Optegrity

User's Guide

Version 5.1 Rev. 0

Optegrity User's Guide Version 5.1 Rev. 0
July 2014

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7100
Fax: (781) 265-7101

Part Number: DOC118-510

Contents Summary

Contents

Preface

Describes this guide and the conventions that it uses.

About this Guide

This guide describes how to use Optegrity to build applications for abnormal condition management. It describes:

- Getting Started:
  - Understanding the various Optegrity modules and architecture.
  - Running Optegrity.
  - Working with Optegrity through its menus.
  - Creating, saving, and loading Optegrity applications.
- Process Maps:
  - Building process maps by creating, configuring, and connecting domain objects to form a graphical representation of a process and configuring domain objects for built-in event detection.

  - Configuring built-in event detection for instruments and process equipment.
  - Creating user-defined domain objects that contain internal datapoints to represent each external tag variable.
- Data Sources:
  - Creating and configuring various types of network interfaces for obtaining data from external data sources, including OPC, PI, databases, JMail, and JMS providers.
  - Configuring external datapoints from CSV files, which obtain values through network interfaces, perform low-level data validation, and provide data for internal datapoints in a domain map.
  - Configuring engineering unit conversions and synonyms for representing external and internal datapoints.
  - Logging internal and external datapoint values.
  - Replaying data to internal and external datapoints for simulation and testing purposes.
  - Creating datapoint simulations for internal and external datapoints.
- Event Detection:
  - Creating generic event detection diagrams that monitor internal datapoints and generate events when certain conditions are met.
  - Initializing and uninitializing process maps.
  - Creating various types of reports for event metrics and system performance.
- Diagnostic Reasoning:
  - Creating generic SymCure fault models to diagnose events, determine root causes, and take corrective actions.
  - Compiling SymCure diagnostic models and sending fault model events for testing these models.
- Alarm and Message Management:
  - Interacting with operator messages through the Message Browser.
  - Interacting with SymCure events through the alarms, root causes, and external actions diagnostic console browsers.
  - Configuring message queues for logging operator messages and events.

- Customization:
  - Creating custom event detection.
  - Customizing various aspects of Optegrity, including creating custom data source integration, custom messaging, and custom menus.
  - Configuring startup parameters for Optegrity.

Version Information

This guide works with Optegrity Version 5.1, which includes these modules:

- G2 Data Source Management (GDSM)
- G2 Data Point Management (GDPM)
- G2 Event and Data Processing (GEDP)
- G2 Event Management (GEVM)
- G2 Run-Time Library (GRTL)
- G2 Reporting Engine (GRPE)
- G2 Engineering Unit Conversions (GEUC)
- SymCure (CDG)

Audience

This guide is for process modelers and application developers who want to learn how to use Optegrity to build abnormal condition management applications. It describes how to create process maps that communicate with external systems, manage data sources, detect events, perform diagnosis, and manage alarms and messages.

Users of this guide should be familiar with process modeling, as well as with Distributed Control Systems such as OPC and PI.

Users of this guide should also be familiar with the general concepts of object-oriented programming, using classes, class hierarchies, and inheritance.

In general, process modelers do not need to be familiar with G2, Optegrity's underlying software environment. However, familiarity with G2 allows modelers to create application-specific custom procedures that the application can execute when various events occur.

For a step-by-step tutorial of an existing Optegrity application, see the *Optegrity Heater Tutorial*.

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, “Burlington, MA”	G2 attribute values and values specified or viewed through dialogs
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
new-argument	User-specified values in syntax descriptions
text-string	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save Properties	GMS and native menu choices
workspace	Glossary terms

Convention Examples	Description
c:\Program Files\Gensym	Windows pathnames
/usr/gensym/g2/kbs	UNIX pathnames
spreadsh.kb	File names
g2 -kb top.kb	Operating system commands
public void main() *gsi_start*	Java, C and all other external code

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure *underlined*. Each value is followed by its type:

g2-clone-and-transfer-objects
(*list*: class item-list, *to-workspace*: class kb-workspace,
delta-x: integer, *delta-y*: integer)
-> *transferred-items*: g2-list

Related Documentation

Optegrity

- *Optegrity Heater Tutorial*
- *Optegrity User's Guide*
- *SymCure User's Guide*

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*

- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System User's Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2-OPC Client Bridge User's Guide*
- *G2-PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2-HLA Bridge User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at *www.gensym-support.com*.

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
Phone	(781) 265-7301	+31-71-5682622
Fax	(781) 265-7255	+31-71-5682621
Email	*service@gensym.com*	*service-ema@gensym.com*

Part I

Getting Started

Chapter 1: Introduction

Describes the modules that make up the Optegrity Bundle.

Chapter 2: Running Optegrity

Describes how to start the server and connect the client.

Chapter 3: Working with Models

Describes how to work with models through the menus and toolbars.

Chapter 4: Creating Optegrity Applications

Describes how to create a new Optegrity application.

1

Introduction

Describes the modules that make up the Optegrity Bundle.

Introduction

Optegrity is an extensible software platform that is used to build abnormal condition management applications for process manufacturing industries. Optegrity applications ensure sustained operational performance and continuous availability of production assets. Its applications detect and resolve abnormal process conditions early—before they disrupt productivity, and weaken product quality and profits.

Optegrity is a powerful application platform for OEMs, value-added-resellers (VARs), system integrators (SIs), consultants, and end users. Combined with existing control systems, data historians, and databases, Optegrity applications work in real time to:

- Proactively monitor process conditions to avoid or minimize disruptions.
- Analyze, filter, and correlate alarms to speed up operator responses.

- Rapidly isolate the root cause of problems to accelerate resolution.
- Guide operators through recovery to enhance safety levels.
- Provide expert guidance so that operators of all skill levels can effectively respond to problems.
- Predict the impact of process disruptions so operators can prioritize actions.

Optegrity consists of a broad set of capabilities for meeting the wide-ranging needs of abnormal condition management, covering the whole life cycle of projects from modeling to validation and deployment. Optegrity monitors and evaluates in real time process conditions received from a DCS (Distributed Control System), provides advisory messages and root cause analysis capabilities. Fault models reason over process maps to reduce the number of operator alarms significantly and to isolate the root cause(s) of problems.

Domain experts such as process engineers can graphically model the logic used to monitor, diagnose, and control the process conditions. Logging and replay capabilities enable domain experts to validate their logic before deploying it or replay historical data for in depth analysis of specific situations. Optegrity also enables domain experts to build libraries of reusable process objects that include a generic formalization of event detection and causal fault models.

Optegrity is a client/server, multi-user environment, enabling multiple users to collaborate during the development and deployment phases of a project. Multiple operators can interact with the server simultaneously to monitor and diagnose in real time the state of the process condition. You can also run Optegrity in a secure G2 environment.

Process Maps and Domain Objects

A **process map** provides a visual representation of your process, where each piece of equipment and each sensor is represented as a **domain object** in your application. External DCS tag variables can be represented as stand-alone domain objects, such as a temperature or flow sensor. They can also be embedded within a domain object, such as a heater.

Optegrity provides built-in domain intelligence for a number of domain objects in the form of:

- **Event detection diagrams**, which monitor real-time or simulated data, and generate events, alarms, and messages when pre-defined conditions specified by the diagrams' logic occur.
- **Diagnostic causal fault models**, which receive events from event detection diagrams and correlate these events to determine root causes and to take corrective actions.

For example, all sensors contain built-in event detection diagrams for the following events: High and Low Limits, Projected High and Low Limits, Noisy, Flatline, and Change. Similarly, heaters detect Heat Release Projected High, Low Efficiency, and Efficiency Severe Change events.

You can extend Optegrity's domain intelligence by defining custom event detection diagrams, fault models, and domain object definitions.

External and Internal Datapoints

Each DCS tag variable is represented in your application as an **external datapoint**. External datapoints obtain data from the external DCS system via an **interface** object, which provides connectivity through a bridge. Optegrity supports connectivity with DCS systems, using G2 OPCLink and the G2-PI Bridge.

External datapoints provide data to a process map via **internal datapoints**. An internal datapoint can be a stand-alone domain object or an embedded datapoint within a domain object.

Internal datapoints provide a link between domain objects and event detection diagrams. They notify event listeners when datapoint values change, and they receive new values from external datapoints. Thus, event detection diagrams monitor internal datapoints to generate events, which may, in turn, trigger diagnostic analysis.

Optegrity Modules

Optegrity consists of the following modules:

This module...	Performs these tasks...
Optegrity	Provides domain object classes for building process maps that integrate with GDSM, GDPM, GEVM, GEDP, and SymCure.
G2 Data Point Management (GDPM)	Configures external datapoints for each DCS tag variable, using CSV (comma-separated value) files. Provides the link between external and internal datapoints within a process map. Configures and performs data validation, data replay, and data logging, all using CSV files.
G2 Data Source Management (GDSM)	Configures the interface object that provides connectivity through a bridge.

This module...	Performs these tasks...
G2 Event Management (GEVM)	Manages low-level notifications and messages, and displays them in various types of message browsers.
G2 Run-Time Library (GRTL)	Manages the user interface.
G2 Event and Data Processing (GEDP)	Defines generic logic for monitoring internal datapoint values and generating low-level notifications, operator messages, and SymCure events that trigger diagnostic reasoning.
SymCure Also known as CDG (Causal Directed Graphs)	Defines generic diagnostic models for domain object classes, which are used to correlate events to determine root causes and to take corrective actions.
G2 Engineering Unit Conversion (GEUC)	Specifies engineering units and conversions for entering and displaying values, as well as a large number of synonyms for those conversions, in both the English and metric systems
G2 Reporting Engine (GRPE)	Manages reports and charts.
Optegrity Events	Contains support for built-in domain object events.
Intelligent Object modules	Provide built-in event detection logic and fault models for various types of domain objects, such as sensors, controllers, heaters, pumps, and so on.

Data Flow and Module Integration

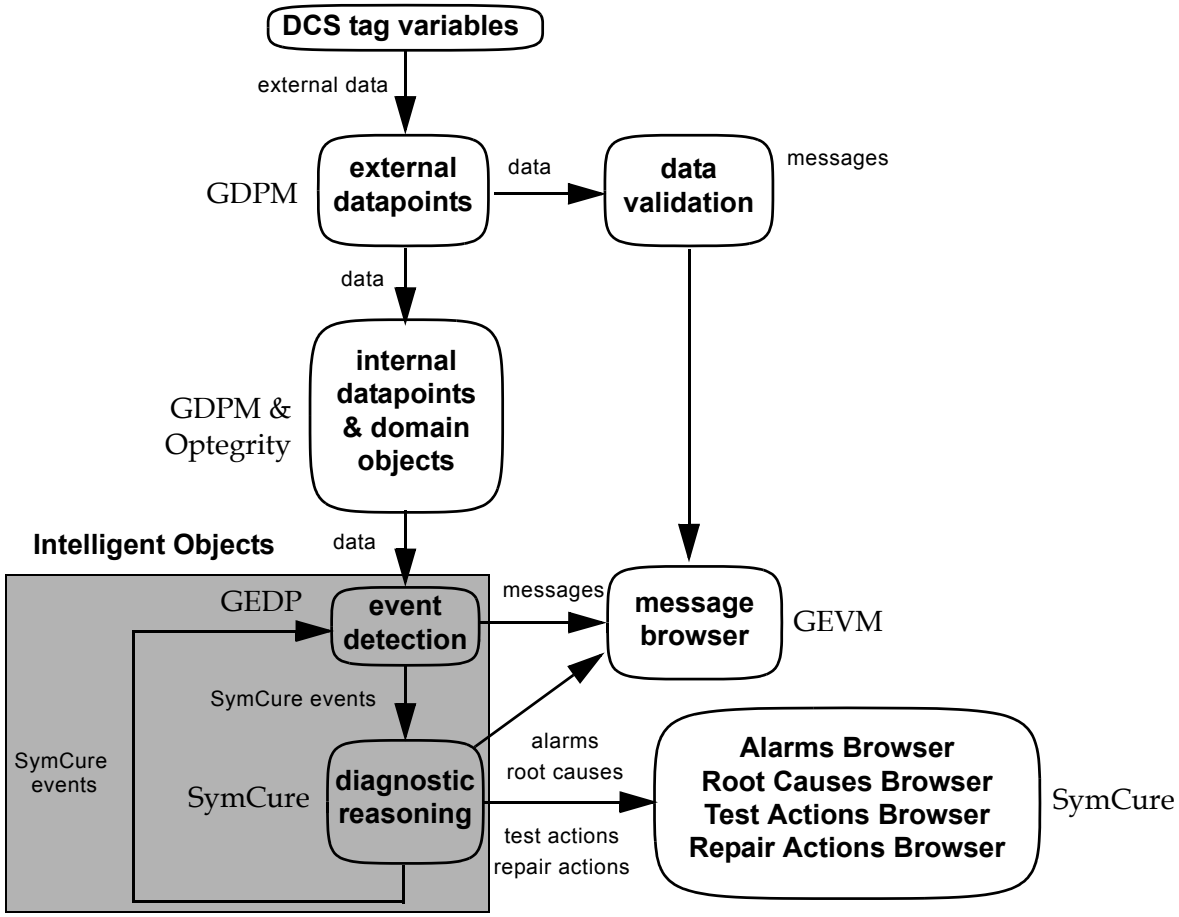
Optegrity integrates its various modules and manages data flow, as follows:

- GDPM creates and configures external datapoints to represent each DCS tag variable. GDPM performs low-level data validation on external datapoints. Messages that result from data validation appear in the Messages browser.
- GDPM defines internal datapoints, which obtain their data from external datapoints. These internal datapoints are embedded in domain objects within a domain map.
- GEDP monitors internal datapoint values and generates SymCure events for specific domain objects. GEDP also generates low-level notifications and operator messages. You can view these messages in the Message browser.

You build generic event detection templates, which apply to classes of domain objects.

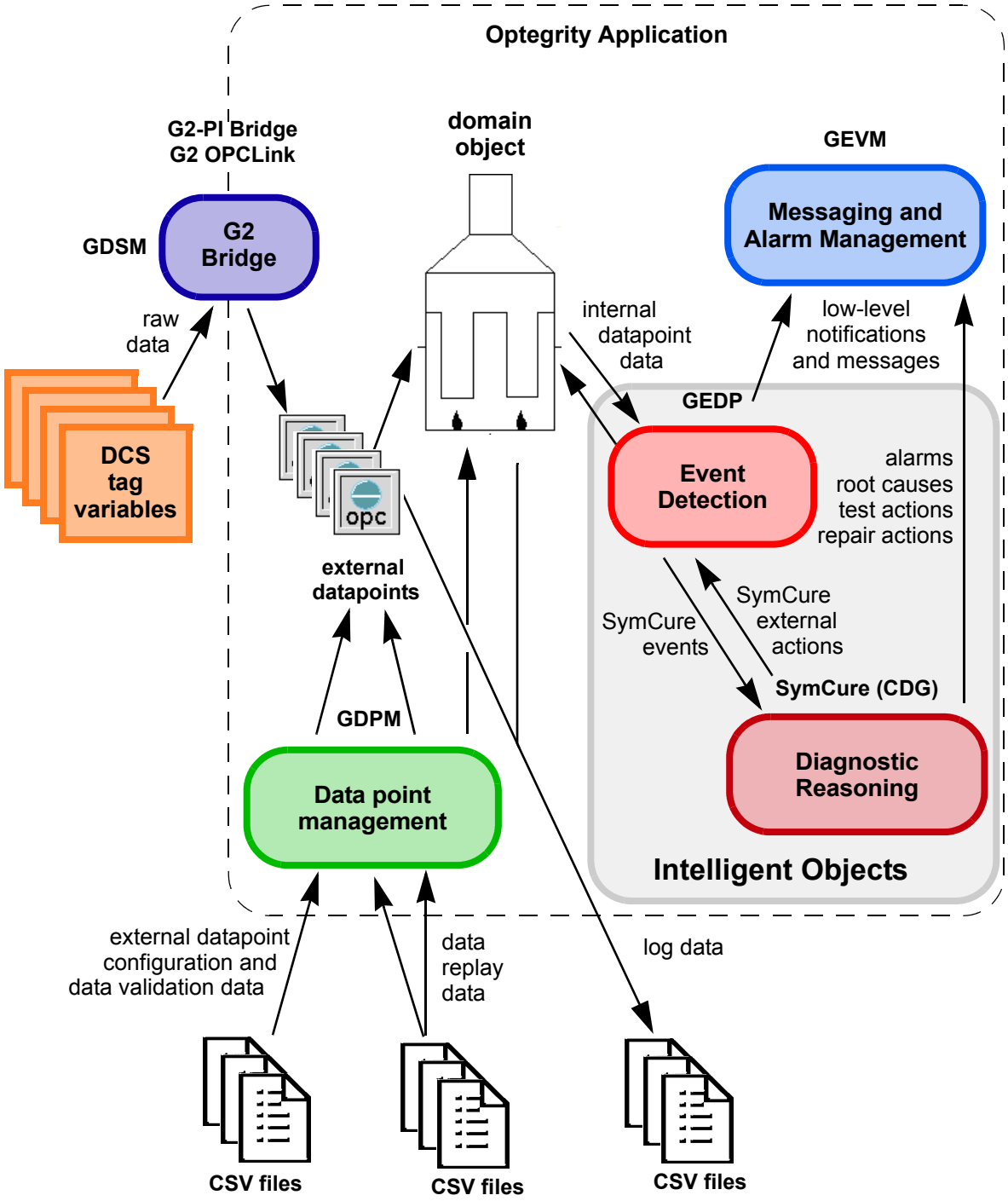
- SymCure correlates events on specific domain objects across your domain map to identify root causes, run tests, and perform repair actions. SymCure also generates messages for alarms, root causes, tests, and repair actions, which are displayed in the diagnostic console browsers and the Messages browser. SymCure fault models can also trigger event detection diagrams to perform actions. You build generic SymCure fault models, which are associated with classes of domain objects.
- The Intelligent Object modules monitor events for the various built-in domain objects.

Here is a graphical representation of data flow and module integration in an Optegrity application:



Architecture

This figure shows the overall architecture of the Optegrity Heater Tutorial:



2

Running Optegrity

Describes how to start the server and connect the client.

Introduction

Optegrity is a client/server application. Optegrity provides a batch file that, by default, starts the G2 server as a hidden process on the local machine at a default port (1111).

To run Optegrity, you must connect the Telewindows client to the server. By default, Telewindows automatically connects to the server running on the local machine on the default port.

You can run Optegrity in a secure G2 environment, which means users must provide a password before Optegrity grants access to a KB. User names and passwords are stored in the `g2.ok` file. For details on how to configure Optegrity to run in a secure G2 environment, see Chapter 54 "Licensing and Authorization" in the *G2 Reference Manual*.

Starting the Server and Connecting the Client

You can start the server and connect the client by using the Start menu.

To start the server and connect the client:

1 Choose Start > Programs > Gensym G2 2011 > G2 Optegrity > Start G2 Optegrity Server.

This menu choice starts the G2 server, using the *StartServer.bat* batch file, located in the *g2* directory of your Optegrity installation directory. It starts the server on the local machine on TCP/IP port number 1111, and it automatically loads the KB named *optegrity.kb*.

When the server has been started, the G2 icon appears in the system tray. When the server is running, the icon looks like this: 

2 Once the server is running, connect the client in one of two ways:

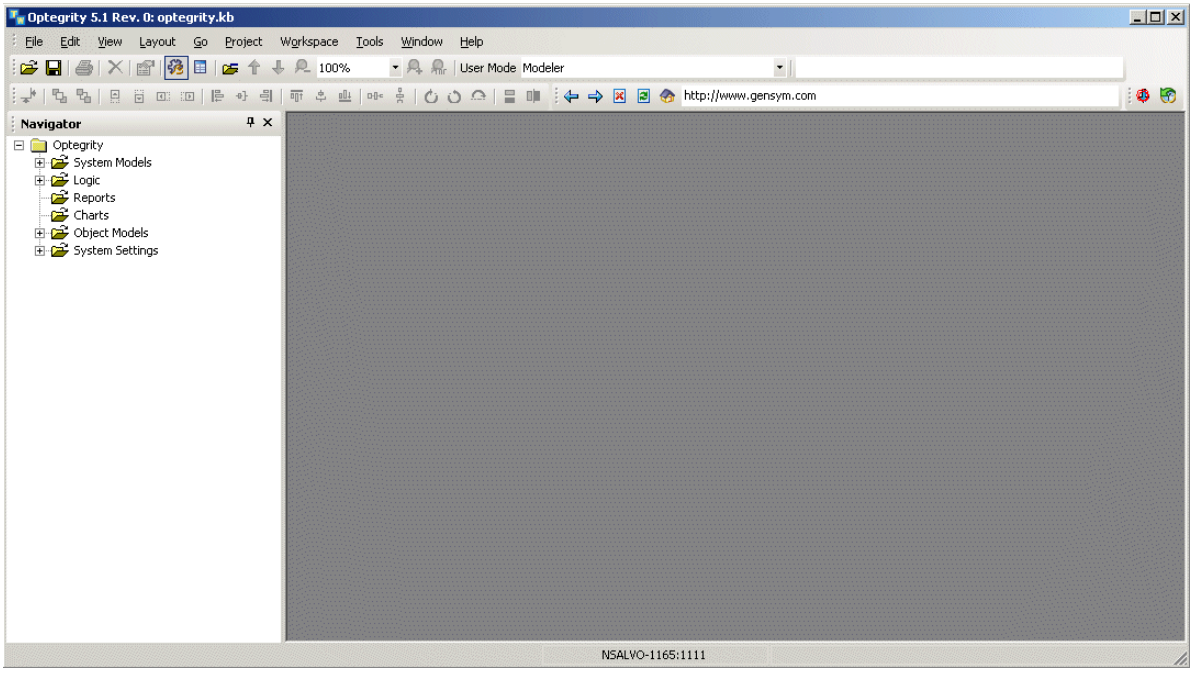
➔ To connect Telewindows to the server running on the default host and port, choose Start > Programs > Gensym G2 2011 > Telewindows Next Generation.

or

➔ To connect Telewindows to the server running on the local host on the current port, right-click the G2 icon in the system tray and choose Connect Telewindows.

The Telewindows client is now connected to the G2 server.

When the client is connected and all files have finished loading, you will see this window:



Optegrity starts in Modeler mode, which is the typical user mode for building models. Optegrity also provides an operator interface for end users, which you access in Operator mode. It also provides user modes for developers and system administrators, in which you can also build models. For details, see Switching User Modes.

Connecting to a Specific Server at Startup

You can run the Optegrity client and server on different computers, or multiple Optegrity servers on the same computer.

You can:

- Connect the client directly to the server.
- Start the server on a specific port.
- Connect the client to a specific server.

Connecting the Client to the Default Server

To connect the client to the default server:

1. Start the Optegrity server from the Start menu.

   By default, the server starts on the local host at port 1111. Each time you start a new server on the same machine, the port number increments by one. For example, if you start another server, the port number would be 1112.

2. To determine the host and port, hover the mouse over the G2 server icon in the system tray.

   For example, *MY-HOST:1111* means the server is running on the machine named *MY-HOST* at port 1111.

3. Right-click the G2 server icon in the system tray and choose Connect Telewindows.

The Telewindows client connects to the specific host and port of that server.

Starting the Server on a Specific Port

To start the server on a specific port:

1. Right-click the Start G2 Optegrity Server menu choice in the Start menu and choose Create Shortcut.
2. Rename the shortcut and/or drag it to your desktop, as needed.
3. Display the properties dialog for the shortcut and click the Shortcut tab.
4. Configure the Target property in the dialog to be the specific port on which to start the server, using the *-tcpport* command-line option.

   For example, to start the server on port 1115, the shortcut would look like this:

   ```
   "C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"
   -kb ..\optegrity\kbs\optegrity.kb -nowindow -tcpport 1115
   ```

Connecting the Client to a Specific Server

To connect the client to a specific server:

1. Create a shortcut to the *twng.exe* file located in the *g2* directory of your Optegrity product installation, either directly or by creating a shortcut from the Telewindows Next Generation menu choice in the Start menu.
2. Display the properties dialog for the shortcut and click the Shortcut tab.

3 Configure the Target by appending the host and port of the Optegrity server to which to connect, using this syntax: *host :port*.

For example, to connect to *my-host* at port *1115*, the shortcut would look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\twng.exe"
   my-host:1115
```

Starting the Server with Your Application Loaded

By default, the server starts up with the default Optegrity application running, *optegrity.kb*. Once you create an application, you might want to create a shortcut to the Optegrity server that automatically loads your application at startup.

To start the server with your application loaded:

1 Copy the Start G2 Optegrity Server shortcut from the Start menu.

You can rename the shortcut and drag it to your desktop, as needed.

2 Display the properties dialog for the shortcut and click the Shortcut tab.

3 Configure the application to load by editing the Target.

For example, to load the application named *f102demo.kb* located in the *\optegrity\examples* directory, the Target should look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"
-kb "c:\Program Files\Gensym\g2-2011\optegrity\examples\
f102demo.kb" -nowindow
```

Starting the Optegrity Server as an NT Service

On Windows platforms, you can run the Optegrity server as an NT service, which means you can control it through the Windows Service Control Panel. It also means you can configure it to start automatically when the computer boots up and restart if the process dies.

You run the Optegrity server as an NT service from a batch file. You can provide the same two arguments that you provide to the batch file or shell script that starts the server:

- A relative or absolute path name to a *.kb* file to load at startup.
- The TCP/IP port number that the server should use to listen for client connections.

To run the Optegrity server as an NT service:

1 Create a shortcut to the *install-dir*`\optegrity\bin\InstallServerAsNTService.bat`.

2 Edit the shortcut to provide these arguments:

install-dir`\optegrity\bin\InstallServerAsNTService.bat` *file port*

For example, this command launches the server as an NT service from the default installation directory with the `f102demo.kb` loaded at port 1112:

```
"c:\Program Files\Gensym\g2-2011\optegrity\bin\
InstallServerAsNTService.bat" "..\examples\f102demo.kb 1112"
```

To uninstall the Optegrity server as an NT service:

➔ Run *install-dir*`\optegrity\bin\UninstallServerAsNTService.bat`.

Exiting Optegrity

To exit Optegrity, you disconnect the client from the server, then shut down the server. By default, you can only exit the server directly from the client in Developer mode.

To disconnect the client from the server:

➔ Choose File > Close.

To exit the server:

➔ Right-click the G2 server icon in the system tray and choose Shut Down G2.

or

1 Choose Tools > User Mode > Developer.

2 Choose File > Exit.

3 Click Yes in the confirmation dialog.

3

Working with Models

Describes how to work with models through the menus and toolbars.

Introduction

To work with Optegrity models, you perform these tasks:

- Use the Project menu.
- Navigate models.
- Interact with workspaces.
- Use the menus.
- Use the Optegrity toolbox.
- Use the G2 Toolbox.
- Interact with objects in the model.
- Use toolbars.
- Switch user modes.
- View messages.

You can also view a summary of command tasks.

Summary of Common Tasks

This section summarizes how to perform common tasks in Optegrity:

To...	Do this...
Display the popup menu for an object on a workspace	Click the right mouse button on the object.
Display the properties dialog for an object on a workspace	Double-click the object, select the object and press the F4 key, or choose Properties from the object's popup menu. You can also select the item, then choose Edit > Properties or click the equivalent toolbar button: 
Display the detail for an object	Choose Show Detail from the popup menu for the object, choose View > Show Details, enter Ctrl + right click on the object, or click the equivalent toolbar button: 
Display the Optegrity toolbox	Choose View > Toolbox - Optegrity.

To...	Do this...
Adjust the size of a workspace and its associated window to fit the contents of the workspace	Choose Shrink Wrap on the workspace, choose Layout > Shrink Wrap, or click the equivalent toolbar button: 
Hide a workspace	Click the Minimize button on the window, choose Hide on the workspace, choose View > Hide, or enter Ctrl + right click on the workspace.

Using the Project Menu

Optegrity provides two basic types of objects, which you use to create an application:

- Configuration objects, which you use to configure various aspects of the model:
  - Network interfaces configure connectivity with various types of bridges.
  - External datapoints configure the external data sources your application monitors and manages.
  - Data logging configures default behavior for logging internal and external datapoints in a process map.
  - Continuous and differential data series configure CSV files from which to replay data.
  - Data replay configures data series for replaying data.
  - Data simulations allow you to simulate external and internal datapoint values.
- Container objects, which have details on which you place objects cloned from palettes in one of the various Optegrity toolboxes:
  - Process maps contain connected domain objects that describe the process you are managing. You create domain objects from the palettes in the Process Modeling toolbox.
  - Event detection templates and diagrams contain GEDP blocks that describe event detection models for domain object classes and instances. You create GEDP blocks from the palettes in the Event Detection toolbox.
  - Generic fault model folders contain diagnostic models and external actions that perform root cause analysis. You create generic fault models from the palettes in the Fault Modeling toolbox.

You create, configure, and interact with Optegrity objects to create a model by using the Project menu.

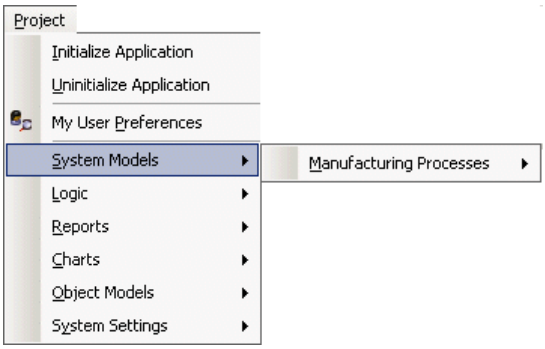
You can also create and interact with objects through the Navigator, and you can search for objects once they exist. For more information, see:

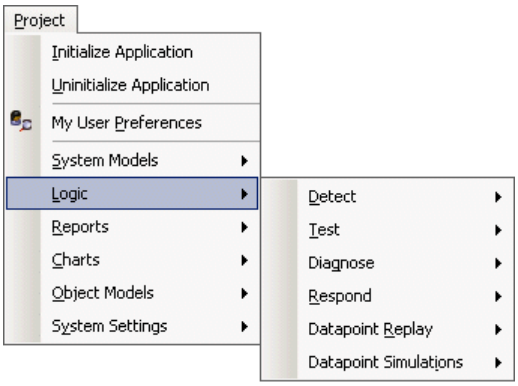
- Using the Navigator.
- Searching for Objects.

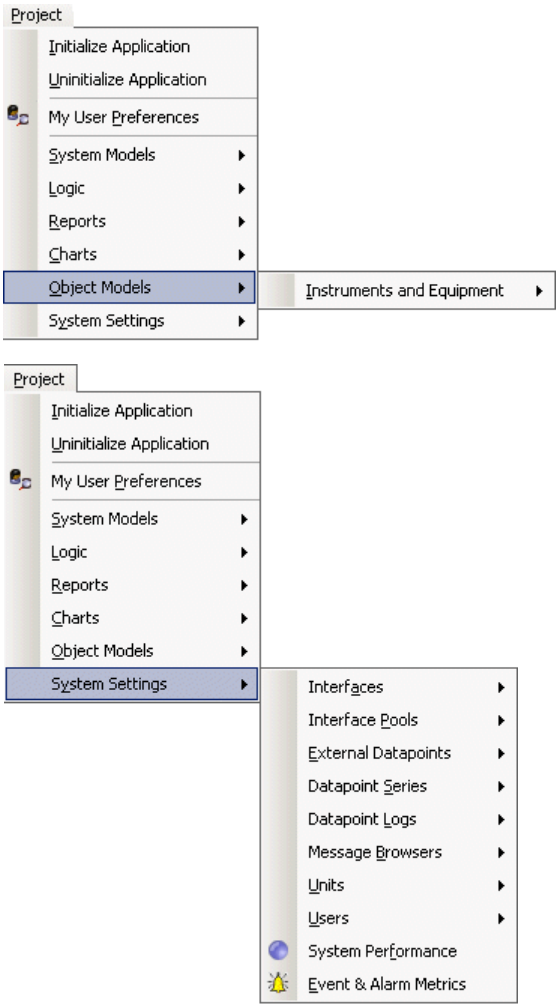
Using the Project Menu

The Project menu allows you to create and manage the various objects you need to build an Optegrity application.These include network interfaces, external datapoints, manufacturing process maps, datapoint series for data replay, datapoint logging, data replay, datapoint simulations, event detection diagrams, and SymCure fault models.

