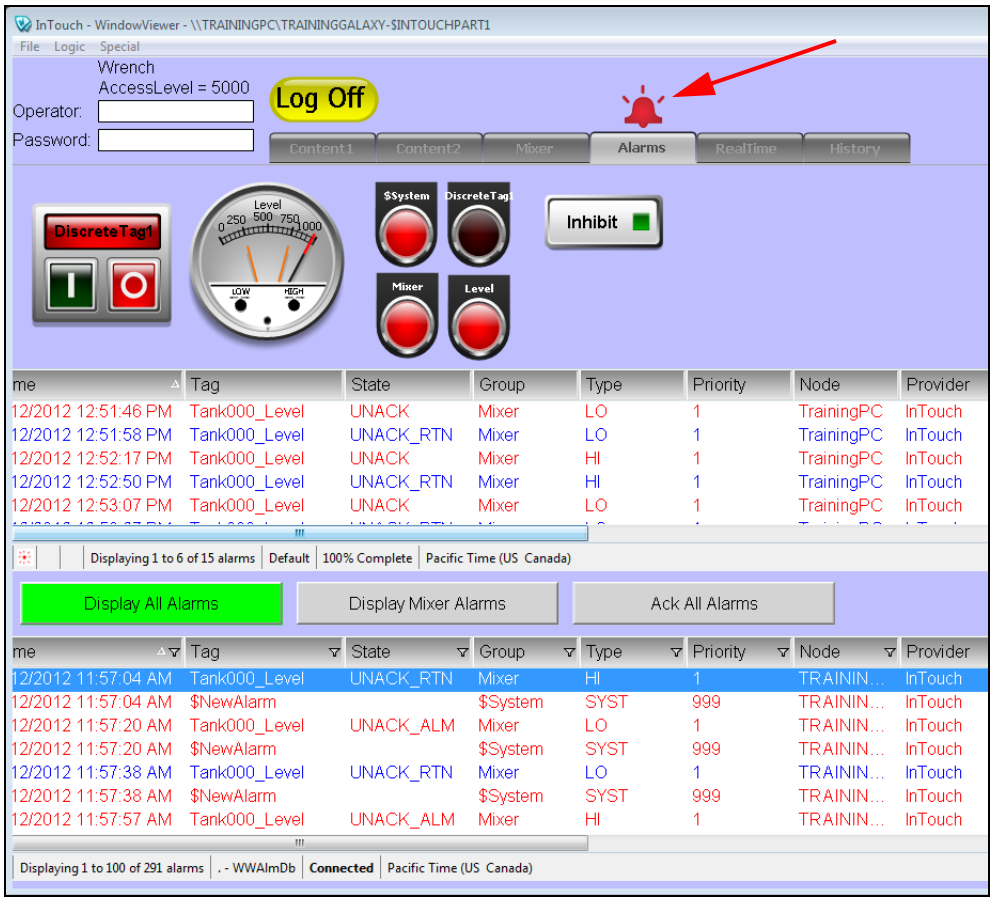


20. **Validate** and click **OK**.
21. Click **OK** to close the **Animation Links** dialog box.

Test What You Have Built

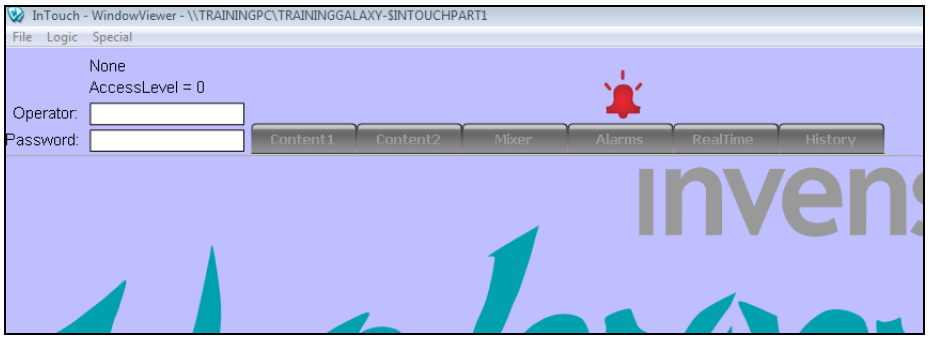
22. Switch to **Runtime**.
23. Log in as **Wrench**, with the password **iom**.
24. Switch to the **Alarms** window, then trigger a **DiscreteTag1** alarm.

    Notice that the alarm indicator bell is visible.



25. Navigate to the **RealTime** window.
26. Click **LogOff** to set the current user to **None**.

    The system returns you to the **Home** window, and the alarm indicator bell remains on the **Menu** window.



27. Click the alarm indicator bell.

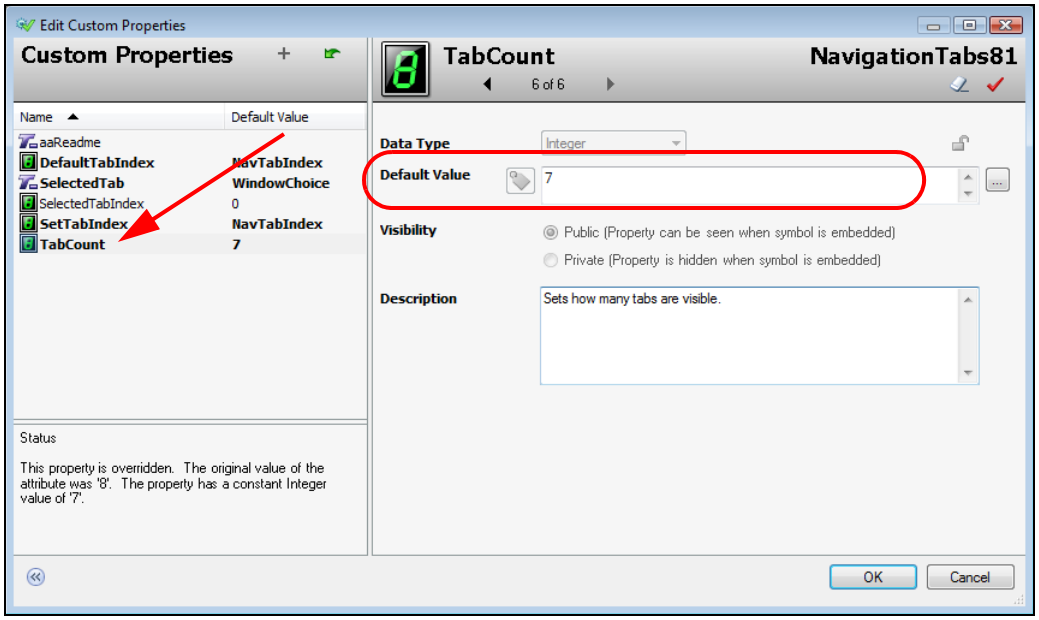
    Notice that the **Alarms** window briefly appears and then the **Home** window returns.

    This happens because the **Show "Alarms"** Action Script in the alarm indicator bell bypasses the **WindowChoice** Data Change script.

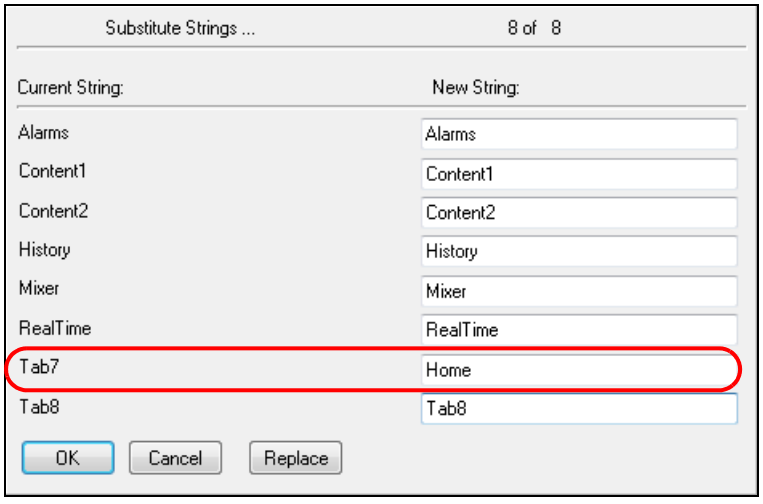
Restrict User Access

Next, you secure the application to restrict user access for unauthorized users, and when no one is logged in.

28. Switch back to **Development**.
29. Double-click **NavigationTabs8** and increase the **TabCount** to **7**.

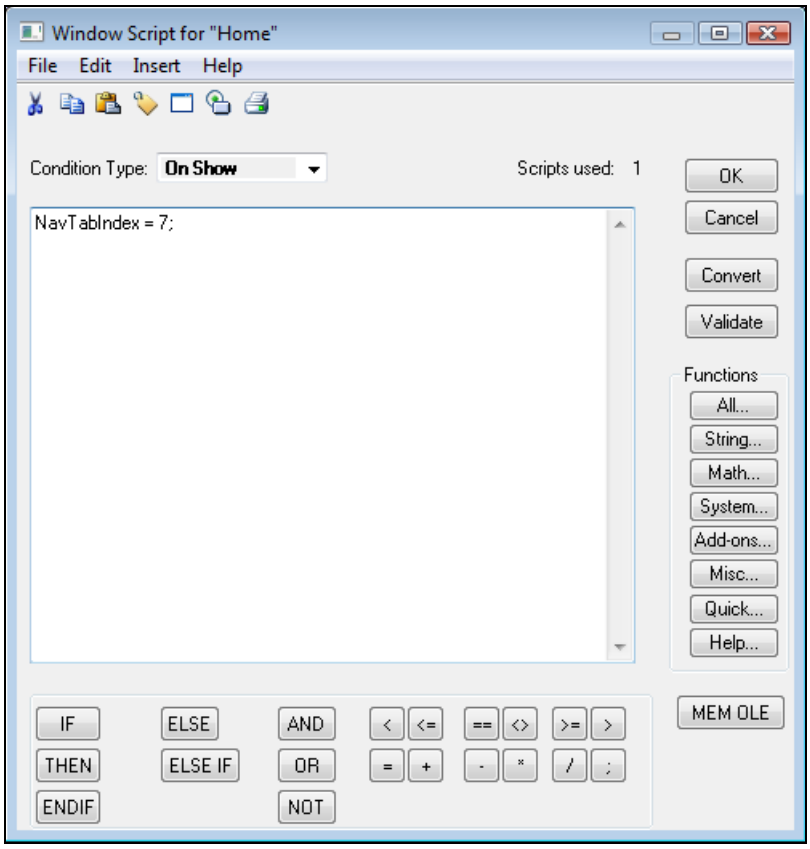


30. Click **OK**.
31. Right-click **NavigationTabs8** and select **Substitute | Substitute Strings** to change **Tab7** to **Home**.



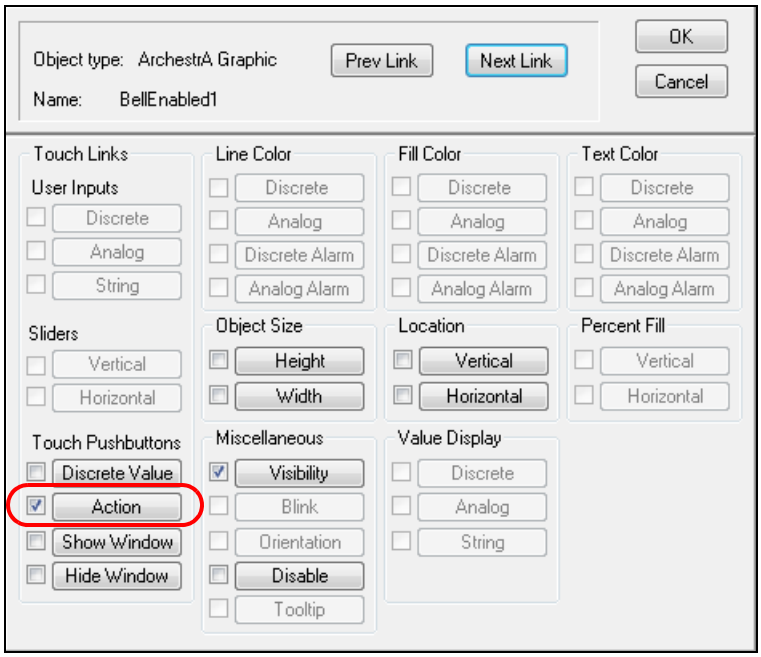
32. Click **OK**.

33. Right-click the **Home** window and select **Window Scripts**.
34. Change the **NavTabIndex** number to **7**.



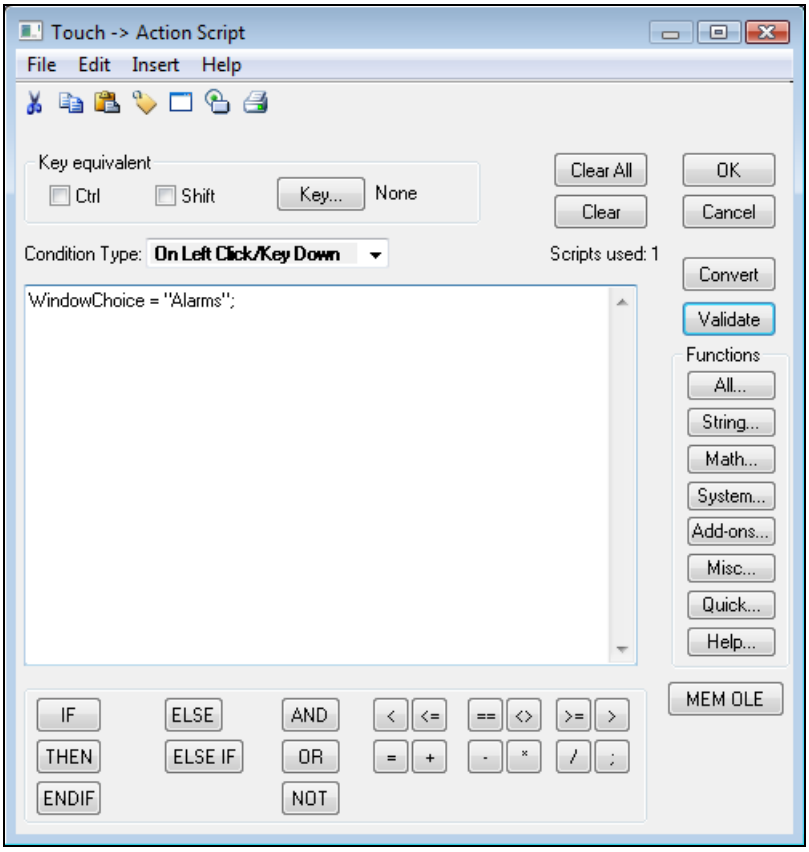
35. **Validate** and click **OK**.

36. On the **Menu** window, right-click the alarm indicator bell and select **Animation Links**.
37. In the **Touch Pushbuttons** area, click **Action**.



38. Replace **Show "Alarms";** with the following:

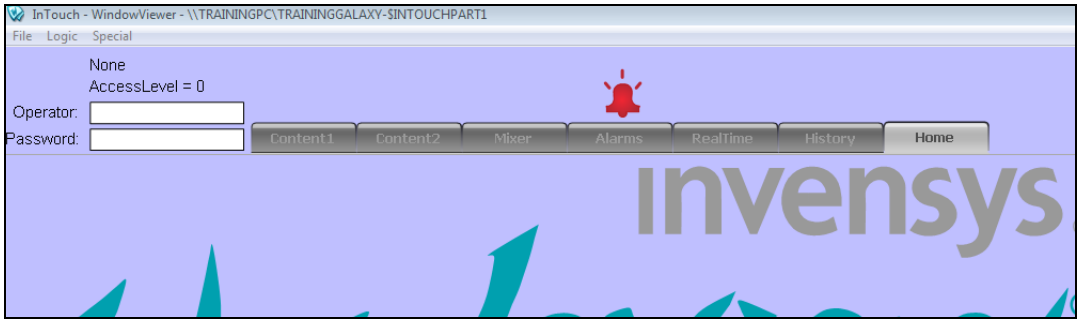
    WindowChoice = "Alarms";



39. **Validate** and click **OK**.
40. Click **OK** again to close the **Animation Links** dialog box.

Test What You Have Built

41. Switch to **Runtime**.
42. Log in as **Wrench**, with the password **iom**.
43. Switch to the **Alarms** window, then trigger a **DiscreteTag1** alarm.
44. Navigate to the **RealTime** window.
45. Click **LogOff** to set the current user to **None**.
46. Click the alarm indicator symbol and verify that the **Alarms** window does not display when no one is logged in.



Lab 20 – Defining Special User Access

Introduction

In this lab, you add application scripts to define different access functionality based on user login. You also restrict access to the WindowViewer menus to further secure the application.

Objectives

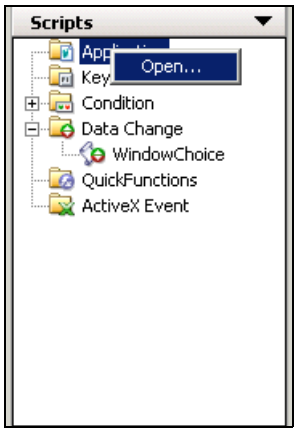
Upon completion of this lab, you will be able to:

- Create an On Startup application script
- Hide menu access in WindowViewer
- Restrict context menu access

Enable or Disable Keys

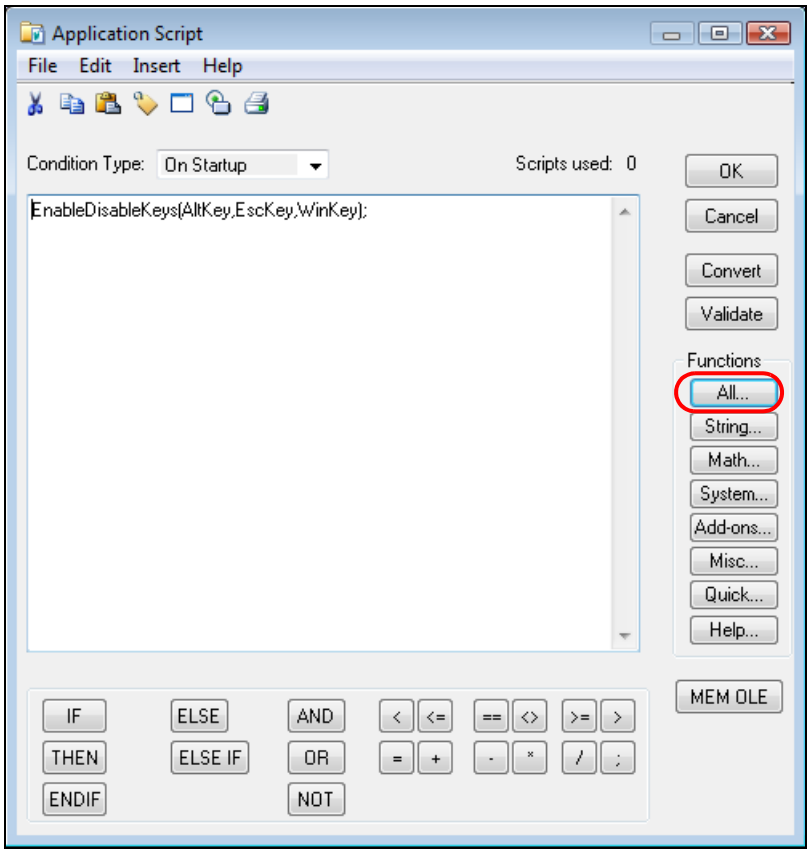
First, you use Enable Disable keys to control access to certain key stroke combinations.

1. Close WindowViewer.
2. In WindowMaker, in the **Scripts** pane, right-click **Application** and select **Open**.



3. In the **Condition Type** drop-down, click **On Startup**.
4. Click the **All** button, then select **EnableDisableKeys**.

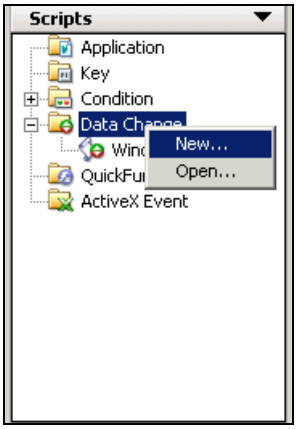
   EnableDisableKeys(AltKey,EscKey,WinKey); is inserted in the script field.



5. Replace each argument as shown:

   EnableDisableKeys(1,0,1);

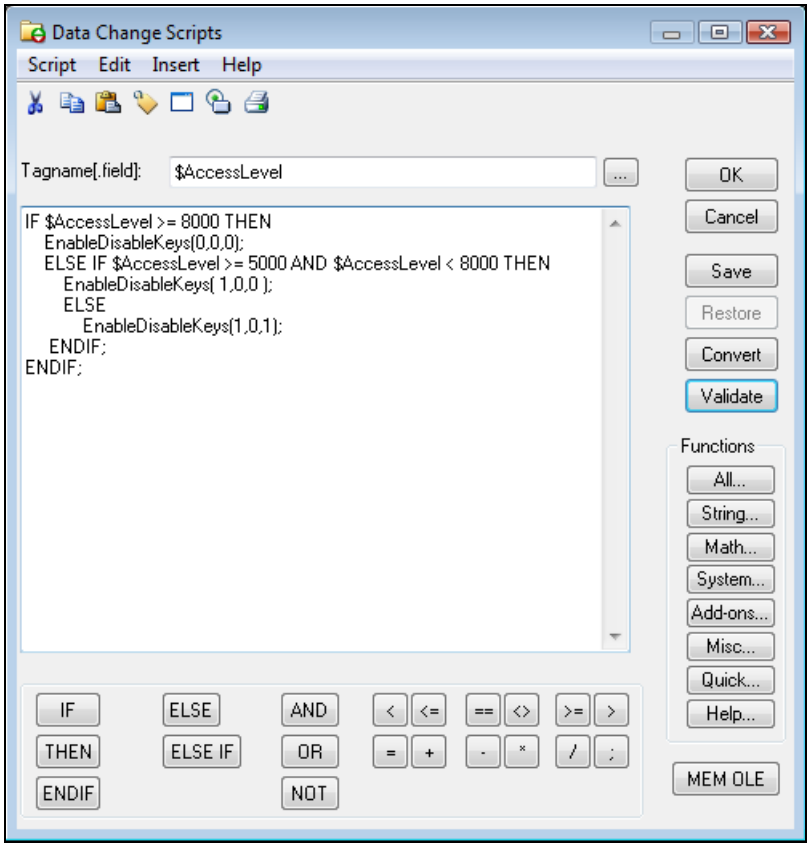
6. **Validate** and click **OK**.
7. Create a new **Data Change** script.



8. In the body of the script, click anywhere to place your cursor in the script area.
9. In **Tagname[.field]**, enter **$AccessLevel**.
10. On the **Edit** menu, click **Clear**.

11. Enter the following **Data Change** script:

```
IF $AccessLevel >= 8000 THEN
    EnableDisableKeys(0,0,0);
    ELSE IF $AccessLevel >= 5000 AND $AccessLevel < 8000 THEN
        EnableDisableKeys( 1,0,0 );
        ELSE
            EnableDisableKeys(1,0,1);
     ENDIF;
ENDIF;
```



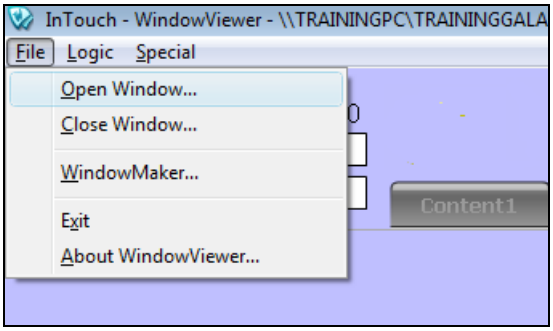
This script enables the Alt, Esc, and Win keys when someone with an access level greater than or equal to 8000 logs in. Users with an access level of 5000, but less than 8000, will have access to the Win key.

12. **Validate** and click **OK**.
13. Open the **Menu** and **Home** windows.

14. Switch to **Runtime**.
15. Press the **Win** key on your keyboard.

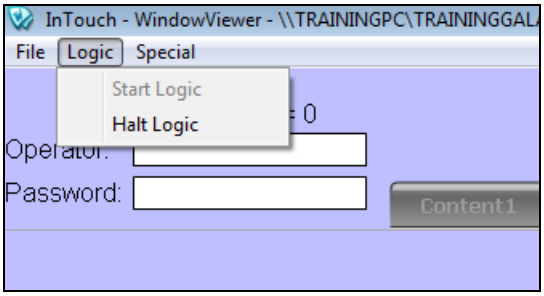
    The **Start** menu does not open.

    Notice that commands are still available from the **File** menu.



    Also notice that the **Logic** menu is accessible.

    Halt Logic stops all scripts from running, which means your security script can be disabled.



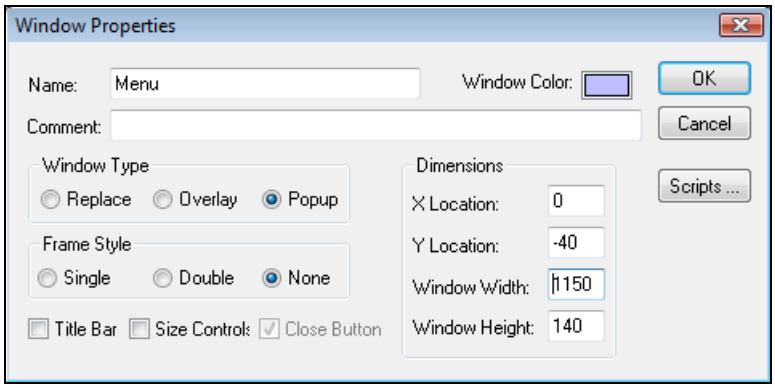
16. Log in as **Wrench**, then press the **Win** key.

    The **Start** menu appears.
17. Close WindowViewer and return to **Development**.

Configure Windows to Hide the Menu

Next, you reposition the Menu window above the menu bar to cover the available options.

18. In WindowMaker, right-click **Menu** and select **Properties**.
19. In the **Window Properties** dialog box, change the **Y Location** to **-40**.
20. Change the **Window Height** to **140**.



Note: These coordinates may vary based on your window resolution.

21. Click **OK**.
22. In the **Menu** window, click and then press **F2** to select all the objects.
23. Press the **Down Arrow** key on your keyboard four times so the **NavigationTabs8** is at the bottom of the window.
24. Switch to **Runtime**.

    Notice that the **Menu** window now covers the WindowViewer menu bar to prevent clicking menu options. Additionally, your **Data Change** script disables the **Alt** key so unauthorized users cannot use keyboard shortcuts like **Alt+F** to access the WindowViewer **File** menu.

Section 3 – Enhancing History with QuickScript

This section describes history tracking enhancements using scripts, and includes Live HistTrend functionality. This section also describes historical pen functionality used to track components of historical pens, Historical Pen Runtime selection and unassignment, and configuration of multiple scripts using Action Script animation.

Overview

This section describes history tracking enhancements using scripting and includes how to force continuous refreshing of a historical trend to current time, which you will call Live HistTrend functionality. This section also describes historical pen functionality used to track components of historical pens and Historical Pen selection and unassignment during runtime, and configuration of multiple scripts using Action Script animation.

There are several internal functions you can use to specify tag trending for each pen display the value at a scooter location, scroll the trend by a percentage, and so on. You can also process a statement in a QuickScript to assign a new tagname to any pen in any historical trend.

For example, the following statement could be used in your QuickScript:

```
MyHistTrendTag.Pen1=MyLoggedTag.TagID;
```

When this QuickScript is executed, Pen1 in the historical trend associated with the **Hist Trend** tagname "MyHistTrendTag" would begin trending the historically logged data for the "MyLoggedTag."

The **HistData** utility program provides access to the historical data files created by InTouch. It is used to move historical data into another program such as Microsoft Excel. This allows you to have quick access to historical data and to create historical storage of data for later access.

The HistData program contains its own I/O. The items are used to specify data you want to access such as start period, duration and sampling interval.

Manipulating Historical Trending at Runtime

You can use QuickScript functions with trend objects or animation link expressions to control a historical trend during runtime. For example, you can use QuickScripts to update a trend to the current time, reassign tags to trend pens, connect pens to the chart, replot the grid, and remove or replot the scooters.

Updating the Trend to the Current Time

You can create a script to update a historical trend to show recent tag data using the **HTUpdateToCurrentTime()** function. The HTUpdateToCurrentTime() function retrieves and shows the data with an end time equal to the current time. The start time is equal to end time minus the width of the chart:

```
HTUpdateToCurrentTime(Hist_Tag);
```

Hist_Tag is HistTrend tag assigned the name of the historical trend. For example, the following statement retrieves and shows data for the Trend1 historical tag at the current time:

```
HTUpdateToCurrentTime("Trend1");
```

If the current time is 3:04 PM and the width of the trend is 60 seconds, the new end time is 3:04 PM. The new trend start time is 3:03 PM.

Changing the Trend Configuration

You can use these script functions to change the tags assigned to the pens of a historical trend:

- **HTSelectTag()** Function
- **HTSetPenName()** Function

The **HTSelectTag()** function opens the Select Tag dialog box for the operator to assign a different tag to a trend pen. Note the Select Tag dialog box only lists the tags that are defined for historical logging with the Log Data option selected in the Tagname Dictionary.

However, it is possible to use the Tag Browser to filter the display to a smaller set of tags. For example, all tags that begin with "A". The function returns the selected tag and can be used as function parameter to assign a tag to a pen.

The following script causes the Select Tag dialog box to appear in WindowViewer. The user can then select a tag from the list. This tag is assigned to pen 1 by the Historical chart named HistTrend:

```
HTSetPenName( "HistTrend", 1, HTSelectTag() );
```

This QuickScript function is the only method to add tags from a distributed history provider during runtime. More on this in the InTouch 2012 R2 – Part 2 course.

You may see the following error message when you attempt to unassign a trend pen:

```
VIEW /UpdateData: Invalid DBS.TAGNAME handle – 0
```

This error occurs if you are trying to unassign a pen that was previously assigned to a remote tag in the form histprovider.tag_name. To resolve this error, create a local tag with the Log Data option selected. Then, use the following script to unassign the pen:

```
HTSetPenName( "HistTrend", 1, "localtag" );
{assigns the pen to a locally logged tag---localtag}

HistTrend.Pen1=None;
{unassigns the pen}
```

Where None is a TagID tag. With no further definition required.