Here is the Project menu with its various submenus:





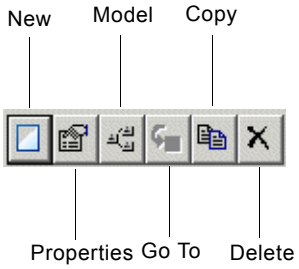


For details, see Using the Project Menu.

Using the Manage Dialog

The Manage dialog allows you to create and configure new Optegrity objects, show model details, copy and delete objects, and perform specific operations.

The Manage dialog provides these toolbar buttons:



The buttons in the Manage dialog are enabled or disabled, as appropriate, for the particular type of object. For example, the Model button is enabled for process maps, event detection models, and fault models, but not for interfaces, external datapoints, datapoint replay, datapoint series, or datapoint logs.

The Go To button is disabled in Modeler mode because, typically, you interact with objects through properties dialogs and model details. You can go to objects directly through the Navigator or search, if desired.

For information about interacting with objects directly, see Interacting with Objects in Developer Mode.

To use the Manage dialog:

1. Choose a submenu from the Project menu and choose Manage.

   If the submenu has additional submenus, choose one of the submenus. The Manage dialog appears, which includes all objects in the submenu.

2. To create a new object, click the New button in the Manage dialog.

   A properties dialog appears for configuring the object. The default name is a unique, system-generated name.

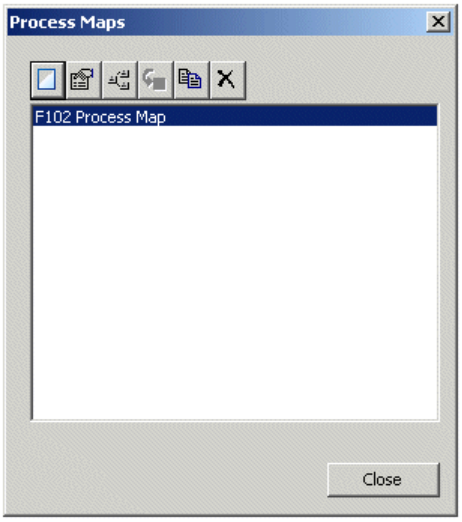
3. Configure the properties, depending on the type of object, and click OK.

   For information on configuring the properties, see the various chapters in this guide.

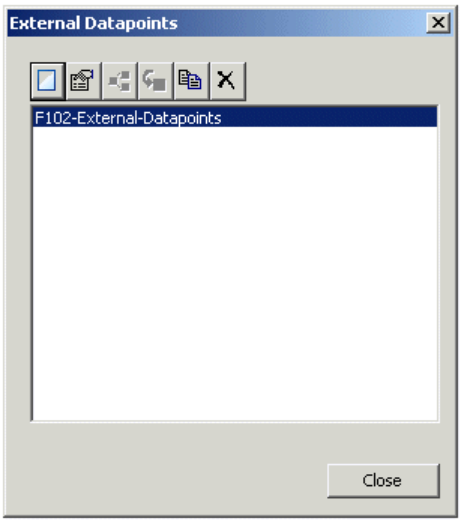
   The object now appears in the Manage dialog.

4 Select an object in the list to enable the toolbar buttons, as appropriate for the type of object.

For example, here is the Process Maps Manage dialog when the *f102demo.kb* application is loaded. Notice that the Model button for showing the detail is active because process maps are containers.



Here is the External Datapoints Manage dialog with the *f102demo.kb* application is loaded. Notice that the Model button is not active because external datapoints is a configuration object, which does not have detail.



5 To display the properties dialog for an object, click the Properties button.

Note that the only way to configure the properties of a container object once it has been created is through the Manage dialog.

6 To display the detail associated with a container object, click the Model button.

This button is only available for container objects; configuration objects do not have detail.

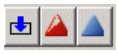
7 To copy an existing object, select the object you want to copy, then click the Copy button.

A properties dialog appears for configuring the copy. The default name is the existing object name with `-copy` appended.

8 To delete an object, select the object you want to delete and click the Delete button.

Performing Specific Operations

The dialog that appears when you choose Manage contains additional buttons for interacting with specific features of the selected object, as appropriate. The following objects have these additional buttons:

This object...	Has these buttons...	Which...
Interface		Connect to and disconnect from the external bridge (Developer mode only). For more information, see Configuring Network Interfaces.
Datapoint Replay		Start, Stop, Pause, and Resume, which control the status of the datapoint replay. For more information, see Replaying Data.
Datapoint Simulations		Activate and deactivate datapoint simulation. For more information, see Simulating Datapoint Values.
Generic Fault Model		Which compiles the generic fault model, and displays errors and warnings that occur during the compile. For more information, see Running SymCure Fault Models.
Message Queue		Shows the browser for the selected message queue. For more information, see Interacting with Operator Messages.

Using the Project Submenus

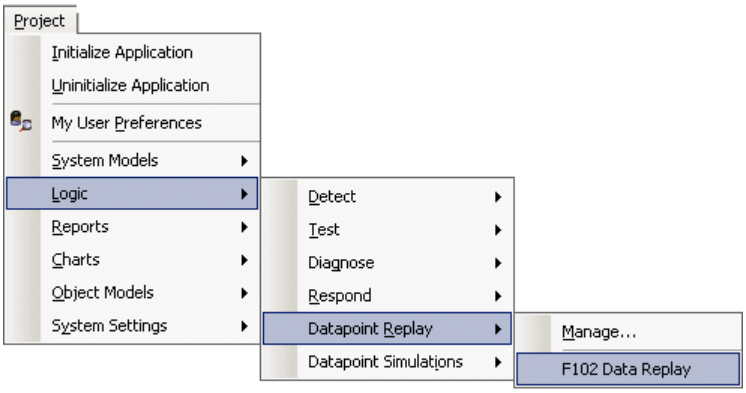
Optegrity provides access to the various objects in a model through submenus of the Project menu. Selecting the menu choice for a configuration object displays the properties dialog for the object. Selecting the menu choice for a container object displays its detail.

To use the project submenus:

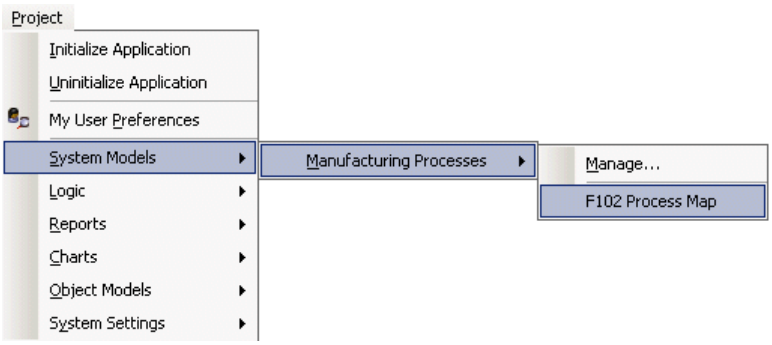
1 Choose a submenu from the Project menu.

If the submenu has additional submenus, choose a submenu until you see a submenu that includes the names of all objects of that type.

For example, here is the Datapoint Replay submenu when the *f102demo.kb* is loaded, which includes F102 Data Replay:

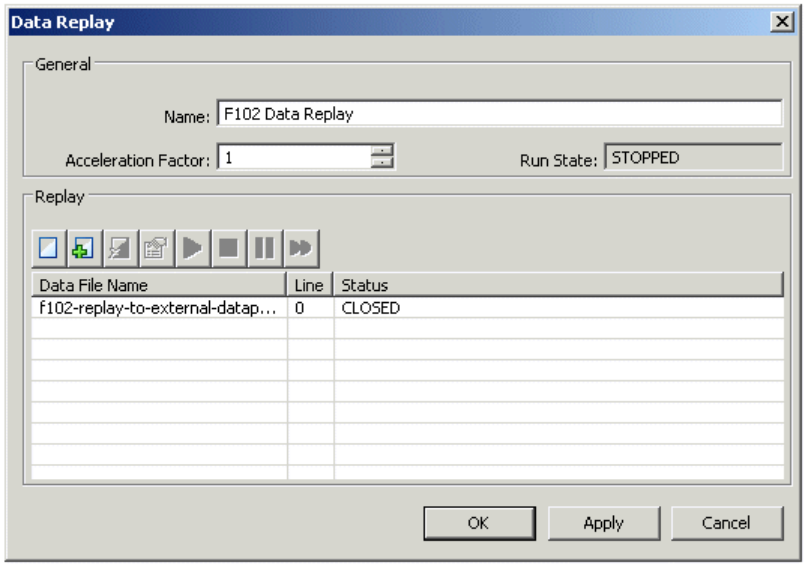


Here is the System Models > Manufacturing Processes submenu for the *f102demo.kb*, which includes F102 Process Map:

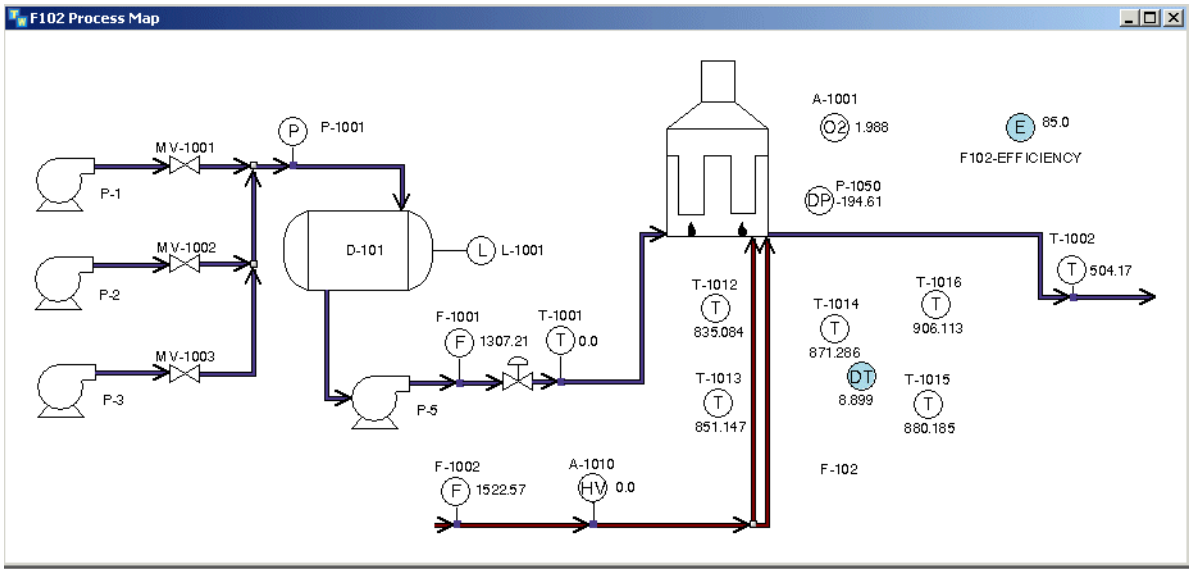


2 Choose an object from the submenu.

For example, choosing F102 Data Replay displays this properties dialog:



Choosing F102 Process Map displays the process map detail:



To configure the properties of a container object, such as a process map, once you have created it, you must use the Manage dialog. For details, see Using the Manage Dialog.

Navigating Applications

To navigate applications, you can:

- Use the Navigator.
- Search for objects.

For information on creating and managing objects through the Project menu, see Using the Project Menu.

Using the Navigator

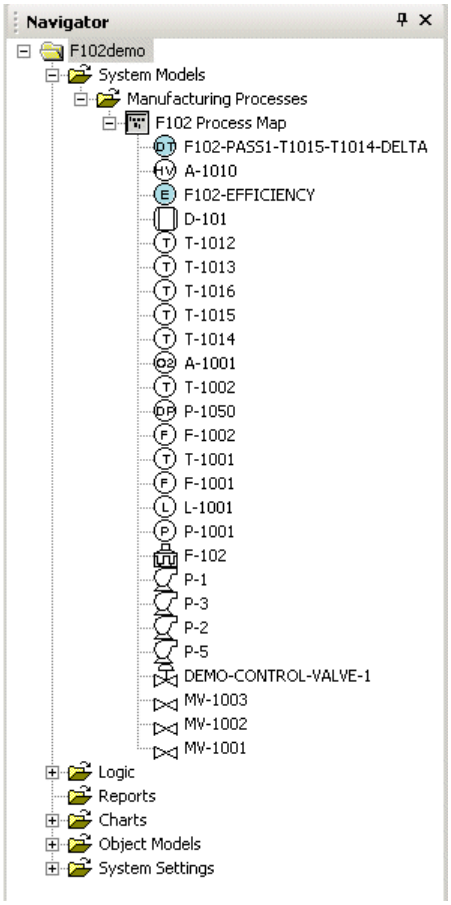
The Navigator displays all the elements of a project.

You can interact with objects in the Navigator, for example, showing its properties or going to the detail, depending on the type of object. You can also create new objects from the Navigator.

To display the Navigator:

➔ Choose View > Navigator or click the equivalent toolbar button () and expand the tree view to the desired level.

Here is the Navigator for the F102demo application with the tree expanded to show the detail of the F102 Process Map:



To interact with objects in the Navigator:

➔ Right-click a node in the Navigator and choose the desired menu choice.

In addition to the menu choices that you normally get when you right-click the object, you can choose Go To to show the selected object. Depending on the type of object, you might go to the object on a detail or you might go to the object in a repository.

You can also choose New Instance on the Manufacturing Processes folder to create a new process map directly from the Navigator.

Searching for Objects

You can search for specific types of objects, by matching text in the label field and/or the target class, depending on the type of object. You can also go directly to named objects.

To search for objects:

1. Choose Tools > Search and choose a category of object to be found.
2. Enter the Keyword text to match and, depending on the type of object, optionally, the Target Class.
3. Configure Search By to search for the keyword only, class only, keyword or class, or keyword and class.
4. Click the Search button.

   The search results include all objects whose label matches the specified text.
5. Select an object and click the Go To button.

An arrow appears next to the found object, if it exists; otherwise, the Search dialog display No Matches Found.

Depending on the type of object, you might go to the object on a detail or you might go to the object in a repository. You can interact with the object through its menu choices, for example, to go its detail or show its properties.

To go to a named object in the model:

➔ Enter the exact name of an object in the Go To type-in box on the toolbar:

A red arrow points to the named object on a workspace.

Interacting with Workspaces

You place all model objects on detail workspaces, which appear their own window. You display and interact with workspaces in these ways:

- Display a detail workspace.
- Hide a workspace.
- Delete a workspace.
- Edit workspace properties.
- Scale a workspace.
- Shrink wrap a workspace to fit the enclosed elements.

- Show the superior object for a workspace.
- Print a workspace.
- Save a workspace as a JPEG file.
- Assign a background image to a workspace.
- Create and access top-level workspaces.
- Initialize domain objects and specific fault models.

Displaying a Detail Workspace

A number of objects define detail, which is a workspace associated with the object on which you place other objects. For example, process maps, event detection diagrams, and fault models all define detail.

To display detail for an object:

➔ Right-click the background of a workspace and choose Show Detail, choose View > Show Details, or click the equivalent toolbar button: ()

Hiding a Workspace

To hide a workspace:

➔ Right-click the background of a workspace and choose Hide or press Ctrl + right-click on the workspace.

Deleting a Workspace

Deleting a workspace permanently deletes it from the server, including all objects on the workspace.

To delete a workspace:

➔ Select a workspace and choose Edit > Delete, right-click the background of a workspace and choose Delete, or click the equivalent toolbar button: ()

Editing Workspace Properties

You can edit the name of the workspace, as well as the background and foreground colors, and the margins around the objects at the edges of the workspace. By default, the background color is white and the foreground color is black.

For information about configuring the background image, see Loading Background Images.

To edit workspace properties:

1. Select a workspace and choose Edit > Properties, right-click the background of a workspace and choose Properties, or click the equivalent toolbar button: ()
2. Configure the Names to be any text.

   The text is converted to a symbol, with hyphens in place of spaces when you accept the dialog.
3. Configure the Workspace Margin by entering the number of pixels.
4. Configure the Foreground Color and Background Color by choosing a color.

   By default, in Operator mode, the workspace does not scale to fit the entire screen.
5. Enable the Auto Scale to Full Screen option to fill the screen in Operator mode, as needed.

The name appears at the top of the workspace when you accept the dialog.

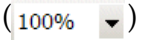
Scaling a Workspace

You can scale a workspace to fit the current window, or zoom a workspace in, out, or to a specific scale.

To scale a workspace:

➔ Choose View > Zoom In or Zoom Out, enter Ctrl + = to zoom in or Ctrl + - (minus) to zoom out, or click the equivalent toolbar buttons: ( )

or

➔ Choose View > Zoom, then choose or enter a zoom scale, or enter a specific zoom scale in the zoom scale on the toolbar: ()

or

➔ Choose View > Zoom to Fit or click the equivalent toolbar button: ( )

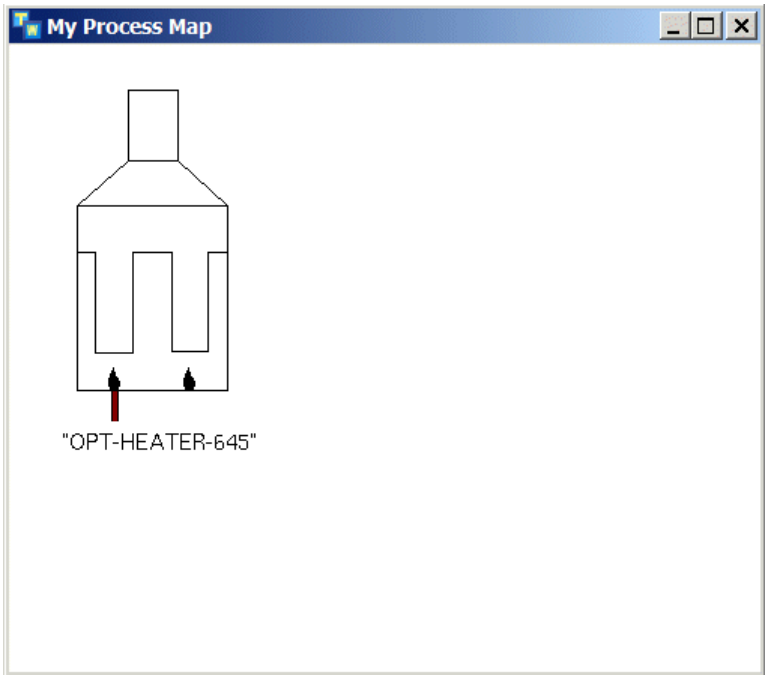
Shrink Wrapping a Workspace

When you move objects on a workspace beyond the visible borders, the borders adjust to fit the objects. When you move objects on a workspace such that the workspace contains extra space at its borders, you can adjust the borders by shrink wrapping the workspace. Shrink wrapping a workspace also adjusts the window size. You can resize the window to make it smaller to add scroll bars to the window.

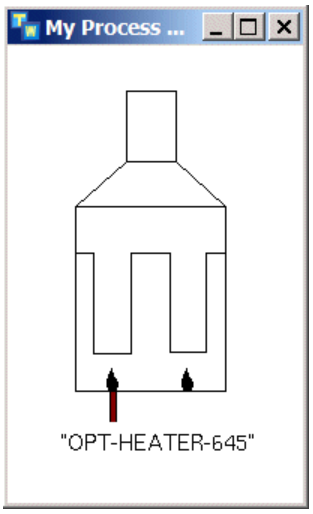
To shrink wrap a workspace:

➔ Select a workspace and choose Layout > Shrink Wrap or click the equivalent toolbar button: ( )

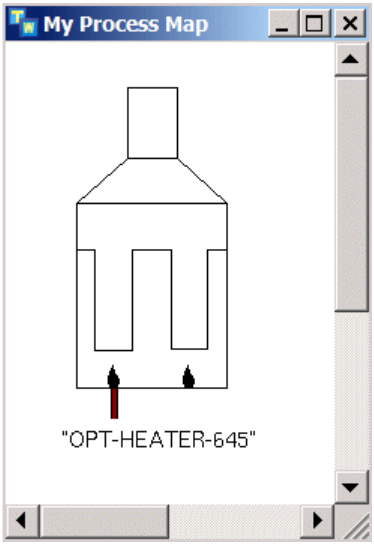
This figure shows a workspace that has extra space around its borders:



This figure shows the result of shrink wrapping the workspace:



This figure shows the result of dragging the object on the workspace so it has extra space around its borders, then adjusting the window size to make it smaller, which adds scroll bars:



Showing the Superior Object of a Detail Workspace

You can show the superior object of a detail workspace, for example, the detail of a Production Site, Production Area, or Process Unit.

To show the superior object of a detail workspace:

➔ Right-click the background of a workspace and choose Go to Superior, or select a detail workspace and choose View > Go to Superior or click the equivalent toolbar button: ()

The workspace with the superior object is now on top with an indicator arrow next to the object.

Depending on the type of object, you might go to the object in a repository. You can interact with the object through its menu choices, for example, to show its properties.

Printing a Workspace

To print a workspace:

➔ Choose File > Print, or enter Ctrl + P or click the equivalent toolbar button (), and configure the Print dialog.

Saving a Workspace to a JPEG File

To save a workspace to a JPEG file:

➔ Choose File > Save as JPEG and specify a file name.

Loading Background Images

You can load one or more JPEG, XMB, or GIF files as the background for a workspace.

To load a single background image:

➔ Choose Workspace > Load Background Image, navigate to the image to use as the background, and click Open.

To remove background images:

➔ Choose Workspace > Delete Background Image.

Creating and Accessing Top-Level Workspaces

Typically, you create new workspaces when you create process maps through the Project menu. However, you can also create new workspaces directly through the

Workspace menu, which are top-level workspaces that you can access by name.

To create a new top-level workspace:

1 Choose Workspace > New.

The workspace is assigned a unique number, which starts with unnamed-workspace.

2 Configure the workspace properties as described in Editing Workspace Properties.

To access the top-level workspace:

1 Choose Workspace > Get or click the equivalent toolbar button: ( )

A list of all top-level workspaces available in the current user mode appears.

2 Select a workspace and click OK.

Initializing Domain Objects and Specific Fault Models

Process map details provide these menu choices for initializing domain objects:

- Initialize Domain Objects — Initializes all domain objects on the process map, which creates a specific GEDP model for each domain object, as well as performing various other required tasks. For details, see Initializing Process Maps.
- Uninitialize Domain Objects — Uninitializes all domain objects on the process map, which performs various tasks, including deleting specific GEDP models for each domain object. For details, see Uninitializing Process Maps.

Using the Menus

The top-level menu bar consists of these menus:

Menu	Description
File	Standard file operations, and print and export operations for workspaces.
Edit	Standard editing operations for objects on workspaces.
View	Display the various toolboxes and toolbars, display the Navigator, zoom workspaces, show details, and show superior objects.

Menu	Description
Layout	Standard layout operations for objects on workspaces, including align, distribute, rotate, reflect, order, nudge, as well as shrink wrapping workspaces.
Go	Standard browser navigation operations and interaction with the server.
Project	Manage system models, object models, reports, charts, system settings, and user preferences.
Workspace	Create new and get existing workspaces, and edit background images for workspaces.
Tools	Find model objects, show users, and switch user modes.
Window	Control window positioning and choose the active window.
Help	Display online help.

The following sections summarize each of these menus.

For information about how to use specific menu choices, see the referenced sections.

For information about additional menu choices available in Developer mode, see the Customizing Optegrity.

Using the File Menu

The File menu allows you to perform basic file and module operations.

Menu Choice	Description
New	Creates a new project. See Working with Projects.
Open	Opens an existing project, replacing the one currently loaded.
Save	Saves the top-level module of the current project.

Menu Choice	Description
Save As	Saves the top-level module of the current project to a user filename. You save models to filenames with a *.kb* extension.
Save as JPEG	Exports the currently selected workspace as a *.jpg* file.
Print	Prints the currently selected workspace to a postscript printer.
Close	Exits the client.

Using the Edit Menu

The Edit menu allows you to perform basic edit operations for objects.

Menu Choice	Description
Delete	Deletes the selected object.
Transfer	Transfers the selected object to the mouse. Click on a workspace to transfer the object.
Clone	Transfers the selected object to the mouse. Click on a workspace to clone the object.
Select All	Selects all objects on a workspace.
Properties	Displays the properties dialog for the selected object.
Colors	Changes the colors of the icon regions of the selected objects.

Using the View Menu

The View menu allows you to show and hide toolboxes and toolbars, and to control the zoom scale.

For details about each of the toolboxes, see Using the Optegrity Toolboxes.

The View menu contains the menu choices in the following table:

Menu Choice	Description
Toolbars > Standard	Toggles the Standard toolbar, which contains standard buttons for file and edit operations.
Toolbars > Layout	Toggles the Layout toolbar, which contains buttons for performing standard layout operations for objects on workspaces.
Toolbars > Web	Toggles the Web toolbar, which contains standard buttons for browsing HTML and text pages.
Toolbars > Fault Modeling	Toggles the Fault Modeling toolbar, which contains buttons for configuring SymCure models.
Toolbars > Operator Toolbar	Toggles the Operator toolbar, which contains buttons that provider tools for operators.
Status Bar	Toggles the status bar, which displays the connection status to the server.
Message Board	Displays the G2 Message Board, which displays text messages.
Message Browser	Displays a message browser of operator messages.
Navigator	Toggles the display of a tree view of all objects in the current project. See Navigating Applications.
Toolbox - Event Detection	Toggles the display of the Event Detection toolbox, which contains blocks for creating GEDP dataflow diagrams for event detection, testing, and response.
Toolbox - External Datapoints	Toggles the display of the External Datapoints toolbox for manually creating external datapoints.
Toolbox - Fault Modeling	Toggles the display of the Fault Modeling toolbox, which contains events and other object used for creating generic fault modeling templates for SymCure diagnostics.

Menu Choice	Description
Toolbox - Process Modeling	Toggles the display of the Process Modeling toolbox, which contains equipment and instruments for creating manufacturing process maps.
Zoom Zoom In Zoom Out Zoom to Fit	Scales the selected workspace.
Hide	Hides the currently selected workspace.
Go to Superior	Displays the superior object of the currently selected workspace.
Show Details	Shows the detail workspace of the currently selected object.

Using the Layout Menu

The Layout menu allows you to interact with objects on workspaces. For details, see Interacting with Objects.

Menu Choice	Description
Order > Bring to Front Send to Back	Controls the stacking order of selected objects on workspaces.
Nudge > Nudge Up Nudge Down Nudge Right Nudge Left	Micro-adjusts the position of selected objects in each direction.

Menu Choice	Description
Align or Distribute > Align Left Align Center Align Right Align Top Align Middle Align Bottom Distribute Horizontally Distribute Vertically	Aligns two or more selected objects along various axes. Distributes three or more selected objects vertically or horizontally.
Rotate or Flip > Normal 90 Clockwise 90 Counterclockwise 180 Flip Horizontally Flip Vertically	Rotates and reflects the selected objects.
Shrink Wrap	Adjusts the borders of the selected workspace to just fit the contained objects.

Using the Go Menu

The Go menu allows you to perform standard browser navigation and interact with the server.

Menu Choice	Description
Back Forward Stop Refresh Home	Provides standard browser operations for HTML and text pages.

Using the Project Menu

The Project menu allows you to interact with all the objects in the current project, as follows:

Menu Choice	Description
Initialize Application Uninitialize Application	Initializes all process maps in the application, which creates specific GEDP diagrams for each domain object with an associated generic diagram template, resets datapoint histories, compiles all SymCure diagrams, and clears all diagnoses from the various message browsers. Uninitialize deletes specific GEDP diagrams that Optegrity creates for domain objects associated with generic diagram templates. See Initializing Process Maps.
My User Preferences	Configures user preferences for the current user. See Configuring User Preferences.
System Models > Manufacturing Processes	Creates manufacturing process maps, which consist of domain objects that are created from the Process Modeling toolbox. See Building a Process Map.

Menu Choice	Description
Logic > Detect > Dataflow Templates Dataflow Instances Test > Dataflow Templates Dataflow Instances Diagnose > Fault Models Diagnosis Managers Diagnostic Console Debug Specific Fault Models Import Enable Tuning Respond > Dataflow Templates Dataflow Templates Datapoint Replay Datapoint Simulations	Creates and manages Optegrity logic models that detect, test, and respond to abnormal conditions, using GEDP dataflow models, and that diagnose faults, using SymCure fault models. Also creates and manages datapoint replay and simulations. For information on dataflow templates, see Creating Generic Dataflow Diagrams. For information on the menu choice in the Diagnose menu, see Creating Generic Fault Models and the *SymCure User's Guide*. For information on data replay, see Replaying Data. For information on data simulations, see Simulating Datapoint Values.
Reports	Creates and manages a variety of reports. See Reporting and Charting.
Charts	Creates and manages various types of charts. See Reporting and Charting.
Object Models > Instruments and Equipment	Creates and manages domain objects. See Creating Domain Object Definitions.

Menu Choice	Description
System Settings	Creates and manages the various system settings described below.
System Settings > Interfaces > OPC PI SQL SMTP JMS HTTP	Creates and manages network and database interface objects for communicating with various types of external systems. See Configuring Network Interfaces.
System Settings > Interface Pools > OPC PI SQL SMTP JMS	Creates and manages network and database interface pools for communicating with various types of external systems. See Using Interface Pools.
System Settings > External Datapoints	Creates and manages external datapoints, which obtain data from external systems via interfaces. See Configuring External Datapoints.
System Settings > Datapoint Series > Continuous Differential	Creates and manages continuous and differential data series for datapoint simulation. See Replaying Data.
System Settings > Datapoint Logs	Creates and manages logging for external and internal datapoints. See Configuring Logging.

Menu Choice	Description
System Settings > Message Browsers > Queues Events Messages Access Tables Templates	Creates and manages custom message browsers and queues. See Using Message Queues.
System Settings > External Datapoints	Creates containers for storing external datapoints, which get their data through a DCS interface. See Configuring External Datapoints.
System Settings > Datapoint Series	Creates continuous and differential datapoint series, which are used for replaying data. See Replaying Data.
System Settings > Datapoint Logs	Creates logs for internal and external datapoints. See Configuring Logging.
System Settings > Units Converter Conversions Synonyms	Creates and manages engineering unit conversions and synonyms, and provides a unit converter. See Converting Engineering Units.
System Settings > Users	Creates and manages user preferences. See Configuring User Preferences.

Menu Choice	Description
System Settings > System Performance	Enables, disables, and configures system performance metrics. See Reporting and Charting.
System Settings > Event and Alarm Metrics	Enables and disables event and alarm metrics. See Reporting and Charting.

Using the Workspace Menu

The Workspace menu allows you to interact with workspaces. For details, see Interacting with Workspaces.

Menu Choice	Description
New	Creates a new workspace.
Get	Displays a list of named workspaces, which you can display.
Load Background Image Delete Background Image	Loads and deletes background images for the selected workspace.

Using the Tools Menu

The Tools menu allows you to browse objects in the model.

Menu Choice	Description
Search	Allows you to search for objects in a model by name or label. See Searching for Objects.

Menu Choice	Description
Show Users	Shows the users currently logged into the server.
User Mode > Administrator System-Administrator Developer Modeler Operator	Changes the user mode. The default user mode is Modeler, which allows you to create models by copying, connecting, and configuring objects, and to run simulations. Operator mode allows end users to view models only. Developer mode allows developers to customize the application. **Note:** In general, you work in Modeler mode. Very occasionally, modelers need to switch to Developer, Administrator, or System Administrator mode to perform particular tasks. See Switching User Modes.

Using the Help Menu

The Help menu allows you to access online help that displays as a window within the client:

Menu Choice	Description
G2 Help Topics	Display the G2 online help.
Optegrity Help Topics	Displays the Optegrity online help.
Server Information	Displays version information about the server.
About G2	Displays the G2 title block, which shows the current version.
About Optegrity	Displays the Optegrity title block, which shows the current version.

You can view PDF versions of the following guides:

- *Optegrity Heater Tutorial*
- *Optegrity User's Guide*

- *SymCure User's Guide*

To view the online manuals:

➔ Choose Start > Programs > Gensym G2 2011 > Documentation > G2 Optegrity and choose the manual you want to view.

Using the Optegrity Toolboxes

The Optegrity toolboxes contain all of the objects that you use to create a model.

Optegrity provides the following toolboxes:

- Toolbox - Event Detection — G2 Event and Data Processing (GEDP) blocks for building dataflow models for event detection, testing, and response.

  See Creating Generic Dataflow Diagrams.

- Toolbox - External Datapoints — OPC and PI external datapoints for representing DCS tag variables; data driver objects for simulating internal or external datapoint data, and a customization procedure for value translations.

  See Configuring External Datapoints

- Toolbox - Fault Models — SymCure generic events and actions for building generic fault models that perform diagnostic reasoning, testing, and repair actions, and include user-defined procedures and methods.

  See Creating Generic Fault Models

- Toolbox - Process Modeling — Built-in process equipment, sensors, and controllers used for building process maps. The Process Modeling toolbox contains additional palettes when the intelligent object libraries are loaded.

  See Building a Process Map.

- Toolbox - G2 — G2 objects for advanced modeling (Developer mode only).

  See Using the G2 Toolbox

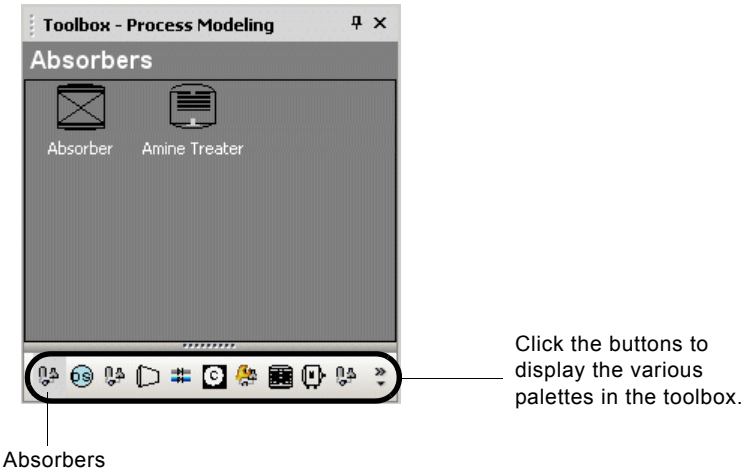
The following examples show the Process Modeling toolbox when all the intelligent object libraries are loaded. For details, see Working with Projects.

To display and interact with the Optegrity toolboxes:

1 Choose a toolbox from the View menu.

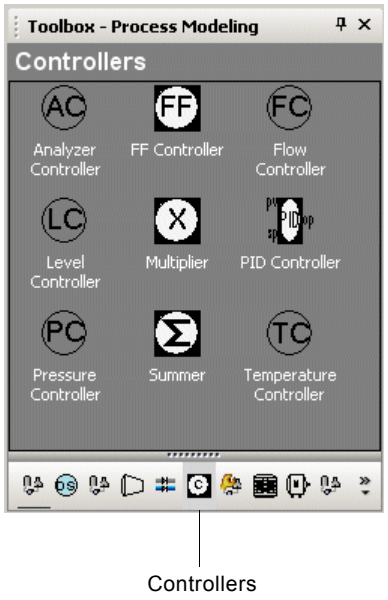
The toolbox appears with the first palette in the toolbox visible. The palettes are organized alphabetically. You access the various palettes in the toolbox by clicking the buttons at the bottom of the toolbox.

Here is the Absorbers palette of the Process Modeling toolbox:



2 To access the various palettes in the toolbox, hover the mouse over a button to display its tool tip, then click the button to display the palette.

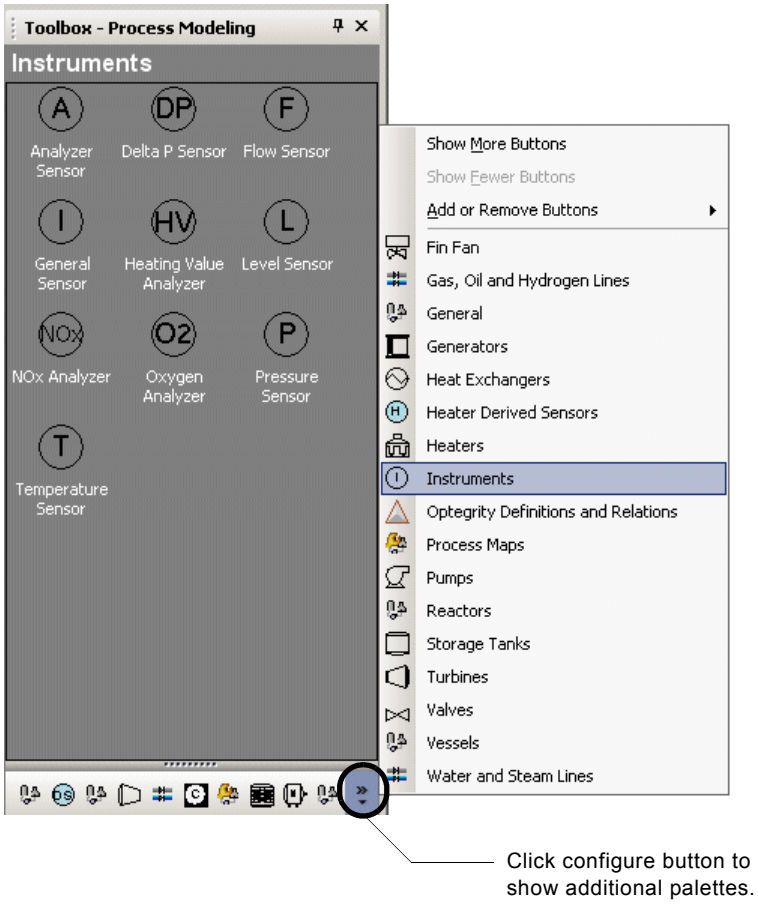
For example, here is the Controllers palette:



Depending on the size of toolbox, the toolbar at the bottom shows only a subset of the available buttons in the toolbox.

3 To display the additional buttons in the toolbox, click the configure button at the far right of the toolbar (»), then choose a palette.

For example, here is how you would display the Instruments palette in the Process Modeling toolbox:



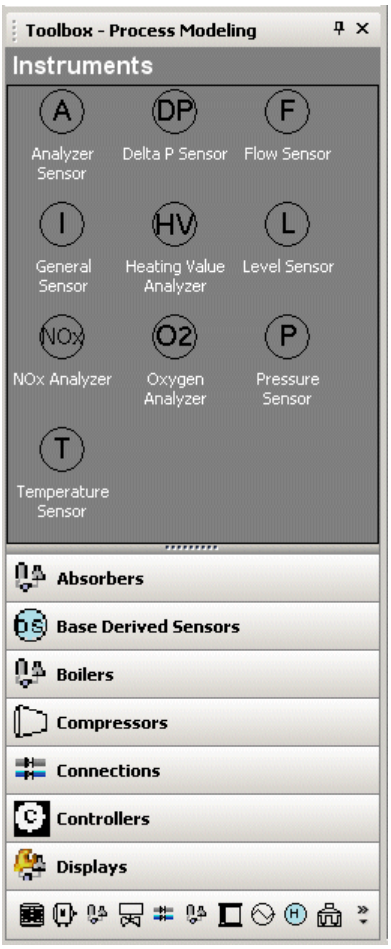
4 To configure the buttons that are visible in the toolbar and associated configuration menu, choose Add or Remove Buttons to display a list of all palettes, then choose a button to add or remove.

For example, if you choose Absorbers, the Absorbers button no longer appears in the toolbar, and if you choose Distillation Columns, the Distillation columns choice no longer appears in the configuration menu. Thus, you can use this menu to narrow the visible palettes in the toolbar and configuration menu.

Once you have configured the buttons you want, you can expand the buttons to show their labels for some or all of the buttons.

5 To show button labels in the toolbox, drag the divider at the bottom of the toolbox up to expose the buttons with their labels.

For example, here is the Process Modeling toolbox with only a subset of the buttons visible in the toolbar and with some button labels showing:



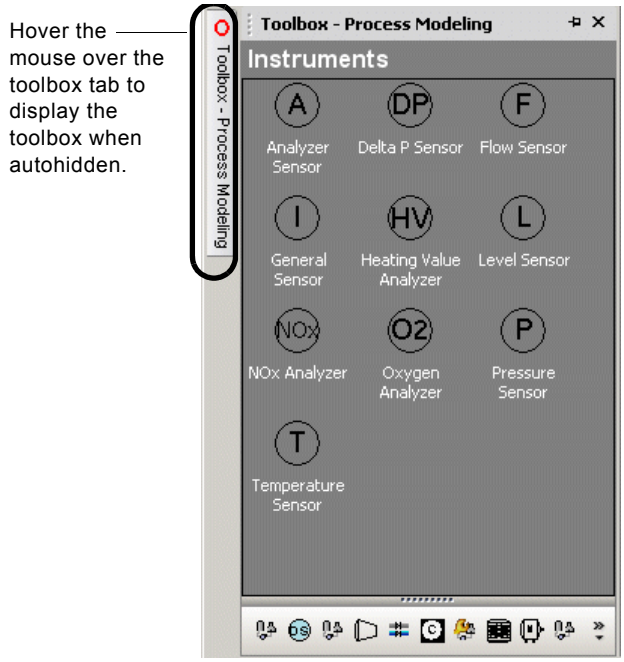
Drag the divider up and down to expose buttons.

Once you have configured the buttons you want to appear in the toolbox, you can auto hide the toolbox by clicking the pin in the upper right corner of the toolbox.

Note Do not close the toolbox or the toolbox reverts to the default set of buttons.

6 Click the pin to autohide the toolbox, and move the mouse over the tab to display the toolbox after it has been hidden.

For example, here is the Process Modeling toolbox



You can display, configure, and autohide multiple toolboxes, as needed, each of which will have its own toolbox tab.

Using the G2 Toolbox

In general, you use the G2 toolbox when customizing models.

For details, see , Customizing Optegrity.

Interacting with Objects

You can interact with objects in a process map by using the Edit menu, the object's popup menu, and the Layout menu. Many of the menu choices have shortcuts and/or equivalent toolbar buttons.

When you create a process map, we recommend that first, you place the domain objects on the workspace, then you align and distribute them, using buttons on the Layout toolbar, then you connect them, as needed.

You configure attributes of objects through properties dialogs.

Selecting Objects

To select one or more objects:

➔ Click an object to select it.

or

➔ Click and drag a rectangular area to select all the objects in the rectangle.

or

➔ Use Shift key and click on an object to add or remove it to or from an existing selection.

or

➔ Use the Alt key and click on a connected network of objects to select all the connected objects.

To select all objects on a workspace:

➔ Choose the Edit > Select All or enter Ctrl + A.

Cutting, Copying, Pasting, and Deleting Objects

When you copy an object, the new object has the same property values as the existing object. If the object has details, the new object has the same details. You can transfer objects from one workspace to another.

To copy and paste objects:

➔ Select one or more objects to copy, choose Edit > Clone, then click on any workspace to paste the selected objects to the workspace.

To cut and paste objects:

➔ Select one or more objects to cut, choose Edit > Transfer, then click on any workspace to paste the selected objects to the new workspace.

To delete objects:

➔ Select an object, then choose Delete from the Edit menu or from the popup menu, press the Delete key, or click the equivalent toolbar button (), then click Yes to confirm the deletion.

Controlling the Layout of Objects

To adjust the order of objects:

➔ Select an object, then choose Layout > Order > Bring to Front or Send to Back or click the equivalent toolbar button: ()

To rotate or flip objects:

➔ Select an object, choose Layout > Rotate or Flip, then choose the desired action from the submenu or click the equivalent toolbar button: 

To align objects:

➔ Select two or more objects, choose Layout > Align or Distribute, then choose the desired align action from the submenu or click the equivalent toolbar button: ()

To distribute objects:

➔ Select three or more objects, choose Layout > Align or Distribute, then choose the desired distribute action from the submenu or click the equivalent toolbar button: ()

To nudge an object up, down, right, or left:

➔ Select an object, choose Layout > Nudge, then choose the desired nudge action from the submenu; or hold down the Ctrl key while pressing the up, down, right, and left arrow keys to nudge the item in the desired direction; or click the equivalent toolbar button: 

For information on the Shrink Wrap toolbar button on the Layout toolbar, see Shrink Wrapping a Workspace.

Displaying the Properties Dialog for an Object

To display the properties dialog for an object:

➔ Double-click the object.

or

➔ Select the object and press the F4 key.

or

➔ Choose Properties from the object's popup menu.

or

➔ Select the object, then choose Edit > Properties or click equivalent toolbar button: ()

Resizing an Object

You might need to resize an object.

To resize an object:

➔ Click an object to select it, and drag the selection handles to resize the object.

Editing Icon Color Regions

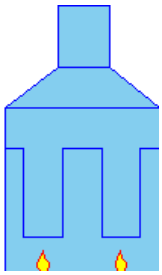
You can edit the color of any named region of any icon.

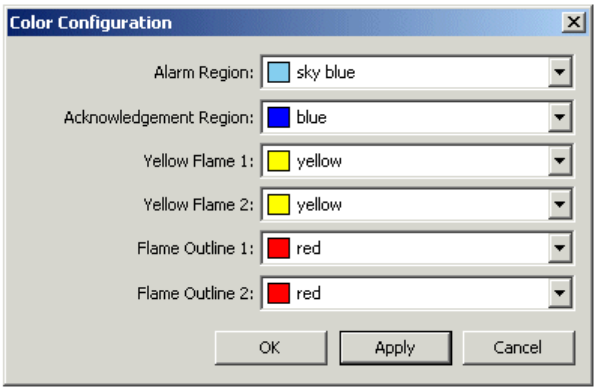
To edit icon colors:

1 Click an object to select it, and choose Edit > Colors.

2 Configure the color of the named icon region for the object, as desired.

For example:





Using the Toolbars

Optegrity provides a number of toolbars that you can use to interact with models.

The toolbars are all docked, by default. You can drag the toolbar to a new location or off the toolbar to make it a floating toolbar.

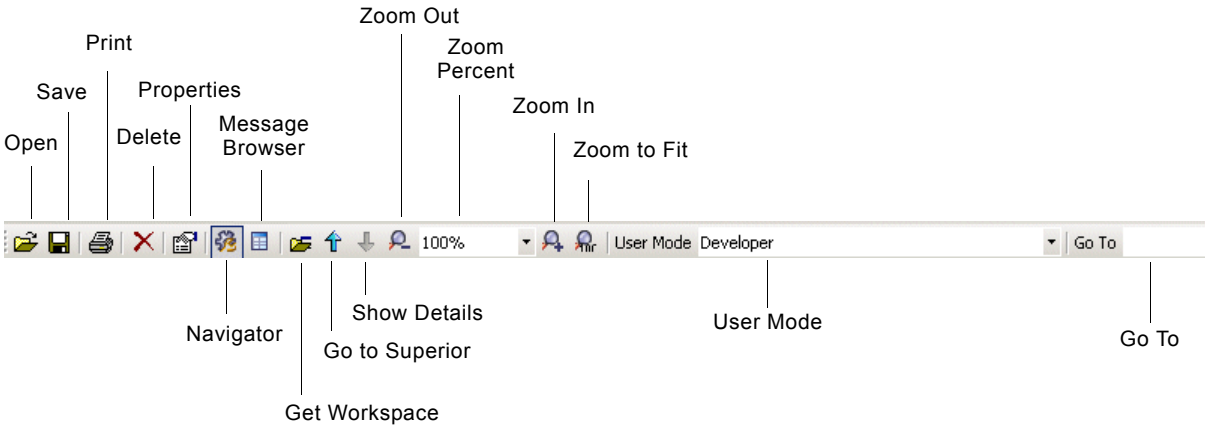
The available toolbars are:

- Standard toolbar
- Web toolbar
- Layout toolbar
- Fault Modeling toolbar

- Operator toolbar
- Status bar

Standard Toolbar

The Standard toolbar contains many of the toolbar buttons that you need to work with the model:



To hide and show the Standard toolbar:

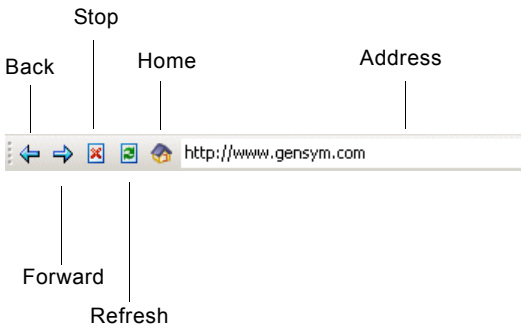
➔ Choose View > Toolbars > Standard.

For information on this button...	See...
Open	Opening a Project.
Save	Saving a Project.
Print	Printing a Workspace.
Delete	Cutting, Copying, Pasting, and Deleting Objects.
Properties	Displaying the Properties Dialog for an Object.
Navigator	Using the Navigator.
Get Workspace	Creating and Accessing Top-Level Workspaces.
Go to Superior	Showing the Superior Object of a Detail Workspace.
Show Detail	Displaying a Detail Workspace.

For information on this button...	See...
Zoom In, Zoom Out, Zoom Percent, and Zoom to Fit	Scaling a Workspace.
User Mode	Switching User Modes.
Go To	Searching for Objects.

Web Toolbar

The Web toolbar provides the standard browser navigation buttons and commands for browsing HTML pages:



To hide and show the Web toolbar:

➔ Choose View > Toolbar > Web.

You can go to any URL, including any HTML file on the World Wide Web or on the file system, or any RTF file.

To go to an HTML file on the World Wide Web, you use the standard HTTP protocol, for example, *http://www.gensym.com*.

To go to an HTML or RTF file on the file system, you use this protocol:

file:\<drive>:\<directory>\<filename>

For example, to go to the readme file, you would use:

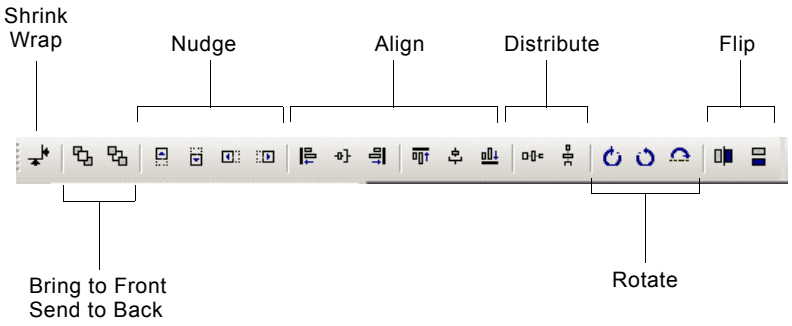
file:\C:\Program Files\Gensym\g2-2011\doc\optegtity\optegtity-readme.html

You navigate by using standard buttons in the Web toolbar or in the Go menu.

You configure the Home button URL in your user preferences. For more information, see Configuring User Preferences.

Layout Toolbar

The Layout toolbar contains toolbar buttons that you need to control the visual layout of objects on a workspace:



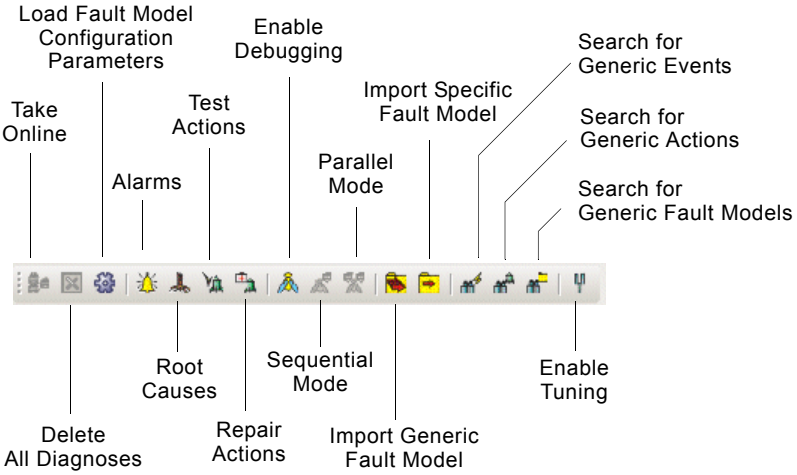
To hide and show the Layout toolbar:

➔ Choose View > Toolbars > Layout.

For information on this button...	See...
Shrink Wrap	Shrink Wrapping a Workspace.
Send to Front, Send to Back, Nudge, Align, Distribute, Rotate, Flip	Controlling the Layout of Objects.

Fault Modeling Toolbar

The Fault Modeling toolbar contains buttons that you use to configure SymCure:



To hide and show the Fault Modeling toolbar:

➔ Choose View > Toolbars > Fault Modeling.

For details, see the *SymCure User's Guide*.

Operator Toolbar

To hide and show the Operator toolbar:

➔ Choose View > Toolbars > Operator.

For details, see Interacting with the Process Model.

Status Bar

The status bar shows various status information, such as the host and port of the client, the current file being loaded, and the progress bar.

By default, the status bar also shows the current message in the operator Message Browser. For information on how to disable this feature, see Configuring User Preferences.

To hide and show the status bar:

➔ Choose View > Status Bar.

Switching User Modes

You build and run applications in one of four built-in **user modes**, or you can define you own user mode. The user mode determines what you can and cannot do when you create your application and run it. For example, the user mode determines whether you can move, edit, and delete objects, and whether you can use the full set of G2 features in your model. For example, the user mode determines the parameters that you can configure.

Optegrity supports the following user modes for these classes of users:

This type of user...	Works in this user mode...	Which allows you to...
Operators and end users	Operator	View pre-built applications without damaging them in any way. Operators cannot open, save, run, or configure applications. Optegrity provides an Operator interface for end users.
Process modeling experts who create applications	Modeler	Create, connect, and configure manufacturing process maps, event-detection models, and fault models. This is the default user mode.
Optegrity application developers	Developer	Create, connect, and configure additional features of Optegrity applications not available in Modeler mode.
Optegrity experts and G2 programmers	System-Administrator Administrator	Customize the behavior of Optegrity.

End users of fully developed applications generally work in Operator mode. Operator mode is restricted so that users may run a model but may not create, configure, or delete objects.

As a model developer, you will almost always be working in Modeler mode. This manual assumes you are working in Modeler mode, unless otherwise stated. Occasionally, as a model developer, you will also need to go into Developer mode to perform certain tasks.

If you are an expert who is customizing Optegrity, you will be working mostly in Developer mode.

The user mode that is available to you depends on your login privileges.

To switch to a different user mode:

➔ Choose Tools > User Mode or configure the User Mode on the toolbar.

Configuring User Preferences

Optegrity allows you to configure different levels of access and default behavior for different categories of users. When a particular user starts Optegrity, the user preference associated with that user restricts the access and provides default behavior, as appropriate for the given user.

You can configure the following preferences:

- The default user mode, which determines the level of access to Optegrity features.
- Subscription to queues, including Messages, Alarms, Root Causes, Test Actions, Repair Actions, and custom queues.
- Message filter to subscribed queues, for filtering messages based on priority, process map, type, category, target, assigned to, age, and acknowledgement status.
- Acknowledgement and deletion permission and behavior in the Message Browser.
- Client disconnection, server shutdown, and modeling configuration permissions, and whether the user is an administrator.
- The default behavior for interacting with objects through menus and showing the logbook.
- Email and mobile email addresses for use with the JMail interface.
- The default view for the home process map and the size of the location history in the operator view.

Specifying User Preferences for Different Types of Users

Optegrity creates a default user preference for the Optegrity server to determine the level of access and default behavior for all users that log into the server. Similarly, Optegrity creates one user preference for each user associated with a G2 login account. The name of the user preference corresponds with the user name specified in the *g2.ok* file. For more information, see Chapter 62 "Licensing and Authorization" in the *G2 Reference Manual*.

If you are logged in as the user named administrator, you are automatically configured to be the Administrative User and can create and configure user preferences for all users. If you are logged in as any other user, you can only configure your own user preferences. You can be logged in either to your windowing system or to the Optegrity server through a secure G2 as administrator.

We recommend that the user preference for the server provide access to all available features, and that it use either Modeler or Developer mode. The user preferences for the clients should provide appropriate levels of access and should use the appropriate user mode, depending on the type of user. For example, you might configure user preferences as follows for these types of users:

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Operators, who interact with messages only	operator	• Disconnect permission • Acknowledge message permission • Show message in operator mode by default • Subscribe to appropriate queues, depending on the model
Modelers, who create process maps, event-detection models, and SymCure diagnostic models	modeler	• Disconnect permission • Configuration permission • Acknowledge message permission • Delete message permission • Subscribe to Messages queue

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Developers, who use G2 to customize models	developer	• Indicate items upon menu selection • Disconnect permission • Shutdown permission • G2 Logbook • Acknowledge message permission • Delete message permission • Subscribe to all queues
Administrators, who configure user preferences for all users, using the Optegrity user interface	system-administrator	The same as developers, plus Administrative User.
Administrators, who configure user preferences for all users, using G2's user interface	administrator	**Note:** You must log in as administrator to enable the Administrative User option.

Configuring User Preferences

In Modeler mode, you can configure these attributes for each user preference. For information about additional attributes that you can configure in administrator mode, see Configuring User Preferences.

Attribute	Description
General	
User Name	The user name of the user that starts either the server or the client, which is read-only. If you are an administrative user, you can create new user preferences for specific users. For details, see Configuring User Preferences.
Default User Mode	The default user mode for the specified user, which is modeler, by default. The options are: operator, modeler, developer, system-administrator, and administrator.
User Interface Theme	The Windows user interface theme. The default value is window-theme-2003.
Email Address Mobile Email	E-mail and mobile e-mail address of the specified user for sending email when a message occurs. For more information, see Delivering Messages by Email.
Home Process Map	A process map to use as the background in the operator interface. The default process map is default view, which is associated with the process map named guif-default-main-view. Click Select to display a list of all process maps in the KB and choose a map to use as the default background.
Telnet Command	The command for launching a Telnet session.
Default Web Location	The default URL when clicking the Home button in the Web toolbar.
Set Default User Mode	Whether the default user mode should be set upon startup.

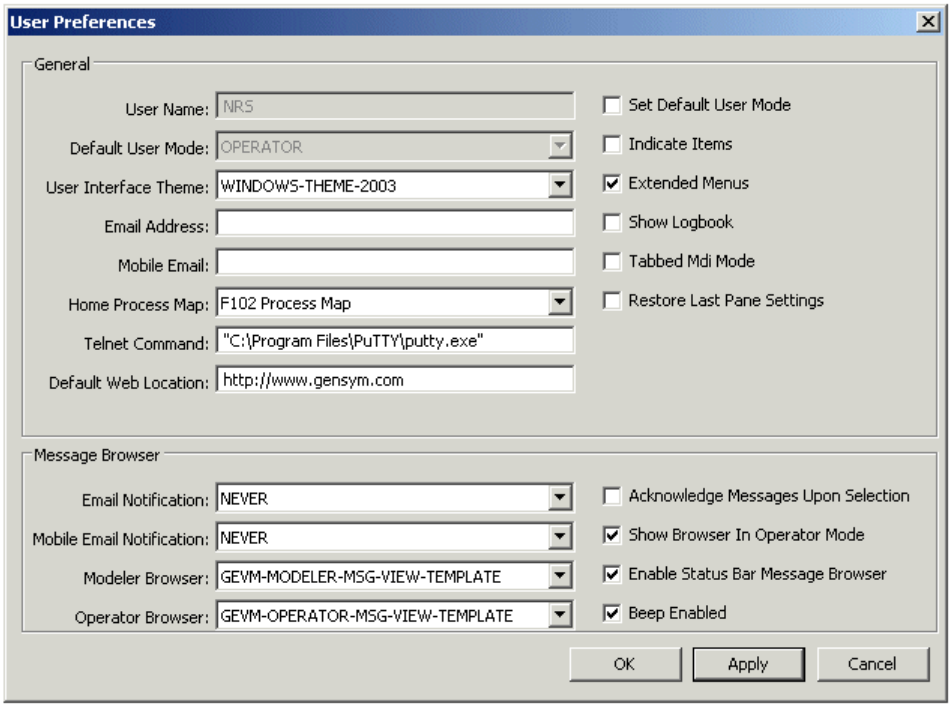
Attribute	Description
Indicate Items	Configures the behavior when choosing items from the Project menu. By default, Optegrity displays the properties dialog or the model detail, depending on the type of item. Developers who are familiar with G2 and prefer to work with the iconic representations of items might want to enable the Indicate Items option, in which case, choosing items from the Project menu goes directly to the item. For more information, see Using the Project Menu.
Extended Menus	Whether to display the complete list of objects in the Project submenus, the default. If your project has many domain models, for example, you might want to disable this option, in which case, selecting Project > System Models > Process Maps displays the Manage dialog for interacting with object.
Show Logbook	Whether to show the G2 Logbook when errors occur. By default, the G2 Logbook does not appear. Modelers or developers who are familiar with G2 might want to enable the Show Logbook option. We recommend that you disable this option for operators and modelers who are not familiar with G2.
Tabbed Mdi Mode	Whether to display workspaces in tabs in the window.
Restore Last Pane Settings	Whether to restore the settings for panes upon connection.
Message Browser	
Email Notification Mobile Email Notification	The format when sending e-mail and mobile e-mail messages. By default, the value is never, which means email messages are not sent. For details, see Delivering Messages by Email.
Modeler Browser	The browser to use in Modeler mode. The default is gevm-modeler-message-view-template, which is the browser that appears when you choose View > Message Browser.

Attribute	Description
Operator Browser	The browser to use in Operator mode. The default is gevm-operator-message-view-template, which is the browser that appears when you are in Operator mode.
Acknowledge Messages Upon Selection	Whether to acknowledge messages automatically when the operator selects a message in the Message Browser view of the operator interface. By default, messages are not automatically acknowledged. When Ack Msg Upon Selection is enabled, Ack Msg Permission must also be enabled.
Show Browser in Operator Mode	Whether to show the Message Browser by default view in the operator interface, or whether to show the process map view. By default, the Message Browser appears as the default view in the operator interface.
Enable Status Bar Message Browser	Whether to show the most recent message in the status bar.
Beep Enabled	Whether to enable beeping when new messages arrive in the Message Browser, as well as when they are acknowledged and deleted. By default, beeping is enabled.

To configure user preferences for yourself:

➔ Choose Project > My User Preferences and configure the user preferences, as needed.

For example, here is the default user preferences dialog appears for the user named nrs:



To configure user preferences for other users:

➔ Choose Project > System Settings > Users and choose the user whose preferences you want to configure.

For details, see Configuring User Preferences.

Delivering Messages by Email

You can configure the user preference for individual users to provide an email address and a mobile email address, then configure rules for when to send email messages when an event occurs.

You can configure Optegrity to format the message as short plain text, suitable for cell phones, for example, plain text with full message contents, or as an HTML document. You can also configure when to send a message, based on when it was created or updated, whether the user is currently connected to the server, and the priority of the message.

To deliver messages by email, you:

- Start the G2 JMail Bridge process.
- Create, configure, and connect a JMail Interface object.
- Configure Optegrity to send email messages.
- View examples.
- Configure startup parameter for sending email.