Examples

The following statement assigns the OutletPressure tag to pen 3 of Trend1:

```
HTSetPenName("Trend1",3,"OutletPressure");
```

The following statement assigns the HistPrv1.Tag1 tag to TrendPen4 of Trend1:

```
HTSetPenName("Trend1",TrendPen4,"HistPrv1.Tag1");
```

Retrieving Information about the Trend and Historical Data

You can create scripts that retrieve information from a historical trend while it is running.

Use the following **Functions**:

- **HTGetPenName()**
- **HTGetTimeAtScooter()**
- **HTGetTimeStringAtScooter()**
- **HTGetValue()** Function
- **HTGetValueAtScooter()**
- **HTGetValueAtZone()**
- **HTScrollLeft()**
- **HTScrollRight()**
- **HTZoomIn()**
- **HTZoomOut()**
- **HTGetPenName()**

The **HTGetPenName()** function returns the name of the tag currently assigned to the specified pen number of the historical trend.

Example

The following statement retrieves the name of the tag assigned to Pen 2 of the Trend1 trend and places the name in the TrendPen message tag:

```
TrendPen=HTGetPenName("Trend1", Trend1.UpdateCount,2);
```

The **HTGetTimeAtScooter()** returns the time in seconds after 00:00:00 hours GMT, January 1, 1970 for the sample at the scooter location specified by the ScootNum and ScootLoc arguments.

Example

The following statement retrieves the time in seconds for the value at the current left scooter location of the Trend1 trend:

```
HTGetTimeAtScooter("Trend1",Trend1.UpdateCount,1,Trend1.ScooterPosLeft);
```

The **HTGetTimeStringAtScooter()** function returns the string containing the time/date for the sample at the specified scooter location.

Remarks

Any changes to the values assigned to the UpdateCount, ScootNum, and ScootLoc arguments cause the expression to be evaluated. This ensures the expression is evaluated after new data retrievals or after a scooter is moved. The format of the string determines the contents of the return value.

Example

The following statement retrieves the date and time for the value at the current scooter location for the right scooter of the Trend1 trend. The value is stored in the NewRightTimeString message tag and is in "Time" format:

```
NewRightTimeString=HTGetTimeStringAtScooter("Trend1",Trend1.UpdateCount,2,Trend1
.ScooterPosRight,"Time");
```

The **HTGetValue()** function returns a value of the requested type for the trend's specified pen.

Remarks

The function returns the requested value as a real value.

Example

The following statement obtains the standard deviation for the pen 2 data retrieved from the PumpPress trend. The value is stored in the LeftHemisphereSD memory real tag:

```
LeftHemisphereSD=HTGetValue("PumpPress",PumpPress.UpdateCount,2,"PenStdDev");
```

The **HTGetValueAtScooter()** function returns a value of the requested type for the sample at the specified scooter position, trend, and pen number. The UpdateCount argument causes the expression to be evaluated after function processing is finished.

Example

The following function returns a 1 if the value is an actual sample or a 0 if the value is invalid for pen 3 of the Trend1 trend for the right scooter's current position:

```
HTGetValueAtScooter("Trend1",Trend1.UpdateCount,2,Trend1.ScooterPosRight,3,
"PenValid");
```

The **HTGetValueAtZone()** function returns a value of the requested type for the data located between the right and left scooter positions for a trend's specified pen.

Remarks

A real value is returned representing the calculated value of the given type. Specifying constant values for the Scoot1Loc and Scoot2Loc arguments have no effect and are only used to trigger the evaluation of the function. The function uses the trend tag's .ScooterPosLeft and .ScooterPosRight Dot Field values directly, regardless of the values you specify for the Scoot1Loc and Scoot2Loc arguments.

Example

The following statement calculates the average value for data between the right and left scooters of the Trend1 trend for pen 1. The value is stored in the AvgValue memory real tag:

```
AvgValue=HTGetValueAtZone("Trend1", Trend1.UpdateCount,Trend1.ScooterPosLeft,
Trend1.ScooterPosRight,1,"PenAverageValue");
```

Panning and Zooming the Trend

You can create QuickScripts containing functions that select specific data from a historical trend during runtime.

The **HTScrollLeft()** function sets the start time of the trend to an earlier time than the current start time by a percentage of the trend's total time span. The effect is to scroll the chart to the left to an earlier time by a specified percentage of the trend’s total time span.

Example

The following statement scrolls the time/date left by 10 percent of the PumpPress trend's total width:

```
HTScrollLeft("PumpPress",10.0);
```

If the current display starts at 12:00:00 PM and the display width is 60 seconds, then the new trend starts at 11:59:54 AM after the function is processed.

The **HTScrollRight()** function sets the start time of the trend to a time later than the current start time by a percentage of the trend's width. The effect is to scroll the date/time of chart to the right by a specified percentage of the trend's width.

Example

The following statement scrolls the PumpPress trend to the right by 20 percent:

```
HTScrollRight("PumpPress",20.0);
```

If the current display starts at 12:00:00 PM and the display width is 60 seconds, then the new trend starts at 12:00:12 PM after the function is processed.

The **HTZoomIn()** function calculates a new chart width and start time. If the trend's scooters are at the left and right sides of the trend, then the new chart width equals the old chart width divided by two. The new start time is calculated based on the value of the **LockString** argument.

If the scooters are not at the left and right sides of the trend, the HTZoomIn() function zooms the trend to the zone defined by the scooters and ignores the LockString argument.

Remarks

If the scooter positions are not at the left and right sides of the trend, the new chart width is the time between .ScooterPosLeft and .ScooterPosRight positions. In this case, the value of LockString is not used. The minimum chart width is one second. The scooter positions are set to .ScooterPosLeft=0.0 and .ScooterPosRight=1.0 after the zoom.

Example

The following statement zooms the display by a factor of two and maintains the same start time for the Trend1 trend. Trend1.ScooterPosRight is equal to 1.0 and Trend1.ScooterPosLeft is equal to 0.0. If the start time before zooming was 1:25:00 PM and the chart width was 30 seconds, the new start time remains at 1:25:00. The new chart width is 15 seconds.

```
HTZoomIn("Trend1","StartTime");
```

The **HTZoomOut()** function calculates a new chart width and start time. The new chart width is the old chart width multiplied by two. The new start time is calculated based on the value of the LockString argument.

Remarks

The current scooter positions have no effect on HTZoomOut(). After the function zoom finishes, the new scooter positions are set to .ScooterPosLeft=0.0 and .ScooterPosRight=1.0.

Example

The following statement zooms out the trend time by a factor of two and maintains the same center time for the Volume trend. If the start time before zooming was 2:15:00 PM and the chart width was 30 seconds, the start time after zooming is now 2:14:45. The chart width is 60 seconds and the center of the trend remains at 2:15:15.

```
HTZoomOut("Volume","Center");
```

Printing the Trend

You can print a historical trend currently visible from a WindowViewer screen using the **PrintHT()** function in a script.

The PrintHT() function prints the historical trend currently visible on the screen. Usually, the PrintHT() function is associated with a screen button included on the historical trend window. Operators click the button to print the visible historical trend with its current values.

Example

This example prints the PumpPress historical trend currently visible on the screen:

```
PrintHT(PumpPress);
```

Several factors determine how fast a trend can be printed. The primary factor is the size of the trend on the printed page. The display mode of the trend also affects printing performance. Min/Max or Average/Scatter trends can be printed more quickly than Average/Bar Chart trends. Also, the longer and wider the lines on the trend are, the longer it takes to print. Another factor that affects printing performance is the background color of the trend. In most cases, a white background prints more quickly than a colored background.

Configuring Trend Printing Options

You can configure options that determine how a trend is printed. To configure historical trend printing:

- On the **Special** menu, click **Configure | Historical Logging**. The **Historical Logging Properties** dialog box appears
- In the **Printing Control** area, specify the percentage of the page to print the trend in the **Default %** of page to print on box. If you enter 50, the trend is printed on half of the page vertically and horizontally. A trend printed at 50 percent takes much less time to print than a full-page trend. As a printing alternative, you can use the **PrintWindow()** QuickScript function
- In the **Max consecutive time to spend printing** field, enter the process time slice in milliseconds. A time slice represents the period allocated to the computer processor to run the print module process in the foreground and print the trend. A longer time slice enables the trend to be printed more quickly at the expense of other processes running on the computer
- In the **Time to wait between printing** field, enter the time in milliseconds the print module waits between processor time slices. A shorter waiting period between processor time slices enables the trend to be printed more quickly
- Click **Select Printer Font**. The **Font** dialog box appears. Select the characteristics of the text appearing in a trend
- Click **OK** to save your printing configuration and close the **Historical Logging Properties** dialog box

Troubleshooting the Trend

You can create QuickScripts to verify if data was successfully retrieved that appears in a historical trend. Use the **HTGetLastError()** function to troubleshoot the trend.

The HTGetLastError() function can be used in a script to determine if an error occurred during the last data retrieval for a specified historical trend pen.

Examples

The following statement retrieves the status of the last data retrieval for pen 3 of the Trend1 trend and assigns the result to the ResultCode integer tag:

```
ResultCode=HTGetLastError("Trend1", Trend1.UpdateCount,3);
```