Starting the G2 JMail Bridge Process

To deliver messages by email, you must start the G2 JMail Bridge process. You identify the host and port to which the bridge is connected for configuring in the JMail Interface object.

To start the G2 JMail Bridge process:

➔ Choose Start > Programs > Gensym G2 2011 > Bridges > G2 JMail Bridge.

The G2 JMail Bridge process appears in the command window.

To determine the bridge port:

➔ Open the command window for the bridge process.

The last line indicates the TPC/IP host and port number, for example:

```
TCP_IP:NSALVO-1165:22080
```

Creating, Configuring, and Connecting the JMail Interface Object

To deliver messages by email, you must create and configure a JMail Interface object, which specifies:

- A name.
- The host and port of the machine running the G2 JMail Bridge.
- Information about the SMTP mail server, including the user name, password, incoming and outgoing SMTP mail host, and SMTP protocol.

If the bridge process is running on the local machine, the host is localhost. The default port number is 22080, 22081, 22082, etc., depending on the number of clients that are currently connected on that port.

Note To configure a JMail Interface object, you must be in Developer mode.

Once you have configured the JMail interface object, you can connect it to the G2 JMail bridge process.

To create, configure, and connect a JMail Interface object:

1. Choose Tools > User Mode > Developer.
2. Choose Project > System Settings > Interfaces > SMTP > Manage and click the New button to create a new JMail Interface object.

   Alternatively, you can choose View > Toolbox - G2, click the Network Interfaces tab, and create a JMail Interface object.
3. In the properties dialog for the JMail Interface object, configure the Interface Name attribute to be any symbol, for example, my-jmail-interface.
4. Configure the Bridge Host and Bridge Port to be the host and port of the machine on which you started the G2 JMail Bridge process.
5. Configure the following additional information:

Attribute	Description
User Name	The user name of the account to which email should be sent.
Password	The password of the user account to which email should be sent.
Incoming Host	The name of the host computer used for incoming email.
Incoming Port	The port number of the host computer used for incoming mail.
Incoming Protocol	The SMTP protocol that the incoming mail host uses. The default is pop3.
Incoming Folder	The folder name of the user account to which to send email. The default is inbox.
Delete Messages on Server	Whether to delete the email message on the mail server after it is sent. By default, messages are not deleted.
Outgoing Host	The name of the host computer used for outgoing email.
Outgoing Port	The port number of the host computer used for outgoing mail.

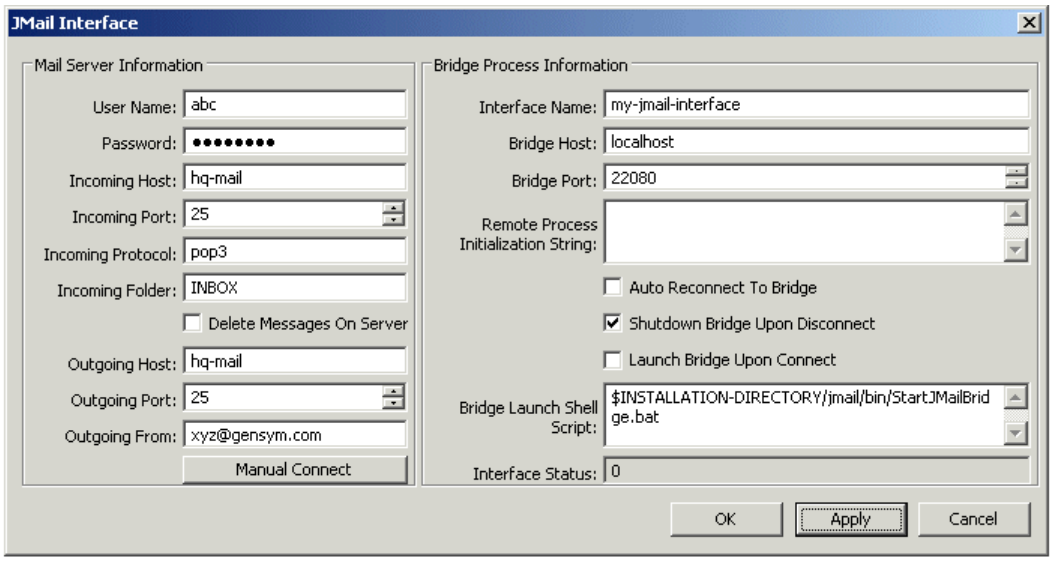
Attribute	Description
Outgoing From	The name to use as the From address when the email message is sent, which cannot contain spaces.
Auto Reconnect to Bridge	Whether to automatically reconnect if the connection goes down.
Shutdown Bridge Upon Disconnect	Whether to shutdown the bridge when the connection is closed.
Launch Bridge Upon Connect	Whether to launch the bridge when a connection is made.
Bridge Launch Shell Script	Pathname to script for launching the bridge.

6 Click Apply to apply these values.

7 Click the Connect button in the dialog to connect the interface to the bridge.

8 Choose Tools > User Mode > Modeler to return to Modeler mode.

For example:



Configuring Optegrity to Send Email Messages

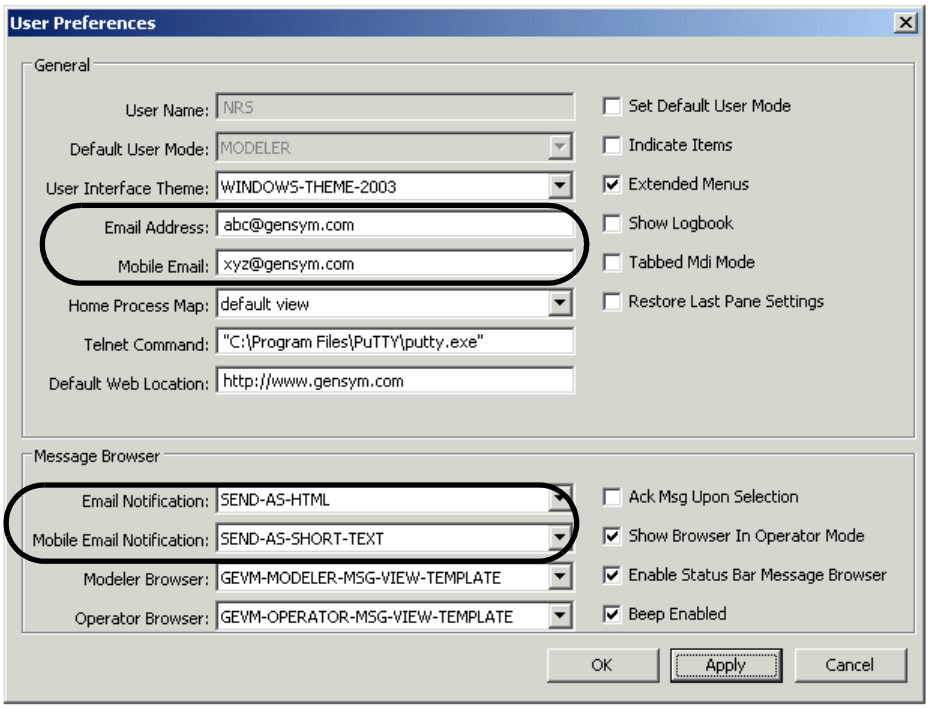
You configure Optegrity to send email messages through the user preferences dialog.

To configure Optegrity to send email messages:

1. Choose Project > My User Preferences.
2. Configure Email Address and/or Mobile Email.
3. Choose the rule to use for each of the configured email addresses, as follows:
   - never — Do not send e-mail messages. This is the default rule.
   - send-as-text — Send the message text and details as plain text.
   - send-as-short-text — Send the message text only as plain text.
   - send-as-html — Send the message text and details as HTML.
   - only-high-priority-as-text — Send the message text and details as plain text only if the priority is 1.
   - only-high-priority-as-short-text — Send the message text as plain text only if the priority is 1.
   - only-high-priority-as-html — Send the message text and details as HTML only if the priority is 1.
   - if-not-connected-send-short-text — Send the message text as plain text only if the user is not connected to the server.
   - if-not-connected-send-as-text — Send the message text and details as plain text only if the user is not connected to the server.
   - if-not-connected-send-as-html — Send the message text and details as HTML only if the user is not connected to the server.

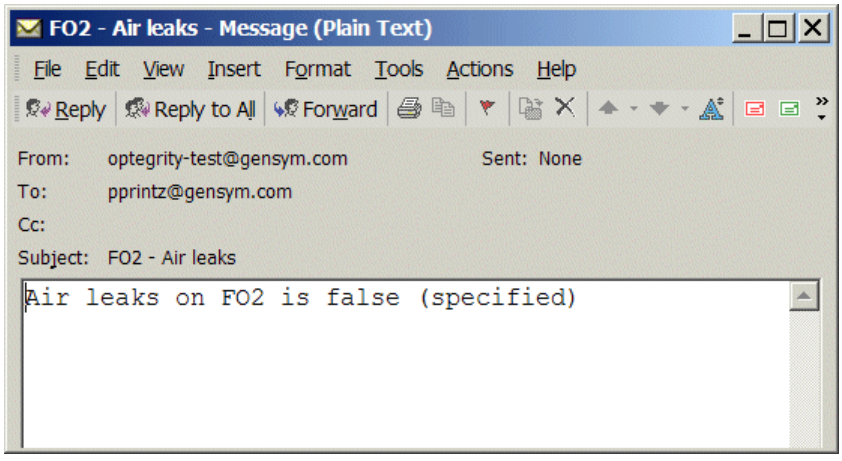
When a message occurs, Optegrity also sends an email to the specified addresses.

Here is the User Preferences dialog with both email addresses and rules configured:

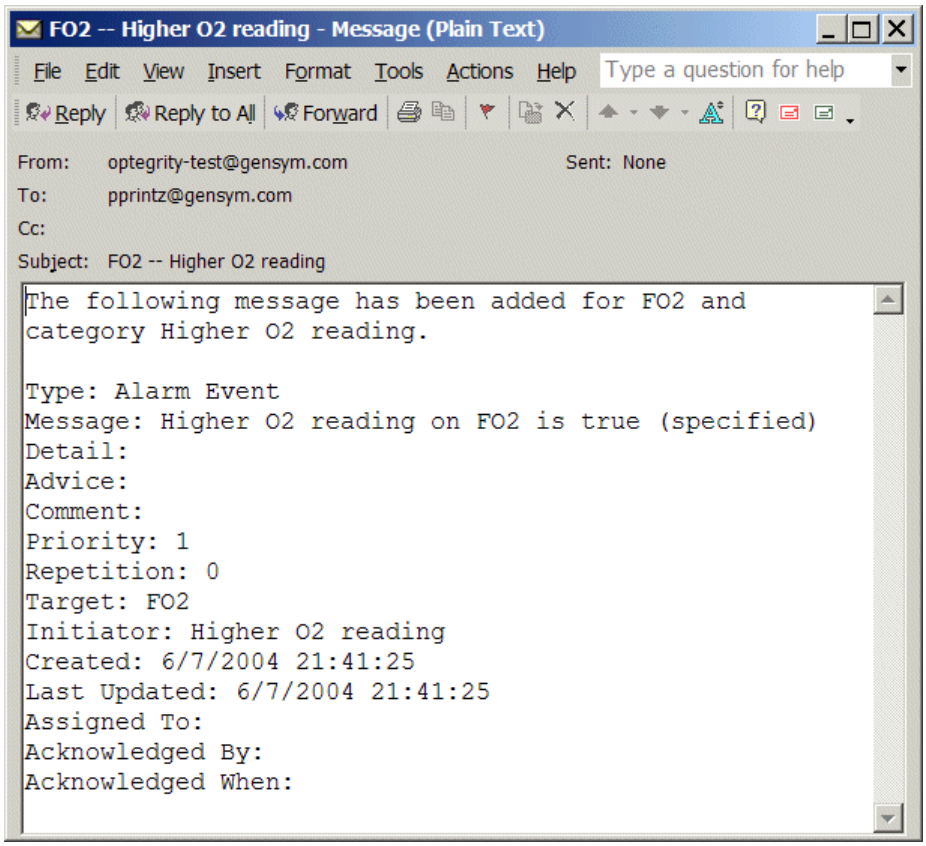


Examples: Sending Email Messages

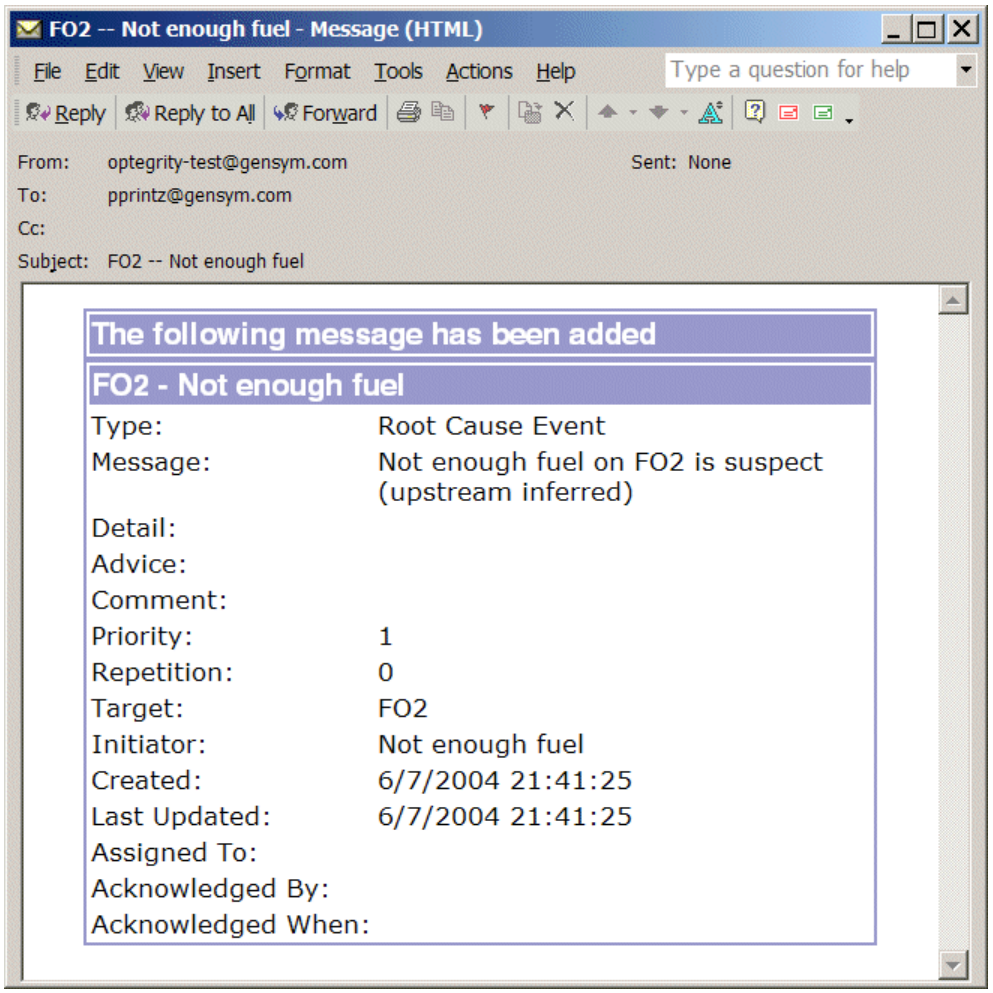
Here is an example of a message that includes the message text only in plain text:



Here is an example of a message that includes the message text and details in plain text:



Here is an example of a message that includes the message text and details in HTML format:



Configuring Startup Parameter for Sending Email Messages

You can configure the following startup parameter in the configuration file:

JMAIL-INTERFACE-NAME=none

Specifies the default JMail interface to use for sending email messages.

For details about using the configuration file, see the *G2 Run-Time Library User's Guide*.

4

Creating Optegrity Applications

Describes how to create a new Optegrity application.

Introduction

This chapter provides a high-level description of the steps you follow to create a complete Optegrity application for abnormal condition management. It also describes how to work with projects.

Unlike traditional G2 applications, you do not need to create a workspace hierarchy to organize your application, and, for the most part, you do not need to place objects on workspaces when you create them. Optegrity takes care of this process for you. Instead, you simply create, configure, and manage objects by using the menus, and you create process maps, event detection diagrams, and fault models by using the various toolboxes.

When creating a new Optegrity application, you should set up your user preferences. User preferences control the default user mode, default permissions, and default behaviors.

Building an Optegrity Application

These are the high-level steps required to build a complete Optegrity application. The details of each step are described in the referenced sections and chapters.

To build an Optegrity application:

1 **Create a new project.**

You create an empty project, which requires all the Optegrity modules that you will need to build an application for abnormal condition management.

See Working with Projects.

2 **Set up user preferences.**

Each user that accesses the application should have a user preference to determine the default user mode, the permissions, and the default behaviors.

See Configuring User Preferences.

3 **Create a process map.**

The process map represents the external process you want to monitor and manage. A process map consists of domain objects, which are created from built-in equipment and instrument definitions.

a Create a graphical representation of your process in a process map, using your domain object definitions, and configure each internal datapoint.

See Building a Process Map.

b Create and configure domain objects for built-in event detection.

See Configuring Built-in Event Detection.

c Create custom domain object definitions for the process equipment, sensors, and controllers in your external system, and create internal datapoints for each external datapoint you want to monitor and manage.

See Creating Domain Object Definitions.

4 **Manage data sources.**

You manage the interface between the Optegrity application and your

a Create and configure OPC or PI interfaces to obtain external data through a bridge.

See Configuring Network Interfaces.

b Use CSV files to create external datapoints that represent each DCS tag variable in your external system, and link those external datapoints to internal datapoints in the process map, either through the CSV file or manually.

See Configuring External Datapoints.

c Configure engineering unit conversions and synonyms, as needed, to define datapoint units.

See Converting Engineering Units.

d Configure the process map to log internal datapoint values.

See Configuring Logging.

e Use continuous or differential CSV files for data replay to replay internal or external datapoint values.

See You replay data to:.

f Use data drivers to simulate internal or external datapoint values.

See Simulating Datapoint Values.

5 Create generic event detection templates and specific event detection diagrams.

You create generic event detection templates for domain object classes, which monitor internal datapoints and generate operator messages and low-level notifications when certain conditions are met.

To use these diagrams in an application, you must initialize the process map, which creates specific event detection diagrams for each instance of the target domain object class.

See Creating Generic Dataflow Diagrams.

See Initializing Process Maps.

6 Create reports to monitor events.

You create event metrics reports to monitor frequency and duration statistics for events. You can also create system performance reports.

See Reporting and Charting.

7 Create generic fault models.

You create generic fault models for domain object classes, which describe causal relationships between fault model events, based on the relationships between domain objects in the process map. These events can be alarms and root causes, which the operator views through a browser, or they can be unspecified events created for modeling convenience.

You can also create generic test actions and repair actions for domain object classes, which are associated with particular events. Test actions help diagnose faults, and repair actions take corrective actions.

You must compile the generic fault models before they can be used in a diagnostic application.

See Creating Generic Fault Models.

See Running SymCure Fault Models.

8 **Manage messages and alarms.**

You can manage operator messages, alarms, root causes, test actions, and repair actions through browsers, and configure the message queues for logging.

See Interacting with Operator Messages.

See Interacting with SymCure Diagnostic Console Browsers.

See Using Message Queues.

9 **Customize startup parameters and other features**.

Optegrity provides numerous initialization parameters, which you can configure at startup to customize the behavior of various aspects of the application.

You can also customize various other aspects of Optegrity, including creating custom message browsers and messages, custom network interfaces, and custom menus.

See Configuring Startup Parameters.

See Customizing Optegrity.

Working with Projects

An Optegrity project consists of a set of related files that form a knowledge base. Each file contains a stand-alone **module**, which together make up a **module hierarchy**. Each module is associated with its own *.kb* file, whose name typically corresponds to the module name. The module hierarchy consists of a top-level module and a number of lower-level modules. The top-level module requires the lower-level modules to run.

When creating a new project, you can choose various intelligent object libraries to provide out-of-the-box event detection for abnormal condition management applications. If you do not plan to use out-of-the-box event detection or if you plan to use only certain intelligent object libraries, disable the libraries you do not need. Only those libraries that you enable are available in the Process Modeling toolbox. By default, all intelligent object libraries are available when you create a new project.

You can:

- Create a new project.
- Save a project.
- Open a project.

Creating a New Project

When creating a new project, Optegrity creates a new, blank project with the name you enter. The new project is saved it in the *projects* directory of your Optegrity installation directory.

Note Creating a new project replaces the existing model in memory. Therefore, before you create a new project, be sure to save the existing project, as necessary.

To create a new project:

1. Choose File > New.
2. Enter the name of the project.

   The project name cannot contain spaces.
3. Ensure that Optegrity is chosen as the selected library.
4. Check or uncheck any additional libraries, depending on how your application needs to access external data.
5. Click OK.

Optegrity displays the Operator Logbook as it creates a new project with the name you specify, then loads the new project and all required modules onto the server. When all modules have been successfully loaded, the menu bar updates. You must wait until the KB has finished loading in the server before you can access your project.

The title bar of the client window displays the default file name for your project, which is your project name with the *.kb* extension.

Saving a Project

Projects are stored in the *projects* directory of your Optegrity installation directory.

Optegrity saves the top-level module only; it does not save the required modules. Unless you are customizing Optegrity, you do not generally need to save the required modules.

To save a project:

➔ To save a project to the project file that was loaded when you started the client, choose File > Save or click the equivalent toolbar button: ()

or

➔ To save the project to a different project file, choose File > Save As, enter a new project name, or choose an existing project name from the list of available projects on the server.

Note To ensure that the title bar of the client window shows the correct file name for your project, we recommend that you always save your project to the default file name.

Opening a Project

To open a project, specify the project name associated with the top-level module in the module hierarchy.

Note Opening a new project replaces the existing application in memory. Therefore, before you open a new project, be sure to save the existing project, as necessary.

To open a project:

1. Choose File > Open or click the equivalent toolbar button () to display the Open Project dialog.
2. Enter or choose the project to open and click Open.
3. Click Yes in the confirmation dialog.

Optegrity displays the Operator Logbook as it loads the project file and all required modules onto the server. When all modules have been successfully loaded, the menu bar updates. You must wait until the KB has finished loading in the server before you can access the application.

Part II

Process Maps

Chapter 5: Building a Process Map

Describes how to configure domain objects to create a process map.

Chapter 6: Configuring Built-in Event Detection

Describes how to configure built-in event detection for domain objects.

Chapter 7: Creating Domain Object Definitions

Describes how to create your own domain object definitions, which are based on the built-in Optegrity equipment and instrument definitions.

5

Building a Process Map

Describes how to configure domain objects to create a process map.

Introduction

A process map provides a graphical representation of your managed system, using domain objects that represent different types of equipment, sensors, and controllers. You construct a process map by creating domain objects from palettes and connecting those domain objects to represent the equipment and topology of your plant. The process map can consist of any combination of built-in and user-defined domain objects.

The simplest type of process map consists of domain objects on the detail of a process map container. You can also define hierarchical process maps. For example, you can create a production site, which consists of one or more production areas, where each production area consists of one or more process units.

SymCure can perform system wide root cause analysis by correlating events across different objects by using containment hierarchies, relations, and connections defined by the domain map. For more information, see Creating Generic Fault Models.

Optegrity provides four types of containers for building process maps:

Container	Description
Process Map	Provides a container in which to create and connect domain objects to create a simple process map. You can also use a process map to subdivide process units when modeling a hierarchical process.
Process Unit	When modeling a hierarchical process, provides a container in which to create individual process units.
Production Area	When modeling a hierarchical process, provides a container in which to create a production area, which consists of one or more process units.
Production Site	When modeling a hierarchical process, provides a container in which to create a production site, which consists of one or more production areas.

Creating a Process Map

To construct a process map, first you create a process map container, then you create process equipment from the various palettes in the Process Modeling toolbox and place them on the detail. Once you have created domain objects, you can connect those domain objects.

After creating and connecting domain objects, you can then create sensors and relate them to the various domain objects, based on the types of events you want to detect.

Creating a Process Map Container

You create a process map container through the Project menu.

You can also create a process map container through the Navigator. For details, see Navigating Applications.

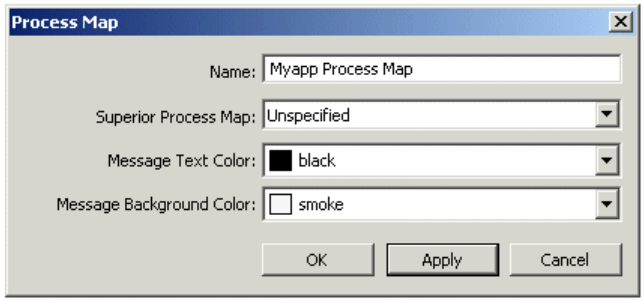
To create a process map container:

1 Choose Project > System Models > Manufacturing Processes > Manage, and click the New button.

The properties dialog for the Process Map container appears.

2 Configure the Name, which is system-generated, by default.

The name can be any text value, with spaces, for example, Myapp Process Map:

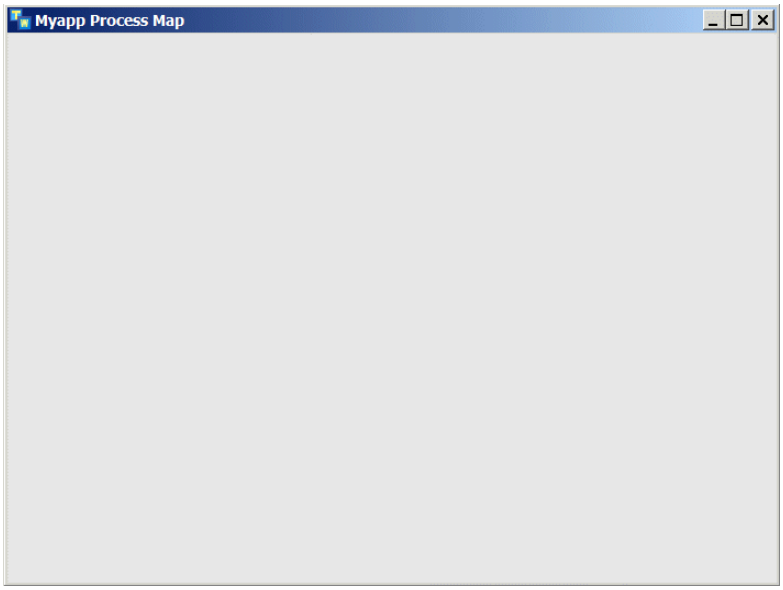


Tip We recommend that you prefix the process map name with your application name.

You can now go to the process map container detail to begin building the process map. You access the detail through the Manage dialog or the Project menu.

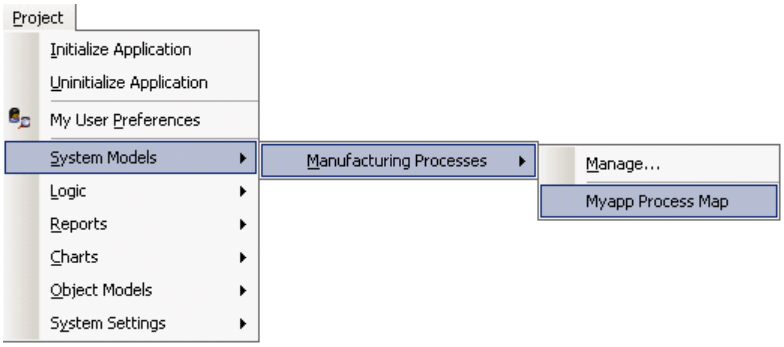
3 When you first create a process map container, you can show its detail by clicking the Show Detail button in the Process Maps Manage dialog.

Here is the resulting process map detail, which is initially empty:



You can also access the process map detail by choosing Project > System Models > Manufacturing Processes, then choosing a process map.

The process map appears in the System Models > Manufacturing Processes submenu, for example:

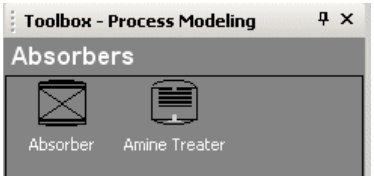


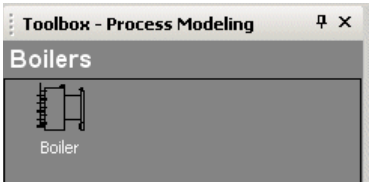
For information on...	See...
Using the Manage dialog	Using the Manage Dialog.
Configuring the process map background	Editing Workspace Properties.

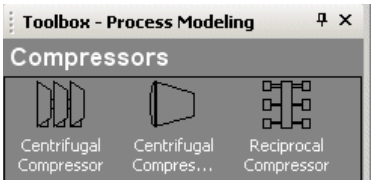
For information on...	See...
Creating a process map on the detail of a process map container	Configuring Domain Objects.
Configuring the superior process map to create a process map hierarchy	Creating a Process Map Hierarchy.
Configuring message colors	Configuring Message Color Based on the Process Map.

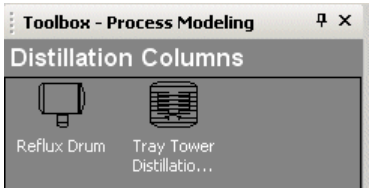
Creating Process Equipment

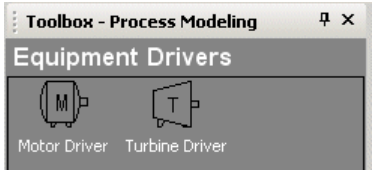
Optegrity provides numerous types of process equipment domain objects that you can use to create a manufacturing process map. You create process equipment from the following palettes in the Process Modeling toolbox:

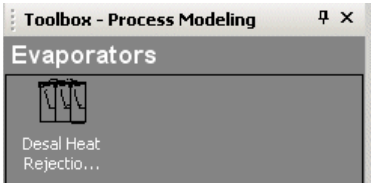


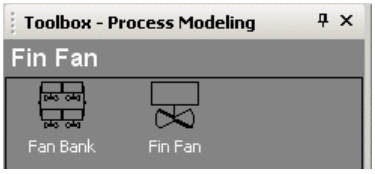


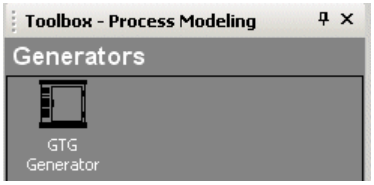


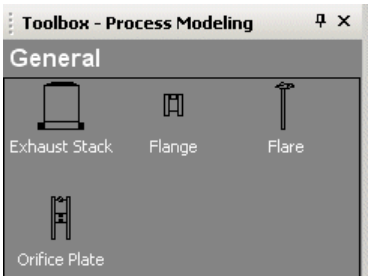


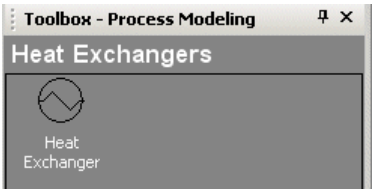


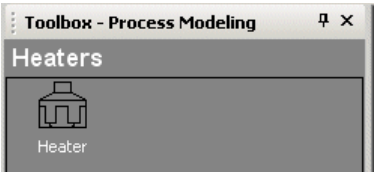


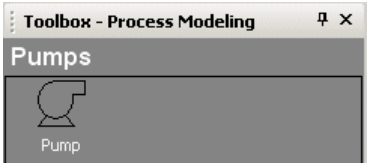


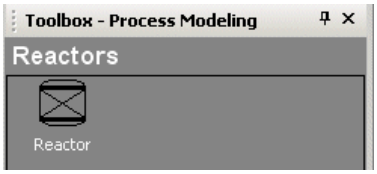


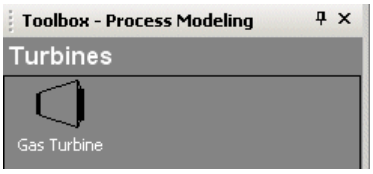


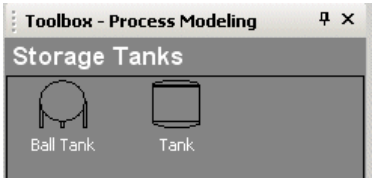


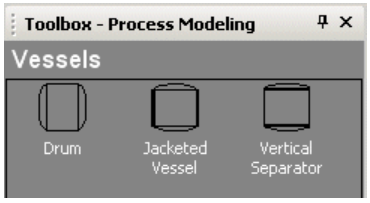


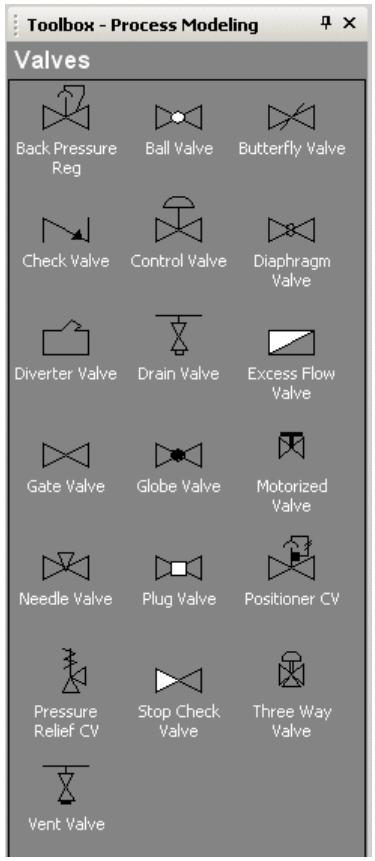












When the intelligent object libraries are loaded, all process equipment can detect a set of built-in sensor events for process flow, inlet and outlet temperature, and inlet and outlet temperature sensors that are related to the piece of process equipment.