In an animation Analog Value Display QuickScript the following statement would be used:

```
HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

Displaying real time values in a Trend, you can create real time trends by two methods. The **Real Time Trend** object provides a standard set of controls to select the data, set a time range, and specify the physical appearance of the graph.

Note: InTouch also includes the 16-Pen Trend Wizard, which is an optional control to create real-time and historical trends. For more information about creating real-time trends with the 16-Pen Trend Wizard, see Chapter 2, Using the 16-Pen Trend Wizard, in the I*nTouch HMI Supplementary Components Guide*.

Lab 21 – Creating a Live Mode for the Historical Trend

Introduction

In this lab, you create a live mode for the Historical Trend wizard. To accomplish this, you use a While Showing window script and a toggle button to enable continuous update of the historical trend wizard you implemented in a previous lab.

Objectives

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

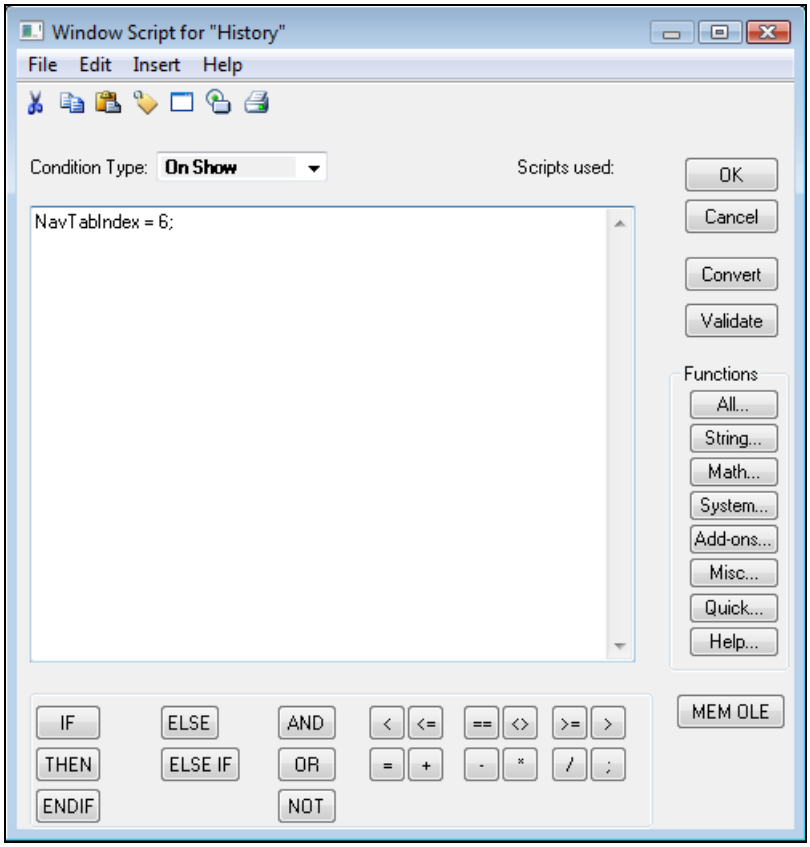
- Implement a live mode for historical trending
- Define a While Showing window script

Create While Showing Window Script

First, you create a script and discrete tag to refresh the historical trend display continuously.

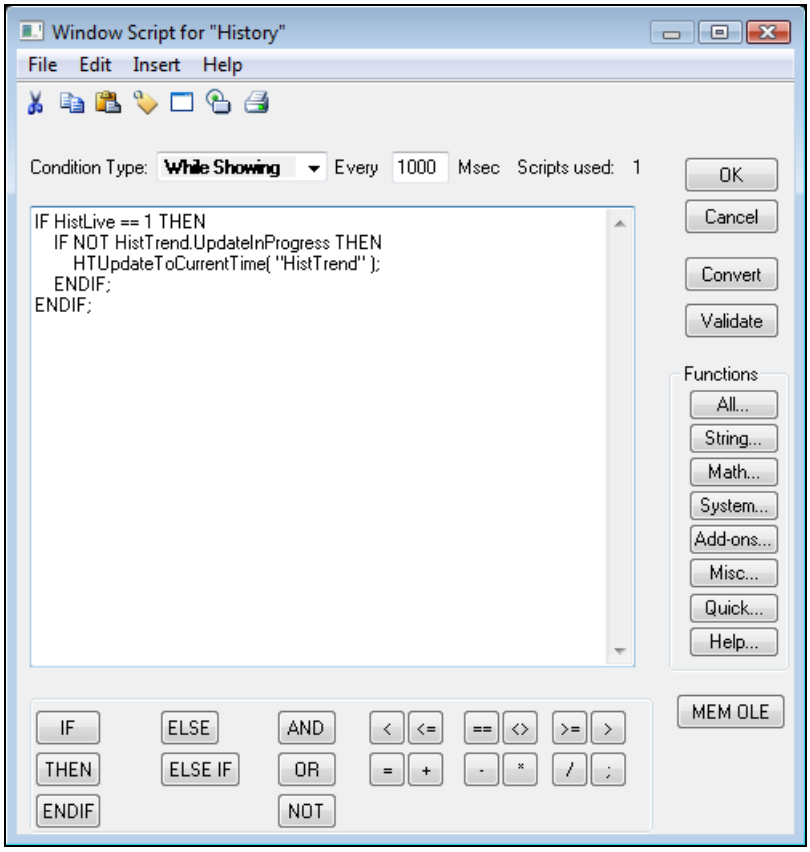
1. Close WindowViewer.
2. In WindowMaker, open the **History** window.
3. Right-click the window and select **Window Scripts**.

   The **Window Scripts** dialog box opens for the **History** window.



4. Click the **Condition Type** drop-down, then select **While Showing**.
5. In the **Every** field, enter **1000**.
6. Enter the following script:

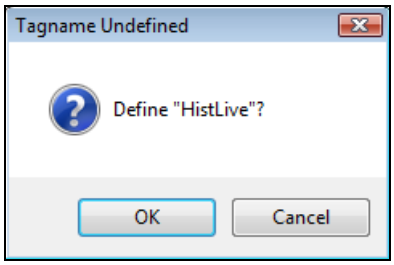
```
IF HistLive == 1 THEN
    IF NOT HistTrend.UpdateInProgress THEN
        HTUpdateToCurrentTime( "HistTrend" );
    ENDIF;
ENDIF;
```



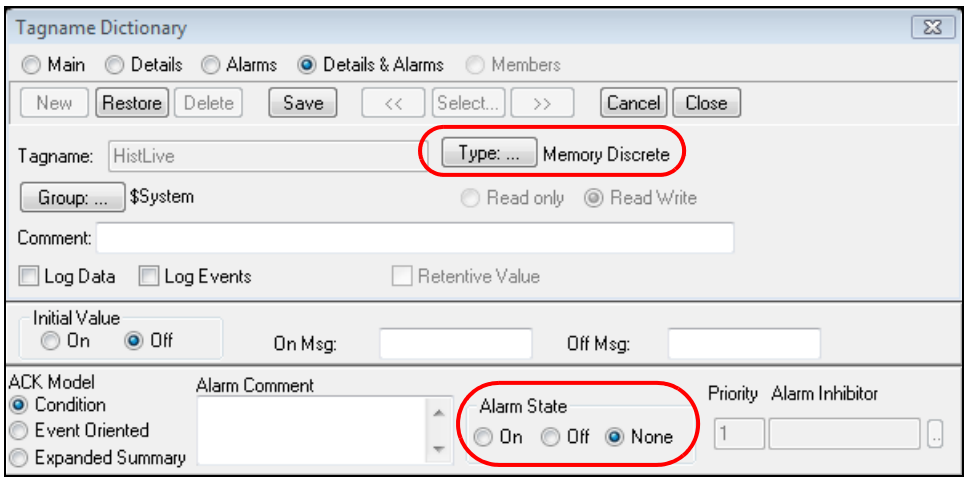
This script monitors the completion status of a historical retrieval request to prevent a history retrieval backlog.

7. Click **Validate**.

   The system prompts you to define **HistLive**,



8. Click **OK**.
9. Click **Type**, then select **Memory Discrete**.
10. Click **OK**.
11. In the **Alarm State** area, verify **None** is selected.

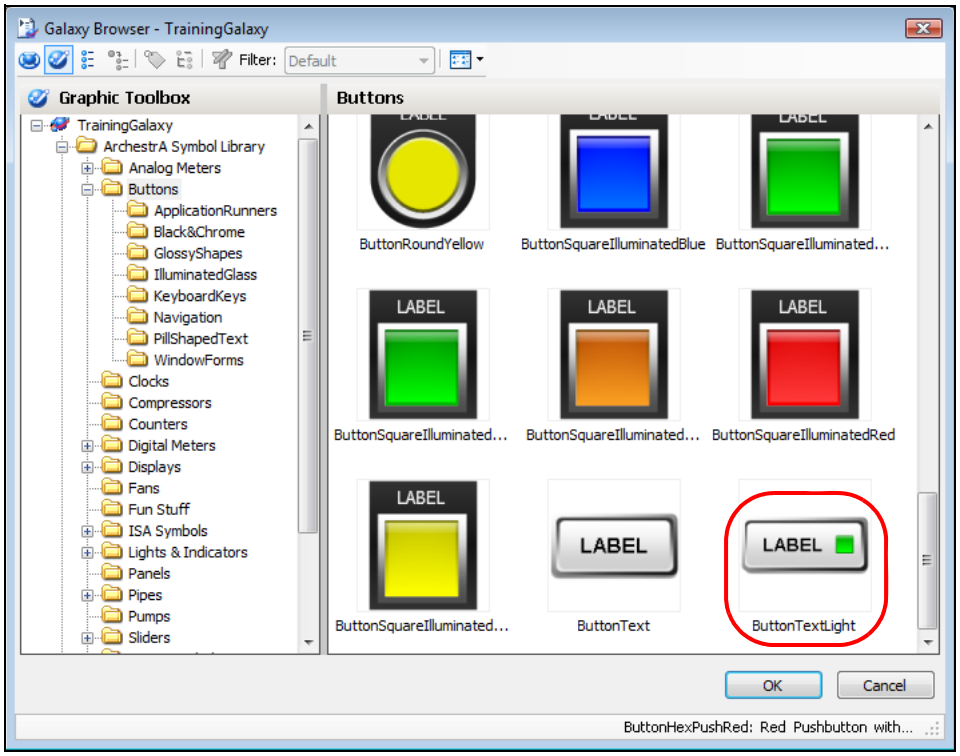


12. Accept all remaining defaults and click **Close**.
13. Click **OK**.

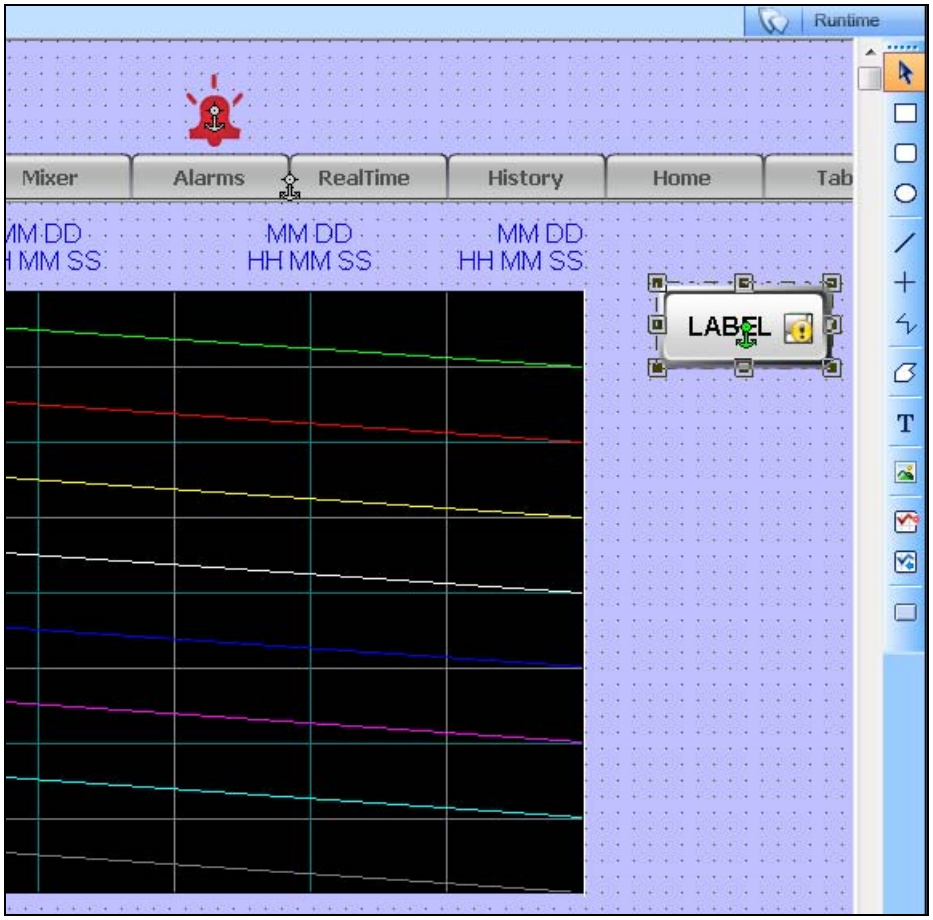
Add a Live Mode Button

Next, you add button to toggle the live mode of the historical trend.

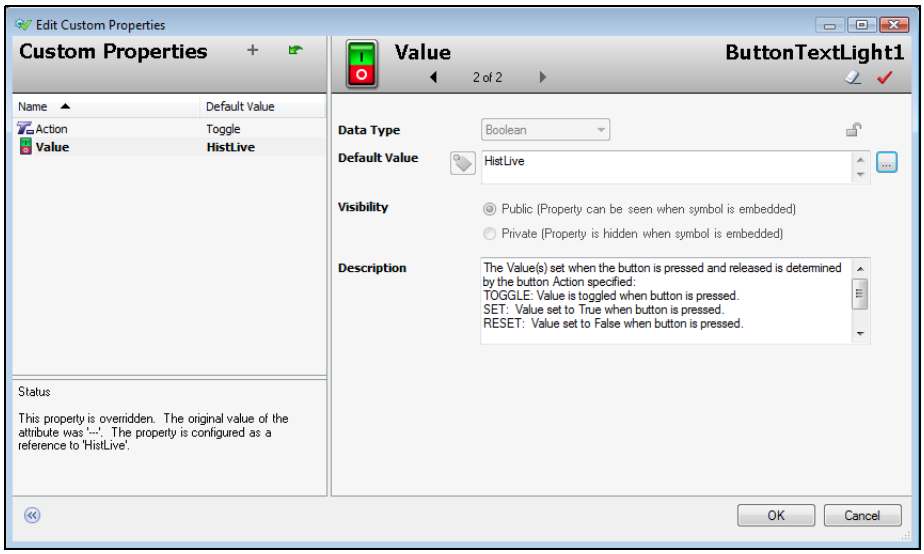
14. Embed **Buttons\ButtonTextLight**.



15. Place the button to the right of the **Hist Trend w/Scooters and Scale** wizard.



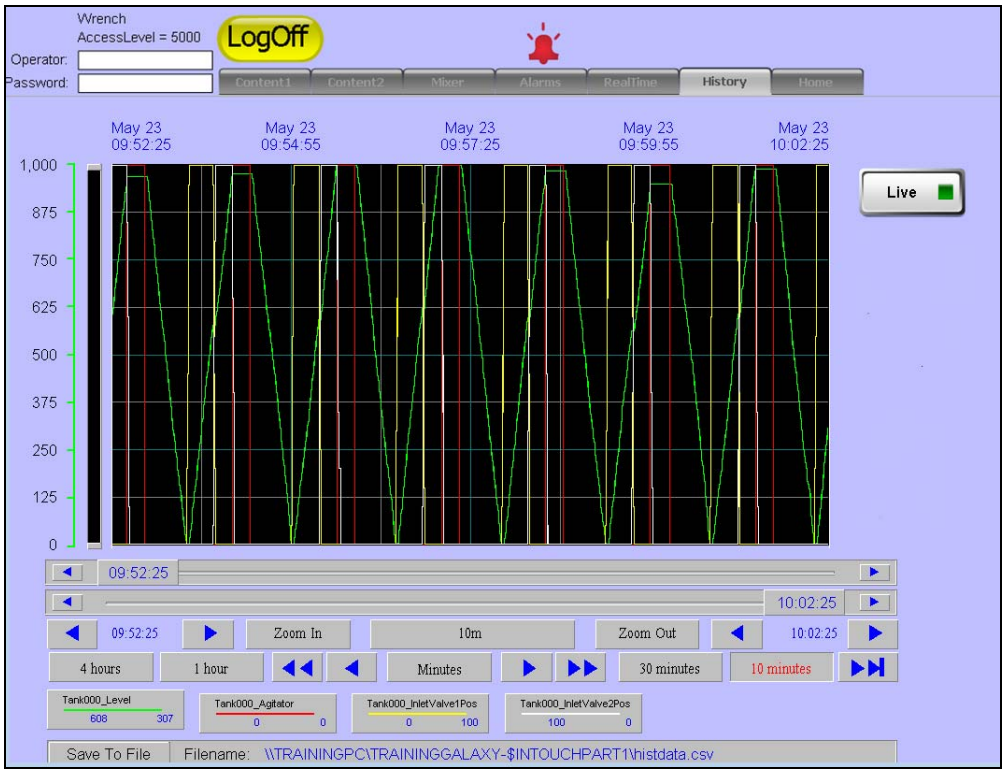
16. Change **LABEL** to **Live**.
17. Double-click the **Live** button and link **Value** to the **HistLive** tag.



18. Click **OK**.

Test the Live Button

19. Switch to **Runtime**.
20. Login as **Wrench**.
21. Navigate to the **History** window.
22. Click **Live**.



Notice that the trend chart updates to the current time. The trend chart continuously retrieves history and scrolls the chart once every second. Also notice that the script overrides most of the navigation features of the wizard.

23. Click the **Live** button again to disable the continuous historical trend updates.

Lab 22 – Changing Historical Pens at Runtime

Introduction

In this lab, you create scripts to clear and select pens for historical trends at runtime.

Objectives

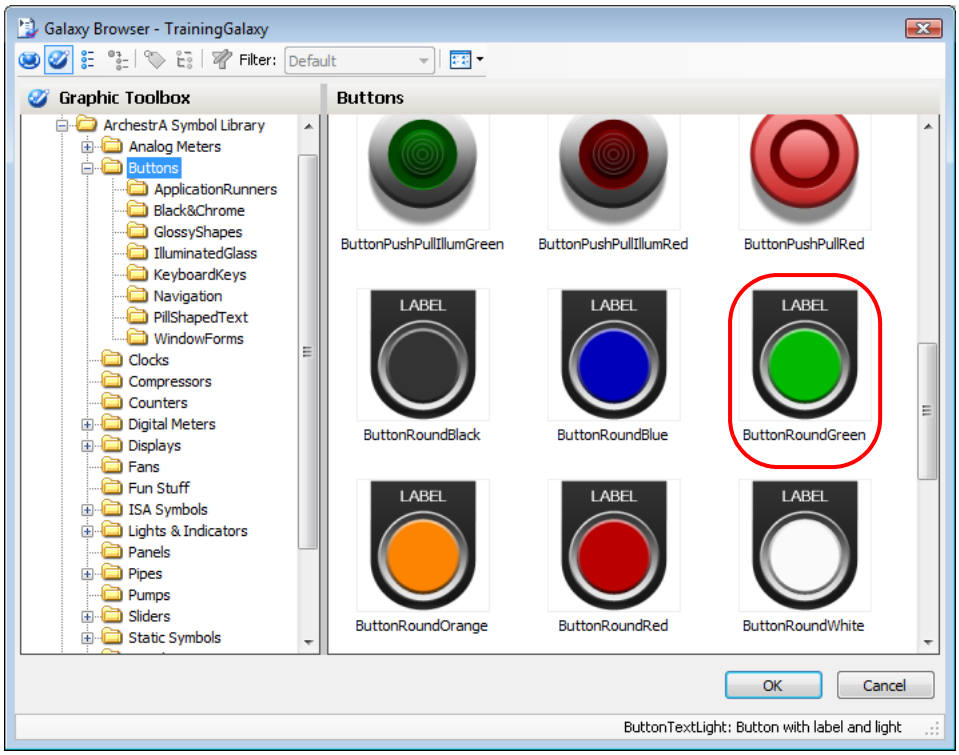
Upon completion of this lab, you will be able to:

- Use the HTSetPenName function to assign tags to historical pens at runtime
- Use the HTSelectTag function to select historical tags at runtime
- Create a script to clear historical pens at runtime
- Configure multiple scripts for one Action Script animation

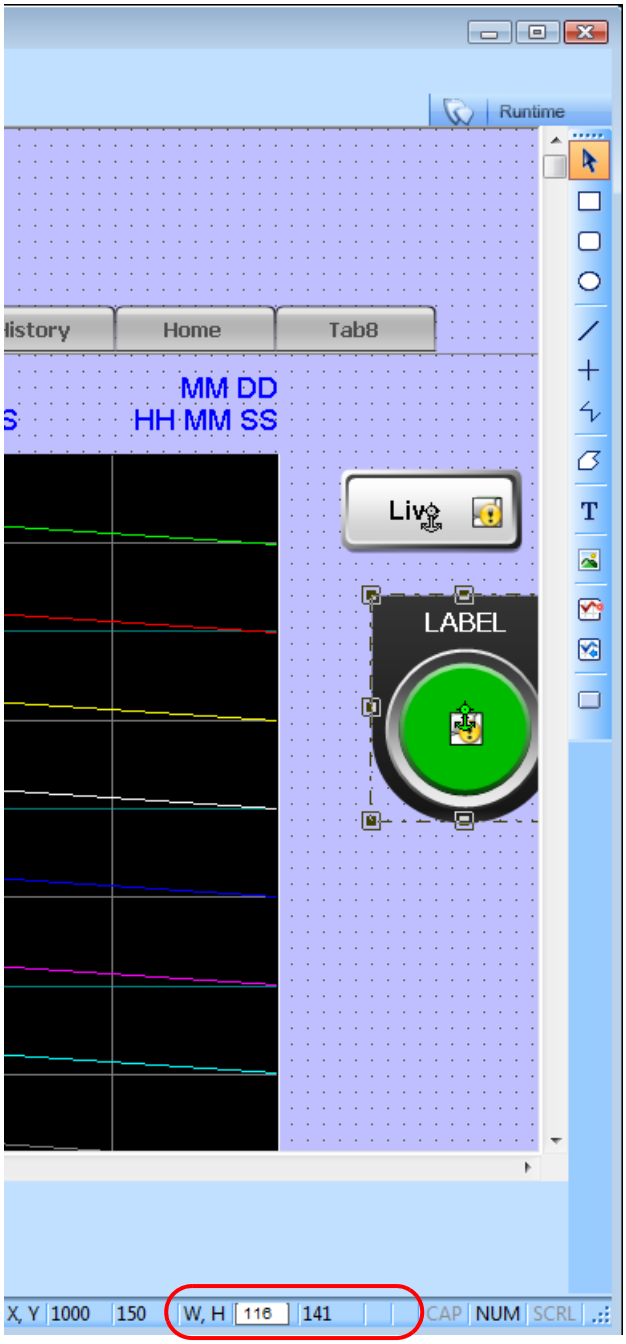
Create a Historical Pen Button

First, you add a button you use to clear and assign historical tags at runtime.

1. Close WindowViewer.
2. In WindowMaker, open the **History** and **Menu** windows.
3. Embed **ButtonRoundGreen** below the **Live** button.



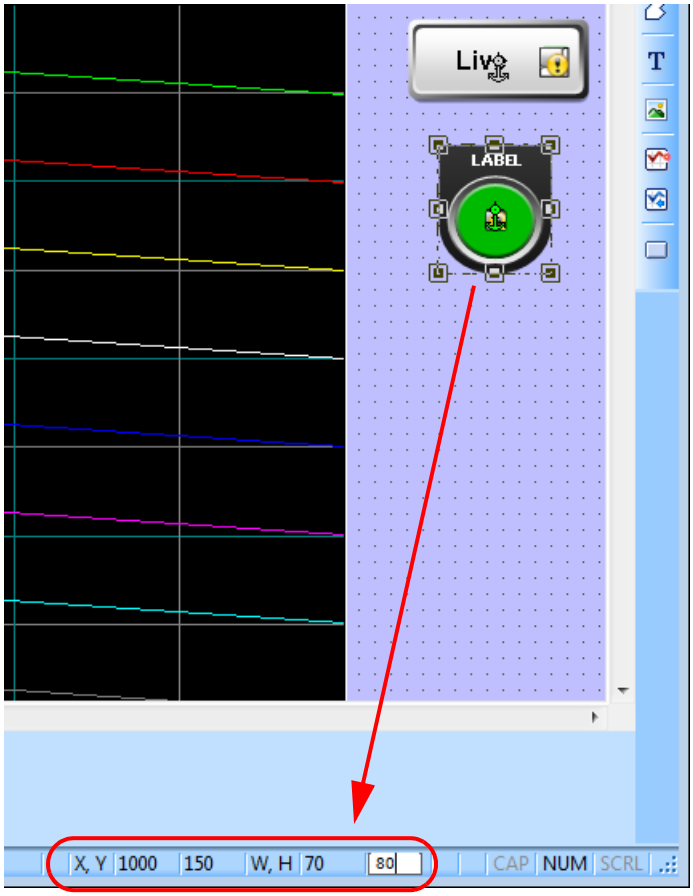
4. Click the graphic, then click in the **W, H** (width and height) area on the status bar to change the dimensions.



5. Enter the following dimensions:

   Width: 70
   Height: 80

Note: Press **Enter** after each dimension change to implement.

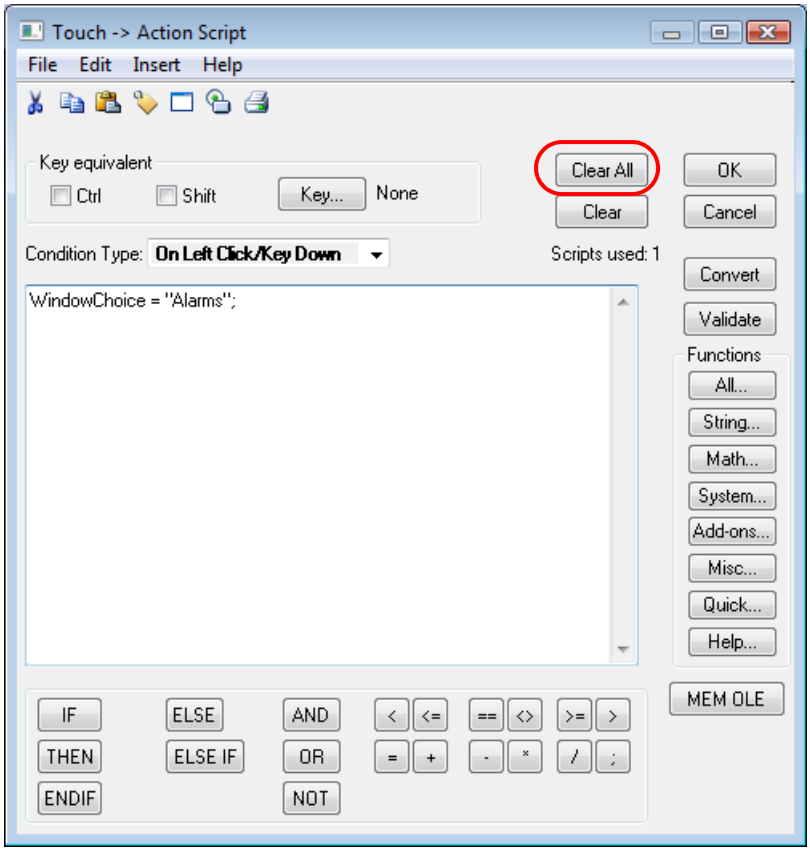


6. Change **LABEL** to **Pen1**.

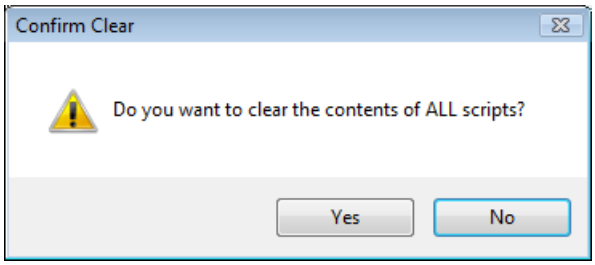
Next, you define an animation to clear a historical tag from the trend at runtime.

7. Right-click the **Pen1** button and select **Animation Links**.
8. In the **Touch Pushbuttons** area, click **Action**.

   The **Action Script** dialog box appears.
9. Click **Clear All** to remove any previous scripts.



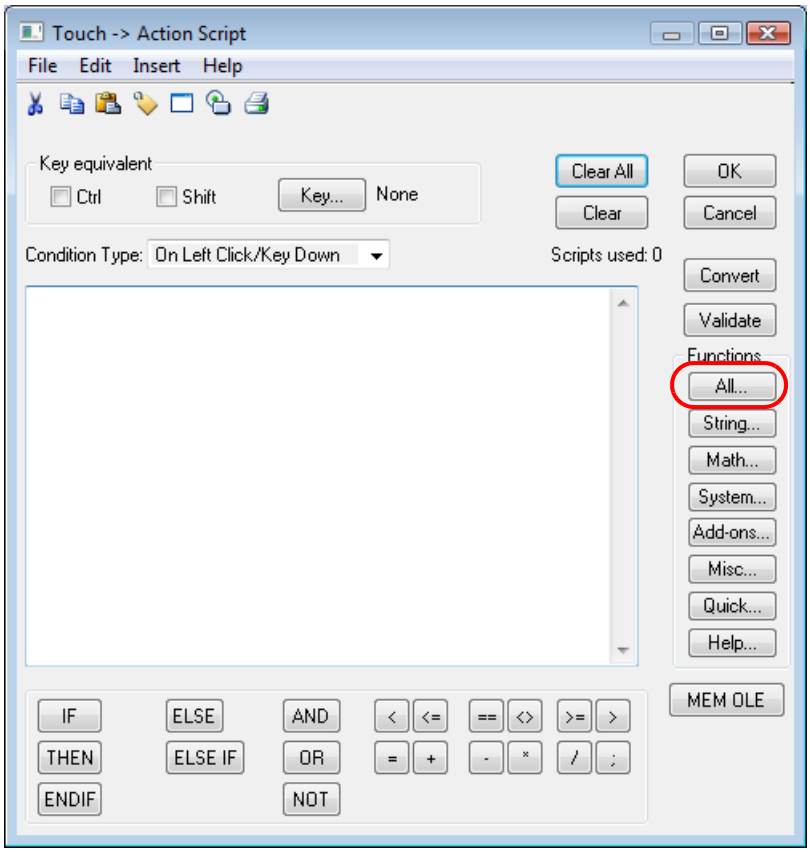
10. The **Confirm Clear** warning message appears.



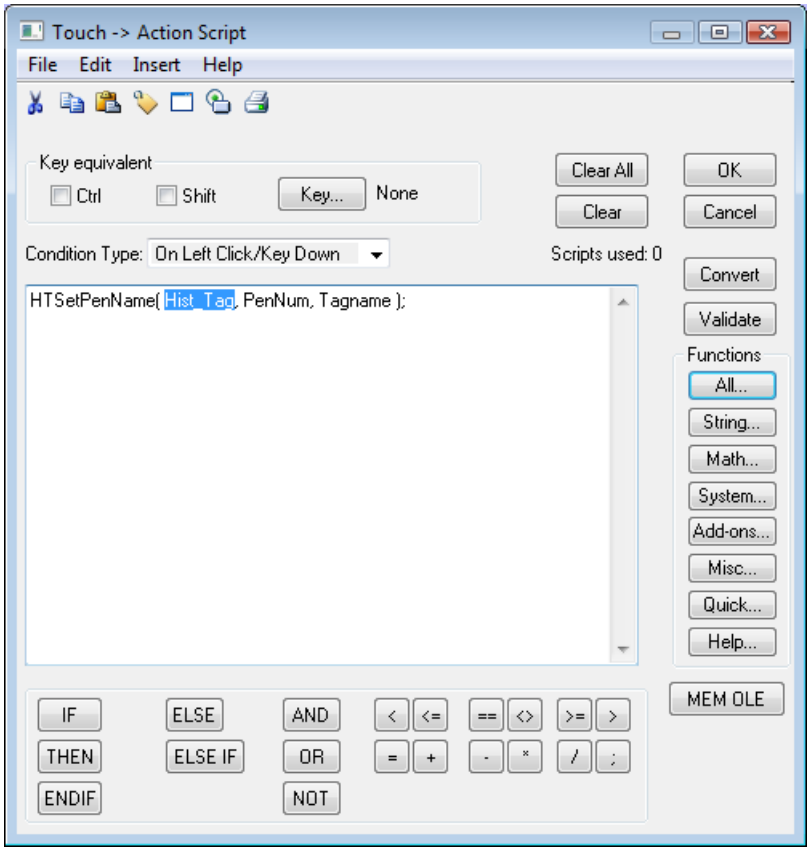
11. Click **Yes**.

Now, you create a script to open a dialog box at runtime for all **Log Data** enabled tags.

12. In the **Condition Type** field, verify that the **On Left Click/Key Down** option is selected.
13. Click the **All** button and select **HTSetPenName**.



The default script is placed in the dialog box.

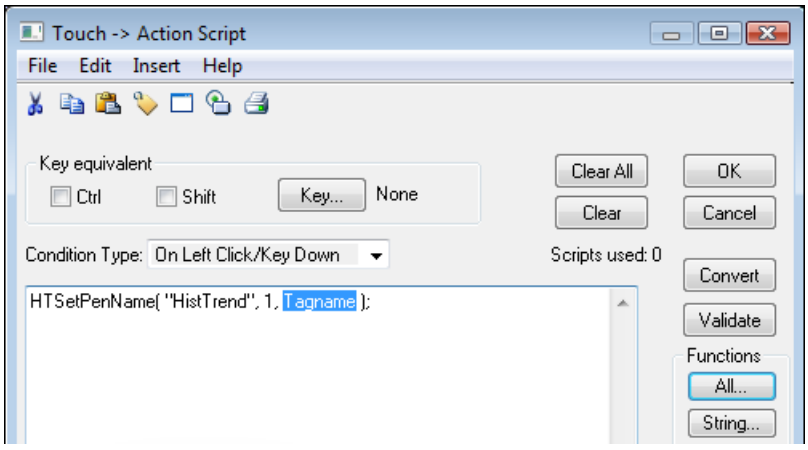


14. Modify the first two parameters as follows:

    Hist_Tag: "HistTrend"

    PenNum: 1

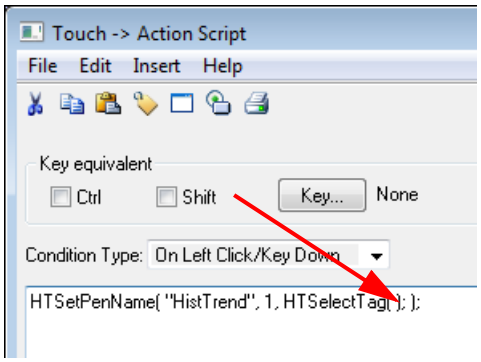
15. Highlight **Tagname**, then click the **All** button.



16. Select **HTSelectTag**.

    HTSelectTag( ); is inserted in your script.

17. Remove the extra semicolon between the close parentheses.

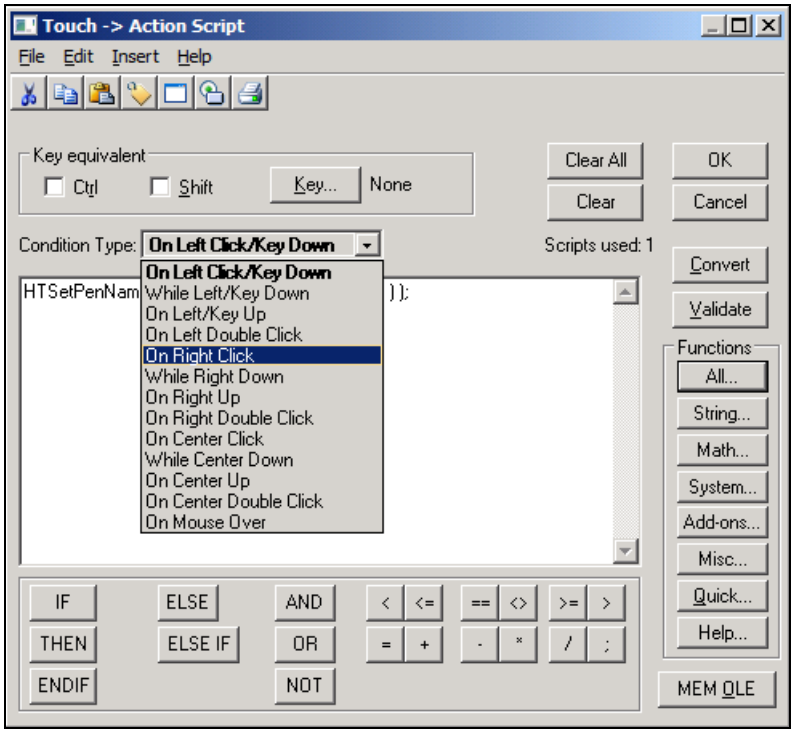


    The completed script looks like this:

    HTSetPenName("HistTrend", 1, HTSelectTag( ));

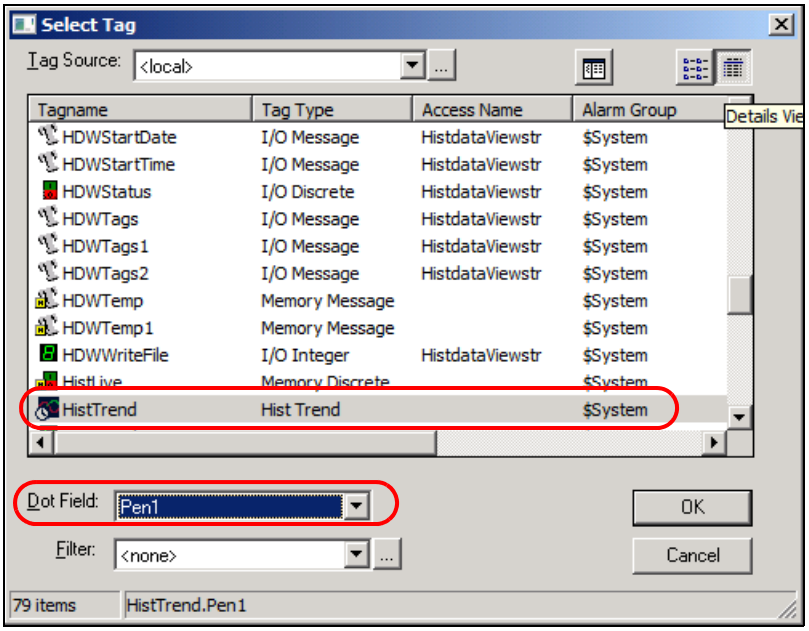
Next, you add a script to clear a pen selection.

18. In the **Condition Type** drop-down, click **On Right Click**.



19. On the toolbar, click **Insert Tag** .

20. In the **Select Tag** dialog box, click **HistTrend**, then from the **Dot Field** drop-down select **Pen1**.

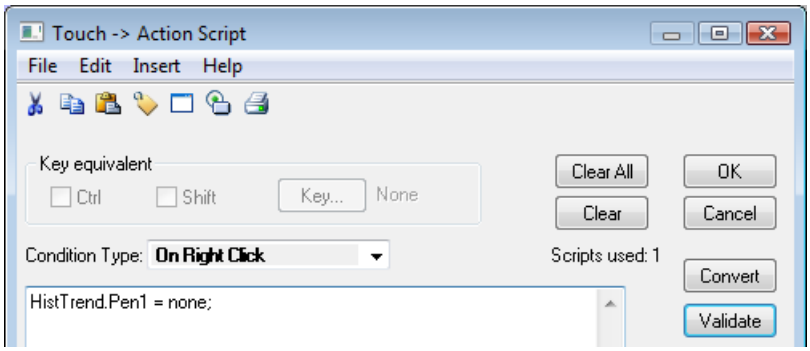


21. Click **OK**.

    HistTrend.Pen1 is inserted in the script area.

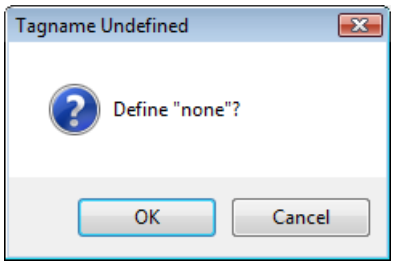
22. Add the following:

    = none;



23. Click **Validate**.

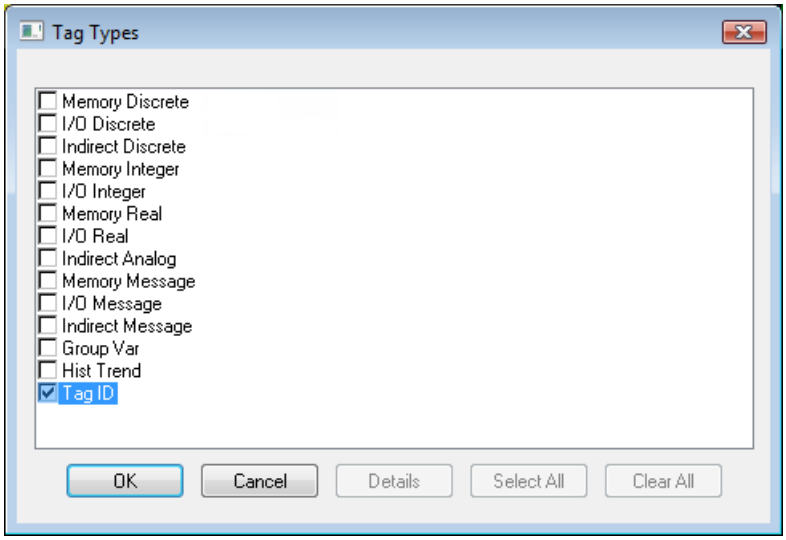
    You are prompted to **Define "none"**



24. Click **OK**.

    The **Tagname Dictionary** appears.

25. Click **Type**, then check **Tag ID**.

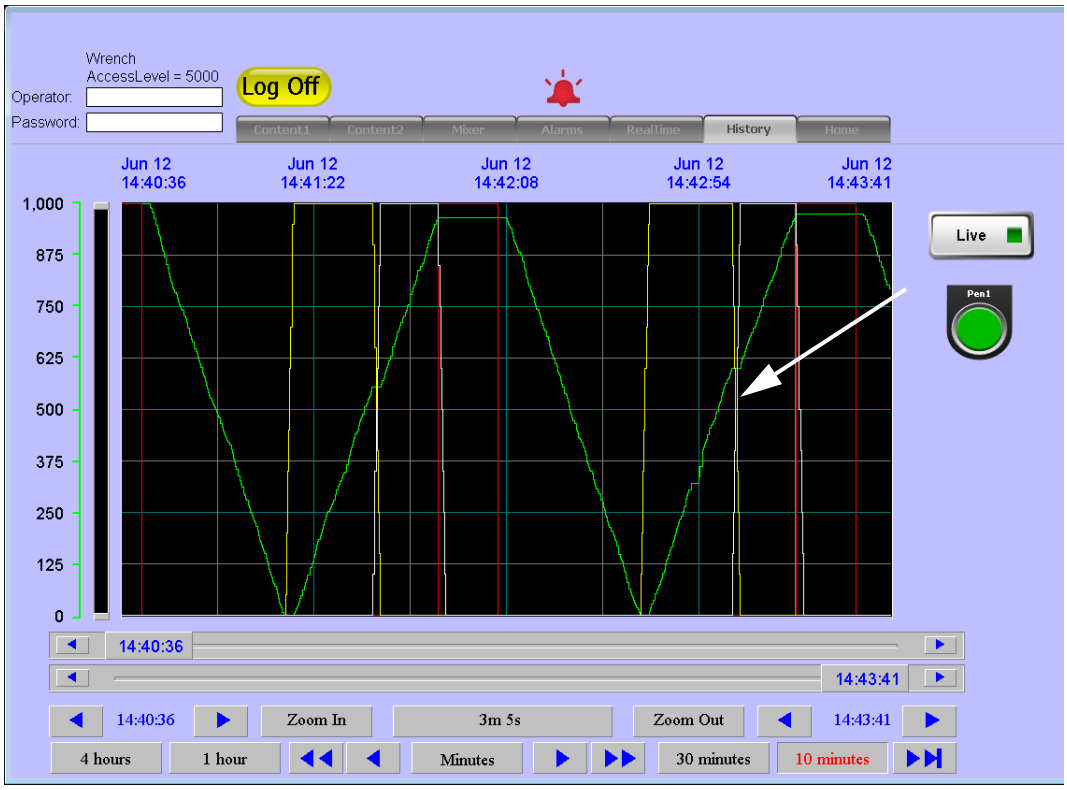


26. Click **OK**.
27. Click **Close**.
28. Click **OK** twice.

Test the History Trend Button

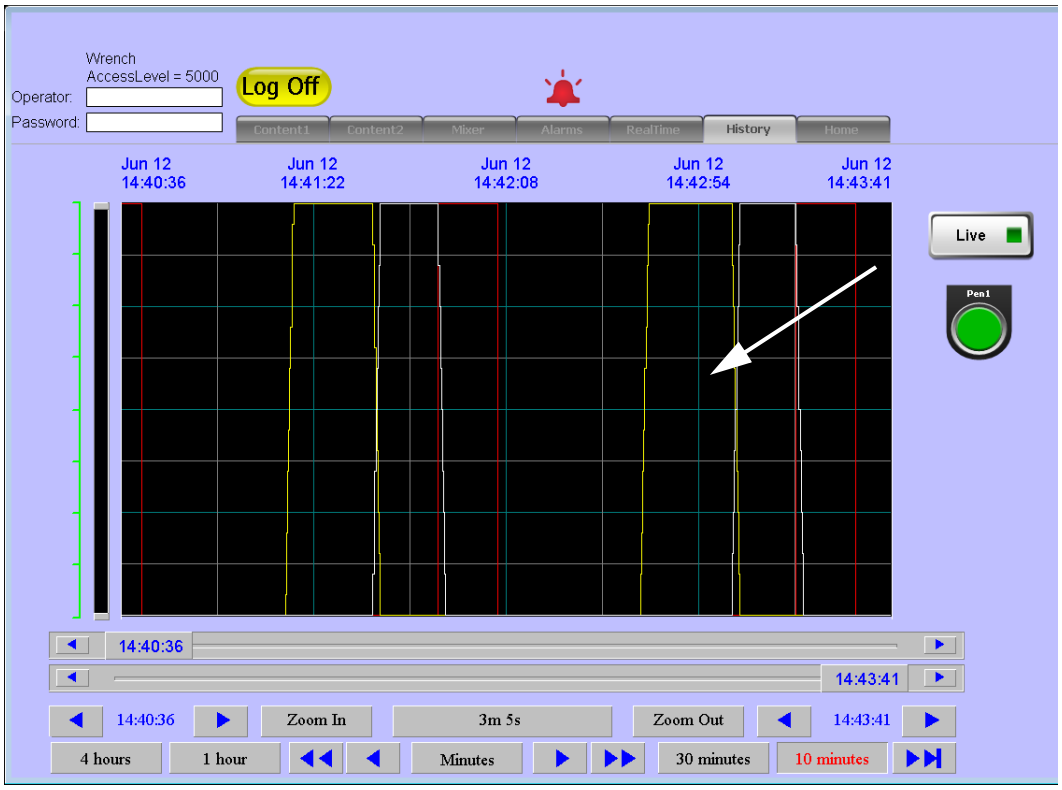
29. Switch to **Runtime**.
30. Log in as **Wrench**.

    On the **History** tab, notice the **Pen1** (green) trace on the chart.



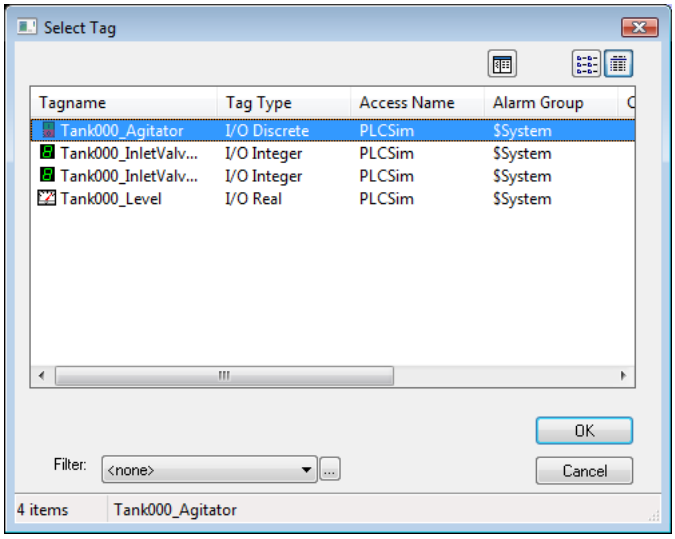
31. Right-click the **Pen1** button.

    Notice the **Pen1** trace is removed from the chart.



32. Click **Pen1**.

    The **Select Tag** dialog box appears showing only the historically logged tags.



33. Click **Tank000_Level**, then click **OK**.

    The **Pen1** trace is restored to the chart.

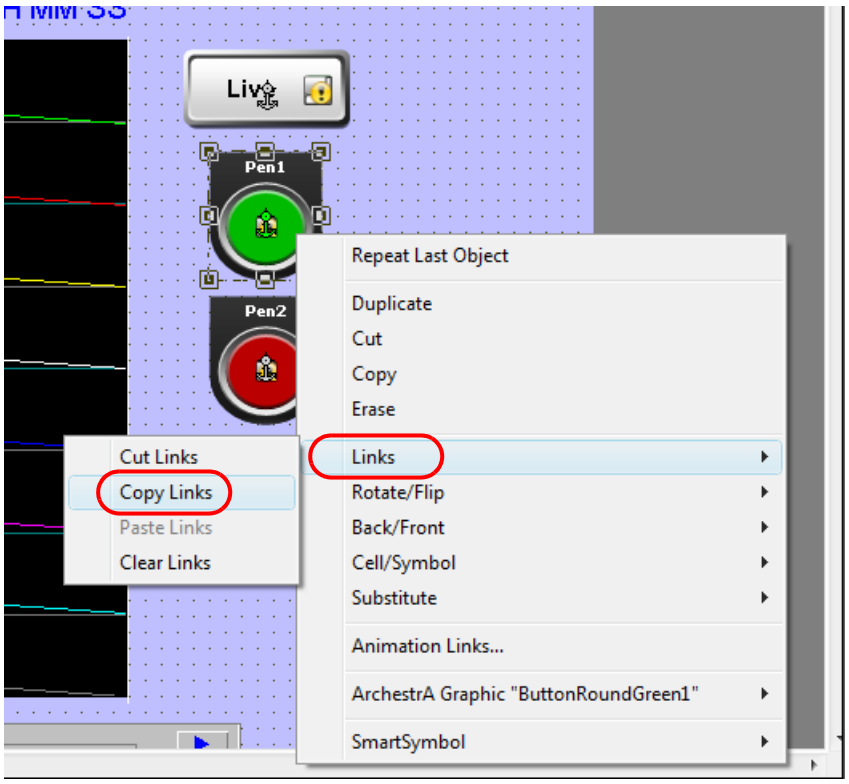
Copy Animations on Additional Pen Buttons

Perform these steps to copy animation links for the remaining history trend pens.

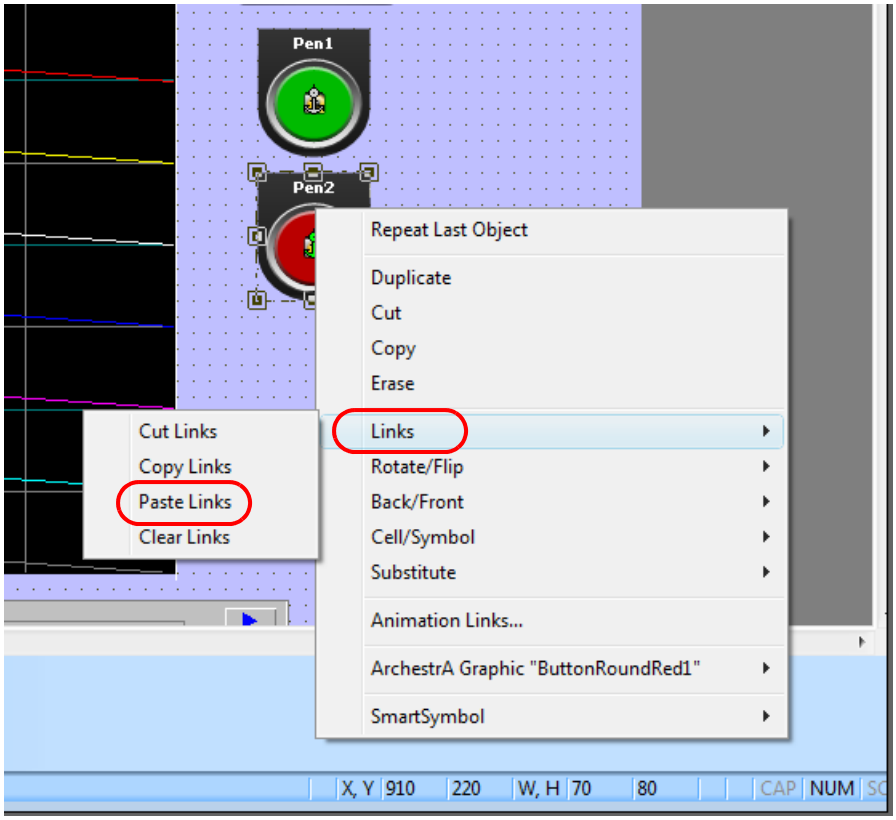
34. Return to **Development**.
35. Embed **ButtonRoundRed** below the **Pen1** button.
36. Change the **Width** to **70** and the **Height** to **80**, then rename the button **Pen2**.

> **Note:** **Ctrl+L** to substitute strings may not work if WindowViewer is running in the background. This is because the **EnableDisableKeys(1,0,1)** function may be disabling the **Ctrl** key. Either close WindowViewer or use the right-click menu option for **Substitute | Substitute Strings**.

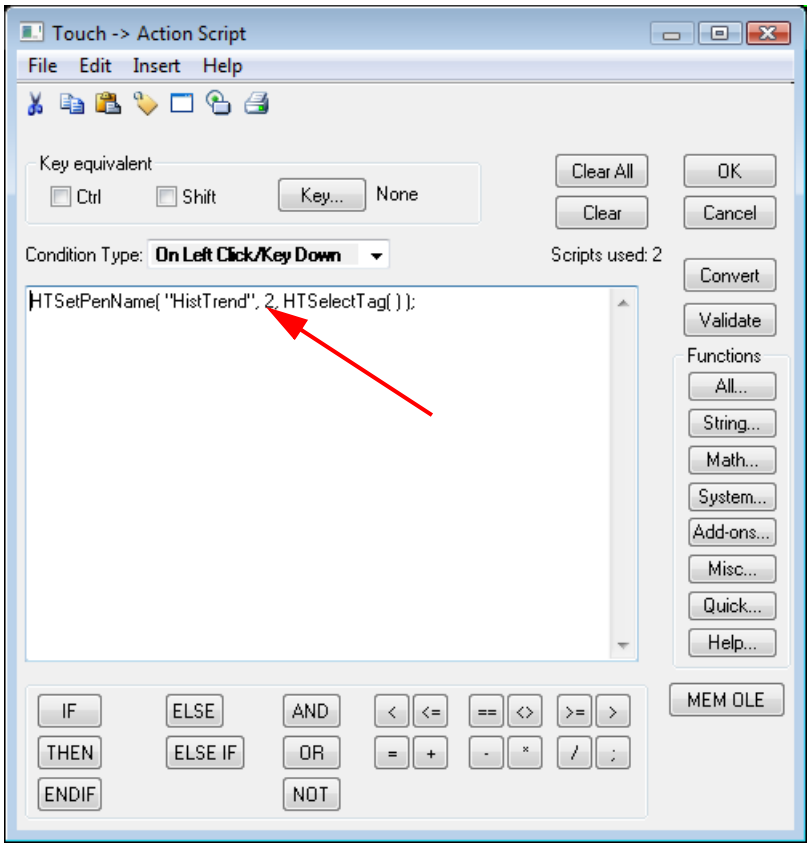
37. Right-click **Pen1** and select **Links | Copy Links**.



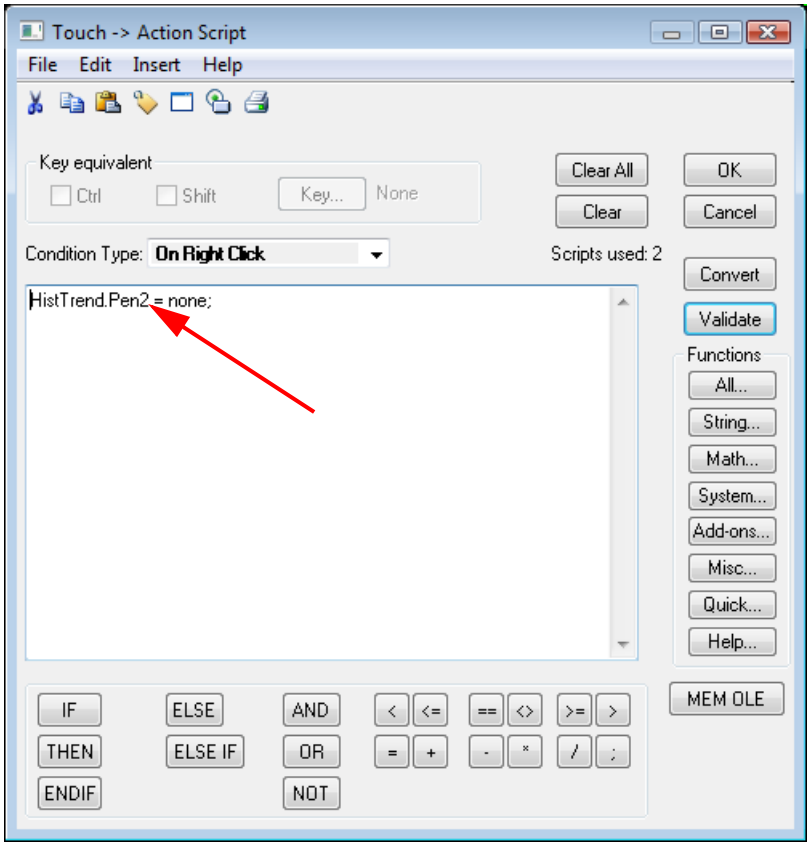
38. Right-click **Pen2** and select **Links | Paste Links**.



39. Right-click **Pen2** and select **Animation Links**, then click **Action**.
40. In the **On Left Click/Key Down** script, replace **1** with **2**.



41. In the **On Right Click** script, change **Pen1** to **Pen2**.



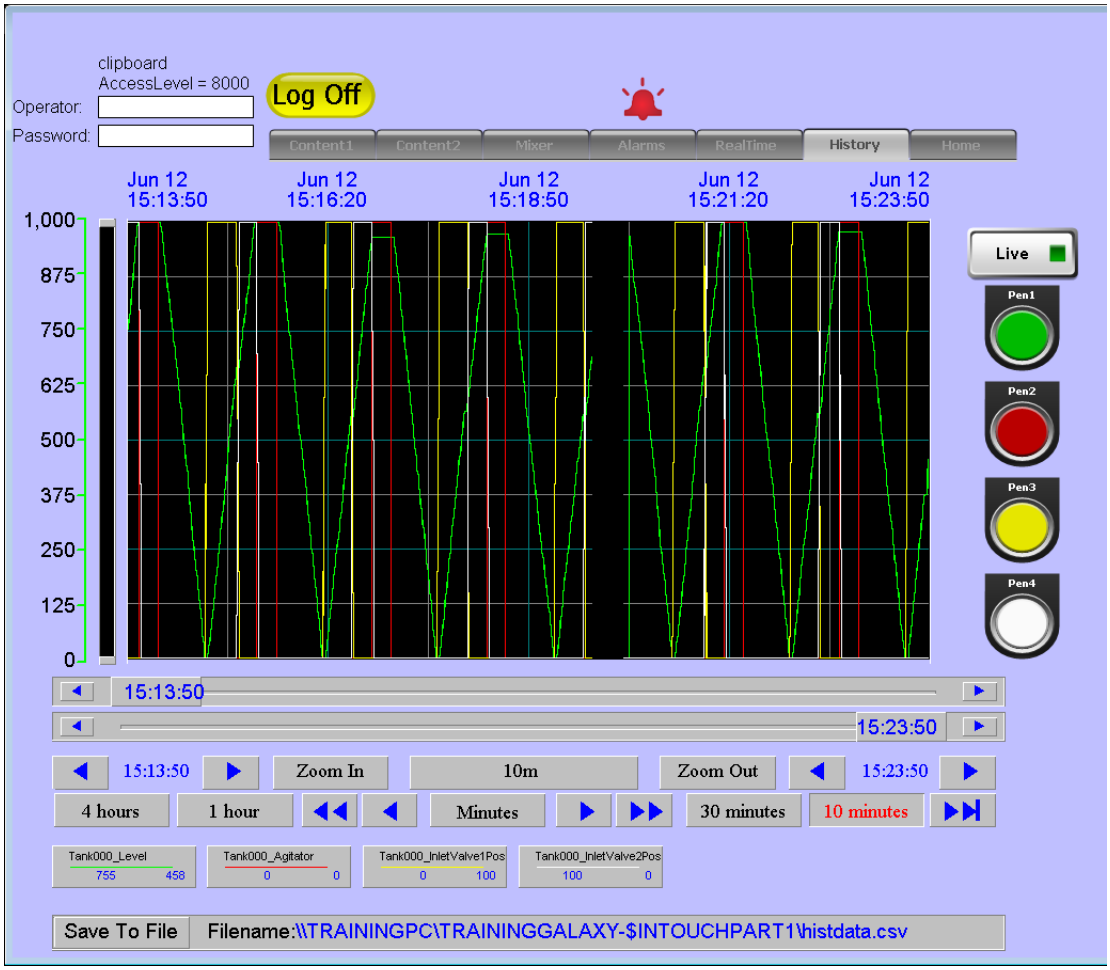
42. In the **Action Script** dialog box, click **OK**.
43. In the **Animation Links** dialog box, click **OK**.
44. Add two more buttons as follows:

Graphic	Label
ButtonRoundYellow	Pen3
ButtonRoundWhite	Pen4

45. Copy and paste animation links from **Pen1** to **Pen3** and **Pen4**.
46. Modify the **Action Script** as follows.

Pen	On Left Click/Key Down	On Right Click
Pen3	Change 1 to 3	Change Pen1 to Pen3
Pen4	Change 1 to 4	Change Pen1 to Pen4

47. Test in **Runtime**.



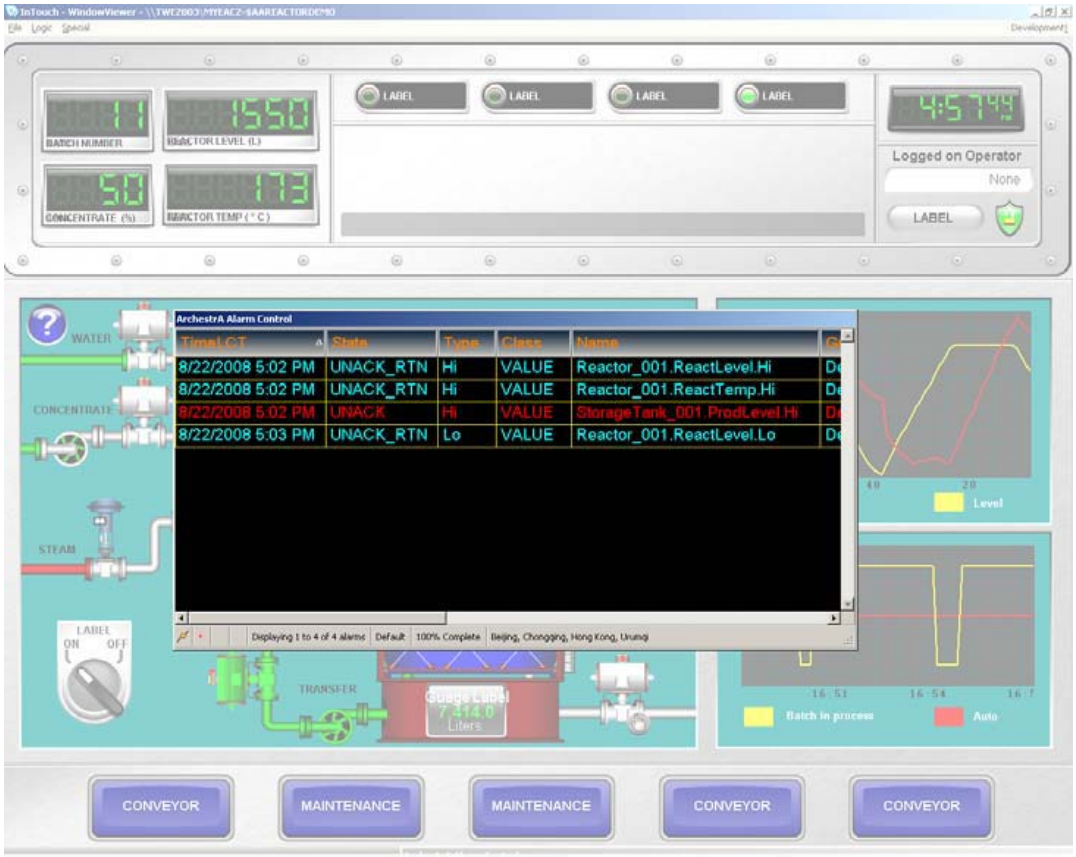
Section 4 – Enhancing Alarms with ArchestrA Symbol Scripting

This section describes symbol enhancements with ArchestrA symbol scripts. It also describes the use of comments to provide script line explanations, and how to restrict access to symbol functionality based on user login.

Overview

Many of the most powerful features of working with the ArchestrA Alarm Client are only accessible through ArchestrA Symbol scripts. For example, you can use the ArchestrA Alarm Client as a faceplate so that when the operator clicks somewhere in the application, it drills down detail in context with the item clicked, and an ArchestrA Alarm Client opens with specific alarms only for that area.

These powerful properties and methods (the ArchestrA script name for functions) are found in the list of properties for the Alarm Client. To access these from any ArchestrA script window, click the **Attribute Browser** icon  just above the script entry area. In the **Galaxy Browser** dialog box, select the **Element Browser** tab and from the elements list in the left panel select the **Alarm Client** element. Then in the right pane, select the property or method desired. Upon clicking **OK** the reference is pasted into the script window.



For additional properties and methods not seen in the basic list, click the show all check box at the bottom of the right pane.

You can also configure the ArchestrA Alarm Control to interact with other ArchestrA symbols and elements or element groups, by mapping its properties to Alarm Client attributes and symbol elements.

You can configure the Alarm Control to require an alarm to be acknowledged even if the condition causing the alarm has passed. This ensures that an operator is aware of events that caused a temporary alarm state but have returned to normal. You acknowledge alarms at runtime using a shortcut menu or through script methods.

Alarm Queries

The Alarm Control supports the standard InTouch alarm query formats, such as:

\galaxy!Area_001

\intouch!Group_A

The following table shows examples of alarm queries:

AlarmQuery	Description
\provider!group	Shows all alarms from the given provider and group. For example: \intouch!Group_A
\provider!group!tagname	Shows all alarms from the given provider, group and tag. For example: \galaxy!Mixing_Area!RotorCtrl
\\node\provider!group	Shows all alarms from the given provider and group from a given node. For example: \\remote\intouch!Group_B
\\node\provider!group!tagname	Shows all alarms from the given provider, group and tag from a given node. For example: \\grnode\galaxy!Packaging_Area!Wrapper1

Section 5 – Enhancing Symbols with ArchestrA Symbol Scripting

This section builds upon the Application and Action Script functions from earlier sections, and discusses elements of script to alter height and width properties at runtime.

Overview

The Alarm Client unites the display features of the InTouch Alarm Viewer control and the historical research concept of the InTouch Alarm DB View control. The Query Favorites of InTouch Alarm Viewer control defines a set of alarm provider, alarm group, an optional node name, and a priority range under one name. The alarm provider, alarm group, and the node name are used for subscribing to a specific alarm group. The priority range on the other hand is used to filter the alarms from the given alarm group.

The Filter Favorites of InTouch Alarm DB View control define a set of any number of criteria you want to filter from the Alarm Database under one name. In summary, Filter Favorites fulfill a purely filtering function, whereas, Query Favorites fulfill a subscription and a filtering function at the same time. The Alarm Control filtering feature unites both these concepts by exclusively using filter conditions and subscribing to the necessary alarm providers on demand. The filter conditions can be reused between different client modes. For example, if you define node name, provider name, alarm group, and a priority range for the current alarms, you can also use this filter to retrieve the historized alarm data of the same source from the Alarm Database instead.

Alarm Queries to Query Filters Translation, as with InTouch alarm controls, you can define queries for current alarms in the \\node\provider!group format, but they are translated by the Alarm Control to a filter after you save. For example, the query string \\GRNode\galaxy!MixingArea is translated to the following filter string:

```
Node = 'GRNode' AND Provider='galaxy' AND Group='MixingArea'
```

Through scripts you can control individual parts of filtering concepts, such as modifying the filter in a tree to query only alarms in the priority range 1 to 250, such as:

- Node = 'GRNode'
- Provider = 'Galaxy'
- Group = 'MixingArea'
- Priority >= '1'
- Priority <= '250'

Alarm Control Properties

This section describes some of the properties and methods of the Alarm Client which are discussed in this course.

AckComment.DefaultValue Property

The AckComment.DefaultValue property is a read-write string property that gets or sets the default acknowledgment comment when the AckComment.UseDefault property is TRUE.

Syntax

```
result = AlarmClient.AckComment.DefaultValue;

AlarmClient.AckComment.DefaultValue = ackComment;
```

Example

```
AlarmClient1.AckComment.UseDefault = 1;

AlarmClient1.AckComment.DefaultValue = "This alarm is acknowledged by John
Smith";
```

Remarks

For more information, see the “Showing Current Alarms” or “Recent Alarms and Events” section.

AckComment.UseDefault Property

The AckComment.UseDefault property is a read-write Boolean property that gets or sets the usage of the default acknowledgment comment.

Syntax

```
result = AlarmClient.AckComment.UseDefault;

AlarmClient.AckComment.UseDefault = useComment;
```

Example

```
AlarmClient1.AckComment.UseDefault = 1;

AlarmClient1.AckComment.DefaultValue = "This alarm is acknowledged by John
Smith";
```

Remarks

For more information, see the “Showing Current Alarms” or “Recent Alarms and Events” section.

AlarmQuery Property

The AlarmQuery property is a read-write string property that gets or sets the alarm query.

Syntax

```
result = AlarmClient.AlarmQuery;

AlarmClient.AlarmQuery = AlmQry;
```

Parameters

AlmQry - Alarm query string in format \\node\provider!group where node is optional.

Example

```
AlarmClient.AlarmQuery = "\intouch!GroupA";
```

Remarks

After you write a new value to the AlarmQuery property, the Alarm Control is updated. If you are using the default query filter, the query is updated with the new node, provider, and group name.

ClientMode Property

The ClientMode property is a read-write integer property that gets or sets the client mode for the Alarm Control. Use one of the following values:

Value	ClientMode
1	Current Alarms
2	Recent Alarms and Events
3	Historical Alarms
4	Historical Events
5	Historical Alarms and Events

Syntax

```
result = AlarmClient.ClientMode;
AlarmClient.ClientMode = clientMode;
```

Example

```
AlarmClient1.ClientMode = 2;
LogMessage("Alarm client set to Recent Alarms and Events");
```

ContextMenu.AckAll Property

The ContextMenu.AckAll property is a read-write Boolean property that gets or sets the appearance of the Ack All option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckAll;
AlarmClient.ContextMenu.AckAll = AckAllVis;
```

Remarks

For more information, see “Configuring the Runtime Shortcut Menu” section.

ShowContextMenu Property

The ShowContextMenu property is a read-write Boolean property that gets or sets the ability to open the shortcut menu at run time.

Syntax

```
result = AlarmClient.ShowContextMenu;
AlarmClient.ShowContextMenu = ContxtMnuAvail;
```

Ack.All() Method

The Ack.All() method acknowledges all alarms in the Alarm Control, including those not shown.

Syntax

```
AlarmClient.Ack.All(AckComment);
```

Parameters

AckComment - A string indicating the alarm acknowledgment comment.

Example

```
AlarmClient1.Ack.All("Alarm is acknowledged");
```

Ack.Selected() Method

The Ack.Selected() method acknowledges all selected alarms.

Syntax

```
AlarmClient.Ack.Selected(AckComment);
```

Parameters

AckComment - A string indicating the alarm acknowledgment comment.

Example

```
AlarmClient1.Ack.Selected("This selected alarm is acknowledged");
```

Requery() Method

The Requery() method refreshes the alarm records in the Alarm Control. For current alarms and recent alarms and events, the control re-queries the Alarm Manager. For historical alarms or events, the control retrieves alarm records from the Alarm Database.

Syntax

```
AlarmClient.Requery();
```

Reset() Method

The Reset() method resets column widths and the column order to their last known design-time settings. The Reset() method also resets the current query filter to the default query.

Syntax

```
AlarmClient.Reset();
```

Show.Context() Method

The Show.Context() method opens the shortcut menu at runtime. This method ignores the ShowContextMenu property setting and always shows the context menu.

Syntax

```
AlarmClient.Show.Context();
```

For a complete list of Methods and Properties for the Alarm Client Control, please reference the *ArchestrA Alarm Control Guide*, which can be found on any computer where InTouch is installed at **C:\Program Files\ArchestrA\Framework\Docs\1033\aaAlarmClientControl.pdf**

Section 6 – Troubleshooting QuickScript

This section describes QuickScript troubleshooting using debugging scripts. It also describes various techniques used to enable or disable portions of scripts using flags or comments.

Overview

You can troubleshoot QuickScripts by setting check points in a script you are having trouble with. You can also turn off script logic using flags by commenting out a script portion. For example, using the LogMessage() function at strategic locations in your script you could capture the current state of logic or tags, then afterward use Log Viewer to review these values.

Debug Scripts writes a message to the Wonderware Logger program each time a QuickScript is executed. You enable the **Debug** option on the WindowViewer menu bar from the **Configuration** options for WindowViewer properties.

The Log Viewer displays error messages in red-highlighted stripes. These indicate malfunctioning processes and should be resolved quickly. Warning messages are displayed in yellow stripes. These indicate major events in the system.

Turn Off Script Logic

You can turn off script logic using one of the following methods:

- Add a flag to a script to enable or disable a portion of the script. This takes the form of an IF THEN ENDIF; section to isolate a portion of your script. For example, to prevent the PumpShutdown tag from being changed in the flowing script you must set the value of the MyDebugModeEnable discrete tag to a value of 1:

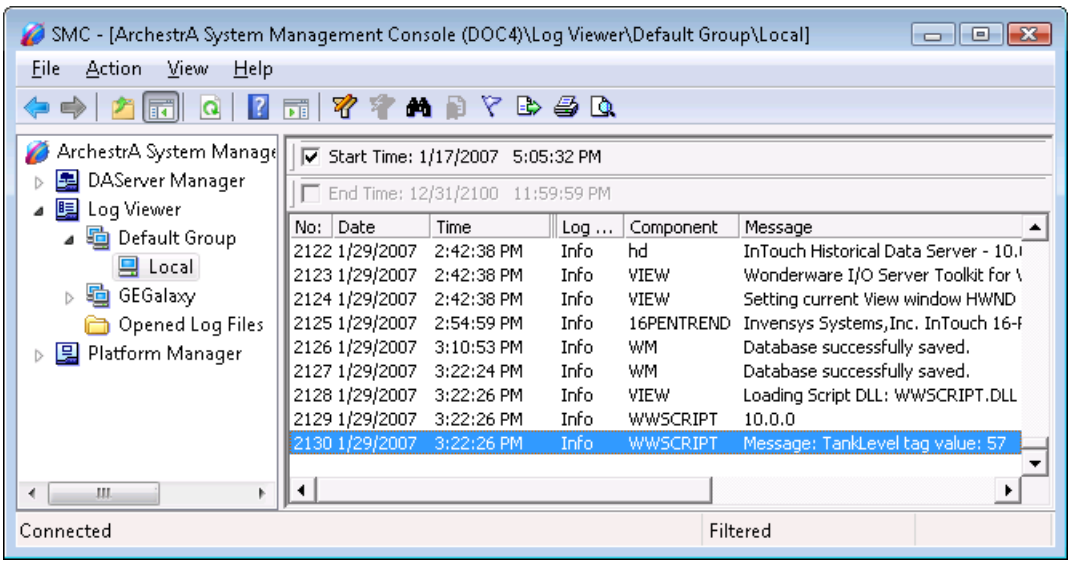
```
IF MyDebugModeEnable == 1 THEN
   {The following will shutdown the pump}
   PumpShutdown = 1;
ENDIF;
```

- Add comments to enable or disable a portion of a script

Note: At least 1 line of executable script must be used. You cannot comment out an entire InTouch Script.

Logger

The Logger is the background process that stores messages and events in the system. This process occurs without any prompting from or interaction necessary from the user. The Logger is installed as a system service, and can be configured to be an Auto Service or Manual Service. As an Auto Service utility, the Logger is started when the PC is booted up. As a Manual Service utility, the Logger requires a manual start through the System Services utility in Windows Control Panel. In either case, the logging process is always started when an ArchestrA component begins functioning.



Log Viewer

The Log Viewer is a utility used to view error and informational messages that are sent by ArchestrA components, such as InTouch. The look and feel and the format of the user interface can be customized to suit individual needs.

When Log Flags are generated they are passed to the Logger where they are stored in the Log Message Storage. They are available for viewing by the LogFlag Viewer. You use the LogFlag Editor to edit flags.

The Log Viewer is located in the ArchestrA System Management Console (SMC) and is installed when you install the InTouch HMI.

The Log Viewer is an ArchestrA utility that records information regarding the activity occurring on the computer. For example, it records start-up data, error conditions, and DAServer information. When running an ArchestrA application, the Logger runs as a system service. The Log Viewer is a Microsoft Management Console (MMC) snap-in that provides a user interface for viewing messages reported to the Logger. The MMC is a host or container utility for the Log Viewer with its own generic functions.

The **LogMessage** function used within your InTouch Scripts can be a significant tool for helping with troubleshooting scripts. By strategically placing LogMessage() functions in your scripts, you can determine the order of script execution, performance of scripts, and identify the value of attributes both before they are changed and after they are affected by the script. Thus, it may be used to find logic errors and script flow or timing issues. For greater detail on this, please see “How InTouch Scripting Works” found on the wdn.wonderware.com website.

By combining LogMessage with rules of string catenation, any string literal, message function, or message type tag (or string custom property) can be added to a message sent to the System Management Console's logger.

Each message posted to the Log Viewer is stamped with the exact date and time. The message always begins with the component Tagname.ScriptName so you can tell what object and what script within the object posted the message to the log; for example, LogMessage("Report Script is Running");

LogMessage() Function

Writes a user-defined message to the ArchestrA Log Viewer.

```
LogMessage("Message_Tag");
```

Parameter

Message_Tag - String to log to the Log Viewer. Actual string or message tagname.

Remarks

This is a very powerful function for troubleshooting InTouch scripting. By strategically placing LogMessage() functions in your scripts, you can determine the order of QuickScript execution, the performance of scripts, and identify the value of tags both before they are changed and after they have been affected by the QuickScript. Each message posted to the Log Viewer is stamped with the exact date and time.

Example

```
LogMessage("Report Script is Running");
```

The above statement would print the following to the Log Viewer:

```
94/01/14 15:21:14 WWSCRIPT Message:Report Script is Running.
```

Note: The **Logger** and **Log Viewer** are automatically installed when an ArchestrA component is installed.

Lab 23 – Enhancing Alarm Client with Scripting

Introduction

In this lab, you enhance your Alarm Client symbols with scripts. You then add security to the symbol to restrict access to some users.

Objectives

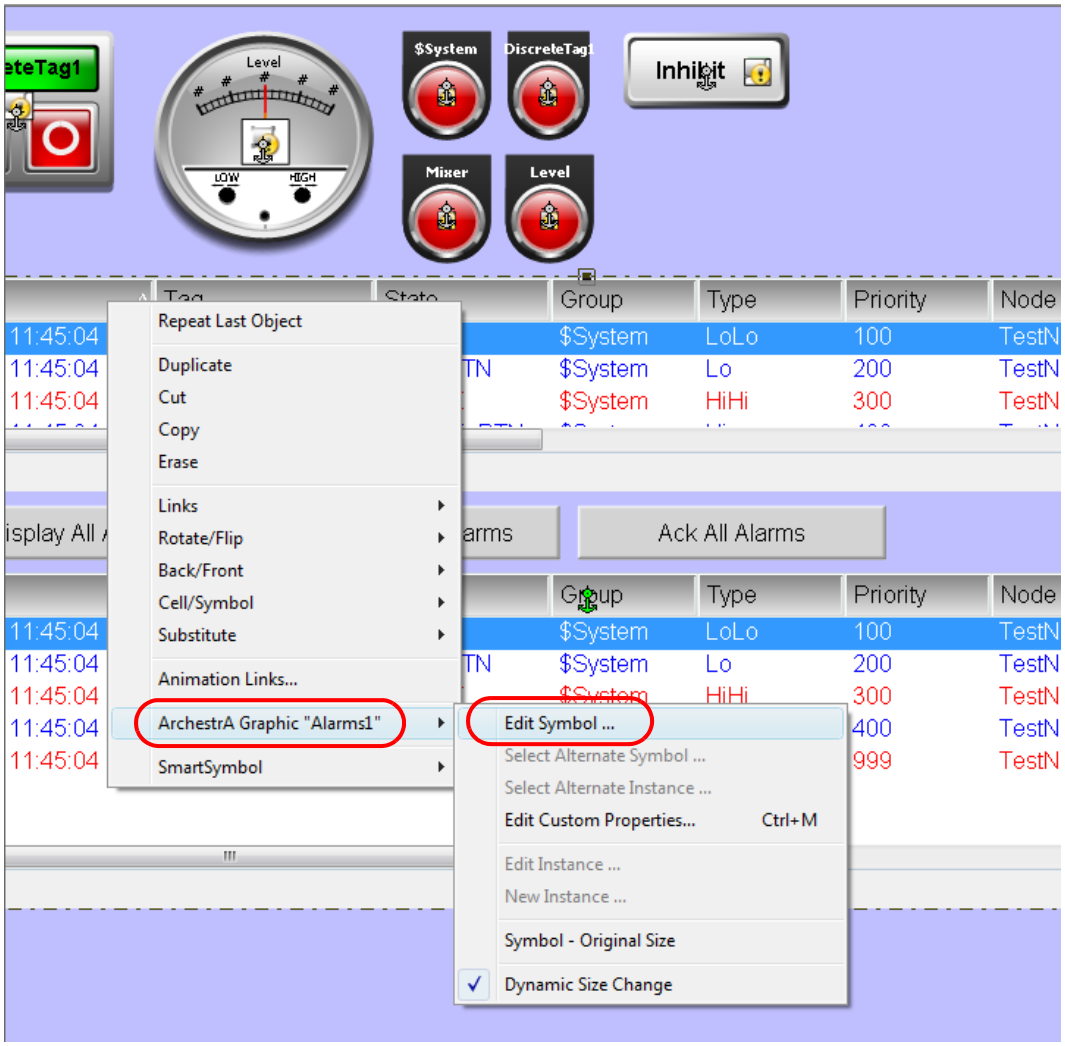
Upon completion of this lab, you will be able to:

- Enhance the Alarm Client with ArchestrA symbol script
- Use ArchestrA symbol script comments to document a script
- Restrict Alarm Client context menu access based on user login

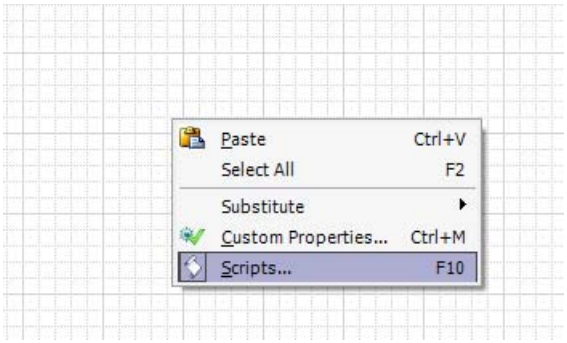
Configure a Predefined Script

First, you create a predefined script using methods and properties.

1. Close WindowViewer.
2. In WindowMaker, open the **Alarms** window.
3. Right-click the **Alarms** symbol and select **ArchestrA Graphic "Alarms1" | Edit Symbol** to open it in the symbol editor.

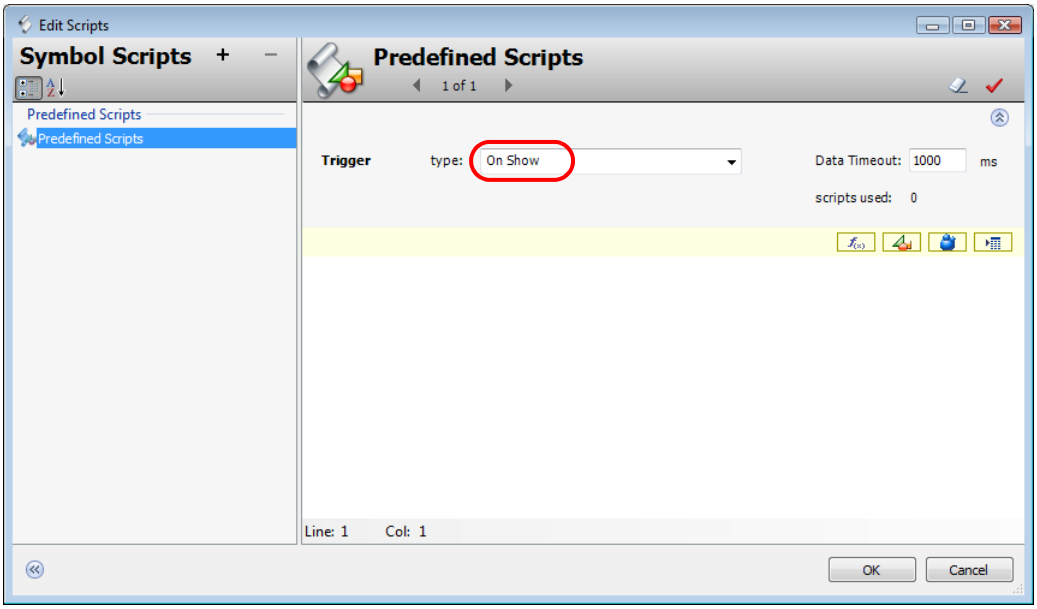


4. On the canvas, right-click a blank area and select **Scripts**.



The **Edit Scripts** window appears.

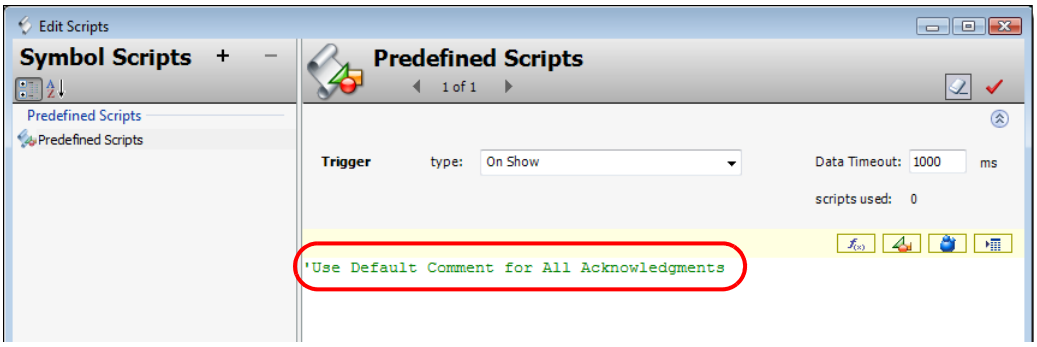
5. Click **Predefined Scripts** and verify the **Trigger type:** is set to **On Show**.



6. In the body of the script, enter the following comment:

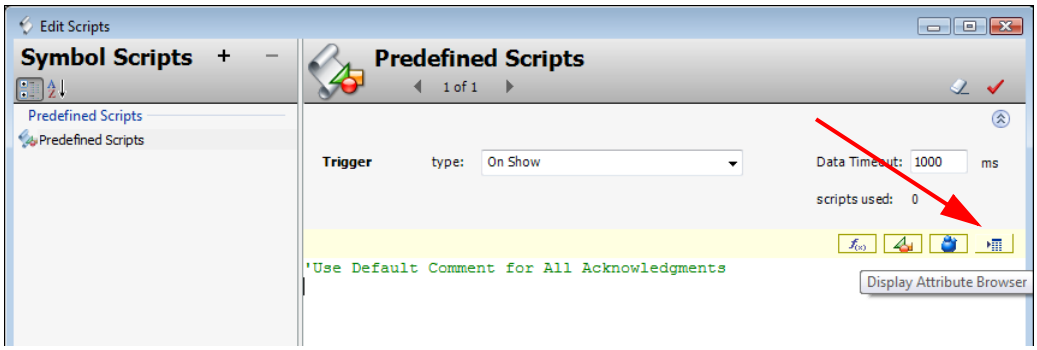
'Use Default Comment for All Acknowledgments

Notice the single quote at the beginning of the comment.

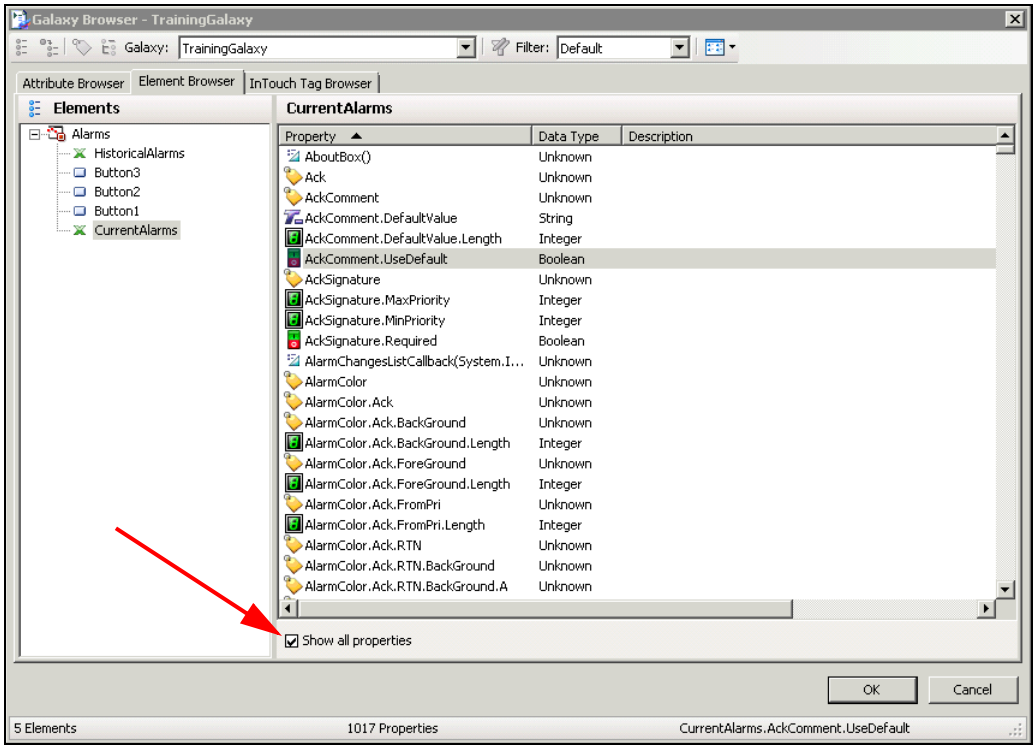


7. Press **Enter** to move the cursor to a new line.

8. Click **Display Attribute Browser** .



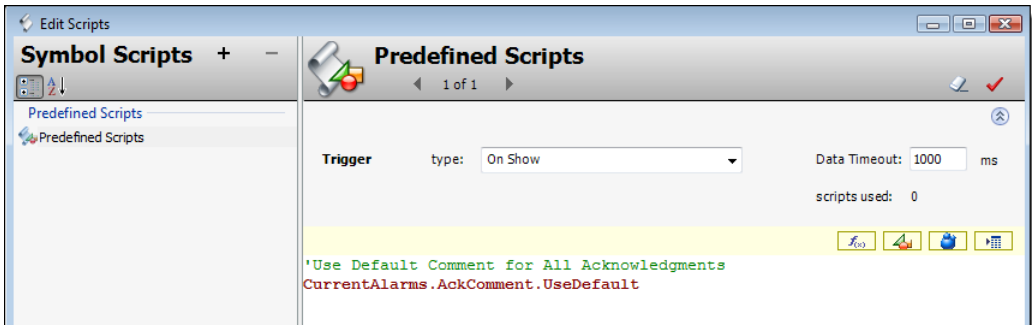
9. In **Galaxy Browser**, click the **Element Browser** tab, then click **CurrentAlarms**.
10. Click **Show all properties**.



A complete list of properties for **CurrentAlarms** appear.

11. Double-click **AckComment.UseDefault**.

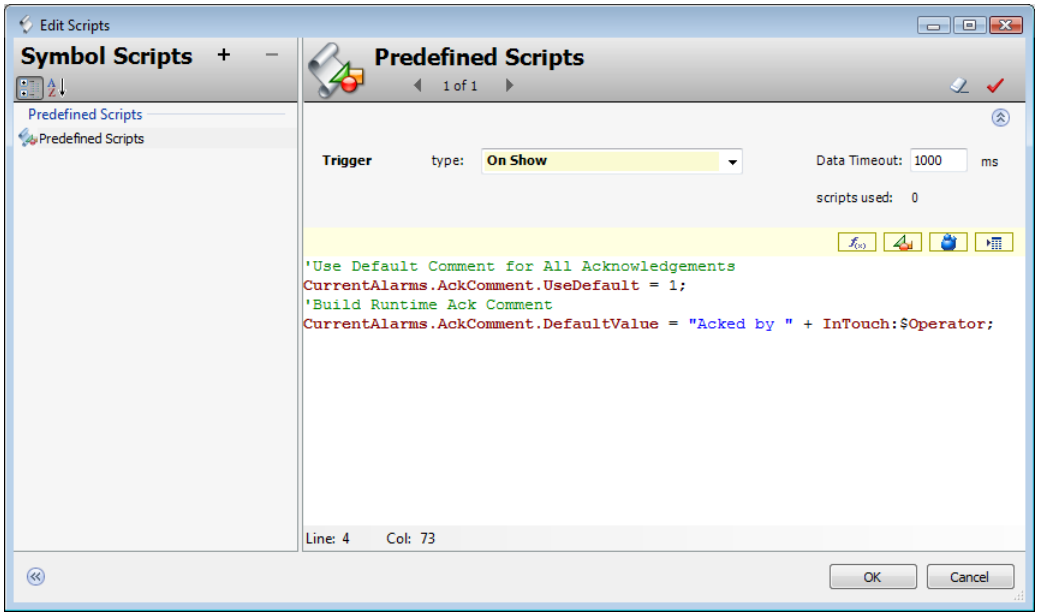
    The property is inserted in the script.



12. At the end of the script, enter **= 1;**, then press **Enter**.
13. Add the following additional comment:

    'Build Runtime Ack Comment
14. Press **Enter**.
15. Click **Display Attribute Browser** .
16. With the **Element Browser** selected, in the **Elements** pane, click **CurrentAlarms**.
17. Check **Show all properties**, then double-click **AckComment.DefaultValue**.
18. Add the following additional information.

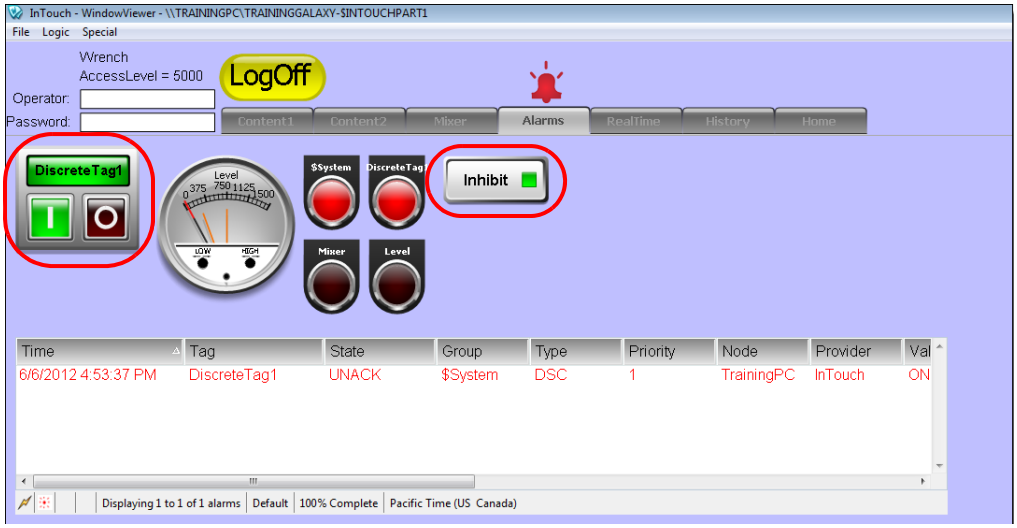
    = "Acked by " + InTouch:$Operator;



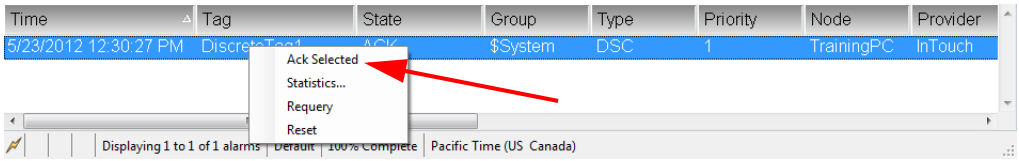
19. **Validate** the script.
20. Click **OK**.
21. Click **Save**.

Test the Alarm Client

22. Return to WindowMaker.
23. Ensure both the **Menu** and **Alarms** windows are open.
24. Switch to **Runtime** and log in as **Wrench**.
25. Navigate to the **Alarms** window, then click the **Inhibit** button.
26. Trigger a **Discrete Tag1** alarm.



27. Click the **DiscreteTag1** alarm record in the display.
28. Right-click the **DiscreteTag1** record and select **Ack Selected**.



Notice that the alarm changes from red to black, but you are not prompted to enter a comment.

29. Scroll to the right and expand the **AlarmComment** field.

    The comment indicates the alarm was acknowledged by **Wrench**.

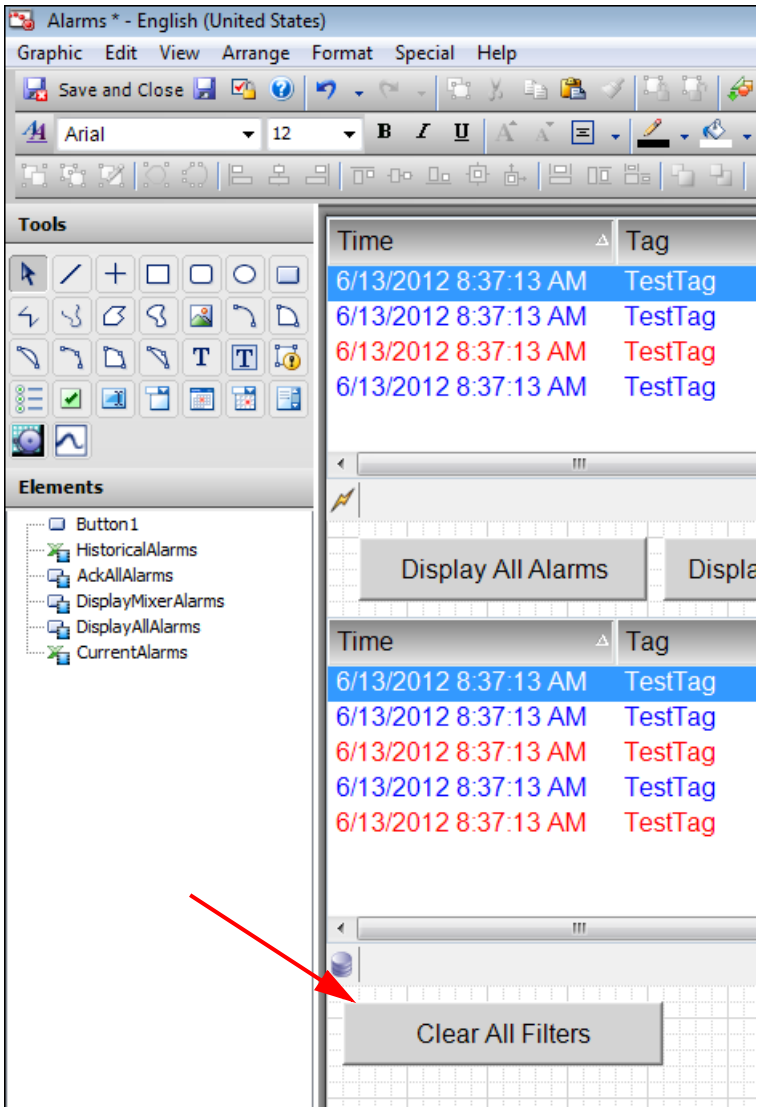


Note: The default comment feature does not apply to functionality on the **Ack All Alarms** button.

Add Scripts for the Historical Alarm Display

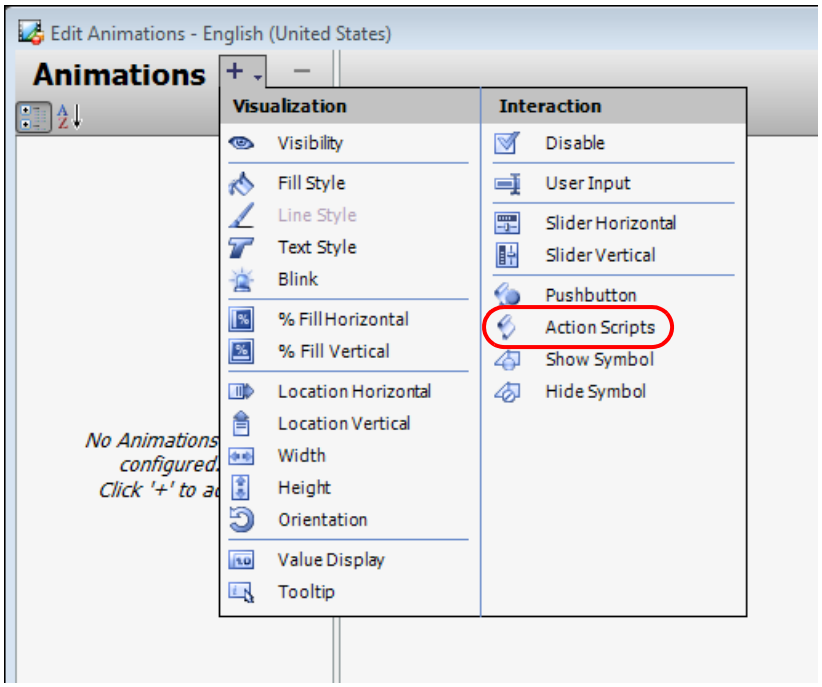
Next, you add a new button to clear existing filters.

30. Switch back to **Development**.
31. Open the **Alarms** symbol in the symbol editor.
32. Add a button below the **Historical Alarm Client** and change the button text to **Clear All Filters**.

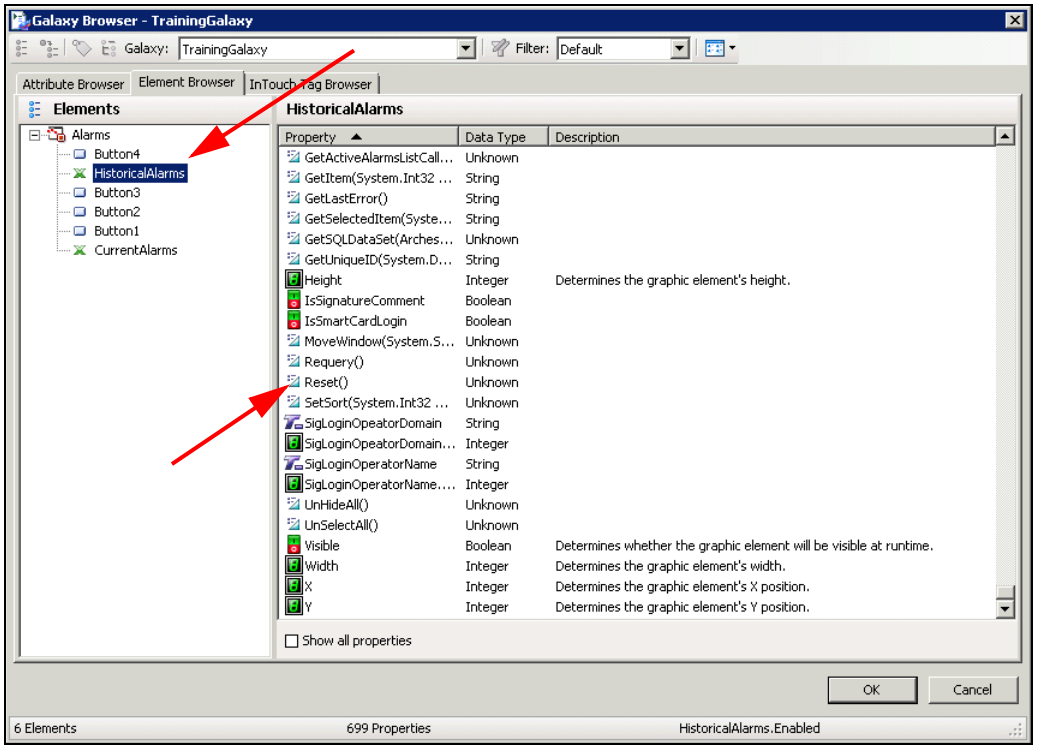


Adjust existing graphics, if necessary.

33. Double-click the **Clear All Filters** button to open the **Animations** window.
34. Click **+** and select **Action Scripts**.

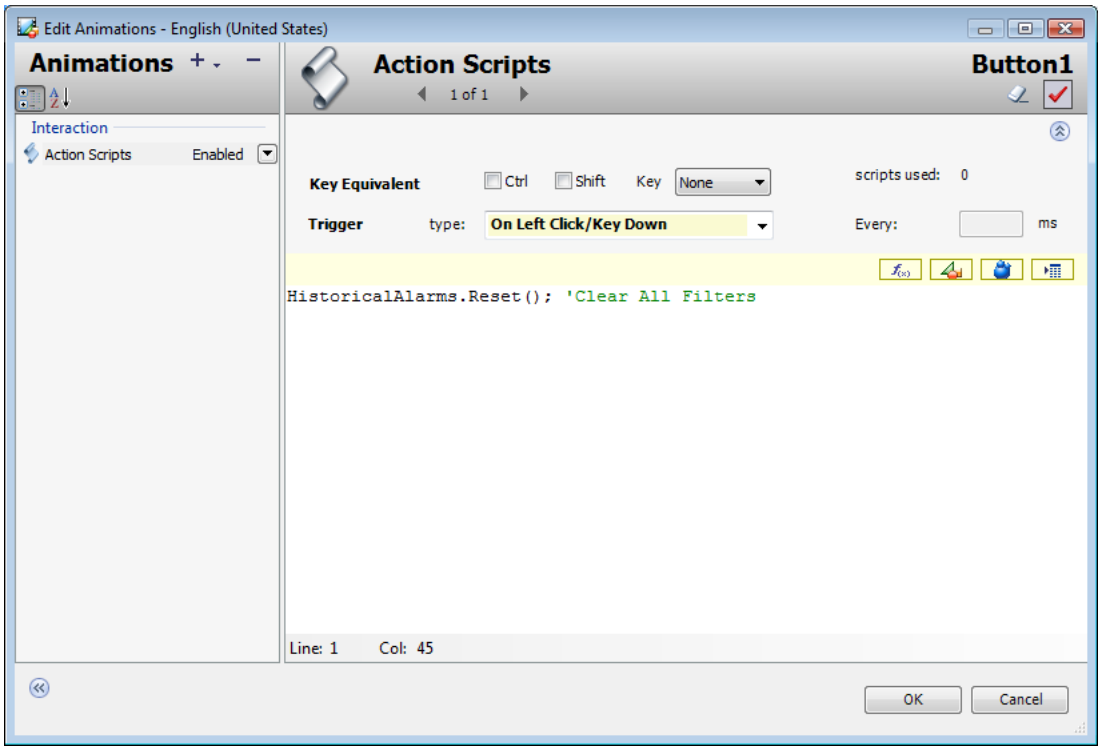


35. In the **Action Scripts** pane, click **Display Attribute Browser** .
36. On the **Element Browser** tab, add the **HistoricalAlarms.Reset()** method to your script.



37. Add the following to the end of the script.

    ; 'Clear All Filters

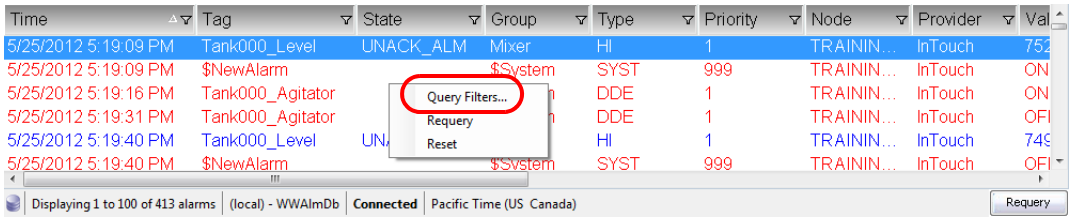


38. **Validate** the script.
39. Click **OK**.
40. Click **Save**.

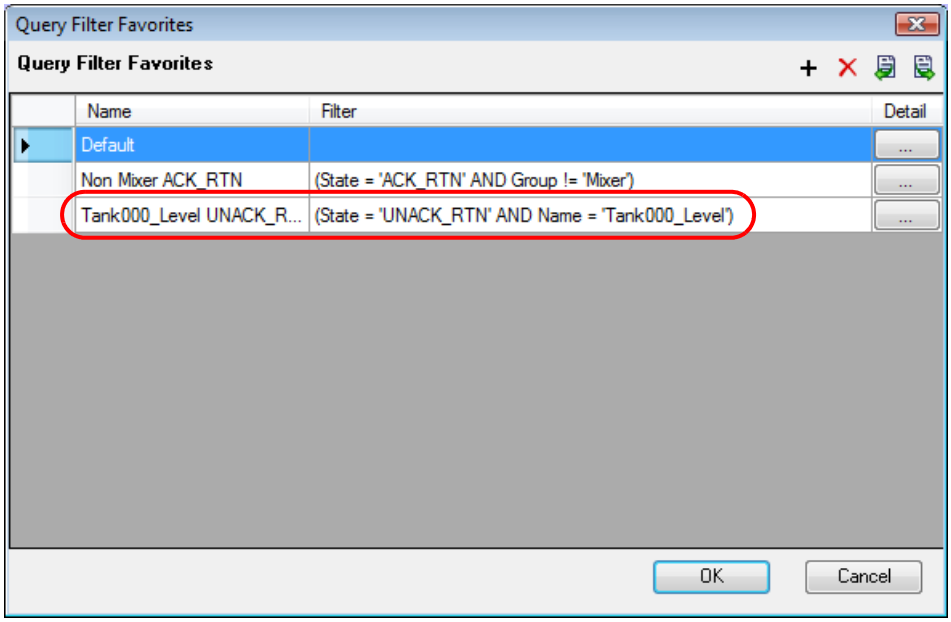
Test the Reset Button

41. Return to WindowMaker and test the reset button in **Runtime**.
42. Verify you are still logged in as **Wrench**.
43. Click the **Alarms** tab.
44. Right-click the **HistoricalAlarms** (bottom Alarm Client) and select **Query Filters**.