Certain process equipment can detect additional built-in events for other types of related sensors, some of which are derived sensor values. For example, a heater can detect events for draft oxygen and pressure, stack NOx, and tube skin temperature related sensors, as well as for heater efficiency and tube skin delta temperature related derived sensors.

Most domain objects define a set of internal datapoints. In general, when detecting built-in events, you do not need to configure the internal datapoints for process equipment. Rather, you configure the internal datapoints for the related sensors values.

To create process equipment:

1. Choose View - Toolbox - Process Modeling and choose one of the palettes.

   For information on displaying and interacting with the Process Modeling toolbox, see Using the Optegrity Toolboxes.

2. Click a domain object in the palette and click on the detail of the process map to place the object on the map.

3. Choose Properties on the domain object.

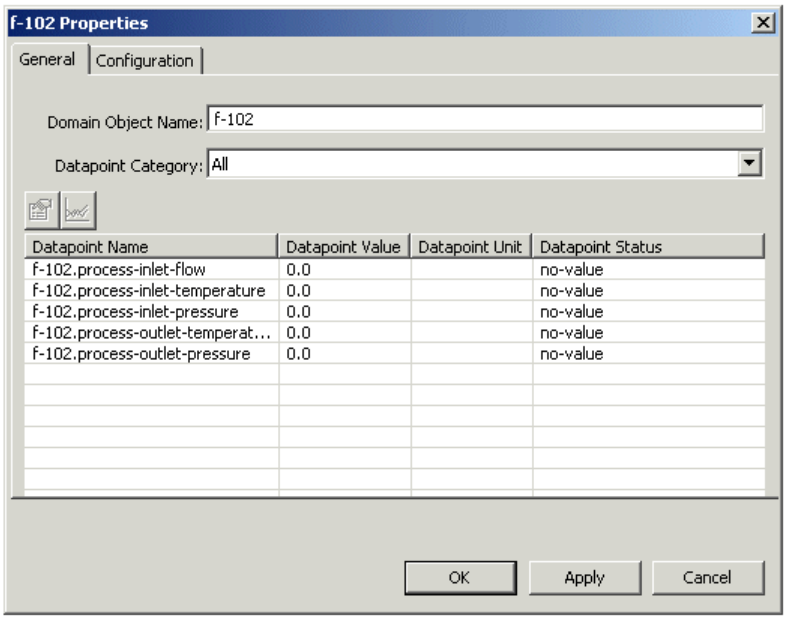
   The properties dialog shows the name of the domain object and its built-in internal datapoints. Optegrity provides a default name for the domain object.

4. Configure the Domain Object Name.

   The name can be any text, with or without spaces, for example, myapp fo2, myapp-fo2, or fo2.

5 Accept the dialog.

For example, here is the properties dialog for a user-defined heater named F-102, which shows the built-in internal datapoints:



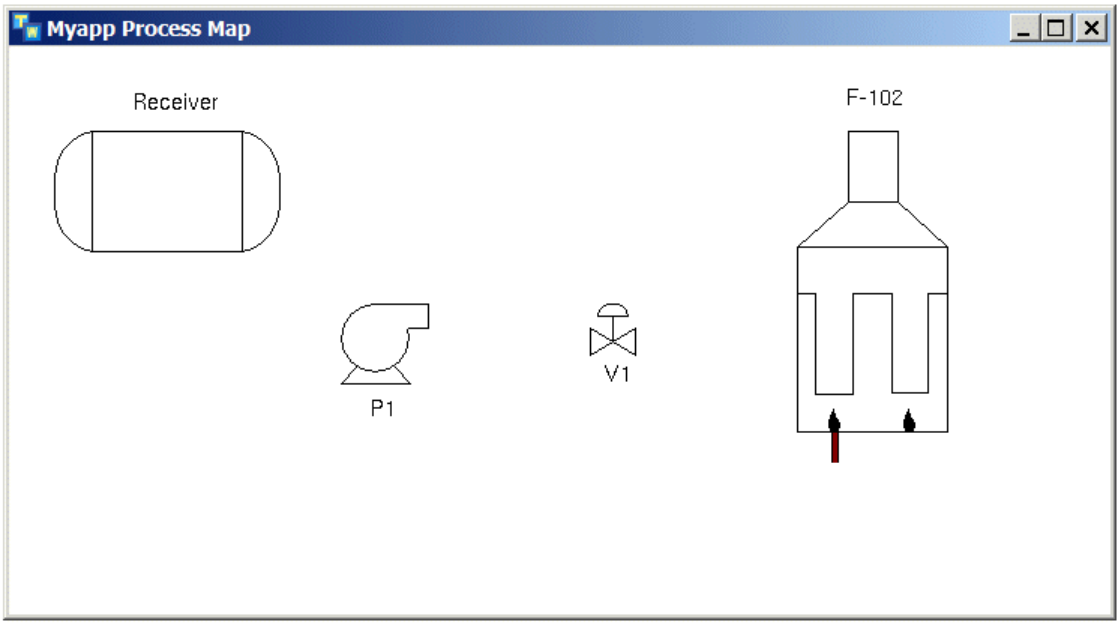
6 Continue displaying palettes and placing objects until you have all the domain objects you need to form your process map.

7 Use the commands in the Layout menu to align and distribute the objects, as needed.

For details, see Controlling the Layout of Objects.

8 From the process map popup menu, choose Shrink Wrap to shrink the size of the process map to just fit the contained domain objects.

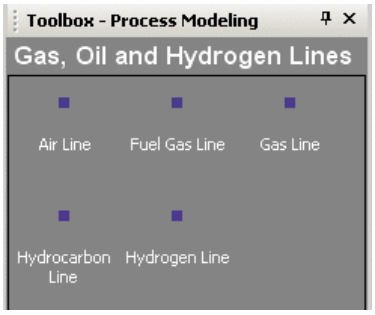
For example, here is a process map with a heater, a valve, a pump, and a tank:

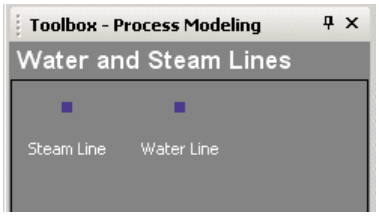


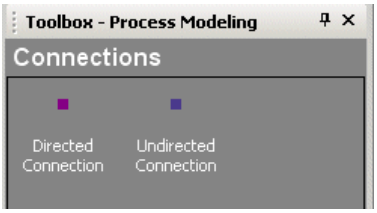
Connecting Process Equipment

Optegrity provides various types of connections that you can use to represent conduits, pipes, or other connection pathways between process equipment in a process map.

You can create gas, oil, and hydrogen lines, water and steam lines, or simple directed and undirected connections. Here are the available connection palettes in the Process Modeling toolbox:







To connect domain objects, you create a connection stub tool from one of the connection palettes and use it to create a connection stub on a domain object. To connect two objects, you drag the connection stub into another object, then you delete the connection stub tool.

You can use general directed and undirected connections to connect instruments in a process map, although, in general, you do not need to connect sensors in a process map. For an example, see Connecting Instruments.

For information about how SymCure uses connectivity to perform system-wide root cause analysis, see the *SymCure User's Guide*.

To connect domain objects:

1. Choose View > Toolbox - Process Modeling and choose one of the connection palettes.

   For information on displaying and interacting with the Process Modeling toolbox, see Using the Optegrity Toolboxes.

2. Click a connection in the palette, then click next to a domain object in the process map to place it next to the object.

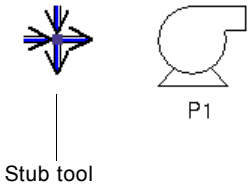
Tip The stub tool remains selected. To unselect it, click in an open area of the process map detail.

---

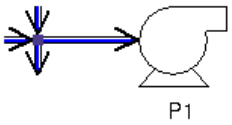
3. Click one of the connection stubs from the stub tool, depending on the connection direction, drag it into the domain object, and click again to create a connection stub on the object.
4. Choose Delete on the connection between the object and the stub tool.

   The connection stub tool remains for creating additional connections.
5. Continue creating connection stubs, as needed.
6. When you are finished creating connections, delete the connection stub tool.
7. Click the newly created stub, drag it into another domain object, and click again to create a connection to another domain object.

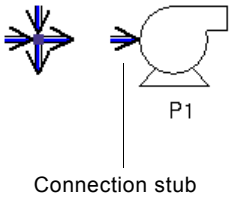
The following series of figures shows the process of creating a water line between the pump and the tank in the previous example. First, you place a connection stub tool on the detail, for example, upstream of the pump:



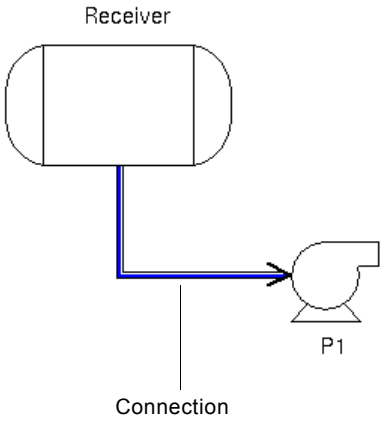
Next, you drag a connection stub into the domain object:



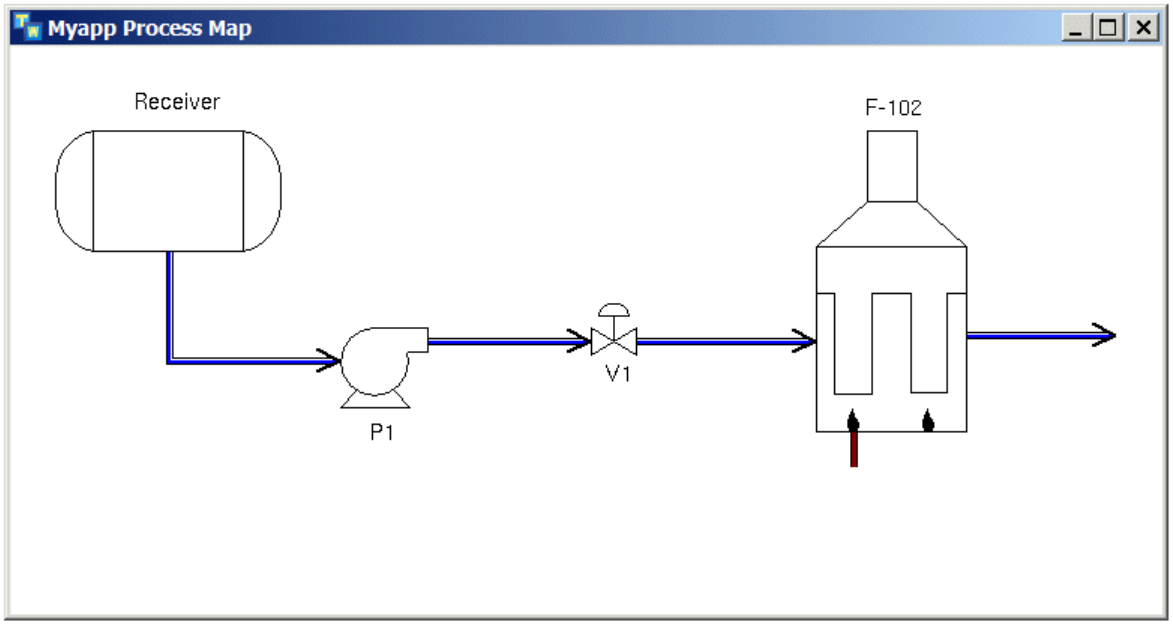
Then, you choose Delete on the connection between the stub tool and the pump, which leaves a connection stub on the pump:



After moving the stub tool out of the way, you then drag the connection stub from the pump into the upstream tank to create a connection:



Here is the process map with the equipment connected, where the tank is upstream of the pump, the pump is upstream of the valve, and the valve is upstream of the heater:



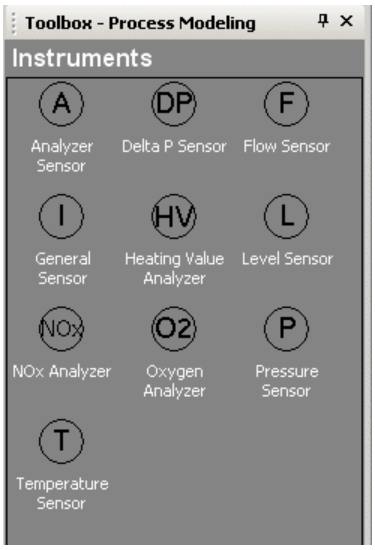
Creating Instruments

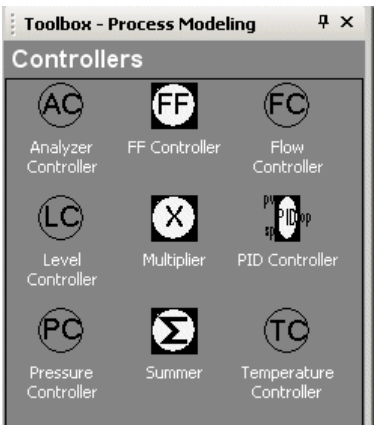
In addition to process equipment, a process map contains various types of instruments, which include sensors, analyzers, and controllers. All sensors and analyzers define PV (process value) as an internal datapoint, and all controllers define PV, SP (setpoint), OP (control output), and Mode as internal datapoints.

When the intelligent object libraries are loaded, all sensors can detect a set of built-in events for PV high, low, projected high, projected low, noisy, flatline, and change. Controllers can detect an additional set of built-in events for OP projected high and projected low, and setpoint error.

You choose instruments, based on the types of events you want to detect. For example, to detect the PV Low event of the inlet temperature of a heater, you create a temperature sensor upstream of the heater in the process map.

Here are the available instrument palettes in the Process Modeling toolbox:





To create instruments:

1 Choose View > Toolbox - Process Modeling and choose one of the instrument palettes.

For information on displaying and interacting with the Process Modeling toolbox, see Using the Optegrity Toolboxes.

2 Click an instrument in the palette and click on the detail of the process map to place it near its associated process equipment.

For example, you would place an inlet temperature sensor for a heater upstream of the heater.

3 Choose Properties on the instrument.

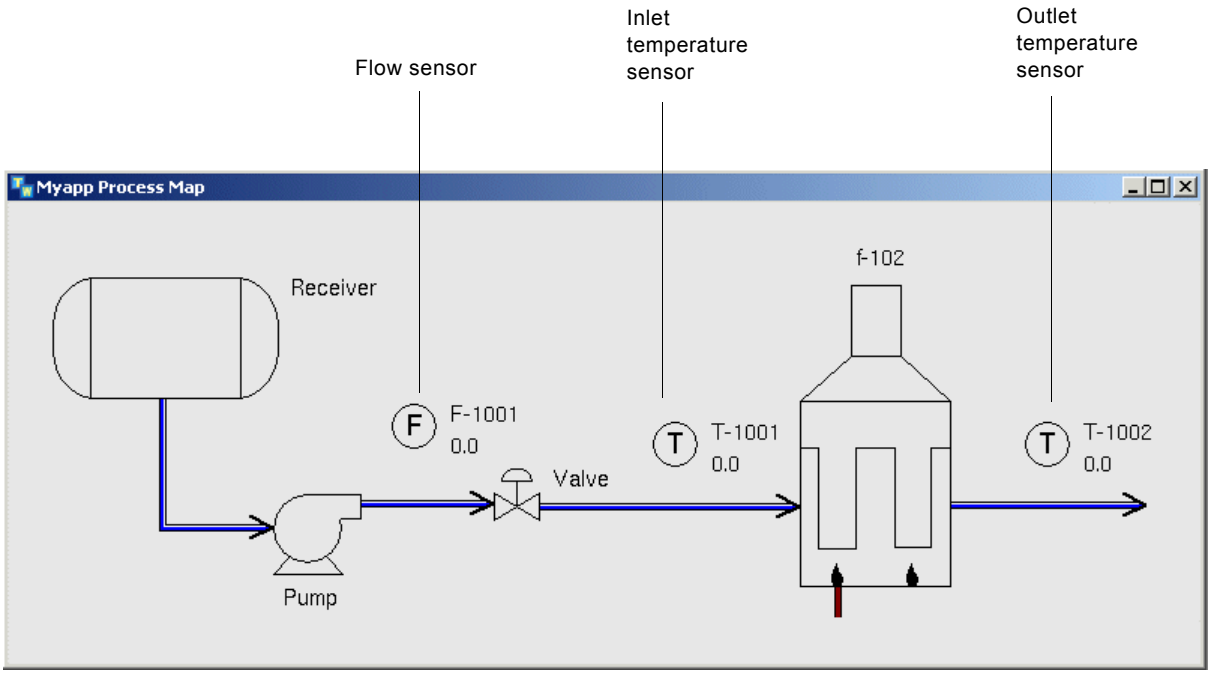
The properties dialog shows the name and built-in internal datapoints.

4 Configure the Domain Object Name.

The name can include spaces.

5 Continue creating instruments until you have all you need for the events you want to detect.

Here is the process map with various sensors configured:



Connecting Instruments

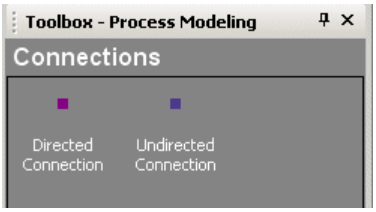
In general, it is not necessary to connect instruments in a process map. However, to provide a more visually accurate representation of your process, you might want to show connection stubs on instruments so that they appear to be connected. You typically show instruments with undirected connections.

You can also connect instruments to the connections between domain objects or to the domain objects themselves. However, note that to connect an instrument to a connection, you must either use the same type of connection, or you must connect the instrument through a connection stub, described below.

To configure the built-in heater efficiency event, you must physically connect sensors to the input fuel gas line. For an example, see Built-in Event Detection for Heaters.

To create a connection stub on an instrument:

1 Choose View > Toolbox - Process Modeling and show the Connections palette:



2 Create an undirected connection and place it near an instrument in your process map.

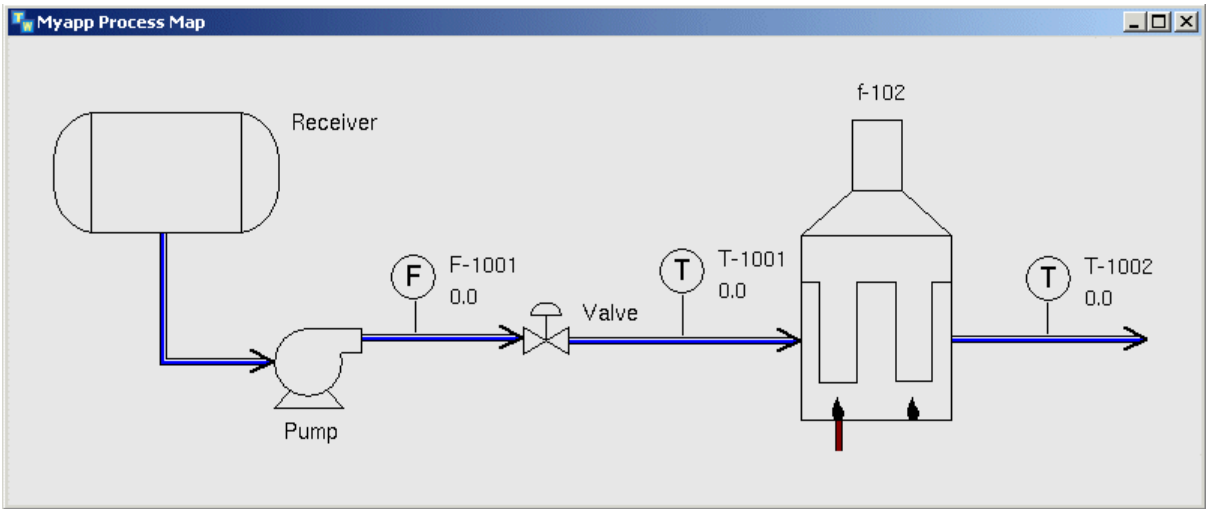
3 Create a connection stub on the instrument in the desired location.

For details, see Connecting Process Equipment.

4 Place the instrument with the connection stub next to a connection or a domain object so that it appears as to be connected.

It is not necessary that the instrument actually be connected; the connection stub is simply providing a visual representation.

For example, here is the process map where the sensors appear to be connected:



Configuring Domain Objects

Once you have created the process equipment and sensors required for event detection, you must configure each domain object in the process map, depending on the type of domain object, as follows:

- Process equipment:
  - For each domain object, you must configure the related sensors for the built-in events you want to detect. For example, to detect the PV Low sensor event for the draft oxygen of a heater, you would configure the Draft Oxygen related sensor of the heater.
  - If the domain object defines its own events, you must configure the datapoint limits for those events. For example, to detect the Efficiency Severe Change event for a heater, you would configure the Change Limit of the Efficiency Change event, along with additional related sensors.
- Instruments:
  - For each instrument, you must configure the source datapoint of each internal datapoint to refer to the external datapoint that provides its data. For sensors, you configure the PV, and for controllers, you configure the PV, SP, OP, and Mode. Typically, you configure the source datapoint automatically when you create external datapoints from a CSV file. You can also configure the source datapoint manually.
  - You can also configure the description, units, data history for plotting values on a trend chart, and logging parameters for internal datapoints.
  - To detect an event for a sensor that is related to a domain object, you must configure the datapoint limits for the events you want to detect. For example, to detect the PV High event, you must configure the Low Limit for the specific PV Low event detection diagram for the particular sensor.

Configuring Related Sensors

When the intelligent object libraries are loaded, all process equipment defines related sensors for Process Flow, Process Inlet Temperature, Process Outlet Temperature, Process Inlet Pressure, and Process Outlet Pressure.

Various other process equipment defines additional related sensors that allow you to detect additional events. For example, heaters define related sensors for Draft Oxygen, Draft Pressure, Stack NOx, Heater Efficiency, Tube Skin Temperatures, and Tube Skin Delta T Temperatures.

To detect one of the built-in events for process equipment, first, you must configure the related sensors of the domain object. For example, in the sample process map, you would configure the Process Flow, Process Inlet Temperature, and Process Outlet Temperature related sensors of the F-102 heater.

For a description of the related sensors of built-in process equipment, see Configuring Built-in Event Detection.

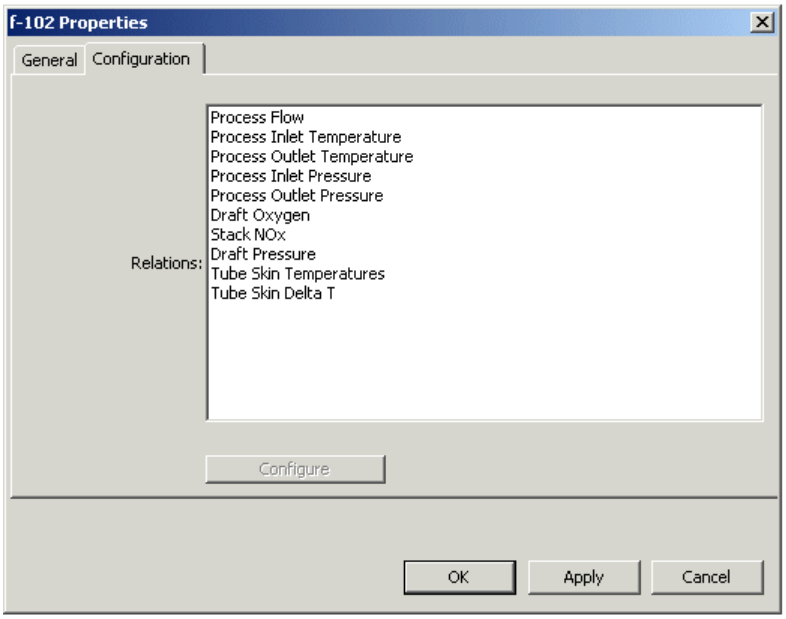
You can only configure related sensors if the intelligent object libraries are loaded or if you have created your own custom events. See Working with Projects and Creating Custom Event Detection.

To configure related sensors:

1 Display the properties dialog for a domain object, such as a heater.

2 Click the Configuration tab.

Here is the Configuration tab of the properties dialog for the F-102 heater:



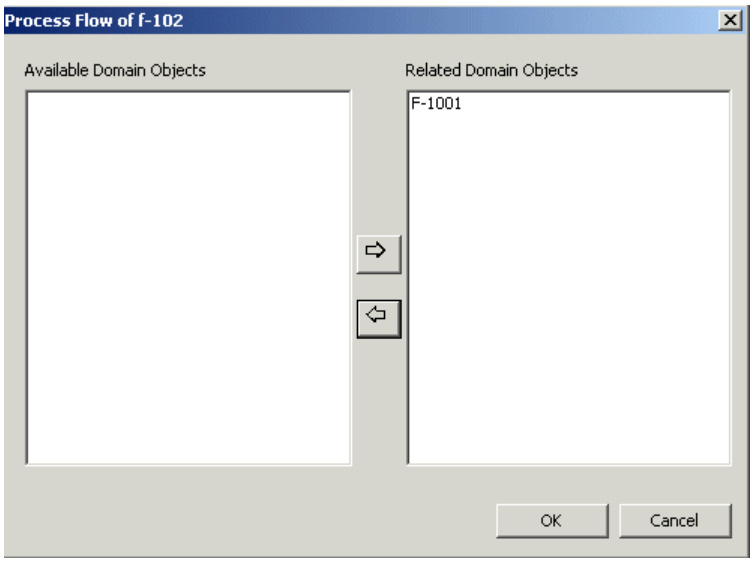
3 Select the relation for the event you want to detect and click the Configure button.

For example, to detect the PV Low sensor event for the process flow of the heater, click Process Flow.

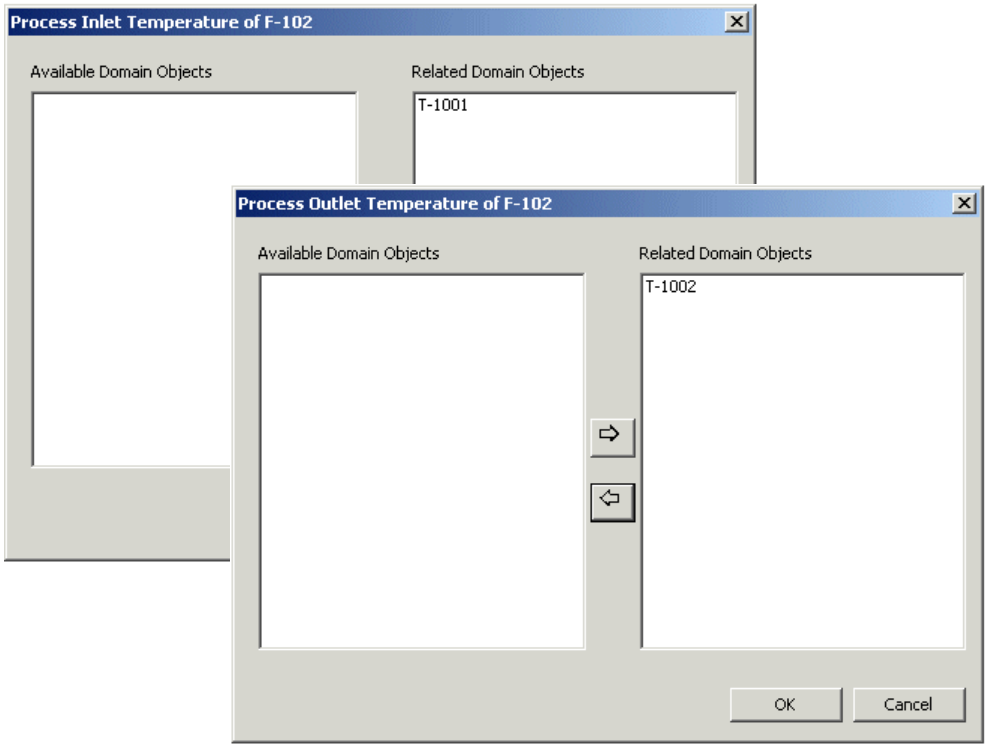
The list of available domain objects includes all sensors on the process map of the required type. For example, when configuring the Process Flow, the list of available domain objects includes only flow sensors. Similarly, when configuring Process Inlet Temperature and Process Outlet Temperature, the list of available domain objects includes only temperature sensors.

4 To configure the related sensor, move the sensor from the Available Domain Objects list to the Related Domain Objects list on the right.

For example, here is the Process Flow of the F-102 heater with the F-1001 Flow Sensor configured as the related sensor:



Here are the Process Inlet Temperature and Process Outlet Temperature of the F-102 heater with the T-1001 and T-1002 Temperature Sensors, respectively, configured as the related sensors:



Configuring Internal Datapoints

When using built-in event detection, there is no need to configure the internal datapoints of process equipment; you only need to configure the internal datapoints of instruments.

You can also create your own event detection diagrams and domain object definitions, which can use the built-in and custom internal datapoints of domain objects.

For information on...	See...
Creating custom event detection templates	Creating Generic Dataflow Diagrams.
Creating custom domain object definitions	Creating Domain Object Definitions.

To configure internal datapoints:

1. In the properties dialog for a domain object, click the row associated with an internal datapoint to display its properties dialog.

   For example, for a sensor, you configure the PV internal datapoint, and for a controller, you configure the PV, SP, OP, and Mode.

2. Configure the Category to be any user-defined symbol to use for filtering datapoints in the domain object properties dialog.

   You can choose the category from a list of existing categories by clicking the Select button, or you can enter a new category.

   In general, you only configure the category for the internal datapoints of a domain object, for example, pressure, temperature, and flow.

3. Configure the Description to provide user-defined information about the internal datapoint.

4. Configure the Units to describe the units of measurement for the datapoint.

   Typically, you configure the units automatically through a CSV file when you create the external datapoints. For details, see Converting Engineering Units.

5. Configure the Number of Historical Values or the Maximum History Age to keep a history of internal datapoint values, which you can plot in a trend chart.

6 You have two options for configuring the Source Datapoint:

- Leave it blank to be fill in automatically when you create the external datapoints from a CSV file.
- Click the button and choose an existing external datapoint from the list as the source datapoint.