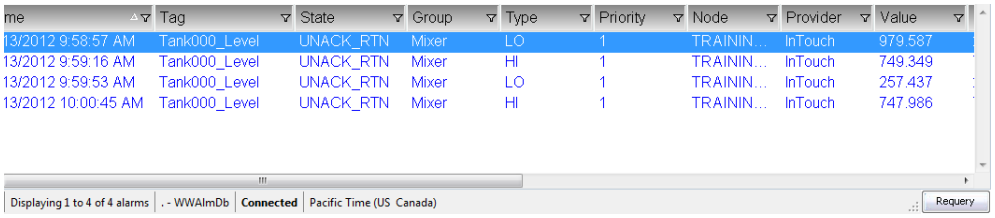
    The **Query Filter Favorites** window appears.



45. Click the **Tank000_Level UNACK_RTN** filter, then click **OK**.

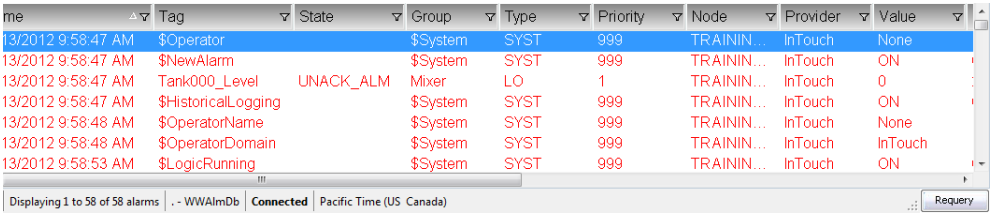


The results show alarms specific to the query.



46. Click **Clear All Filters**.

    The system reverts back to the unfiltered display.



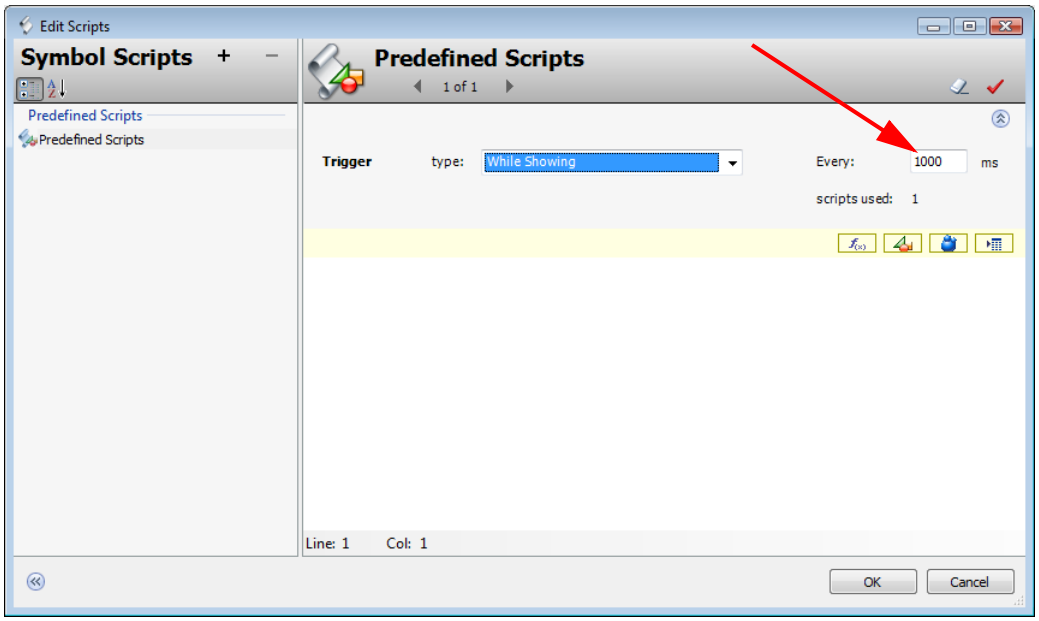
47. Return to **Development**.

Restrict Access to a Symbol

Next, you show the context menu security on the client control to show the context menu for supervisors and not show the content menu for all other users.

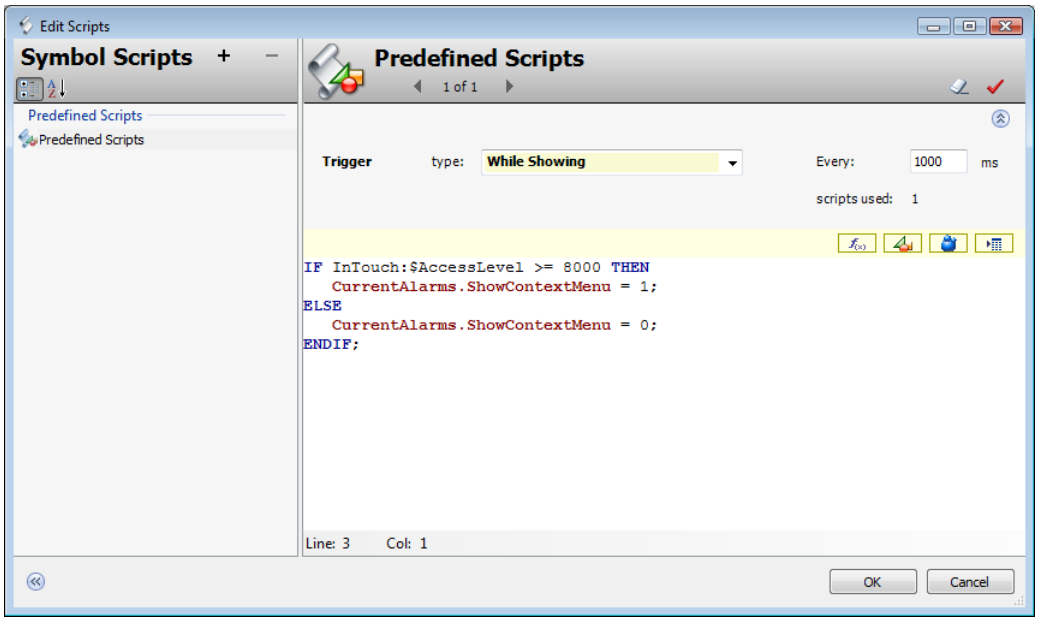
48. Open the **Alarms** symbol in the symbol editor.
49. Right-click the canvas and select **Scripts**.

    The **Edit Scripts** window appears.
50. In the **Trigger Type** drop-down, select **While Showing**, then confirm the default is set to **Every 1000 ms**.



51. Click in the body of the script and enter the following:

```
IF InTouch:$AccessLevel >= 8000 THEN
    CurrentAlarms.ShowContextMenu = 1;
ELSE
    CurrentAlarms.ShowContextMenu = 0;
ENDIF;
```



Note: You can navigate to the **Element Browser** and locate the **ShowContextMenu** property.

52. **Validate** the script and click **OK**.
53. **Save and Close** the symbol.

Test Supervisor Security

54. From WindowMaker, switch to **Runtime**.
55. Verify you are logged in as **Wrench**.
56. Right-click **CurrentAlarms** (top Alarm Client).

    Notice the context menu is not available based on your access level.
57. Log in as **Clipboard**, then right-click the **CurrentAlarms** client.

    Notice that the context menu is now available.

Lab 24 – Managing Limited Screen Real Estate

Introduction

In this lab, you use the built-in element properties and write scripts to resize a graphic at runtime. This allows you to initially reduce the size of a graphic on the window, then with script enlarge it to see details when the mouse hovers over it.

Objectives

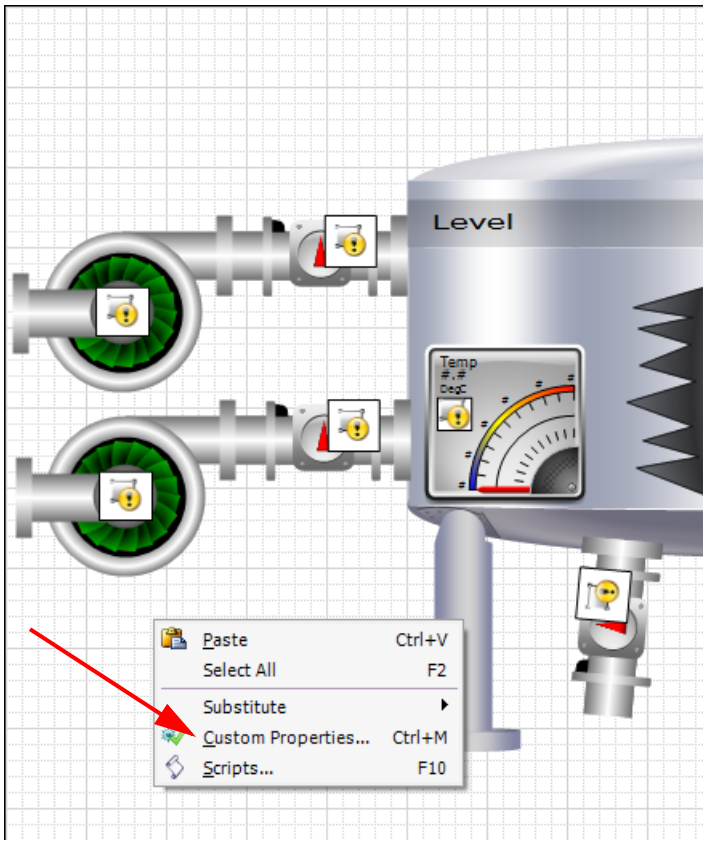
Upon completion of this lab, you will be able to:

- Write scripts to modify symbol height and width properties at runtime
- Create Named scripts in an ArchestrA symbol
- Set TreatAsIcon to enable additional animations for an embedded symbol

Define Additional Custom Properties

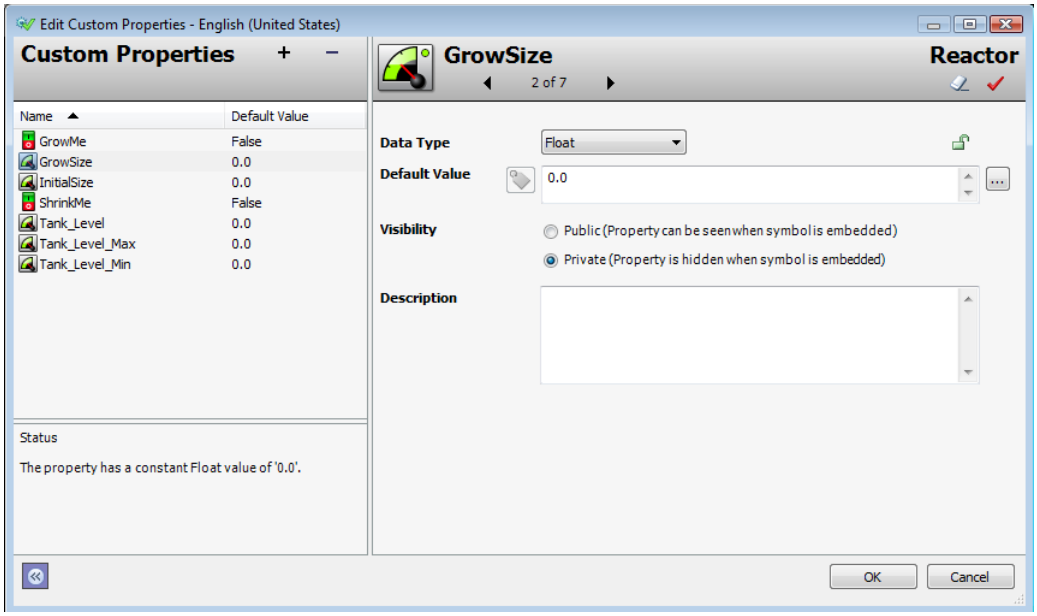
First, you add custom properties to the Mixer symbol.

1. In WindowMaker, open the **Mixer** window.
2. Right-click the **Reactor** symbol and open it for editing.
3. On the canvas, right-click and select **Custom Properties**.



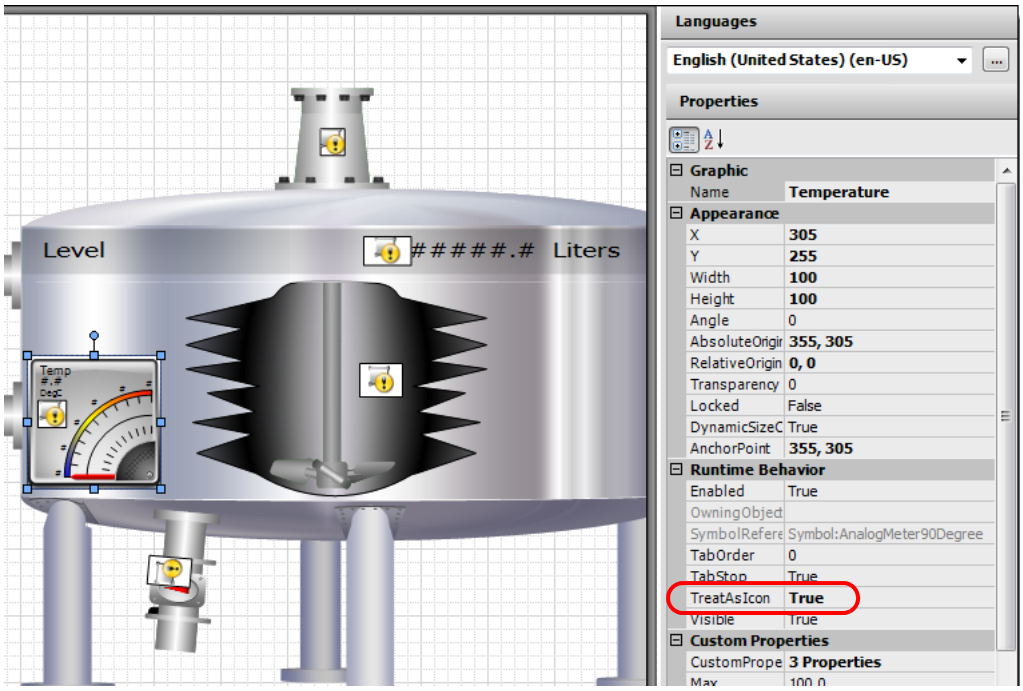
4. Click **+** and add the following **Custom Properties**:

Name	Data Type	Visibility
GrowMe	Boolean	Private
ShrinkMe	Boolean	Private
InitialSize	Float	Private
GrowSize	Float	Private



5. Click **OK**.

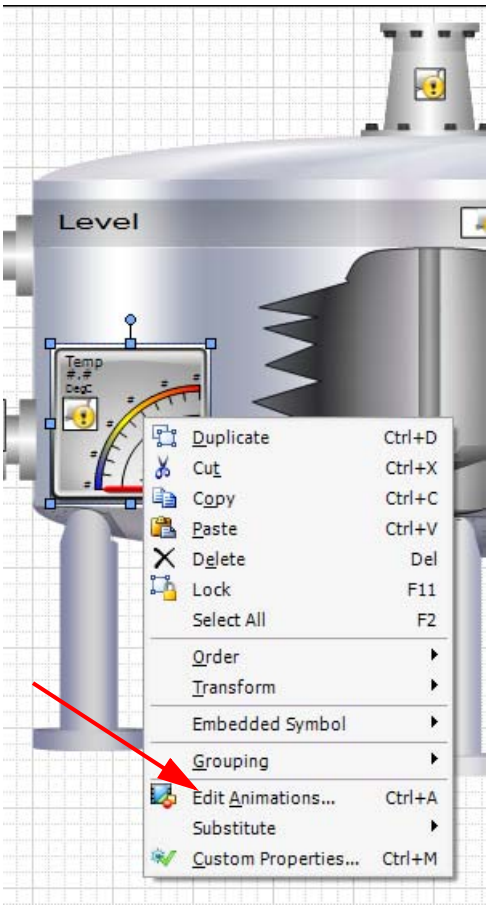
6. Click the **Temperature** element.
7. In **Properties** pane, **Runtime Behavior** area, change the **TreatAsIcon** to **True**.



This allows you to add additional animations to the symbol.

Next, you add Action and Named scripts to control grow and shrink properties.

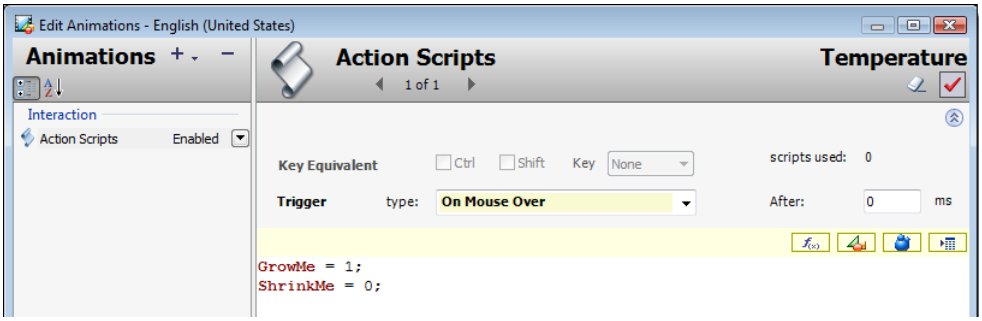
8. Right-click the **Temperature** element and select **Edit Animations**.



9. Click **+** and select **Action Scripts**.
10. Select the **Trigger type | OnMouseOver** and enter the following script:

    GrowMe = 1;

    ShrinkMe = 0;

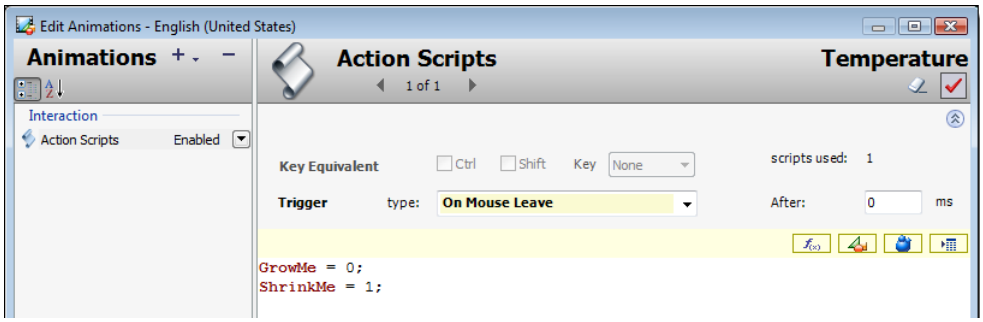


11. **Validate** the script.

12. In the same **Action Script**, select **Trigger type | OnMouseLeave** and enter the following script:

    GrowMe = 0;

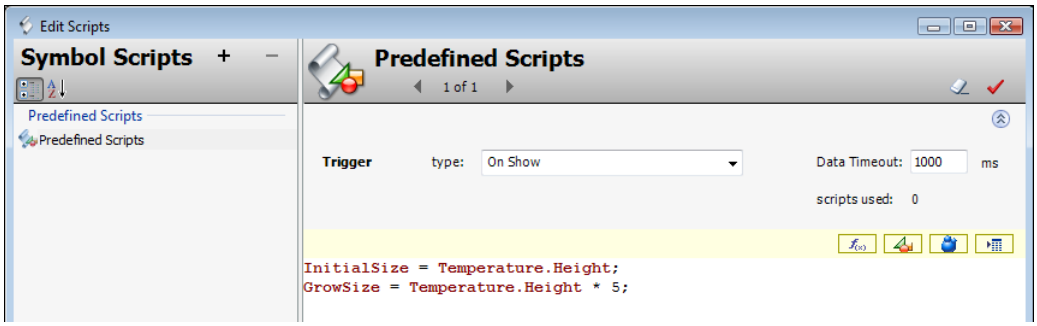
    ShrinkMe = 1;



13. **Validate** ✓ the script.
14. Click **OK**.
15. On the canvas, right-click and select **Scripts**.
16. Click **Predefined Scripts** and ensure **OnShow** is selected.
17. Enter the following script:

    InitialSize = Temperature.Height;

    GrowSize = Temperature.Height * 5;



18. **Validate** ✓ the script.

19. Click **+** and add the following **Named Script**:

Name: Grow

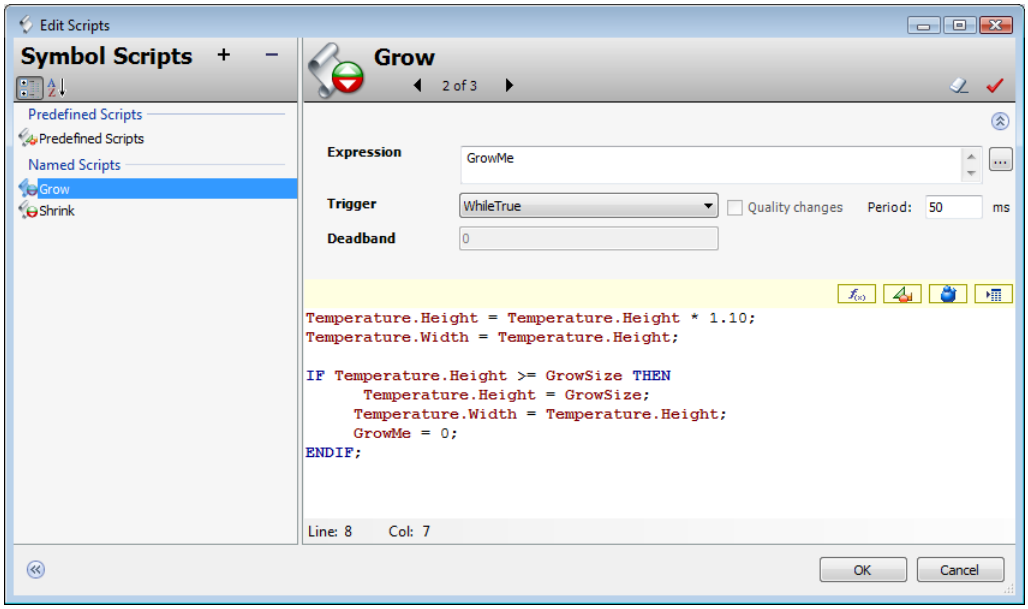
Expression: GrowMe

Trigger: WhileTrue

Period: 50 ms

Script:

```
Temperature.Height = Temperature.Height * 1.10;
Temperature.Width = Temperature.Height;
IF Temperature.Height >= GrowSize THEN
   Temperature.Height = GrowSize;
   Temperature.Width = Temperature.Height;
   GrowMe = 0;
ENDIF;
```



20. **Validate** the script.

21. Click **+** and add an additional **Named Script**:

    Name: Shrink

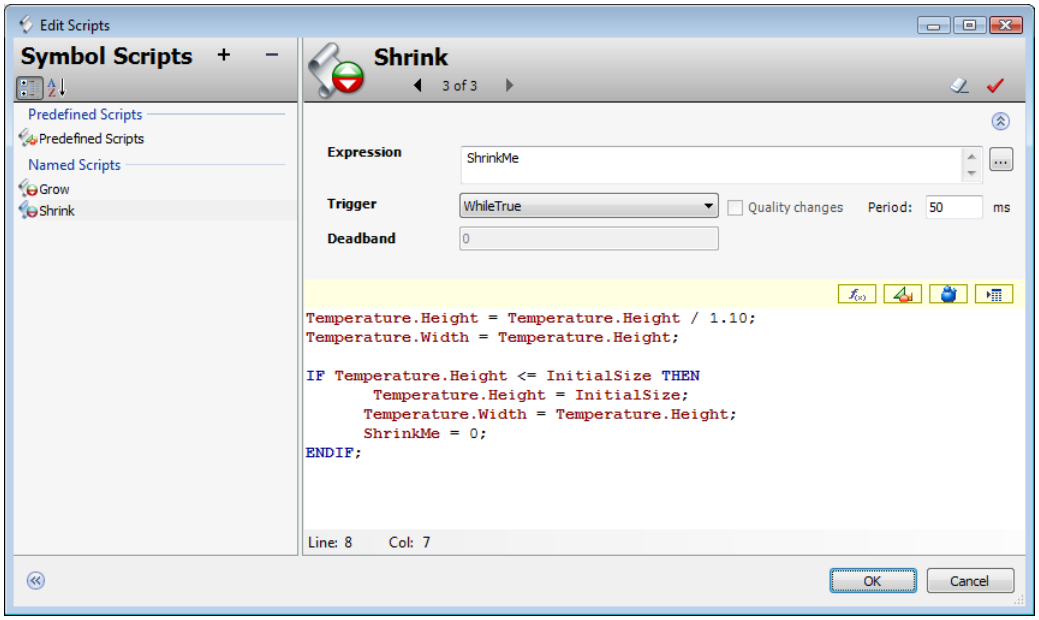
    Expression: ShrinkMe

    Trigger: WhileTrue

    Period: 50 ms

    Script:

```
Temperature.Height = Temperature.Height /1.10;
Temperature.Width = Temperature.Height;
IF Temperature.Height <= InitialSize THEN
   Temperature.Height = InitialSize;
   Temperature.Width = Temperature.Height;
   ShrinkMe = 0;
ENDIF;
```

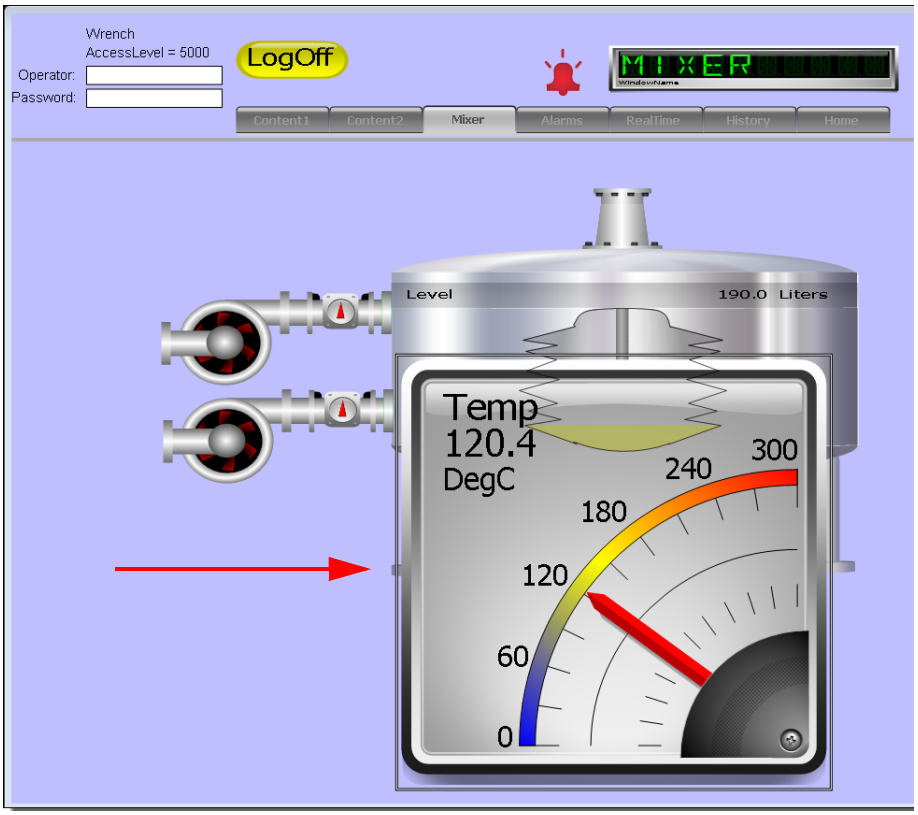


22. **Validate** ☑ the script.
23. Click **OK**.
24. Click **Save and Close**.
25. Add an appropriate **Check In** comment and click **OK**.

Test Grow and Shrink

26. Return to WindowMaker, then switch to **Runtime**.
27. Log in as **Wrench**.
28. Hover your mouse over the temperature symbol.

    The temperature symbol grows in size.



29. Move your mouse away from the temperature symbol.

    The temperature symbol returns to the original size.



invensys
Learning Services

Module 8 – Application Distribution

Section 1 – Application Deployment 8-3

Lab 25 – Deploying the Application 8-5

Section 2 – Application Publishing 8-23

Module Objectives

- Introduce application deployment
- Deploy an InTouch application for runtime operators
- Publish an InTouch application for runtime operators

Section 1 – Application Deployment

This section describes how to deploy your InTouch application for runtime operators.

Overview

When you are ready to test or move the application into production, you deploy your application to a platform in the Galaxy. This copies the objects from the development environment to the runtime environment and makes your objects live and functional. After you have deployed subsequent changes made to your application, these changes are not copied to the runtime environment until you redeploy them from the ArchestrA IDE.

You can deploy and test your objects at any time during development. To deploy your InTouch application, you must deploy a couple of objects to the runtime environment that your InTouch application depends on for interaction and support with the Galaxy. A WinPlatform is the base object needed on any PC that will participate in the Galaxy, but InTouch itself requires a ViewEngine to handle file synchronization and copying. The ViewEngine hosts your InTouchViewApp on the runtime platform.

Also, to deploy your InTouch application you must check your application into the Galaxy (Save and Close). This requires both WindowViewer and WindowMaker on the development environment to be closed.

Deploying Objects

You deploy object instances for three reasons:

- Testing
- Place the application into production to process field data
- Update an existing application with changes you made

When you are ready to deploy, make sure the following conditions are met:

- Bootstrap software is installed on the target computer(s)
- The objects being deployed are not in an error state in the Galaxy database
- You created, configured, and checked in the objects to the Galaxy
- Objects are assigned to a host
- The object's host is already deployed. Objects are deployed in hierarchy order that is the object host deploys before a subordinate object which depends on it.