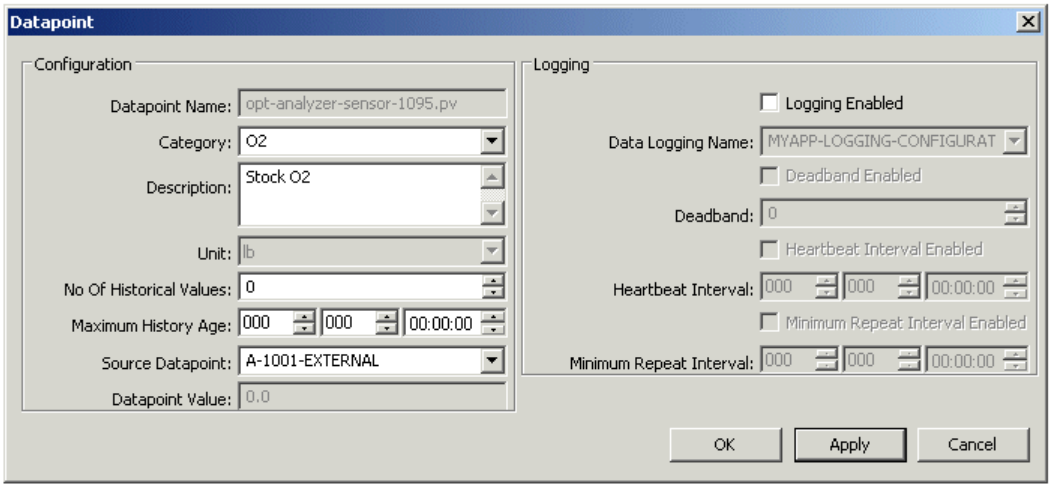
For information about creating external datapoints and configuring the Source Datapoint, see Configuring External Datapoints.

If you configure the Source Datapoint, the Datapoint Value is read-only. Otherwise, you can configure the Datapoint Value manually, and the value persists when the application is initialized.

7 Configure the logging specification for the internal datapoint.

For more information about configuring logging, see Configuring Datapoints for Logging.

Here is the default properties dialog for the a-1001.pv internal datapoint of the A-1001 oxygen analyzer of a heater:



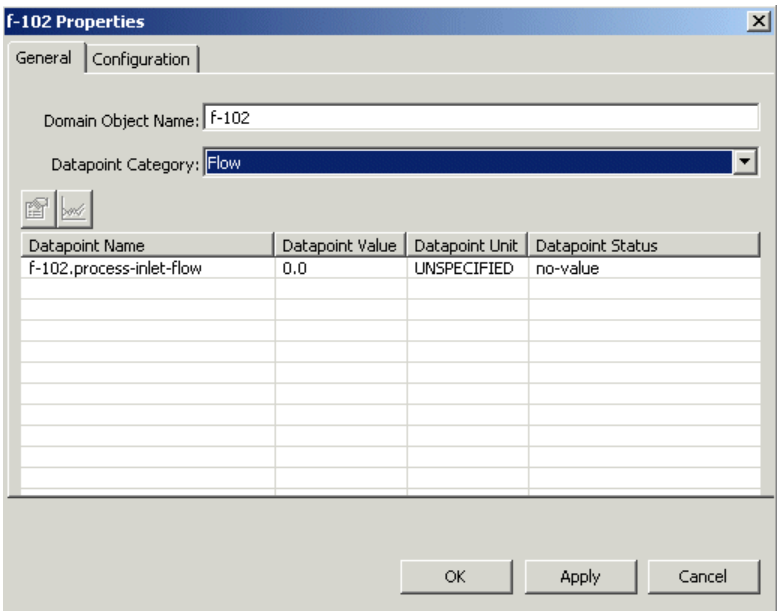
8 Accept the dialog for the internal datapoint.

9 To plot the data history of a domain object that you have configured to keep a history, select the datapoint and click the Plot History button in the domain object properties dialog.

When you initialize the process map and data values arrive, the trend chart plots the data. For an example, see Displaying Trend Charts of Datapoint Values.

10 To filter internal datapoints, based on category, choose a category from the Datapoint Category dropdown list in the domain object properties dialog.

Here is the domain object dialog for the F-102 heater with only the internal datapoints in the Flow category visible:



Configuring Built-In Event Detection

When the intelligent object libraries are loaded, all instruments have a set of built-in generic event detection templates, which describe the datapoint to monitor and the logic for detecting the event. For example, all sensors define the PV Low event detection diagram, which monitors the PV of the associated sensor, detects a low limit, applies a fuzzy truth value, filters out unchanged values, and sends a SymCure Low event if the value is true. The SymCure Low event triggers diagnostic reasoning, based on a built-in fault model to determine root causes.

When you initialize the process map, Optegrity creates specific event detection diagrams for all built-in generic event detection templates in the process map, and it automatically enables event detection for those diagrams. You configure limits in the specific event detection diagrams to determine when the event occurs in your particular process. For example, to detect the PV Low sensor event of the Process Flow related sensor of the F-102 heater, you configure the specific PV Low event detection diagram for the F-102 heater by configuring the Low Limit of the F-1001 flow sensor.

For information on configuring built-in event detection diagrams, see Configuring Built-in Event Detection.

To configure built-in event detection:

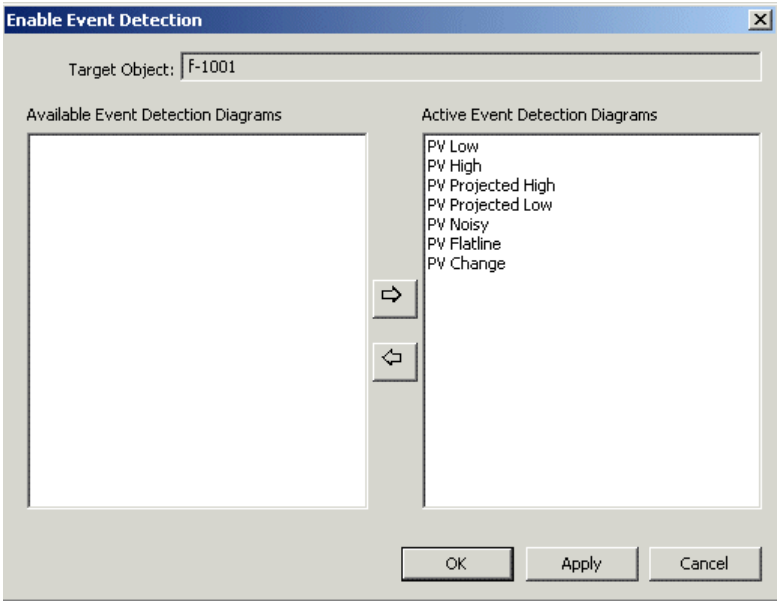
1 Choose Project > Initialize Application to create the specific event detection diagrams for each built-in event in your project.

You can also choose Initialize Domain Objects on the process map detail to initialize only the domain objects in the process map.

2 Choose Enable Dataflow Event Detections on the related sensor whose events you want to configure.

The built-in events automatically appear in the Active Domain Object Events list on the right. You can disable events as needed for the particular sensor, or just leave all the events active and only configure limits for those you want to detect.

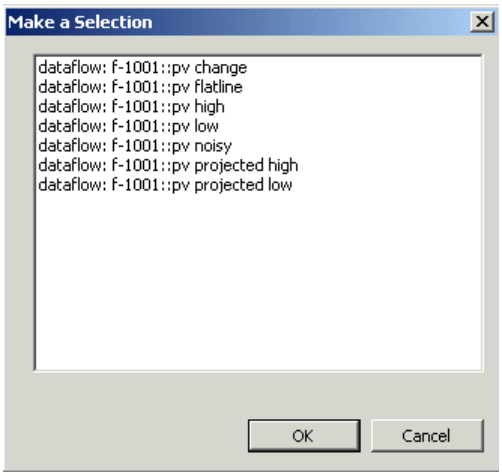
For example, here is the Enable Event Detection dialog for the F-1001 flow sensor related sensor of the F-102 heater:



3 Close this dialog, and choose Show Logic on the domain object whose event you want to configure.

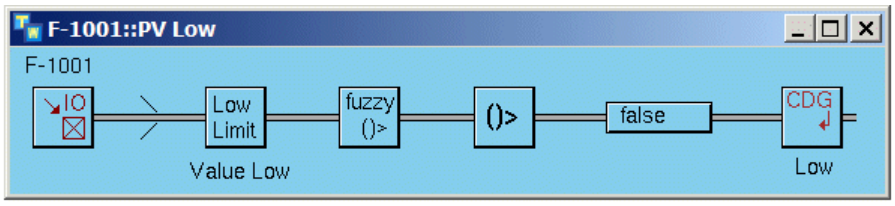
This list includes the specific event detection diagrams for all active events of the domain object. The specific event detection diagram name concatenates the domain object name and the generic event detection template name.

For example, here is the list of specific event detection diagrams for the F-1001 flow sensor:



4 Choose the specific event detection diagram whose limits you want to configure.

For example, to configure the low limit of the PV Low event of the F-1001 flow sensor, choose F-1001::PV Low to display this specific event detection diagram:

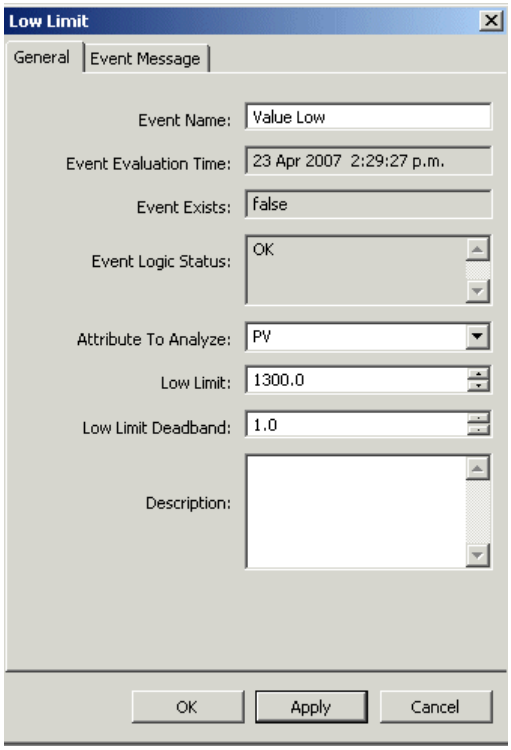


This diagram uses various GEDP blocks to monitor the PV of the associated sensor, detect a low limit, convert the value to a fuzzy truth value, filter unchanged events, display the discrete value, and send a SymCure Low event when the low limit is detected.

For more information, see Part IV, Event Detection.

5 Display the properties of the GEDP block that detects the limit and configure the values, as appropriate.

For example, here is the properties dialog for the Low Limit block with the Low Limit and Low Limit Deadband values configured:



The Low Limit Deadband defines a truth range, below which the block passes a fuzzy truth value of true. The Low Limit block passes a fuzzy truth value of 1.0, which is true, if the detected value is between the Low Limit and the Low Limit plus the Low Limit Deadband, in this example, between 1300 and 1301.

6 Accept the dialog.

The built-in PV Low event is now configured to generate an event when the F-1001 flow sensor related sensor of the F-102 heater goes too low.

7 Continue configuring the limits in the specific event detection diagrams for each domain object whose built-in events you want to detect.

For example, you might configure these specific event detection diagrams for these related sensors of the F-102 heater: PV Projected Low::T-1001 (Process Inlet Temperature), PV Projected High::T-1002 (Process Outlet Temperature).

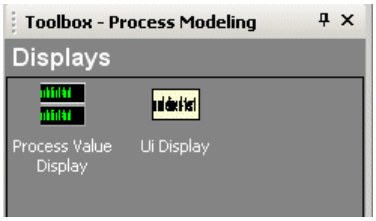
Creating Datapoint Displays

To see internal datapoint values update, you can create datapoint displays on a process map. You can create a datapoint display that shows just the value, or the datapoint name and its value.

To create a datapoint display:

1 Choose View > Toolbox - Process Modeling and show the Displays palette.

The palette contains two styles of datapoint displays:



2 Click a datapoint display and place it on a process map.

Note You can only create datapoint displays on a process map.

3 Choose Properties and configure the Description to be a text description of the datapoint.

4 Choose the Source Datapoint to display from the list of available internal datapoints in the process map.

For example, to display the PV of the F-1001 flow sensor, the Source Datapoint would be f-1001.pv.

When you run the simulation, the displays show the datapoint values as they update. For an example, see Replaying Data from CSV Files.

Creating a Process Map Hierarchy

The simplest type of process map consists of domain objects on the detail of a Process Map container. You can also define process maps hierarchically, based on Production Site, Production Area, and Process Unit containers. Production sites, production areas, and process units are all considered to be both process maps and domain objects, which means they appear in lists that contain both types of objects, such as the process map view of the operator interface.

You can also configure process map hierarchies by explicitly specifying the superior process map. This approach enables domain objects to be visible in more than one process map. For example, a sensor might be the output of equipment on one process map and the input of equipment on another.

Operators can navigate across process maps hierarchically in the process map view of the operator interface.

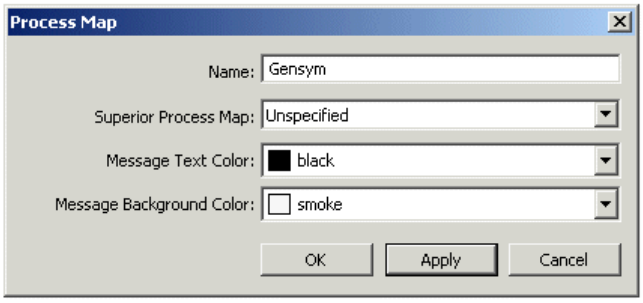
You can configure the process map so that messages that occur on domain objects in the map take on the assigned color. To do this, you must also set a parameter in the configuration file.

For information on reasoning across process maps, based on containment, see "Configuring Causal Connections" in Chapter 4 "Creating Generic Fault Models" in the *SymCure User's Guide*.

To create a process map hierarchy:

1 Choose Project > System Models > Manufacturing Processes > Manage, and click the New button to create a top-level process map, which will contain your production sites.

For example, here is a process map named Gensym, which will contain all of Gensym's production sites:



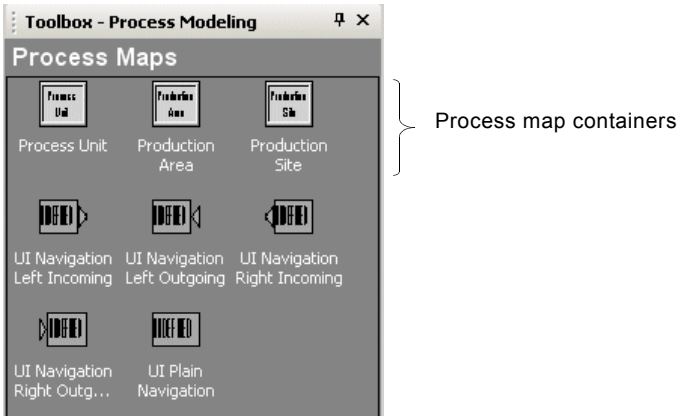
You can use Superior Process Map to configure process map hierarchies explicitly by choosing the superior process map.

Tip You can define SymCure fault models that reason across process maps that are defined hierarchically by using the cdg-contained-in and cdg-the-container-of relation types. This feature works for production sites, production areas, process units, and any domain object. However, SymCure requires that a contained object must be placed on the subworkspace of its container object for its built in containment relations to be applied.

---

2 Choose Project > System Models > Manufacturing Processes and choose the process map you just created to show its detail.

3 Choose View > Toolbox - Process Modeling and show the Process Maps palette:



4 Create one or more Production Site containers and place them on the top-level process map detail.

5 Show the detail of a Production Site, then create one or more Production Area containers and place them on the Production Site detail.

6 Show the detail of a Production Area, then create one or more Process Unit containers and place them on the Production Area detail.

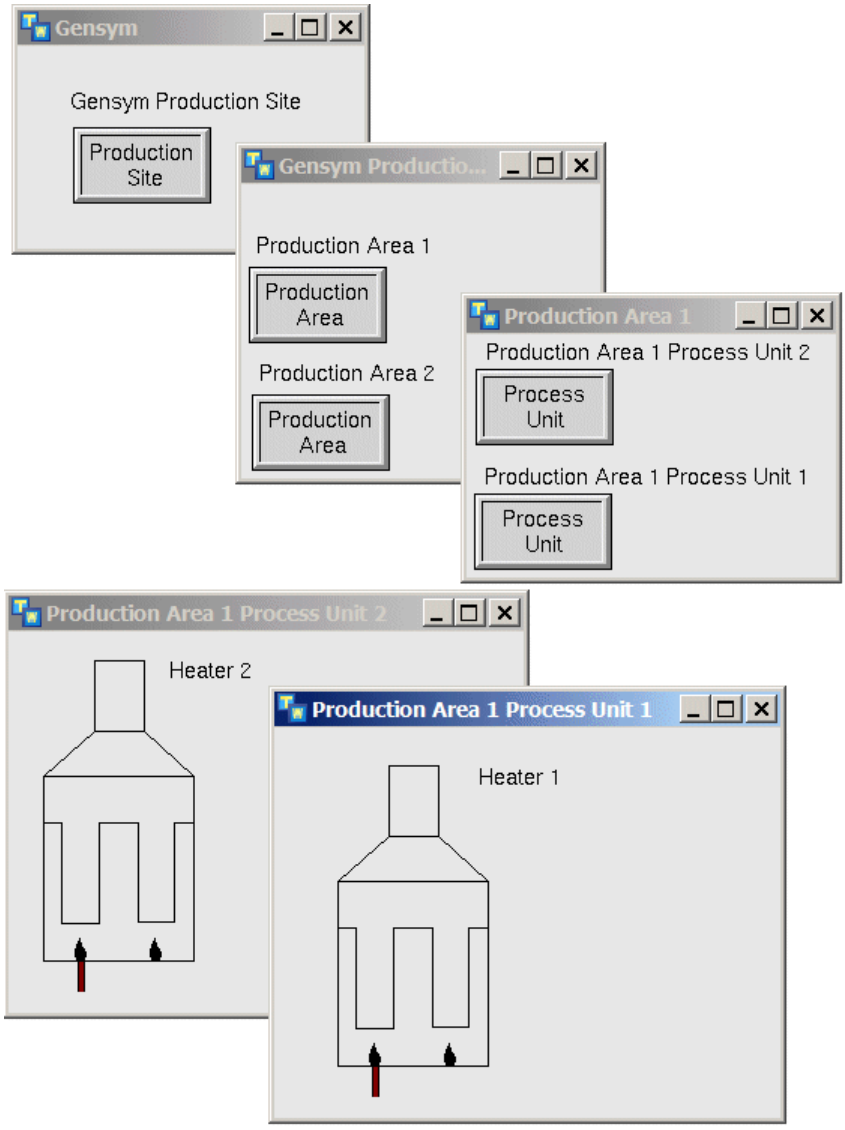
7 Configure the Name of each process map container you created.

Note You must configure the name in order to access the detail through the Project > System Models > Manufacturing Processes menu.

---

8 Show the properties of each container and configure the Background Color, as desired.

Here is an example of a process map hierarchy:

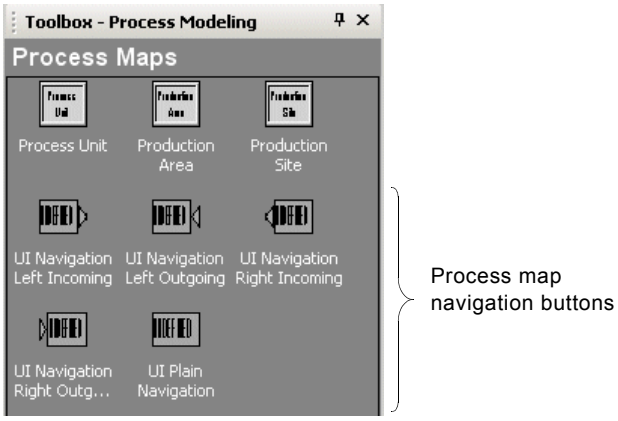


Navigating Across Process Maps

You might want to create navigation buttons between process maps to break up large maps into several smaller maps or to provide navigation between hierarchical process maps.

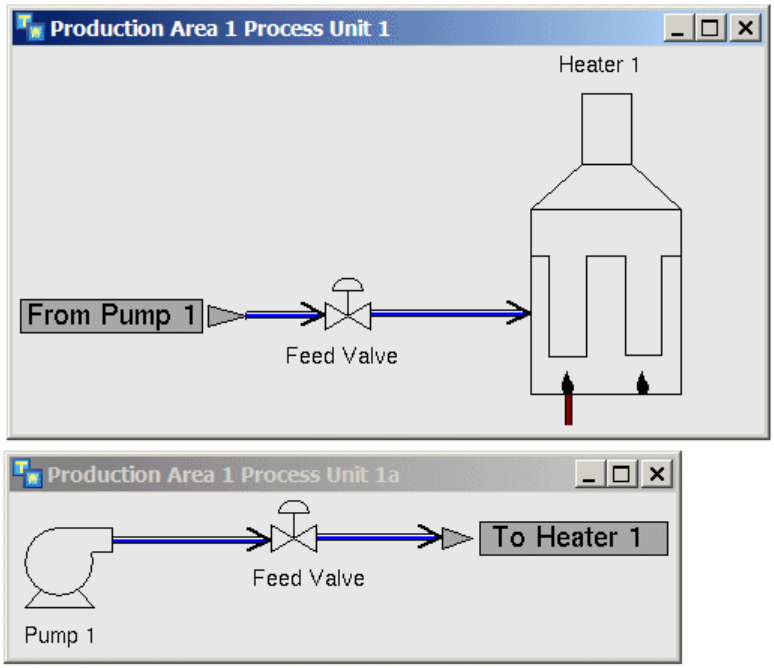
To create navigation buttons between process maps:

1. Create two process maps between which you want to navigate.
2. Choose View > Toolbox - Process Modeling and show the Process Maps palette:



3. Clone one of the navigation buttons from the Process Maps palette and place it on one of the process maps.
4. Choose Properties and configure the Name to be the text to display in the navigation button.
5. Configure the Process Map to go to by choosing the other map from the drop down list.
6. Clone and configure a navigation button on the other process map to navigate to the first map.

This figure shows navigation buttons between two process maps:



To navigate across process maps:

➔ Choose Go To Process Map on a navigation button.

To navigate hierarchically across process maps:

➔ Click the Go to Superior button on a process map, whose Superior Process Map has been configured.

Configuring Message Color Based on the Process Map

By default, messages that occur on domain objects in a process map use colors that are based on the priority of the message.

To configure message color based on the process map, you configure the message color of each process map container in the hierarchy. You must also configure a parameter in the *config.txt* file located in the *g2i\kbs* directory of your Optegrity installation directory.

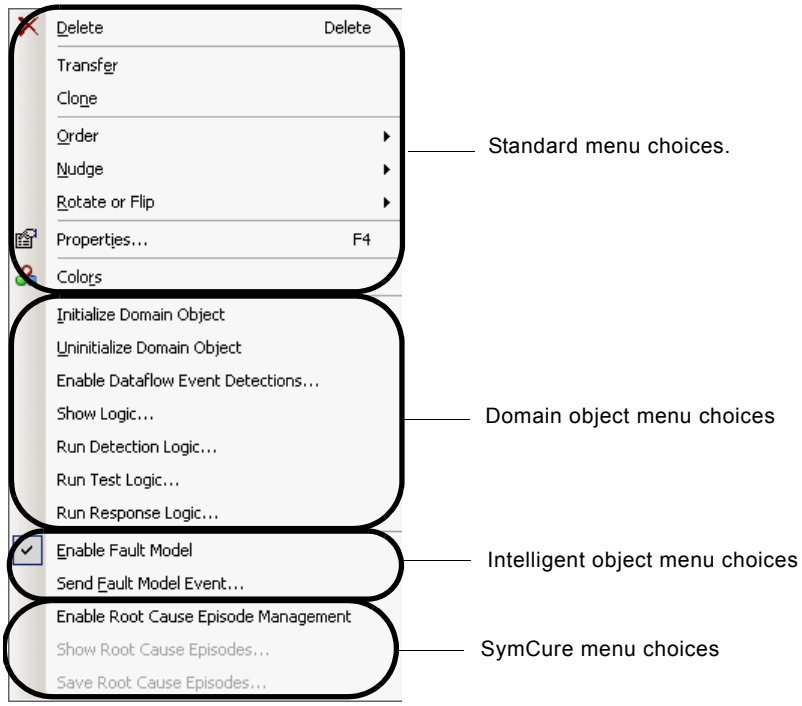
For more information on configuring message color, see Message Color.

To configure message color based on the process map:

1. Display the properties dialog of a process map through the Process Maps Manage dialog.
2. Configure the Message Text Color and Message Background Color of each process map to specify colors for messages that occur on domain objects in that process map.
3. In the configuration file, set the message-color-based-on parameter to process-map.

Interacting with Domain Objects

Domain objects provide the following popup menu choices:



All domain objects provide these popup menu choices:

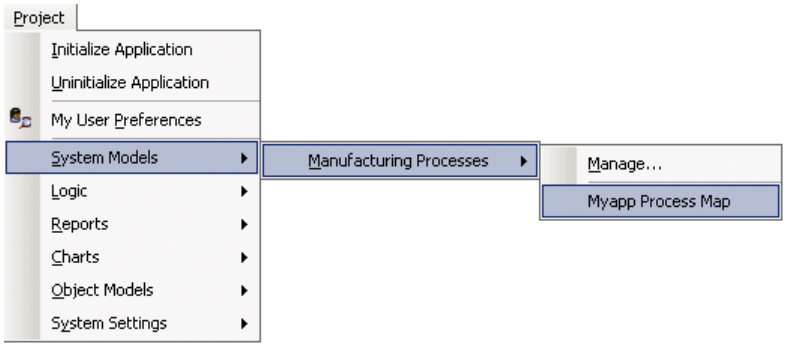
Menu Choice	Description
Initialize Domain Object	Initializes the domain object, which creates a specific GEDP diagram for the domain object, as well as performing various other required tasks. For details, see Initializing Process Maps.
Uninitialize Domain Object	Uninitializes the domain object, which performs various tasks, including deleting specific GEDP diagrams for the domain object. For details, see Uninitializing Process Maps.
Enable Dataflow Event Detection	Displays a dialog that allows you to activate and deactivate built-in and user-defined dataflow event detection diagrams. See Configuring Built-In Event Detection.
Show Logic	Shows the specific event detection, test, and response diagrams associated with the domain object. For details, see Showing Specific Dataflow Diagrams.
Run Detection Logic	Runs the specific dataflow event detection diagrams associated with the domain object.
Run Test Logic	Runs the specific dataflow test diagrams associated with the domain object.
Run Response Logic	Runs the specific dataflow response diagrams associated with the domain object.
Enable Fault Model	Enables SymCure diagnostics for the domain object. For details, see Enabling Fault Models.
Send Fault Model Event	Simulates sending a SymCure fault model event for the domain object. For details, see Sending Fault Model Events.
Enable Root Cause Episode Management Show Root Cause Episodes Save Root Cause Episodes	For information about these menu choices, see Chapter 7, "Debugging SymCure Applications" in the *SymCure User's Guide*.

Managing Process Maps

To manage process maps:

1 Choose Project > System Maps > Manufacturing Processes.

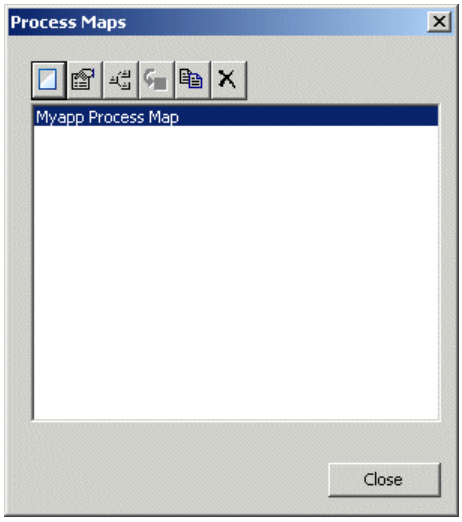
All process map containers appear in the submenu, for example:



2 To display the detail of a process map, choose one from the Manufacturing Process Maps submenu of the Domain Models menu.

3 To display a dialog for managing all process maps, including displaying the properties dialog, choose Manage.

Here is the Process Maps Manage dialog:



For information on using this dialog and the Project menu to manage process maps, see Using the Project Menu.

6

Configuring Built-in Event Detection

Describes how to configure built-in event detection for domain objects.

Introduction

When the intelligent object libraries are loaded, Optegrity provides built-in event detection for various domain object classes. This chapter describes how to configure built-in event detection for:

- Instruments
- Controllers
- Base Derived Sensors
- Heaters

- Compressors
- Equipment Drivers

Optegrity defines built-in generic fault models for domain objects that use the built-in generic event detection templates.

The built-in generic event detection templates use the Fetch Intelligent Object GEDP block to get the domain object instance of the target class. The diagrams then use one of the built-in event blocks, such as the Low Limit or High Limit sensor event block, to test an internal datapoint of the domain object, for example, the pv of a sensor.

The built-in generic event detection templates are scheduled to evaluate on a regular basis. Most diagrams are configured to evaluate once every 15 seconds, with some exceptions. However, you can configure the specific diagrams to evaluate at any interval required by the frequency of your application's data acquisition.

When you initialize your application, Optegrity creates specific event detection diagrams for each built-in event detection template. By default, these diagrams are persistent, which means you can configure event limits in the specific diagrams and the configurations will persist when the application is initialized or uninitialized.

The built-in event detection templates provide typical events that you might want to detect for various domain object classes. However, you can also create custom event detection templates for built-in or custom domain object classes. For more information, see Creating Custom Event Detection.

Note You cannot delete the built-in event detection templates or generic fault models.

For general information on configuring domain objects for event detection, see Configuring Domain Objects.

The following sections assume the intelligent object libraries are loaded.

Built-in Event Detection for Instruments

Instruments provide a generic methodology for monitoring and diagnosing common problems with process sensors, online analyzers, and controllers. Instruments perform these functions by analyzing current and historical process values. Recognizing a sensor or online analyzer problem or failure can be critical to preventing process upsets.

All instruments can detect these events:

- PV High
- PV Low

- PV Projected High
- PV Projected Low
- PV Change
- PV Flatline
- PV Noisy

All events generate operator messages when the event occurs. Some events also generate a SymCure fault model event when the event is true, which triggers diagnostic reasoning to determine root causes.

For information about the generated SymCure events, see Built-in Generic Fault Models for Sensors.

In the description of configuring each of the following events, you must first create and configure an instrument from the Instruments or Controllers palette of the Process Modeling toolbox, then you must initialize the process map. For details, see:

- Creating Instruments.
- Configuring Internal Datapoints.
- Initializing Process Maps.

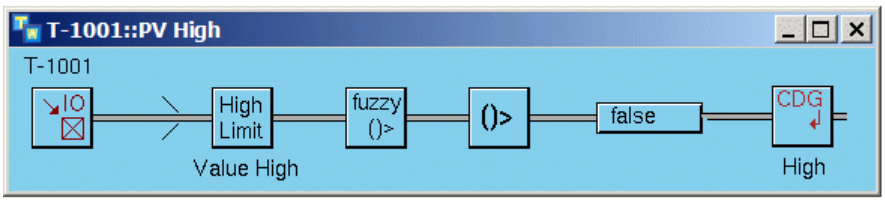
PV High

The PV High event detects when a process value exceeds a high PV limit and generates an operator message when the event is true.

To configure the PV High event:

1 Choose Show Logic on an instrument and choose the PV High event.

Here is the specific event detection diagram for the PV High event of the T-1001 sensor:



The event detects process values above a specified limit, converts the value to a fuzzy truth value, filters out unchanged values, and generates a SymCure High event when the high limit is detected.

2 Display the properties dialog for the High Limit block in the specific event detection diagram.

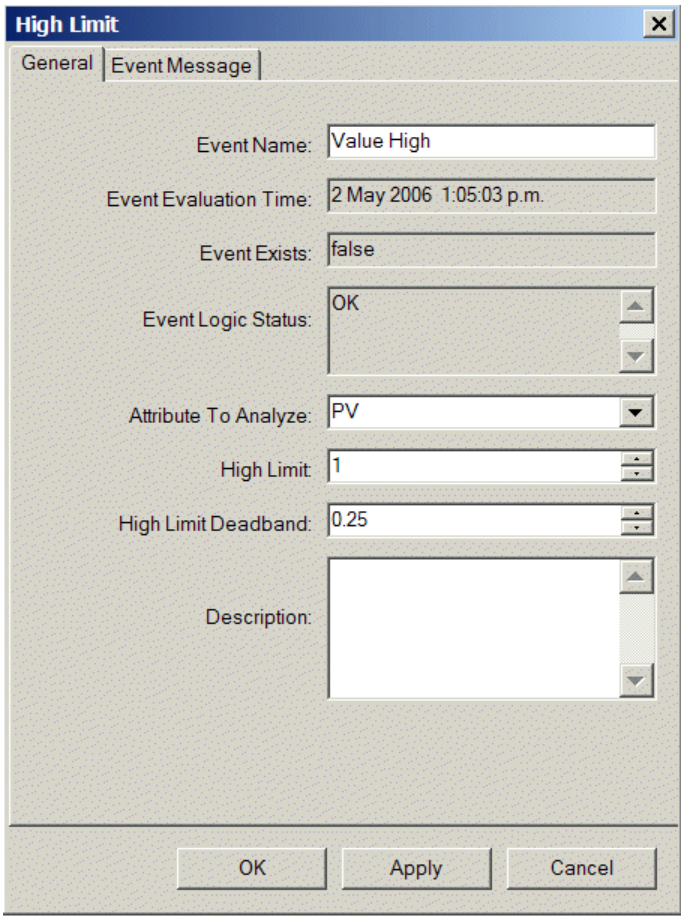
3 Configure the High Limit to be the high PV limit for the sensor.

4 Configure the High Limit Deadband to be a range for determining when to pass a fuzzy truth value.

The High Limit block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the PV is between the High Limit minus the High Limit Deadband and the High Limit.

The Discretize Fuzzy Value block that follows the High Limit block acts as a filter to prevent event chattering. It passes a discrete value of true if the input is above the Max Threshold and a discrete value of false if input is below the Min Threshold. By default, the Max Threshold is 1.0 and Min Threshold is -1.0, which means that the input must be 1.0 before it passes a value of true and -1.0 before it passes a value of false.

Here is the properties dialog for the High Limit block that detects process values between 0.75 and 1.0:



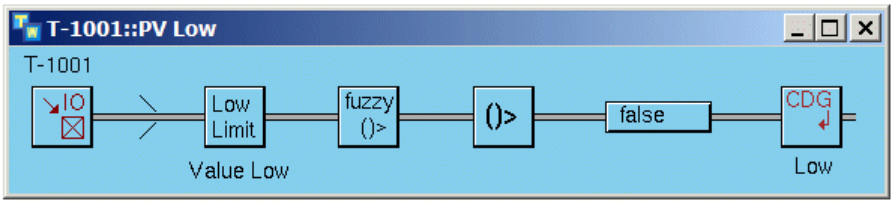
PV Low

The PV Low event detects when a process value falls below a low PV limit and generates an operator message when the event is true.

To configure the PV Low event:

1 Choose Show Logic on an instrument and choose the PV Low event.

Here is the specific event detection diagram for the PV Low event of the T-1001 sensor:



The event detects process values below a specified limit, converts the value to a fuzzy truth value, filters out unchanged values, and generates a SymCure Low event when the low limit is detected.

2 Display the properties dialog for the Low Limit block in the specific event detection diagram.

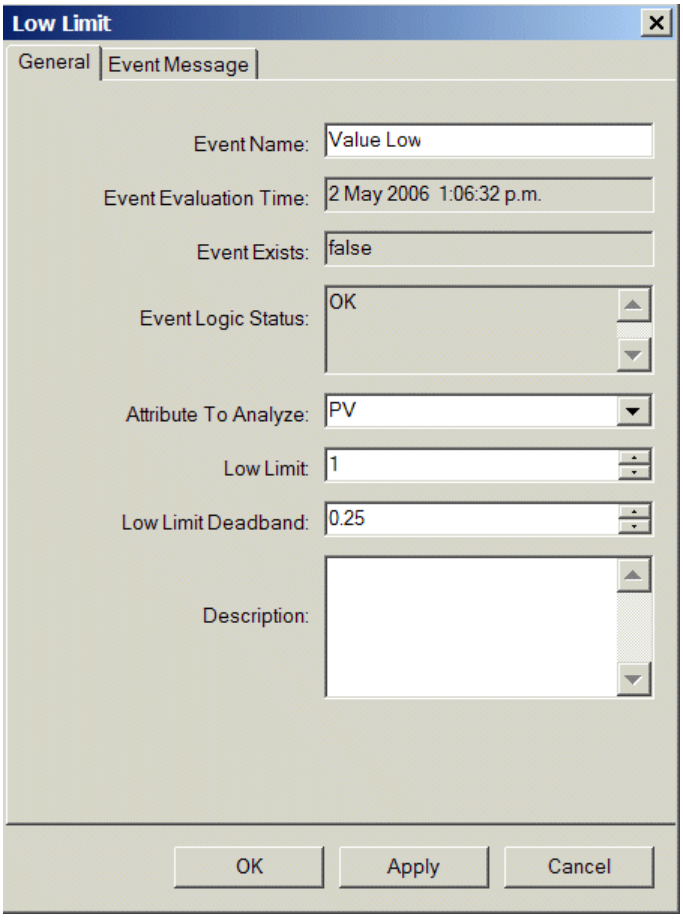
3 Configure the Low Limit to be the low PV limit for the sensor.

4 Configure the Low Limit Deadband to be a range for determining when to pass a fuzzy truth value.

The Low Limit block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the PV is between the Low Limit plus the Low Limit Deadband and the Low Limit.

For a description of the Discretize Fuzzy Value block, see PV High.

Here is the properties dialog for the Low Limit block that detects process values between 1.0 and 1.25:



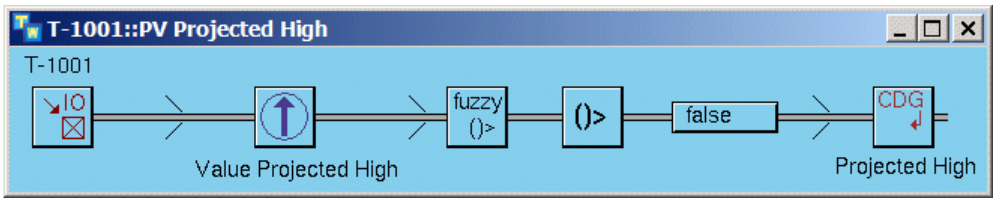
PV Projected High

The PV Projected High event detects a process value that is about to violate a high limit, based on a linear regression of future values. The event generates an operator message when the event is true and sends a SymCure Projected High event on a sensor.

To configure the PV Projected High event:

1 Choose Show Logic on an instrument and choose the PV Projected High event.

Here is the specific event detection diagram for the PV Projected High event of the T-1001 sensor:



2 Display the properties dialog for the Projected High block in the specific event detection diagram.

3 Configure the High Limit to be the projected high PV limit for the sensor.

4 Configure the High Limit Deadband to be a range for determining when to pass a fuzzy truth value.

5 Configure the Response Time to be the time period for event detection, in minutes.

6 Configure the Minimum History Points to be the minimum number of history points required for event detection.

7 Configure the Pearson R Limit to be the minimum absolute Pearson R correlation value required for validating the regressed slope.

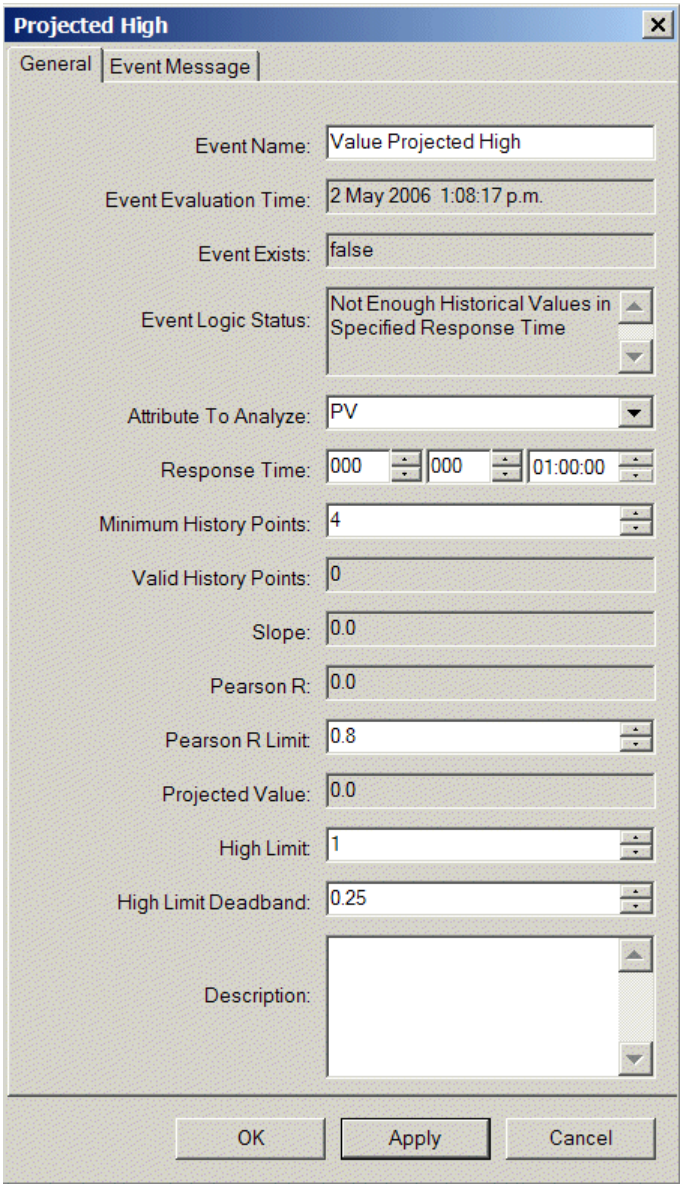
The Pearson R is a quantitative measure of how well a straight line fits the historical data within the response time. The Pearson R is a value between -1.0 (perfect negative slope correlation) and 1.0 (perfect positive slope correlation).

The block calculates the Projected Value by multiplying the calculated Slope by the Response Time and adding the result to the current value of the attribute being evaluated.

The Projected High block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the Projected Value is between the High Limit minus the High Limit Deadband and the High Limit, and the Pearson R value is above the Pearson R Limit.

For a description of the Discretize Fuzzy Value block, see PV High.

Here is the properties dialog for the Projected High block that detects projected high values between 0.75 and 1.0:



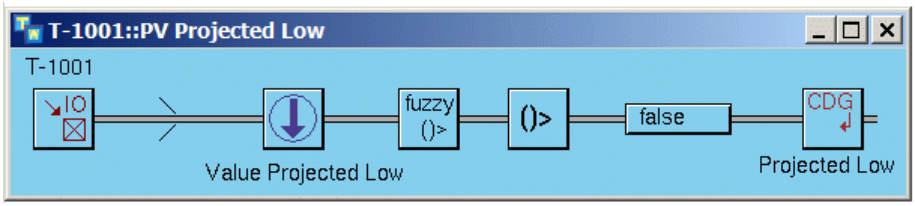
PV Projected Low

The PV Projected Low event detects a process value that is about to violate a low limit, based on a linear regression of future values. The event generates an operator message when the event is true and sends a SymCure Projected Low event on a sensor.

To configure the PV Projected Low event:

1 Choose Show Logic on an instrument and choose the PV Projected Low event.

Here is the specific event detection diagram for the PV Projected Low event of the T1 sensor:



2 Display the properties dialog for the Projected Low block in the specific event detection diagram.

3 Configure the Low Limit to be the projected low PV limit for the sensor.

4 Configure the Low Limit Deadband to be a range for determining when to pass a fuzzy truth value.

5 Configure the Response Time to be the time period for event detection, in minutes.

6 Configure the Minimum History Points to be the minimum number of history points required for event detection.

7 Configure the Pearson R Limit to be the minimum absolute Pearson R correlation value required for validating the regressed slope.

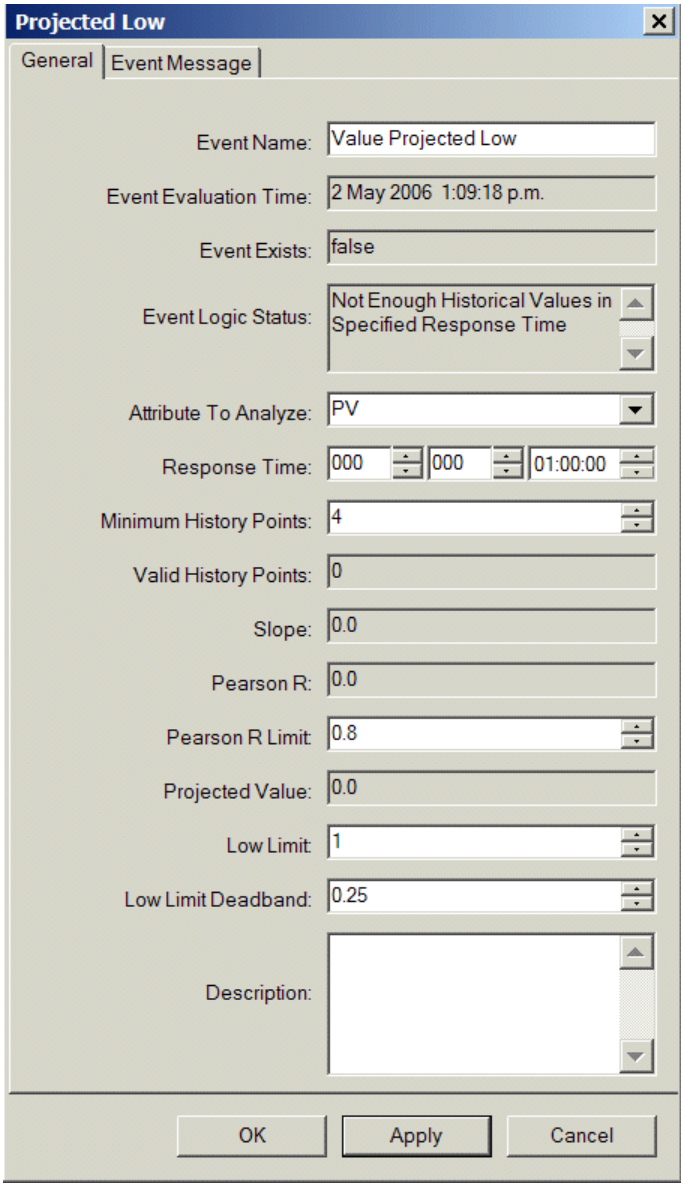
The Pearson R is a quantitative measure of how well a straight line fits the historical data within the response time. The Pearson R is a value between -1.0 (perfect negative slope correlation) and 1.0 (perfect positive slope correlation).

The block calculates the Projected Value by multiplying the calculated Slope by the Response Time and adding the result to the current value of the attribute being evaluated.

The Projected Low block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the Projected Value is between the Low Limit plus the Low Limit Deadband and the Low Limit, and the Pearson R value is above the Pearson R Limit.

For a description of the Discretize Fuzzy Value block, see PV High.

Here is the properties dialog for the Projected Low block that detects projected low values between 1.0 and 1.25:



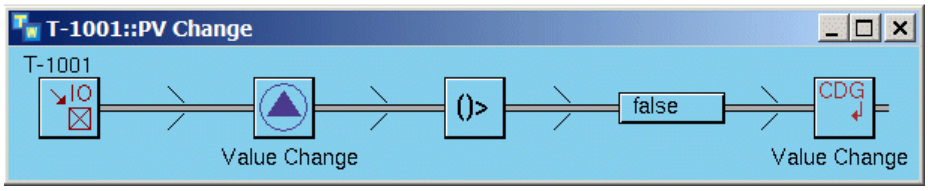
PV Change

The PV Change event detects a process value that has changed by more than a specified amount within a given time period. The event generates an operator message when the event is true and sends a SymCure Value Change event on a sensor.

To configure the PV Change event:

1 Choose Show Logic on an instrument and choose the PV Change event.

Here is the specific event detection diagram for the PV Change event of the T-1001 sensor:



The event filters out unchanged values and generates a SymCure Value Change event when the change limit is detected.

2 Display the properties dialog for the Change block in the specific event detection diagram for a sensor.

3 Configure the Change Limit to be the high limit for process value changes.

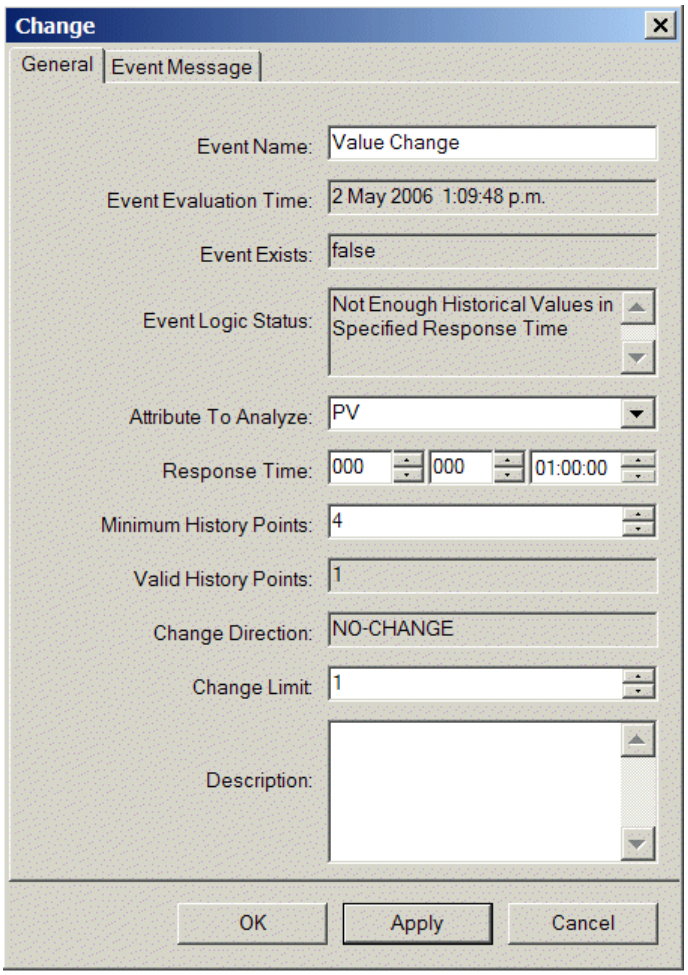
4 Configure the Response Time to be the time period for event detection, in minutes.

5 Configure the Minimum History Points to be the minimum number of history points required for event detection.

The Change block passes a value of true if the PV changes by more than the Change Limit within the Response Time.

The block calculates the Change Direction to indicate the direction of change, which is Increased, Decreased, or No-Change.

Here is the properties dialog for the Change block that detects a change in process value of more than 1 within one hour:



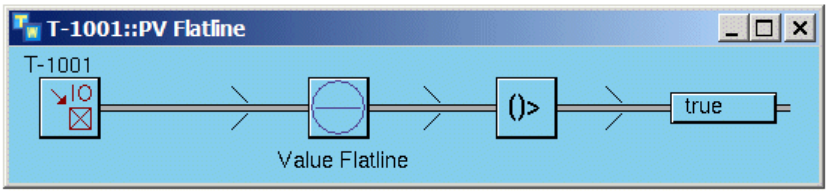
PV Flatline

The PV Flatline event detects a process value that has not changed within a given time period. The event generates an operator message when the event is true.

To configure the PV Flatline event:

1 Choose Show Logic on an instrument and choose the PV Flatline event.

Here is the specific event detection diagram for the PV Flatline event of the T-1001 sensor:



The event filters out unchanged values.

2 Display the properties dialog for the Flatline block in the specific event detection diagram for a sensor.

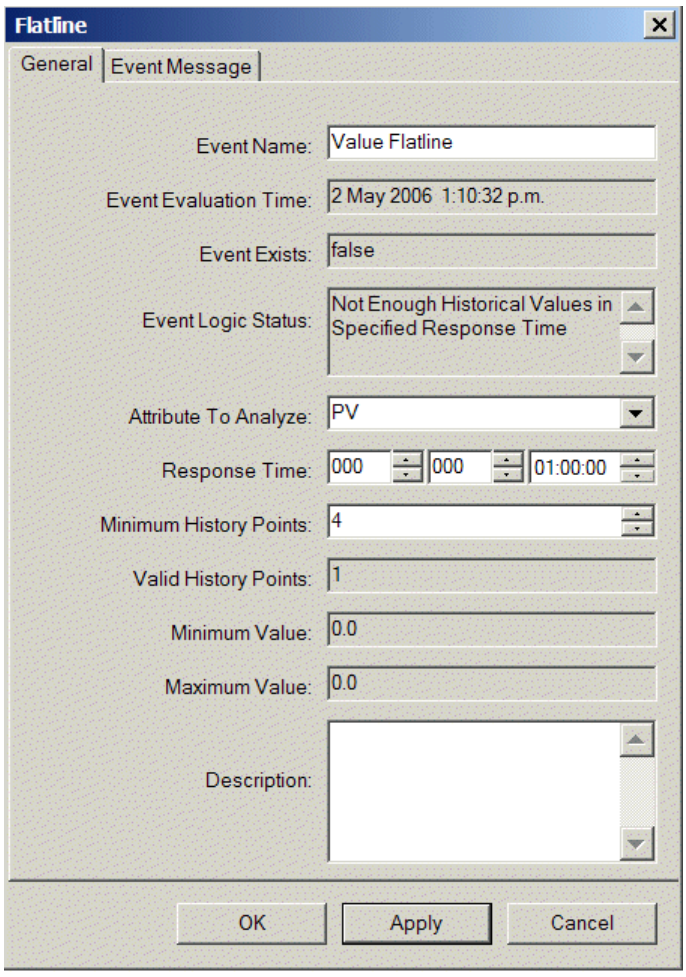
3 Configure the Response Time to be the time period for event detection, in minutes.

4 Configure the Minimum History Points to be the minimum number of history points required for event detection.

The block passes a value of true if the PV has not changed within the Response Time.

The block calculates the Minimum Value and Maximum Value of the PV during the Response Time.

Here is the properties dialog for the Flatline block that detects no changes within an hour:



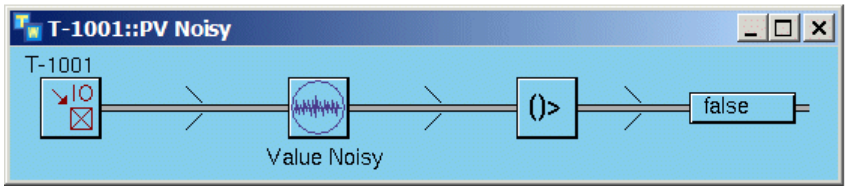
PV Noisy

The PV Noisy event detects a process value whose standard deviation has violated a limit within a given time period. The event generates an operator message when the event is true.

To configure the PV Noisy event:

1 Choose Show Logic on an instrument and choose the PV Noisy event.

Here is the specific event detection diagram for the PV Noisy event of the T-1001 sensor:



The event filters out unchanged values.

2 Display the properties dialog for the Noisy block in the specific event detection diagram for a sensor.

3 Configure the Standard Deviation Limit to be the limit for changes in the standard deviation of the process value.

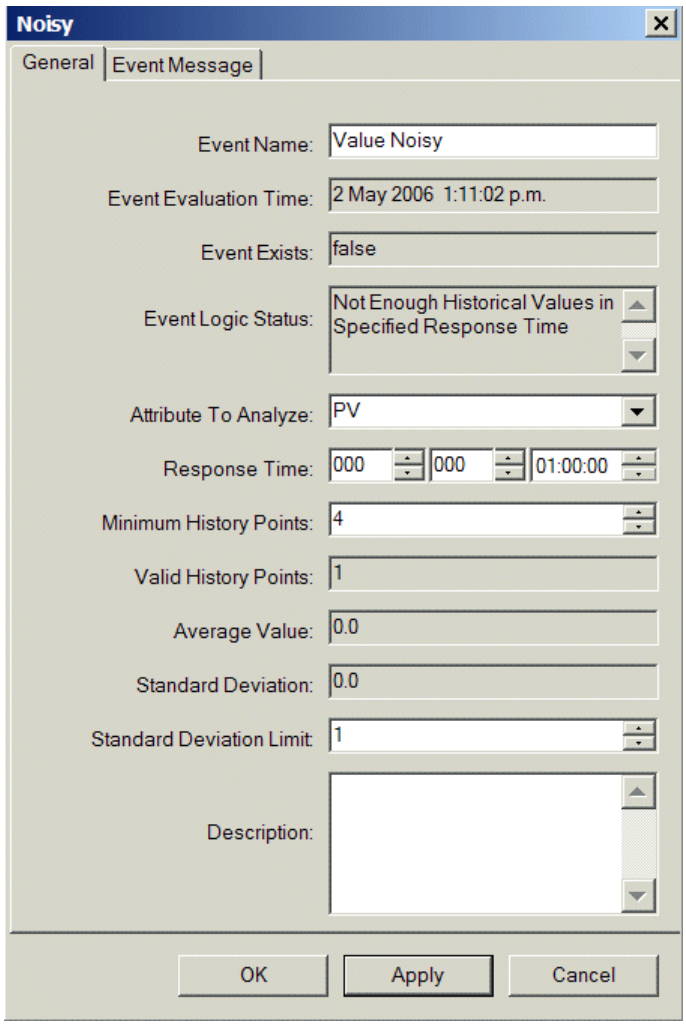
4 Configure the Response Time to be the time period for event detection, in minutes.

5 Configure the Minimum History Points to be the minimum number of history points required for event detection.

The block calculates the Average Value and Standard Deviation during the Response Time.

The block passes a value of true if the Standard Deviation changes by more than the Standard Deviation Limit within the Response Time.

Here is the properties dialog for the Noisy block that detects a noisy signal if the standard deviation of the process value within an hour is greater than 1:



Built-in Event Detection for Controllers

Controllers provide a generic methodology for monitoring and diagnosing common problems with controllers. Controllers perform these functions by analyzing current and historical process values. Recognizing a controller problem or failure can be critical to preventing process upsets.

Controllers can detect all the events that a sensor detects, as well as these events:

- OP Projected High
- OP Projected Low
- Setpoint Error

In the description of configuring each of the following events, you must first create and configure a controller from the Controllers palette of the Process Modeling toolbox, then you must initialize the process map. For details, see:

- Creating Instruments.
- Configuring Internal Datapoints.
- Initializing Process Maps.

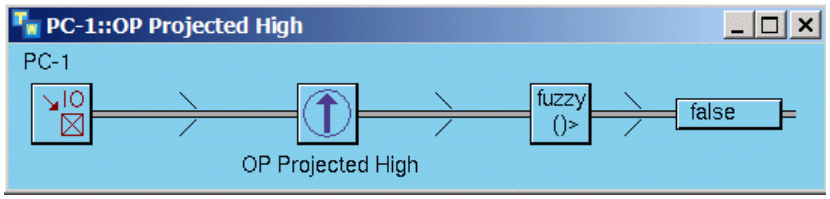
OP Projected High

The OP Projected High event detects an OP value that is about to violate a high limit, based on a linear regression of future values. The event generates an operator message when the event is true.

To configure the OP Projected High event:

1 Choose Show Logic on a controller and choose the OP Projected High event.

Here is the specific event detection diagram for the OP Projected High event of the PC-1 controller:



The event converts the projected high value to a fuzzy truth value.

2 Display the properties dialog for the Projected High block in the specific event detection diagram.

3 Configure the High Limit to be the projected high OP limit for the controller.

4 Configure the High Limit Deadband to be a range for determining when to pass a fuzzy truth value.

5 Configure the Response Time to be the time period for event detection, in minutes.

6 Configure the Minimum History Points to be the minimum number of history points required for event detection.

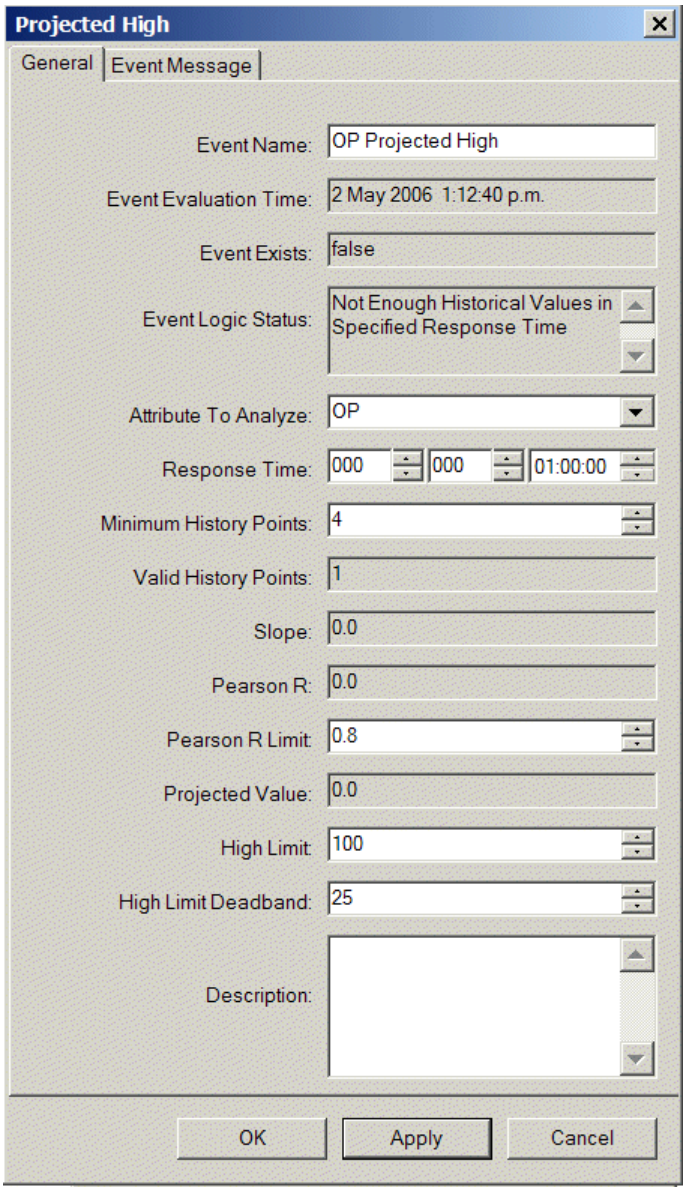
7 Configure the Pearson R Limit to be the minimum absolute Pearson R correlation value required for validating the regressed slope.

The Pearson R is a quantitative measure of how well a straight line fits the historical data within the response time. The Pearson R is a value between -1.0 (perfect negative slope correlation) and 1.0 (perfect positive slope correlation).

The block calculates the Projected Value by multiplying the calculated Slope by the Response Time and adding the result to the current value of the attribute being evaluated.

The Projected High block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the Projected Value is between the High Limit minus the High Limit Deadband and the High Limit, and the Pearson R value is above the Pearson R Limit.

Here is the properties dialog for the Projected High block that detects projected high values between 75.0 and 100.0:



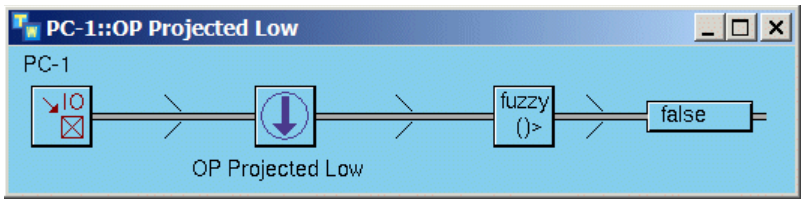
OP Projected Low

The OP Projected Low event detects an OP value that is about to violate a low limit, based on a linear regression of future values. The event generates an operator message when the event is true.