Development and Runtime

The development environment is where the configuration of the InTouch application and Galaxy environment takes place, such as using WindowMaker to create, configure, and modify your application. 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 runtime Galaxy objects and InTouch WindowViewer takes place. Access to I/O, alarm generation, historical data, script execution, and visualization of data, occurs in the runtime environment. The runtime environment is comprised of execution components (Platforms and Engines) and client components (InTouch, TagViewer, and I/O & DAServers).

Deploy an InTouch Application

Deployment is comprised first of creating runtime instances of the three derived templates needed. First you must make an instance of your WinPlatform derived template and make one small configuration change on this instance - it must be assigned to the network node name that will serve as the runtime environment.

To deploy the instance of the InTouch application, create a new derived template from the ViewEngine object and make an instance of this and place it under the WinPlatform instance that was just created. Finally, derive an instance of your InTouchViewApp and place this under the ViewEngine instance. You now have a hierarchy of WinPlatform, ViewEngine, and InTouchViewApp, the necessary three instances for deployment. This procedure is repeated for each runtime node that will host your InTouch application within the Galaxy.

Any node which will receive a WinPlatorm must have prerequisite boot software installed, which allows that node to participate in the Galaxy and prepares it to receive a WinPlatform deployment. Any node that will host an InTouch runtime environment, must also have a copy of InTouch WindowViewer and an appropriate runtime license installed.

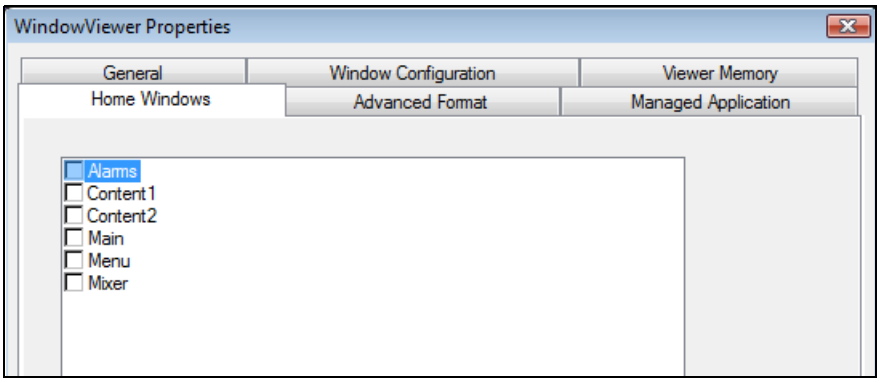
Redeploying or Undeploying Objects

Redeploying is a bit of a misnomer, as there is no actual option to redeploy an object. When the latest copy of an object is ready for deployment, the deploy option is selected. This automatically handles shutting down the currently deployed instance, synchronizing the changes, and starting it up again.

While you are testing, you frequently redeploy your application to test the changes made in a runtime environment, this can be done even on the same development node. To get the current copy of your InTouch application to the runtime node, deploy the instance of the InTouchViewApp again. A separate undeployment step is not required unless the intention is to remove the current InTouch application; for example, to deploy an entirely different InTouch application to that node.

Configuring a Home Window

Home Windows are windows designated in your application to open when WindowViewer first starts. Configure the Home Window on the **WindowViewer Properties** dialog box, **Home Windows** tab. The Home Windows selections have no effect when you use the fast switch from WindowMaker development to start WindowViewer. Home Windows are automatically opened when you start WindowViewer directly from either its icon or its menu command.



Lab 25 – Deploying the Application

Introduction

Before a managed InTouch application can be used in a production environment, it must be deployed to a ViewEngine running on WinPlatform. Operators will be able to interact with your application on their workstations using WindowViewer, and receive updates when the application is deployed.

In this lab, you create derived templates and deploy an InTouch application in WindowViewer using the ArchestrA IDE.

Objectives

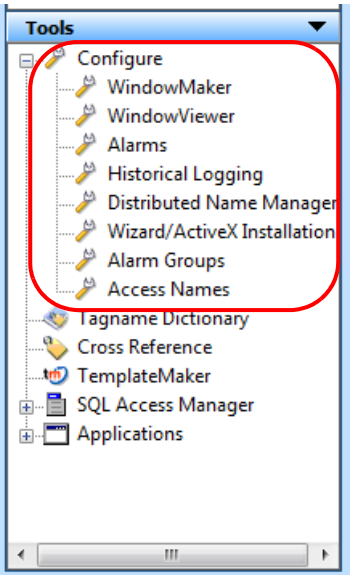
Upon completion of this lab, you will be able to:

- Configure how WindowViewer will receive application updates
- Derive a template and an instance of a ViewEngine
- Derive an instance of an InTouch application

Configure WindowViewer

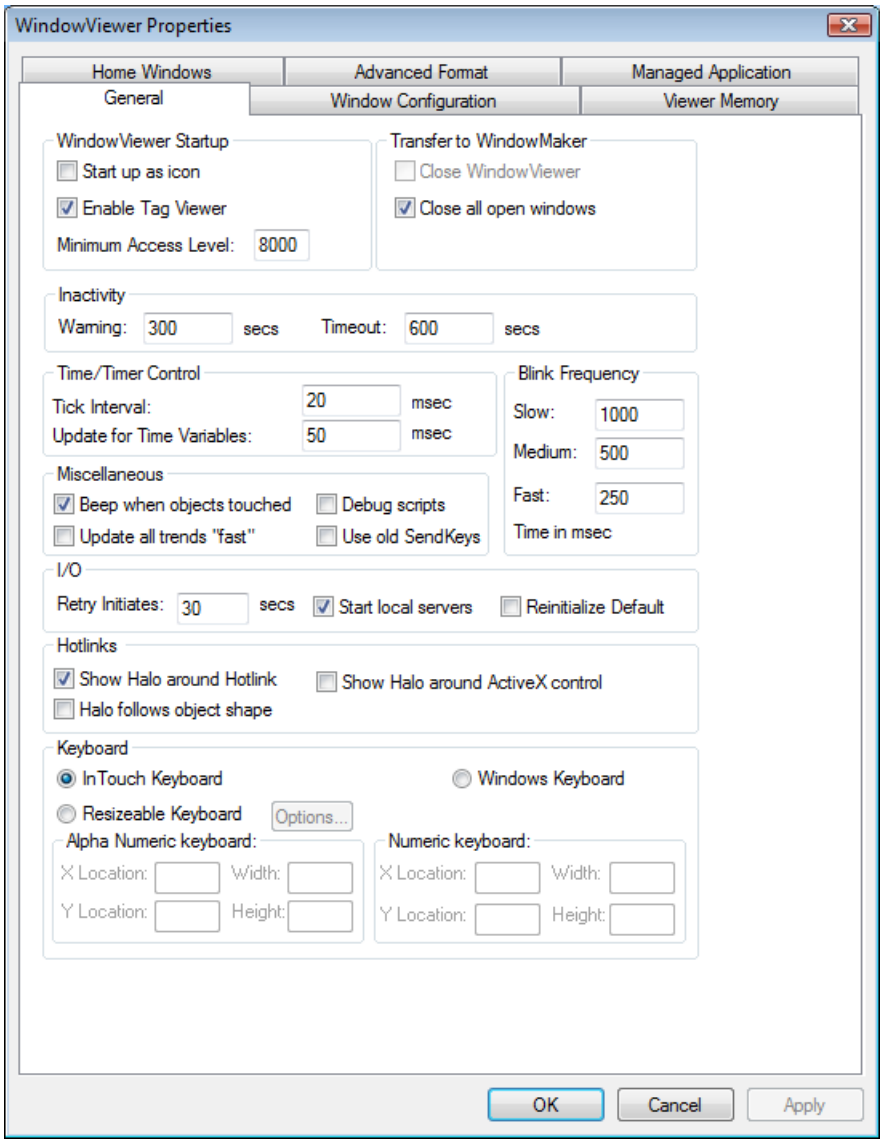
First, you verify WindowViewer configuration options for deployment to a visualization node.

1. Close WindowViewer.
2. In WindowMaker, in the **Tools** pane, expand **Configure**.

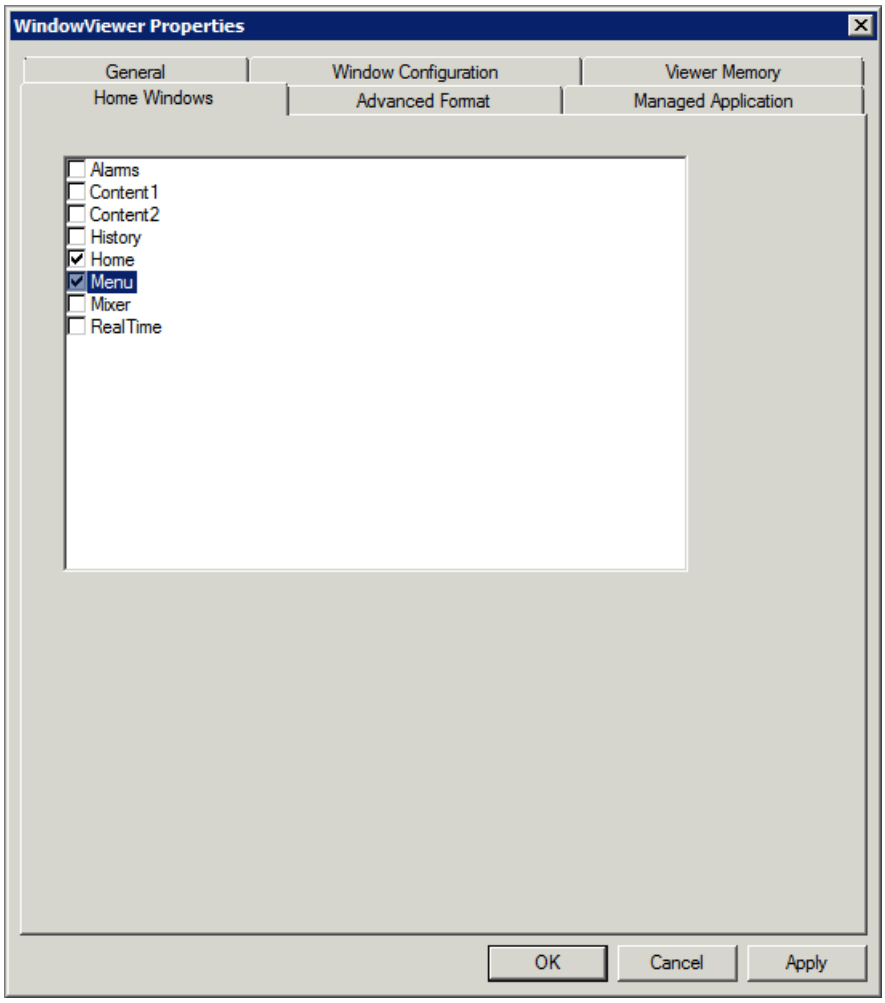


3. Double-click **WindowViewer**,

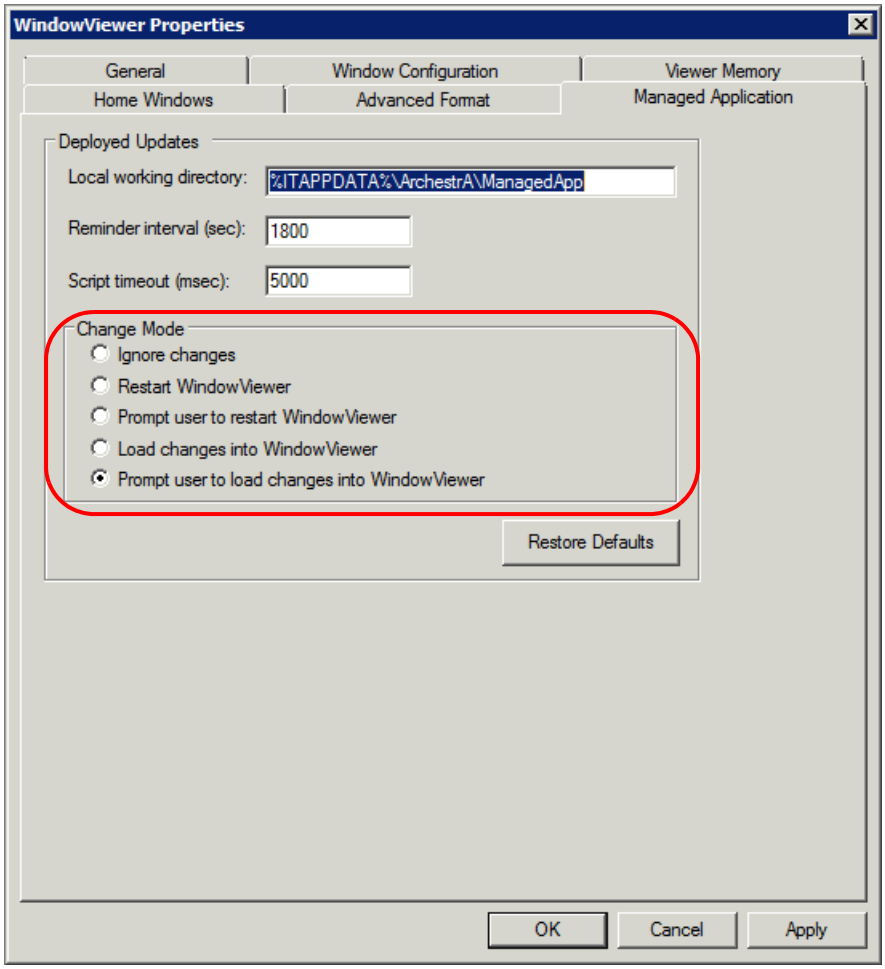
   The **WindowViewer Properties** window appears.



4. Click the **Home Windows** tab.
5. Check the **Home** and **Menu** windows.



6. Click the **Managed Application t**ab.
7. In the **Change Mode** area, verify that the **Prompt user to load changes into WindowViewer** is selected.



8. Click **OK**.
9. Close WindowMaker.
10. Enter an appropriate **Check In** comment.
11. Click **OK**.
12. When **Check In** is complete, click **Close**.

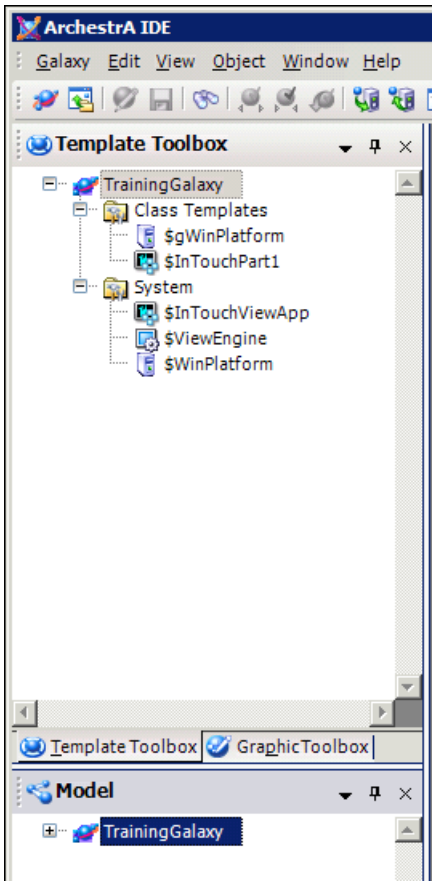
Derive Templates

Recall in Lab 1 you already derived a WinPlatform and created an instance of it for your primary development node. Now you reuse the WinPlatform derived template to create a new instance for our visualization node.

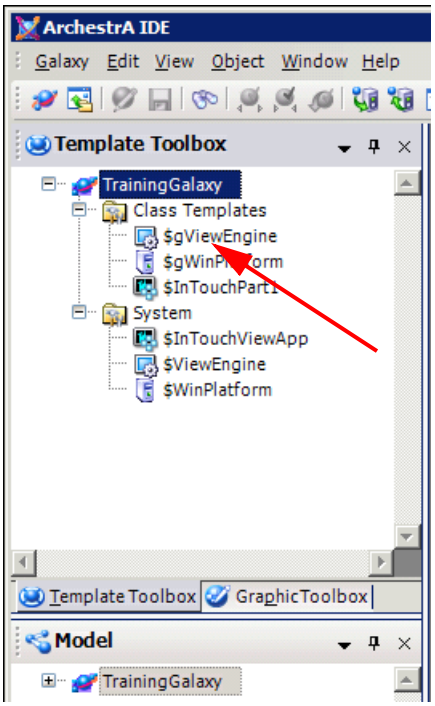
Note: It is considered a best practice to create at least one level of derivation before creating an instance of any object.

---

13. In ArchestrA IDE, **Template Toolbox**, expand the **Class Templates** and **System** toolsets.

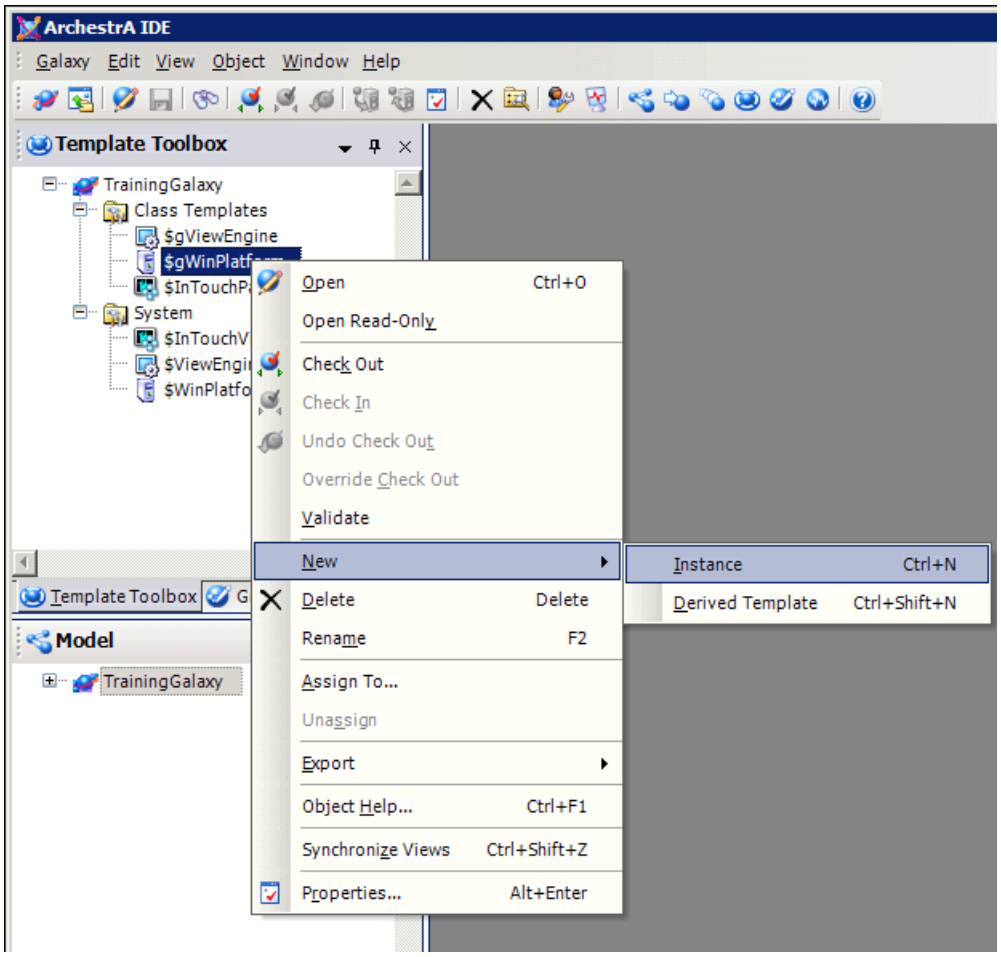


14. In the **System** folder, right-click **$ViewEngine** and select **New | Derived Templat**e.
15. Name the new template **$gViewEngine**, then move it to the **Class Templates** toolset.

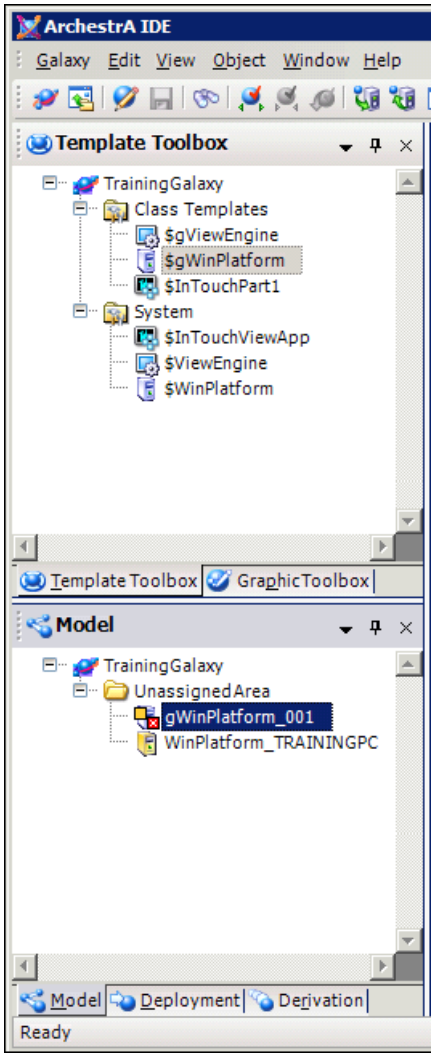


Next, you create instances of the objects.

16. In **Template Toolbox**, right-click **$gWinPlatform** and select **New | Instance**.

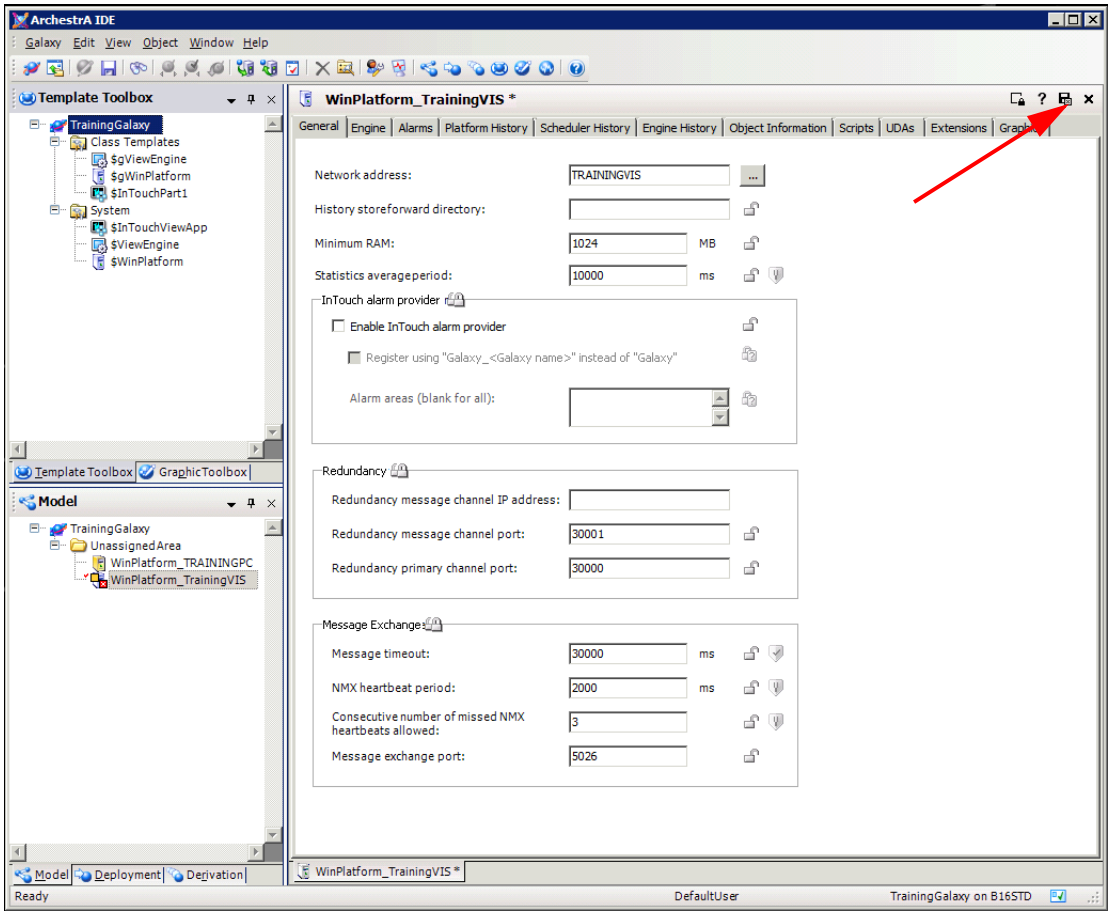


The instance appears in the **Unassigned Area** of the **Model** view.



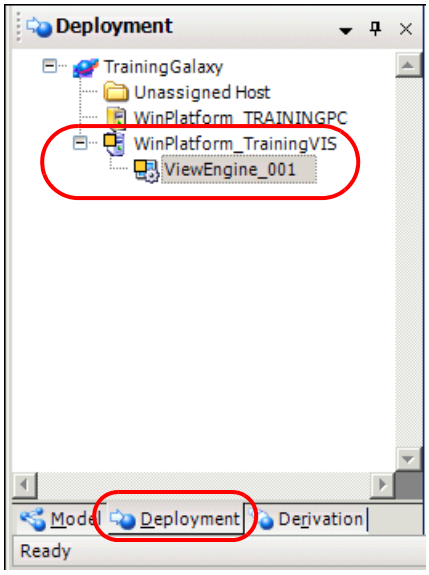
17. Name the instance **WinPlatform_TrainingVIS**.
18. Double-click **WinPlatform_TrainingVIS** to open the configuration editor.

19. On the **General** tab, in the **Network address** field, enter the visualization system node name provided by your instructor.
20. On the top-right of the configuration editor, click the **Save and Close** button.



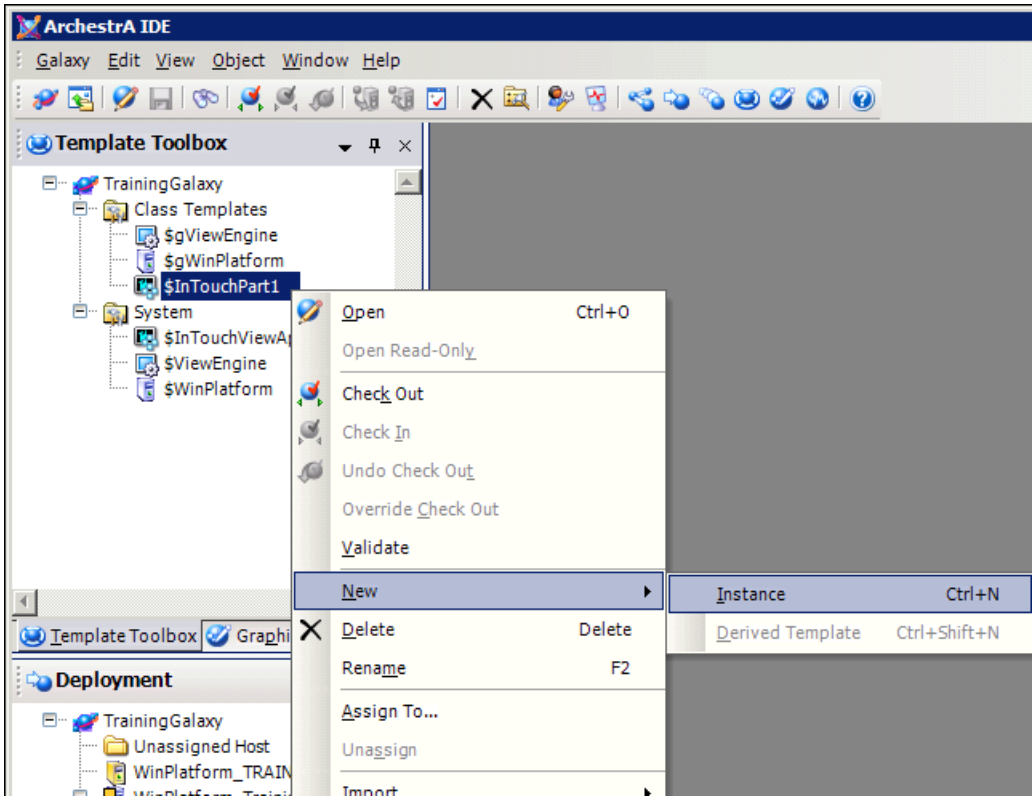
21. Enter an appropriate **Check In** comment and click **OK**.

22. Create a new instance for **$gViewEngine**.
23. Name the new instance **ViewEngine_001**.
24. Click the **Deployment** view.
25. Drag **ViewEngine_001** below **WinPlatform_TrainingVIS**.

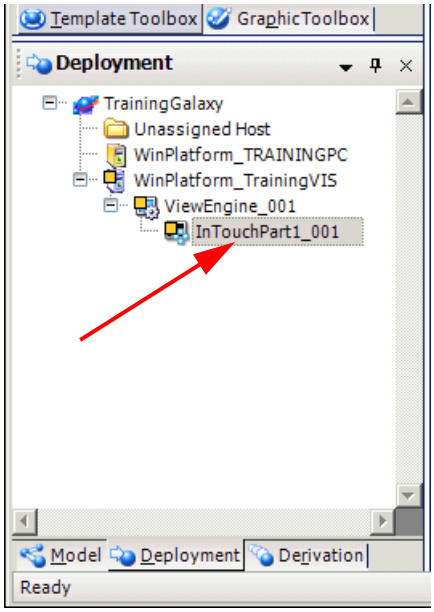


Now, you create an instance of your application.

26. In **Template Toolbox**, right-click **$InTouchPart1** and select **New | Instance**.

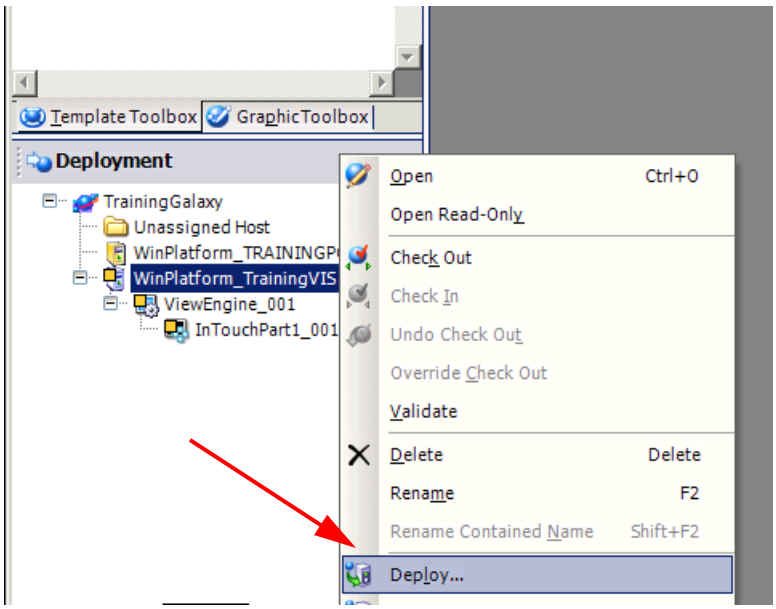


27. Accept the default name and drag it below **ViewEngine_001**.



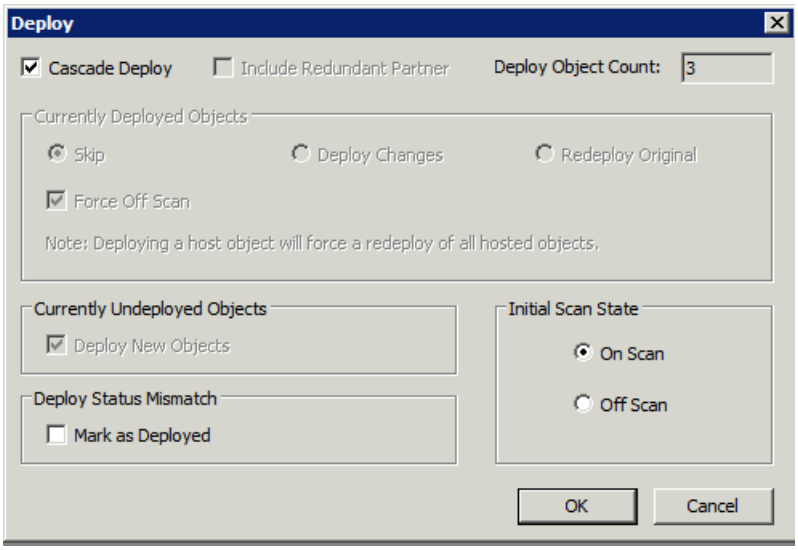
Notice the icons for your **WinPlatform**, **ViewEngine**, and **InTouch** application show the undeployed  state.

28. Right-click **WinPlatform_TrainingVIS** and select **Deploy**.



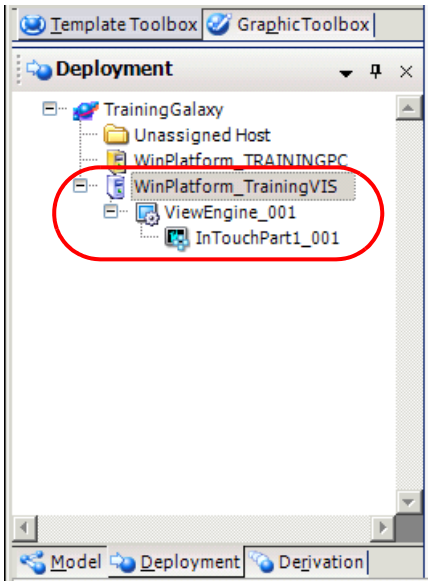
The **Deploy** dialog box appears.

29. Leave all defaults and click **OK**.



30. When **100% completed**, click **Close**.

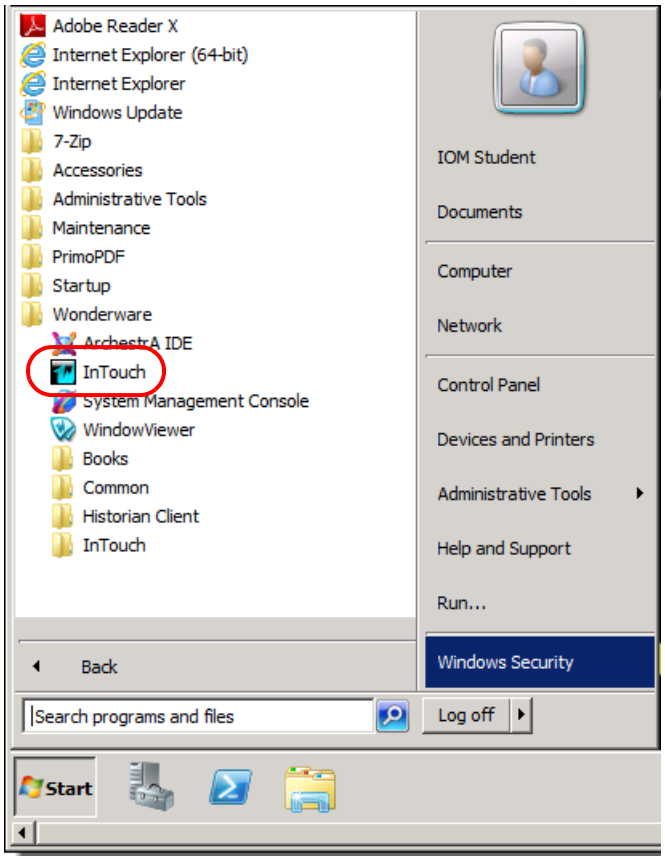
    Notice icons now indicate the application is deployed.



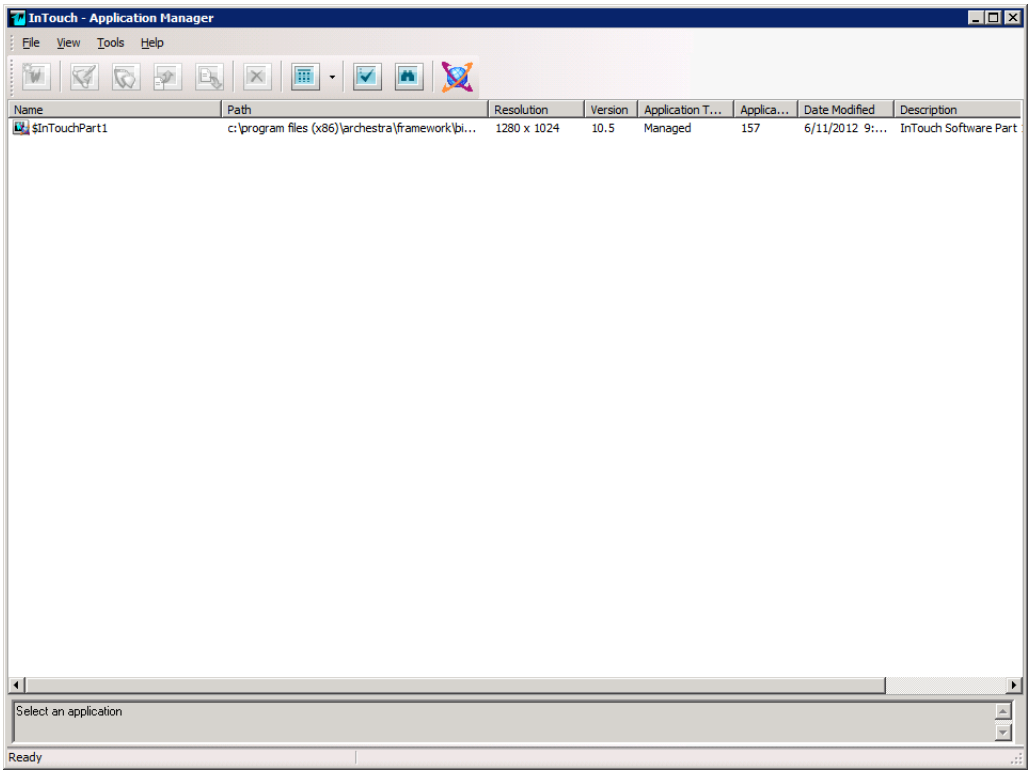
Test the Application

Now, you test your application on the visualization node. Your instructor will provide a visualization node and instructions on how to access it.

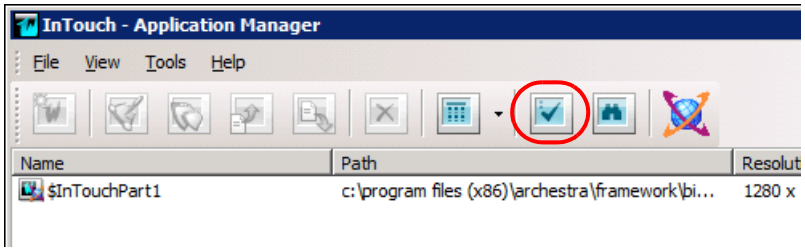
31. In the **TrainingVIS** visualization node, click **Start | All Programs | Wonderware | InTouch | InTouch** to start the **InTouch Application Manager**.



Your Managed application is displayed in the **InTouch Application Manager** window.

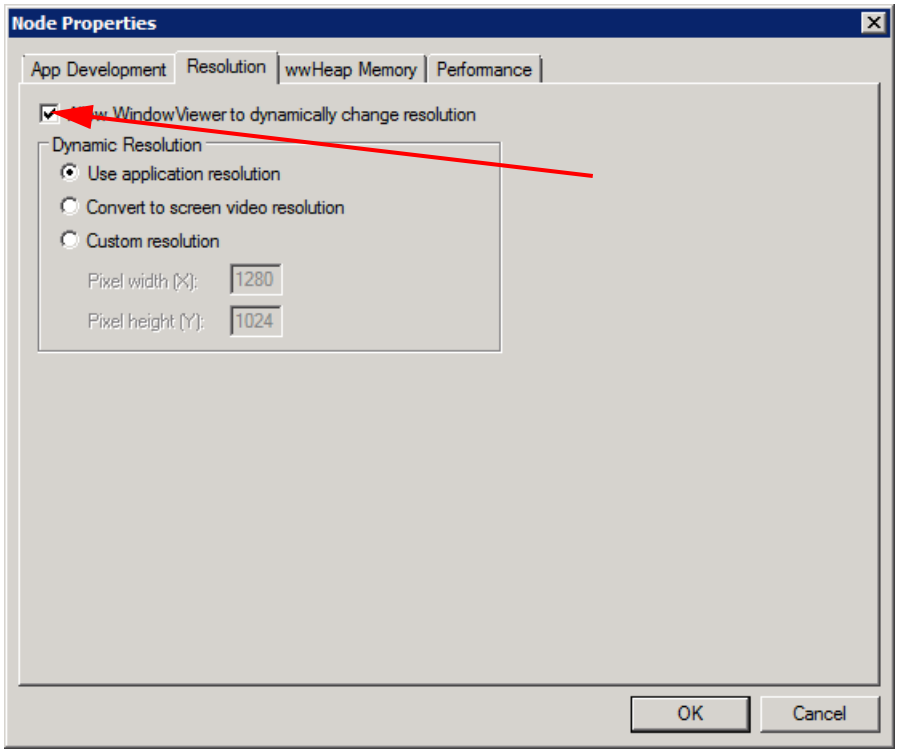


32. On the toolbar, click **Node Properties**.

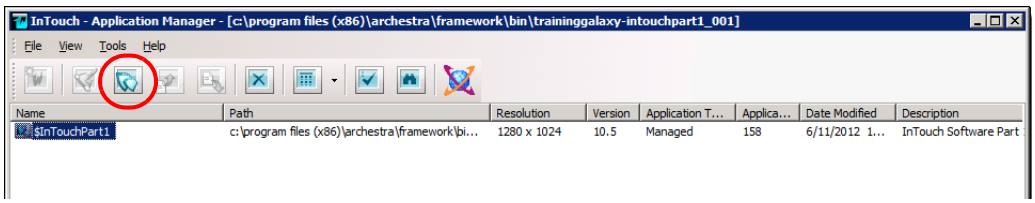


The **Node Properties** dialog box appears.

33. On the **Resolution** tab, check **Allow WindowViewer to dynamically change resolution**.

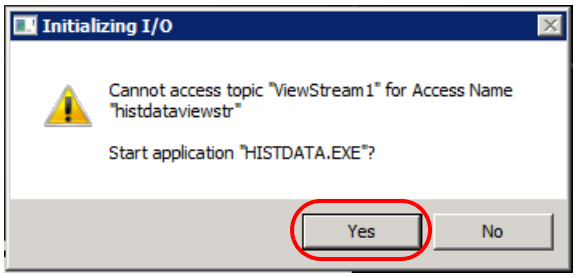


34. Leave remaining defaults and click **OK**.
35. In **InTouch Application Manager**, click **$InTouchPart1** to select the managed application.
36. Click the **WindowViewer** button.



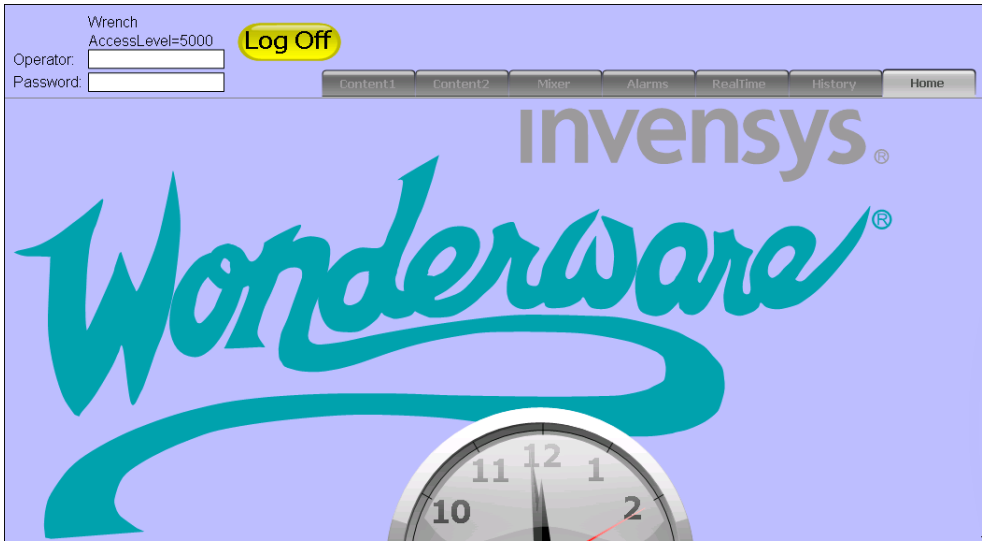
The **Initializing I/O** dialog box appears.

37. Click **Yes** to start **HISTDATA.EXE**.



The WindowViewer application appears.

38. Log in as **Wrench**.



39. Navigate through the tabs to see that all windows are functional.

Note: You cannot switch to WindowMaker from a deployed instance of application.

Section 2 – Application Publishing

This section discusses how to publish your InTouch application for your production environment.

Overview

Publishing an InTouch application converts the database of a Managed application into a traditional file-based InTouch application. You can use a published version of your InTouch Application in your production environment. This is a static release of the application, which is no longer connected to the Galaxy and cannot reflect changes made to the original application or any of its dependencies such as ArchestrA symbols.

Unlike a deployed instance of the application, changes made in your development environment are not pushed out to the published version. In essence, publishing your application extracts a copy of the files from the database and turns the application into a flat file application. Once in this form, it can be zipped, copied, and run from the InTouch.exe application manager tool.

InTouch File Content

In a flat file (instead of live database managed) format, the InTouch application has a copy of all of the ArchestrA Symbols snapshotted at the instance they were published. A copy of each of these symbols is extracted and placed in a subfolder under the main published InTouch application folder.

To give you an idea of what's under the hood with a flat file version of an InTouch application, you will look at one case - that of a simple InTouch Application Window, such as a popup window. When an application window is created, multiple files are created to hold the information needed by both WindowMaker and WindowViewer to display the window as it was defined in WindowMaker. This means that each window will have more than one file used to define it, WindowMaker requires a file called **WIN****.WIN**, which holds window definition information such as width, height, and frame style.

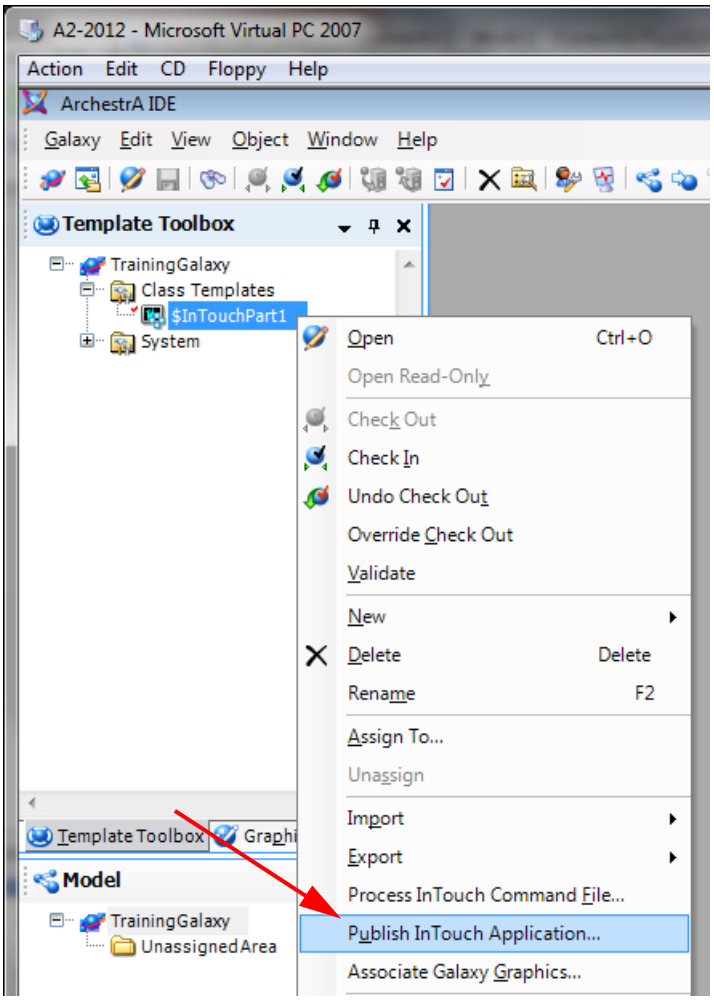
When changes are made to the window in WindowMaker, the new copy of the win file is resaved as a **.WIN** file and the old file extension is changed to **.WBK**. When you switch to runtime, WindowViewer compiles its own version of the .WIN file for high performance use, this file has the **.WVW** extension. Since these files are compiled automatically by WindowViewer, if an application gets very large and difficult to copy, its footprint can be reduced dramatically by removing these .WVW extension files. This will make startup longer, however, every .WIN file must be recompiled before the application can start. The more windows in the application, the longer this process will be (for a first time startup). It should be noted, if the display resolution is different between the node where the development occurred and the runtime node, these files will be recompiled anyway.

By way of one additional example, if a window contains a real-time trend, a file is created with a **.TRN** extension which accompanies the .WIN file.

For a complete and detailed list of InTouch application files, see InTouch documentation.

Publish an IDE-Managed InTouch Application

To publish an IDE-Managed InTouch application, from the **Template Toolbox** select the **Publish InTouch Application** option from the application's right-click menu. At the **Browse** dialog box, point to or create the folder to publish the application to. The IDE will display a notification message upon successful completion of the operation.

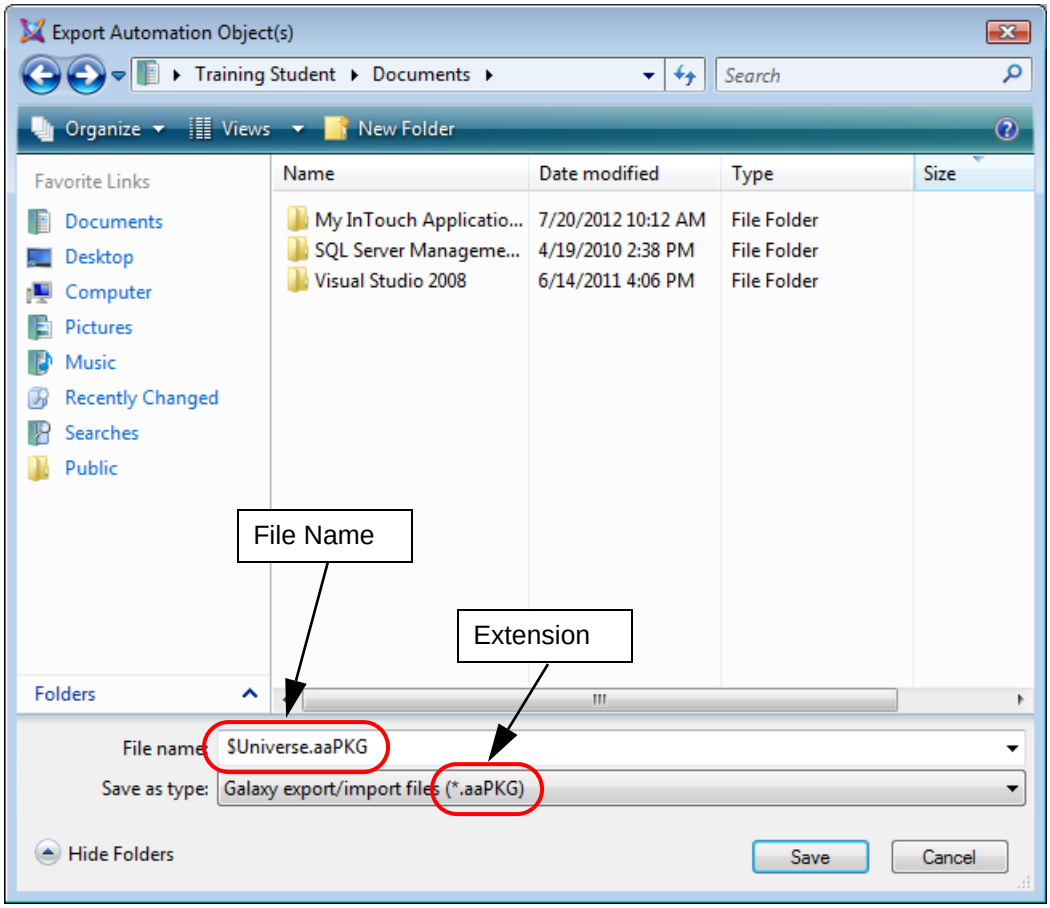


Published vs. Managed InTouch Application

A published application does not serve as a backup copy of your application. A published application does not allow managed InTouch application edits, or edits to ArchestrA Symbols. Publishing can be used as a tool to obfuscate intellectual property and prevent the theft of your application design, but only if you build with ArchestrA Graphic Symbols approach. Even if WindowMaker is started on a published application, only primitive drawing and animation changes can be made. For example, an application developer cannot even change the tags linked to an ArchestrA symbol, much less alter the design or appearance of any symbols.

Also, a published InTouch application cannot be restored back into the Galaxy. Only a proper .aaPKG backup file may be restored in this way, or used in another Galaxy. Backing up an application directory can be as easy as copying and pasting the entire contents of an application into another directory. Backing up files creates a record of your development and ensures recovery of work in case of some unexpected event.

Once an IDE-Managed InTouch application is checked in, you can export the object to back it up from the IDE. The result is the .aaPKG file.



At the **Export Automation Object(s)** dialog box, notice the **File name** field reflects the name of your application and the **.aaPKG** extension of the filename in the **Save as Type** field. This extension indicates that this is the backup of an IDE-Managed InTouch application. You then use the .aaPKG file to import the application to another Galaxy.



invensys
Learning Services

Module 9 – Putting It All Together

Section 1 – Building a New Process 9-3

Lab 26 – Putting It All Together 9-5

Module Objectives

- Implement the key concepts learned throughout the InTouch – Part 1 class
- Create a new process using I/O tags
- Develop a three-state valve

Section 1 – Building a New Process

In this section, you review important features of InTouch software covered in this class. You then apply what you have learned as you design, build, and animate a process using InTouch.

Overview

Now that you have been introduced to Wonderware InTouch software, this is your opportunity to build a process window on your own. If time permits, enhance your process window with Navigation, Alarms, History, Security, and enhance it with scripting. This section contains no new learning; so let us review what you have learned.

By now, you have:

- Created a new Managed InTouch application and created windows and navigation
- Defined and created InTouch tags of different types, and applied I/O communication
- Created and customized ArchestrA graphics and used this is a wide variety of ways
- Used many animations and data presentation features to interact with and view your data
- Created and applied InTouch and ArchestrA scripting to expand and enhance your application
- Applied InTouch alarming, real-time trending, historical logging, and security

Take advantage of this opportunity to use your newly gained knowledge in a practical way while you have an instructor available to assist you and to facilitate making your ideas come to life. This is a unique opportunity to work on a practical project with an experienced application designer on hand to provide any assistance and guidance you need, as well as technical support. Challenge yourself and this exercise will cement your understandings and prepare you for your role as an InTouch application designer when you return to your facility.

Lab 26 – Putting It All Together

Introduction

The following lab is designed for you to build your own InTouch process window. Before you begin, study the process workflow and follow the preparation instructions in this lab.

Objectives

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

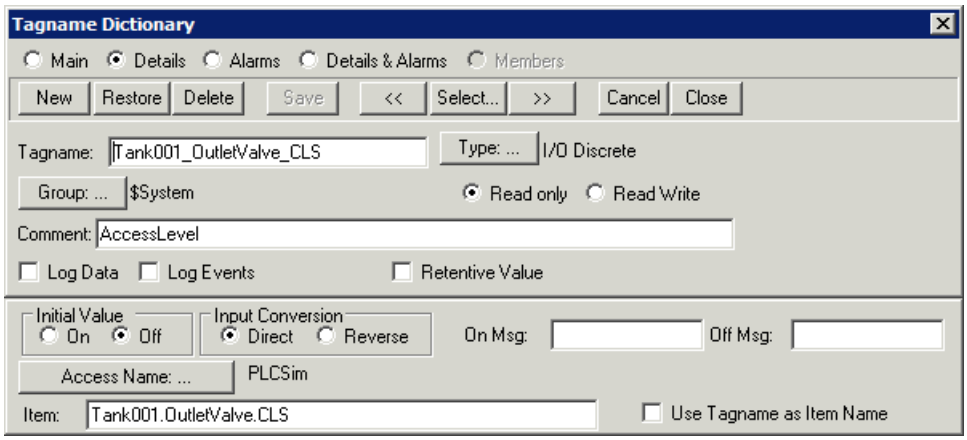
- Build a visual display of a process using what you have learned and created in all the previous labs

Create the I/O Discrete Tags

This lab uses the second set of reactor tags starting with Tank001_ that you created in In Lab 4. In addition to these tags, you create 6 additional Open and Close Limit Switch tags to represent the 3 valves.

1. In WindowMaker, open **Tagname Dictionary**.
2. Create an **I/O Discrete** tag, as follows:

   Tagname: Tank001_InletValve1_CLS
   Item: Tank001.InletValve1.CLS



3. Using the above steps, create the **I/O Discrete** tags listed below.

Tag	Item
Tank001_InletValve1_OLS	Tank001.InletValve1.OLS
Tank001_InletValve2_OLS	Tank001.InletValve2.OLS
Tank001_InletValve2_CLS	Tank001.InletValve2.CLS
Tank001_OutletValve_OLS	Tank001.OutletValve.OLS
Tank001_OutletValve_CLS	Tank001.OutletValve.CLS

I/O Process Simulation Workflow

The process you model must represent the following I/O values. However, keep in mind the discrete and analog values from the I/O Simulation Process can represent anything. You are not limited to creating a mixer tank with a level and temperature going up and down. Use your imagination and be creative. You can use these I/O points to represent any process. Try to imagine a creative way to use these inputs and outputs to do something similar to your own area of business.

- **2 Inlet Valves**
- **2 Transfer Pumps**
- **1 Agitator**
- **1 Outlet Valve**
- **1 Tank Level transmitter**
- **1 Tank Temperature transmitter**

The details of the process are as follows:

- The process begins with **InletValve1** opening and **TransferPump1** running.
- The **TankLevel** begins to increase.

- When **TankLevel** reaches 60%, **InletValve1** closes and **TransferPump1** stops running.
- **InletValve2** opens and **TransferPump2** runs.

- The **TankLevel** continues to fill to between 95% and 100%.
- **InletValve2** closes and **TransferPump2** stops running.
- **Agitator1** then runs for 15 seconds.

- **OutletValve** then opens and the **TankLevel** decreases (drains) to 0%.
- The process begins again.

- The **TankTemperature** fluctuates: cooler when the tank is less than half full, and warmer when the tank is more than half full.

Required Processes

The list of tags you use to animate and graphically represent the I/O Simulation Process, is shown below. The list is combined from the tags you created in Lab 4 using DBLoad along with the 6 tags you created for OLS and CLS in this lab.

- **Tank001_Agitator**
- **Tank001_InletValve1_CLS**
- **Tank001_InletValve1_OLS**
- **Tank001_InletValve2_CLS**
- **Tank001_InletValve2_OLS**
- **Tank001_Level**
- **Tank001_OutletValveCLS**
- **Tank001_OutletValveOLS**
- **Tank001_Temperature**
- **Tank001_TransferPump1**
- **Tank001_TransferPump2**

The Challenge

This lab leverages the I/O Simulation Process which runs continuously and contains a complete simulation. Your task is to represent the process through graphics and animation using your imagination and creativity.

Truth Table

A challenge presented in this lab is to represent a 3 state valve for Inlet1, Inlet2, and Outlet using the OLS and CLS tags created in this lab. The following truth table explains the three states of each valve: Open, Closed, and Transition. The Transition state occurs between each transition from Open to Closed, and from Closed to Open.

Note: Though OLS True and CLS True is a fault state for the 3StateValve, the I/O Simulation Process used in this course does not have a fault state for its valves.

Truth Table:

State	OLS	CLS
Open	True (1)	False (0)
Closed	False (0)	True (1)
Transition	False (0)	False (0)
Fault	Not Used By I/O Process Simulation	

Here are some suggestions to help you think about your approach to representing your new process.

- You might create a new window for your process and include a way to navigate to your new window
- You might add your process to enhance an existing window in your application
- You might create a new ArchestrA Symbol that contains your entire process or you might build your process a la carte using existing graphics from the library directly on the window
- You might add security for your process
- You might add alarms for your process
- You might add trending for your process
- You might redeploy your application to test your new process on the visualization node