To configure the OP Projected Low event:

1 Choose Show Logic on a controller and choose the OP Projected Low event.

Here is the specific event detection diagram for the OP Projected Low event of the PC-1 controller:



The event converts the projected low value to a fuzzy truth value.

2 Display the properties dialog for the Projected Low block in the specific event detection diagram for a controller.

3 Configure the Low Limit to be the projected low OP limit for the controller.

4 Configure the Low Limit Deadband to be a range for determining when to pass a fuzzy truth value.

5 Configure the Response Time to be the time period for event detection, in minutes.

6 Configure the Minimum History Points to be the minimum number of history points required for event detection.

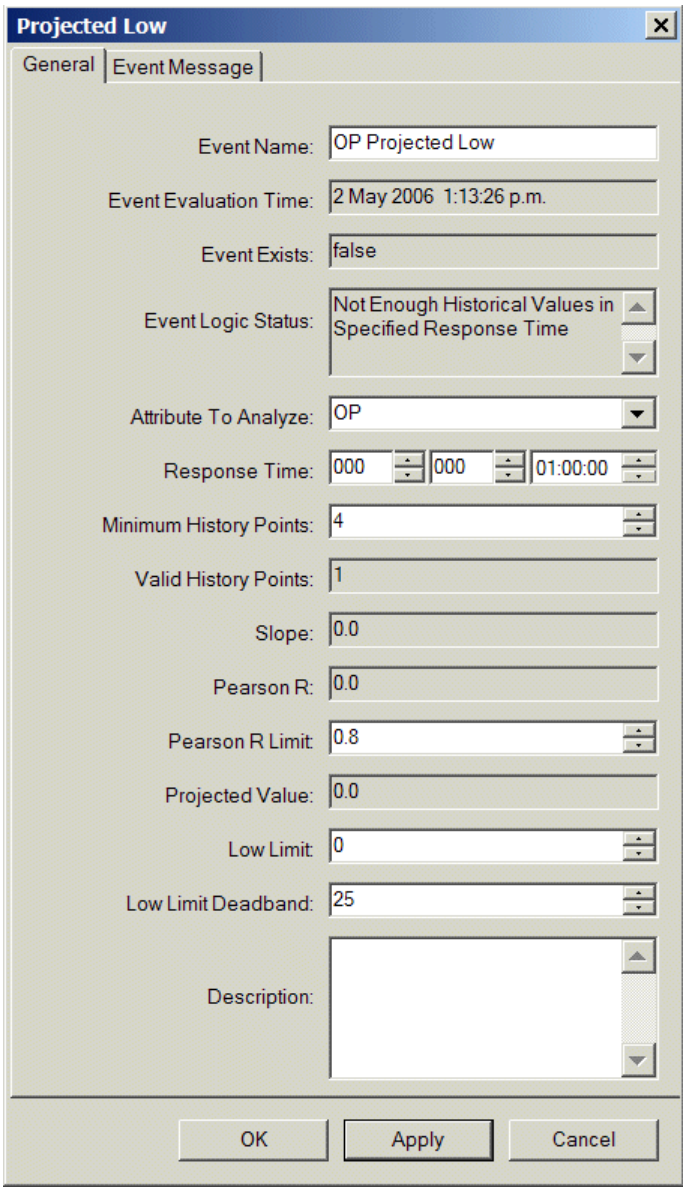
7 Configure the Pearson R Limit to be the minimum absolute Pearson R correlation value required for validating the regressed slope.

The Pearson R is a quantitative measure of how well a straight line fits the historical data within the response time. The Pearson R is a value between -1.0 (perfect negative slope correlation) and 1.0 (perfect positive slope correlation).

The block calculates the Projected Value by multiplying the calculated Slope by the Response Time and adding the result to the current value of the attribute being evaluated.

The Projected Low block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the Projected Value is between the Low Limit plus the Low Limit Deadband and the Low Limit, and the Pearson R value is above the Pearson R Limit.

Here is the properties dialog for the Projected Low block that detects projected low values between 0 and 25:



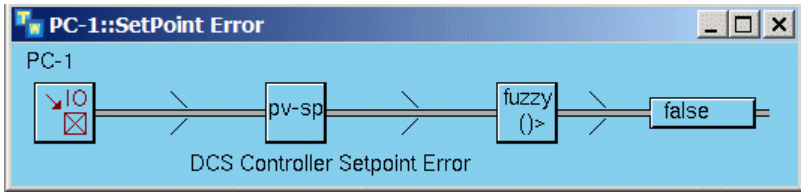
Setpoint Error

The Setpoint Error event detects when the average of the absolute value of the difference between the PV and SP violates a specified limit. The event generates an operator message when the event is true.

To configure the Setpoint Error event:

1 Choose Show Logic on a controller and choose the Setpoint Error event for the controller.

Here is the specific event detection diagram for the Setpoint Error event of the PC-1 controller:



The event converts the average setpoint error to a fuzzy truth value.

2 Display the properties dialog for the Setpoint Error Event block in the specific event detection diagram for a controller.

3 Configure the Average Error Limit to be the error limit for the controller.

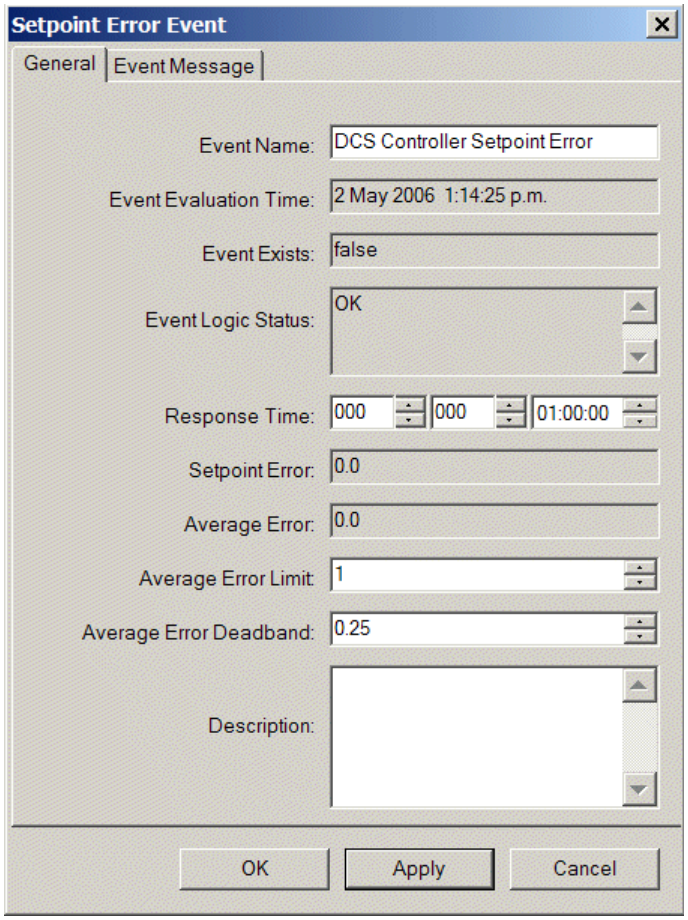
4 Configure the Average Error Deadband to be a range for determining when to pass a fuzzy truth value.

5 Configure the Response Time to be the time period for event detection, in minutes.

The block calculates the Setpoint Error to be the absolute value of the difference between the SP and the PV of the controller, and the Average Error to be an average of the Setpoint Error during the Response Time.

The Setpoint Error Event block passes a fuzzy truth value between -1.0 (false) and 1.0 (true) if the Average Error is between the Average Error Limit minus the Average Error Deadband and the Average Error.

Here is the properties dialog for the Setpoint Error Event block that is configured to detect an average setpoint error between 0.75 and 1.0:



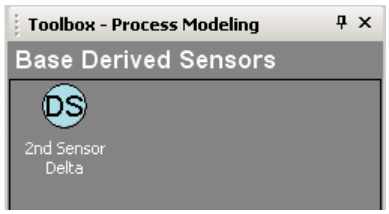
Built-in Event Detection for Base Derived Sensors

The 2nd Sensor Delta derived sensor defines the 2nd Sensor Mismatch event to detect the difference between two sensor values. The event occurs if the absolute value of the difference exceeds a high limit. The event generates an operator message when true.

To detect the 2nd Sensor Mismatch event, you create a 2nd Sensor Delta derived sensor and configure the PV to be the difference between two related sensors. You then configure the High Limit of the 2nd Sensor Mismatch event detection diagram to specify the high limit for the delta.

To configure the 2nd Sensor Delta event:

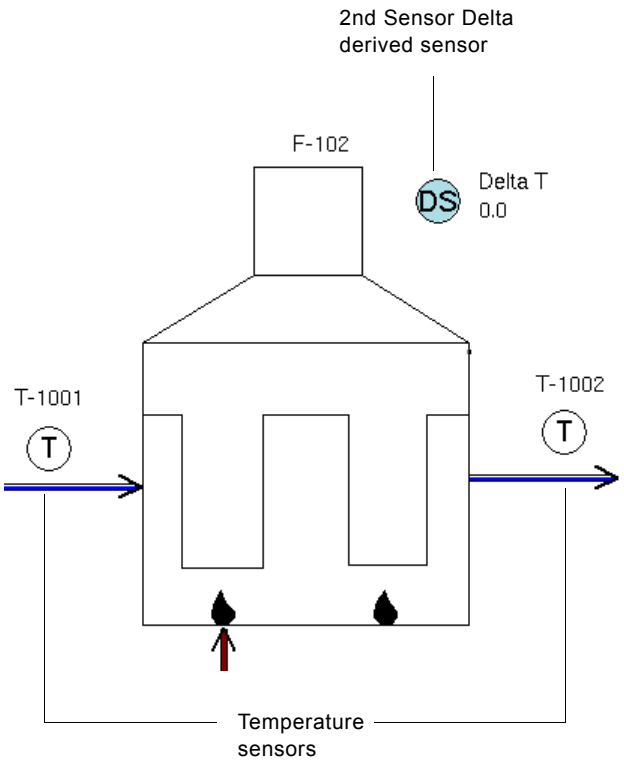
1 Create a 2nd Sensor Delta derived sensor from the Base Derived Sensors palette of the Process Modeling toolbox:



2 Create two or more sensors or controllers from the Instruments or Controllers palette of the Process Modeling toolbox and place them on your process map.

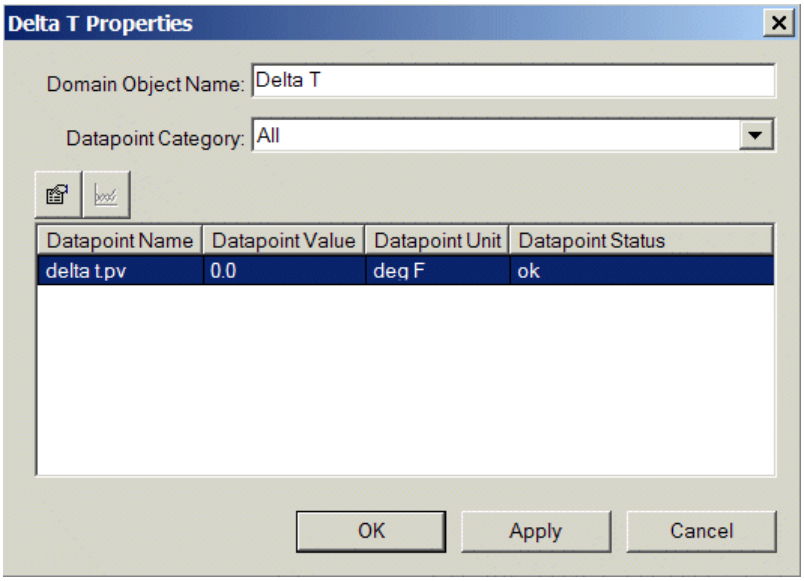
For details, see Creating Instruments and Configuring Internal Datapoints.

For example, here is the F-102 heater with an inlet and outlet temperature sensor, and a 2nd Sensor Delta derived sensor:



3 Display the properties dialog of the 2nd Sensor Delta derived sensor and click the PV internal datapoint.

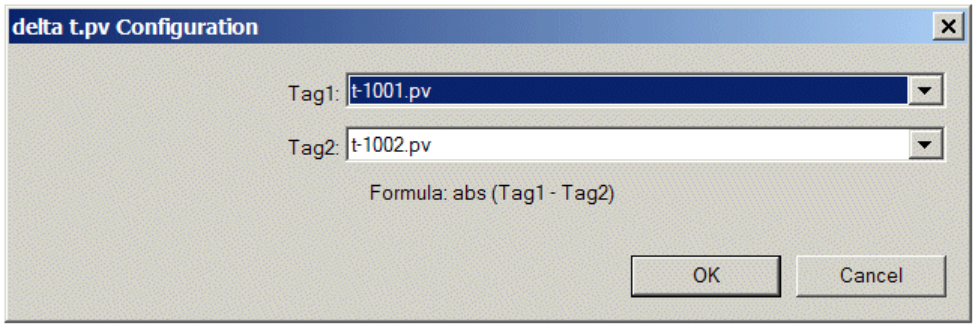
Here is the properties dialog for the Delta T 2nd Sensor Delta derived sensor with the PV internal datapoint selected:



4 Click the Properties button and configure Tag1 and Tag2 to be the sensors for which to compute the difference.

The derived sensor computes the absolute value of the difference by using the formula abs(Tag1 minus Tag2).

For example, here is how you would configure the 2nd Sensor Delta derived sensor to calculate the delta between the T-1001 and T-1002 temperature sensors:

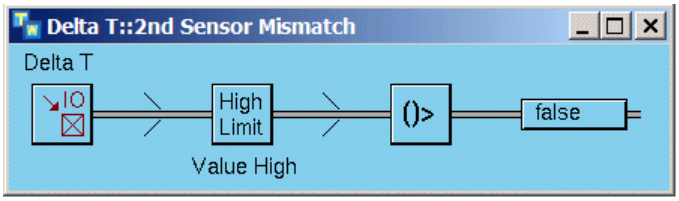


5 Initialize the process map.

For details, see Initializing Process Maps.

6 Choose Show Logic on the 2nd Sensor Delta derived sensor to display the specific 2nd Sensor Mismatch event detection diagram.

Here is the specific 2nd Sensor Mismatch event detection diagram for the Delta T 2nd Sensor Delta derived sensor:



This event detection diagram monitors the PV of the Delta T derived sensor, which is the difference between the two temperatures specified in Tag1 and Tag2. The event occurs when the absolute value of the delta exceeds the specified high limit.

7 Display the properties dialog for the High Limit block in the specific event detection diagram and configure the High Limit and High Limit Deadband, as needed.

For information on configuring these values, see PV High.

The 2nd Sensor Delta derived sensor is now configured to detect the 2nd Sensor Mismatch event for the specified temperature sensors.

Built-in Event Detection for Heaters

Heaters provide a generic methodology for monitoring and diagnosing common problems with heaters. The heater performs event detection by analyzing current and historical process values from process sensors and derived sensors that are related to the heater.

A heater can detect these events:

- Related Sensor Events
- Tube Skin Delta T
- Efficiency Severe Change
- Low Efficiency
- Heat Release Projected High

All events generate operator messages when the event occurs. Some events also generate a SymCure fault model event, which triggers diagnostic reasoning to determine root causes.

For information about the generated SymCure events, see Built-in Generic Fault Models for Heaters and Built-in Generic Fault Models for Process Equipment.

In the description of configuring each of the following events, you must first create and configure a heater from the Heaters palette of the Process Modeling toolbox, then you must initialize the process map. For details, see:

- Connecting Process Equipment.
- Initializing Process Maps.

Related Sensor Events

All process equipment can detect built-in sensor events for these related sensors: Process Flow, Process Inlet Temperature, Process Outlet Temperature, Process Inlet Pressure, and Process Outlet Pressure.

In addition, a heater can detect built-in sensor events for these related sensors:

- Draft Oxygen
- Draft Pressure
- Stack NOx
- Tube Skin Temperatures

For general information on configuring the common related sensor events, see Configuring Related Sensors.

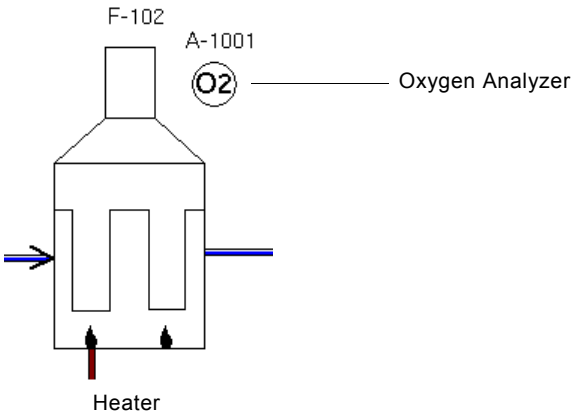
Draft Oxygen

To configure the draft oxygen related sensor event:

1 Create and configure an oxygen analyzer from the Instruments palette of the Process Modeling toolbox and place it on your process map near a heater.

For details, see Creating Instruments and Configuring Internal Datapoints.

For example, here is the F-102 heater with the A-1001 oxygen analyzer:



2 Display the properties dialog for a heater and click the Configuration tab.

3 Select Draft Oxygen from the Relations list and click the Configure button.

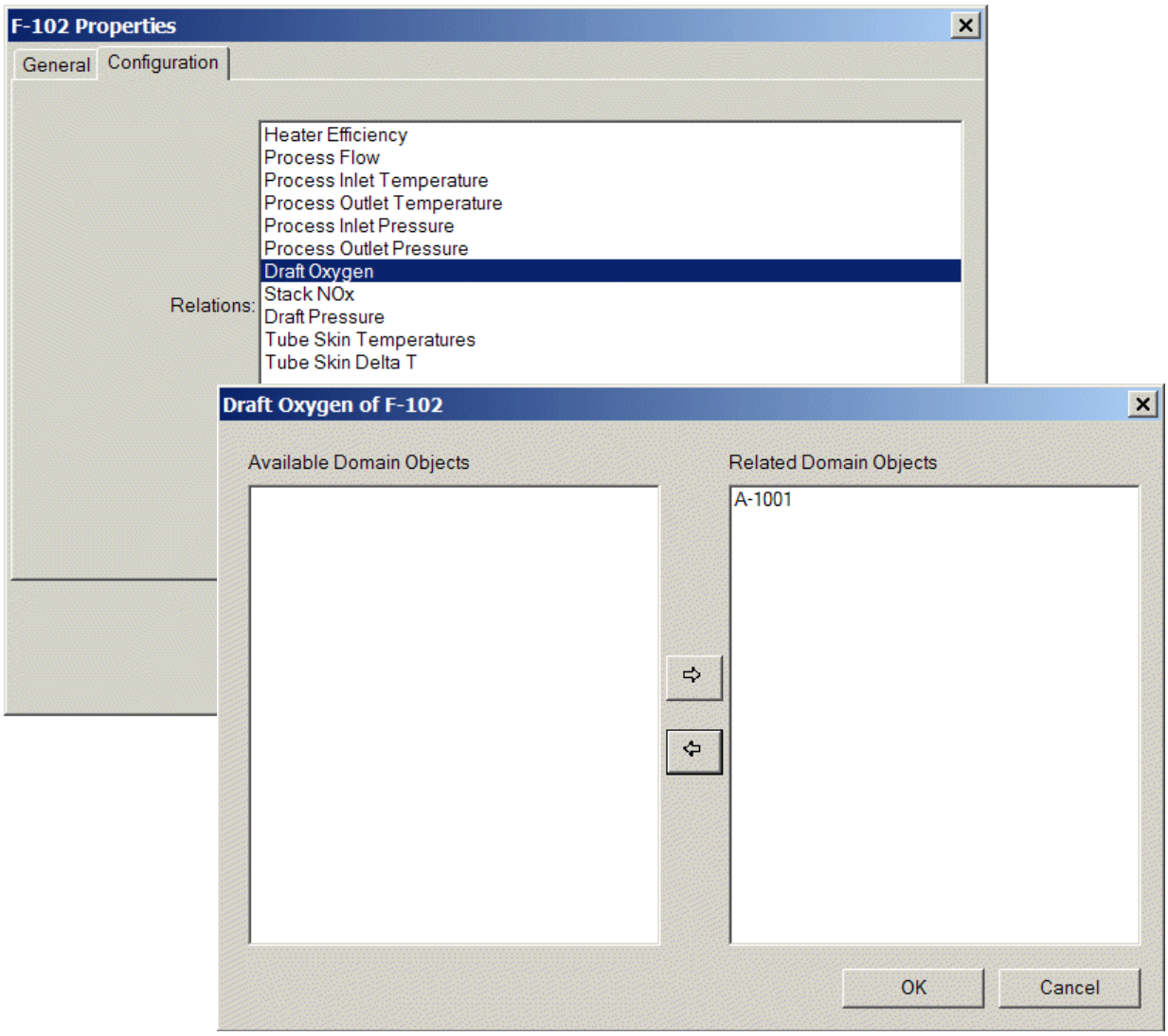
4 Move the oxygen analyzer to the Related Domain Objects list.

The oxygen analyzer is now a related sensor of the heater.

5 Configure the sensor events of the oxygen analyzer, as needed.

For details, see Built-in Event Detection for Instruments.

For example, here is how you would configure the Draft Oxygen of the F-102 heater to be the A-1001 oxygen analyzer:



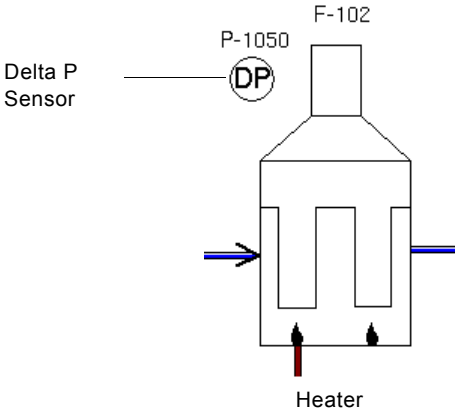
Draft Pressure

To configure the draft pressure related sensor event:

1 Create a delta P sensor from the Instruments palette of the Process Modeling toolbox and place it on your process map near a heater.

For details, see Creating Instruments and Configuring Internal Datapoints.

For example, here is the F-102 heater with the P-1050 delta P sensor:



2 Display the properties dialog for a heater and click the Configuration tab.

3 Select Draft Pressure from the Relations list and click the Configure button.

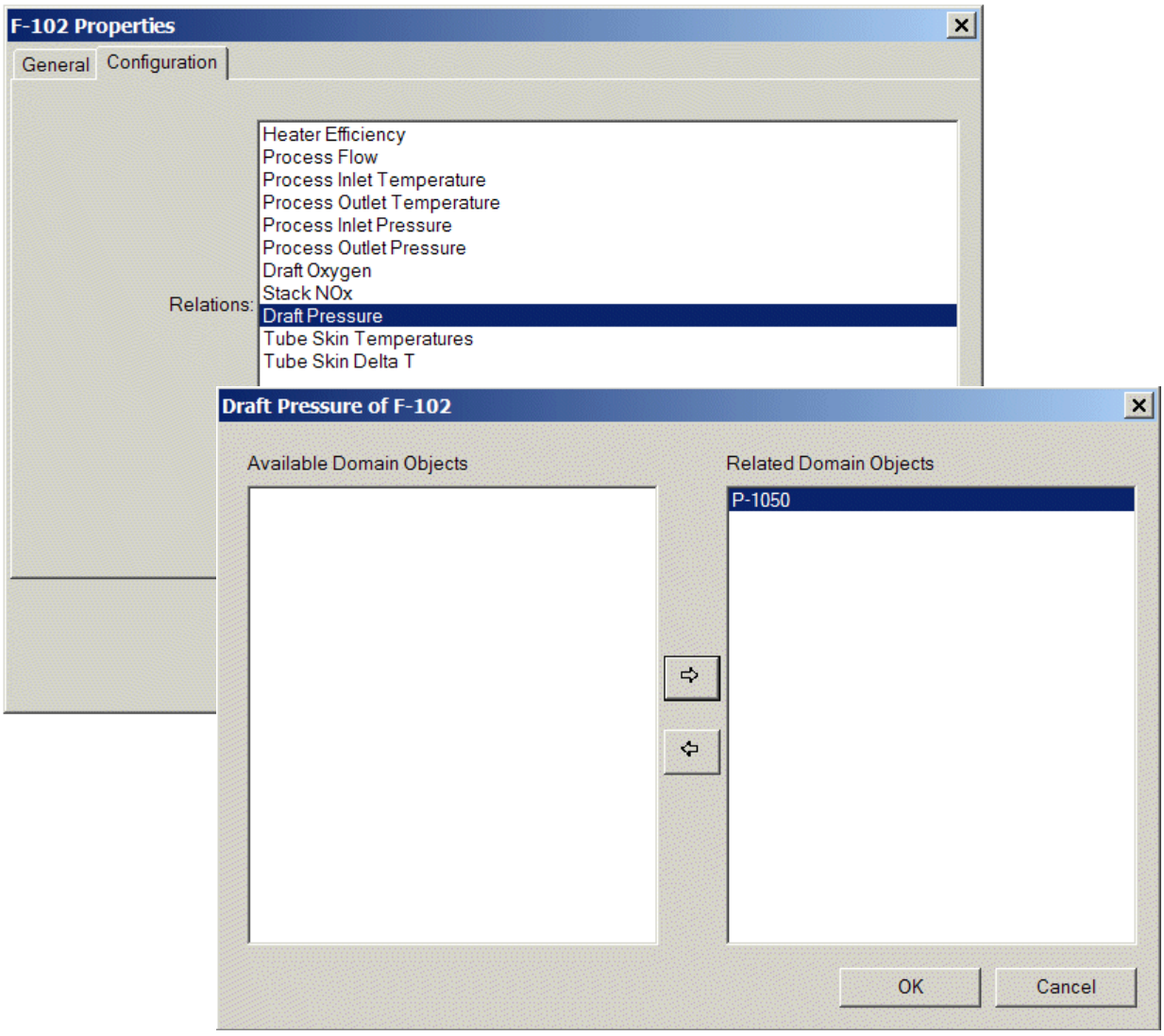
4 Move the delta P sensor to the Related Domain Objects list.

The delta P sensor is now a related sensor of the heater.

5 Configure the sensor events of the delta P sensor, as needed.

For details, see Built-in Event Detection for Instruments.

For example, here is how you would configure the Draft Pressure of the F-102 heater to be the P-1050 delta P sensor:



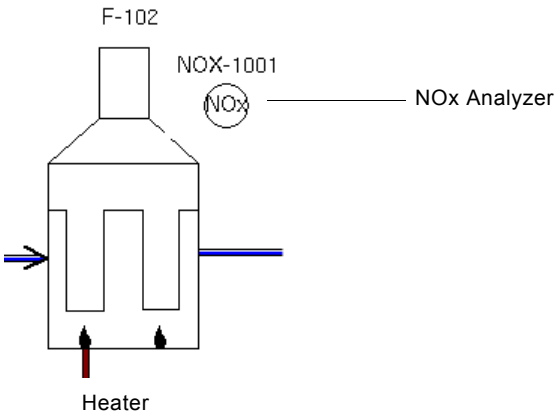
Stack NOx

To configure the stack NOx related sensor event:

1 Create a NOx analyzer from the Instruments palette of the Process Modeling toolbox and place it on your process map near a heater.

For details, see Creating Instruments and Configuring Internal Datapoints.

For example, here is the F-102 heater with the NOX-1001 NOx sensor:



2 Display the properties dialog for a heater and click the Configuration tab.

3 Select Stack NOx from the Relations list and click the Configure button.

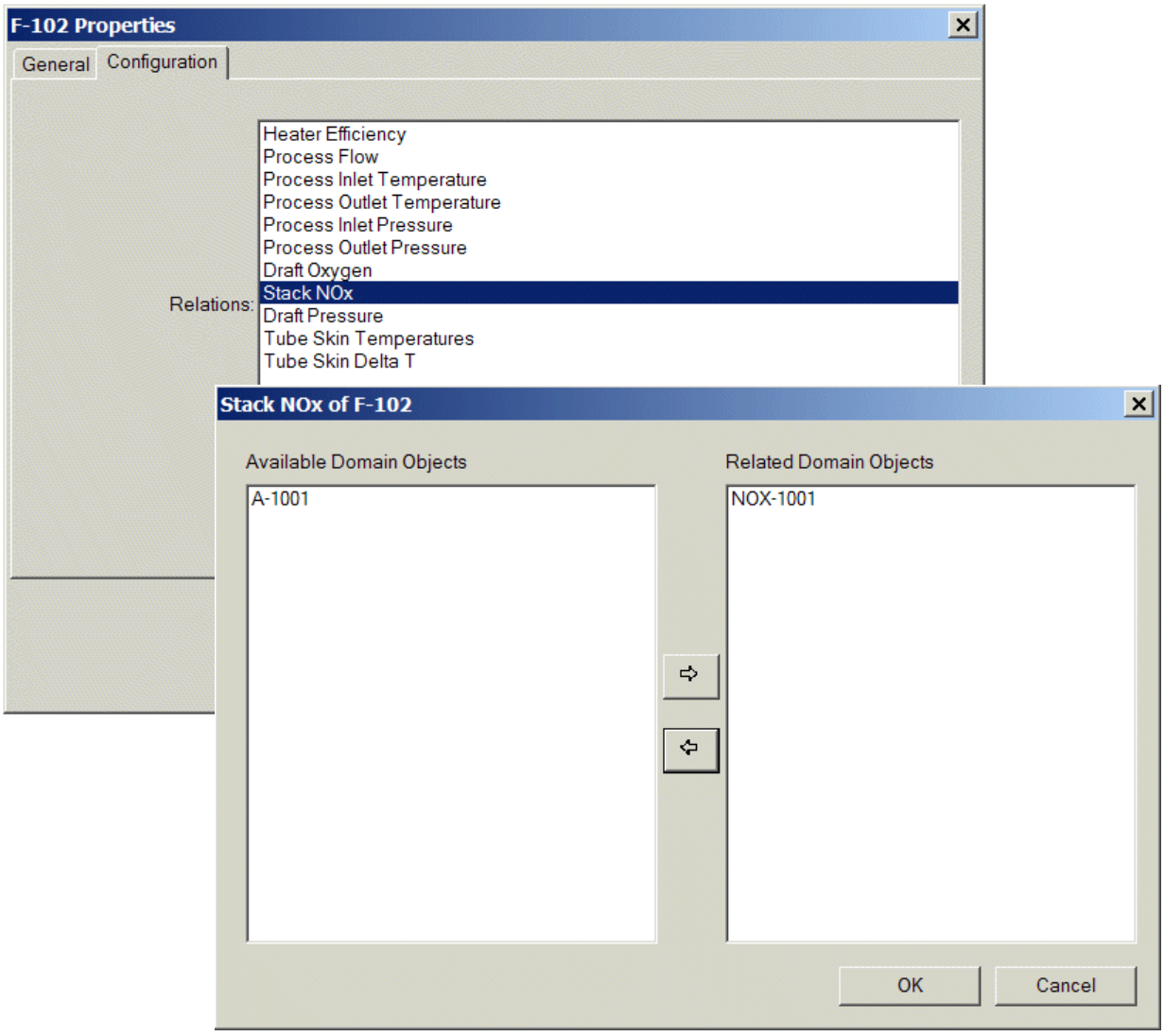
4 Move the NOx analyzer to the Related Domain Objects list.

The NOx analyzer is now a related sensor of the heater.

5 Configure the sensor events of the NOx analyzer, as needed.

For details, see Built-in Event Detection for Instruments.

For example, here is how you would configure the Stack NOx of the F-102 heater to be the NOX-1001 NOx analyzer:



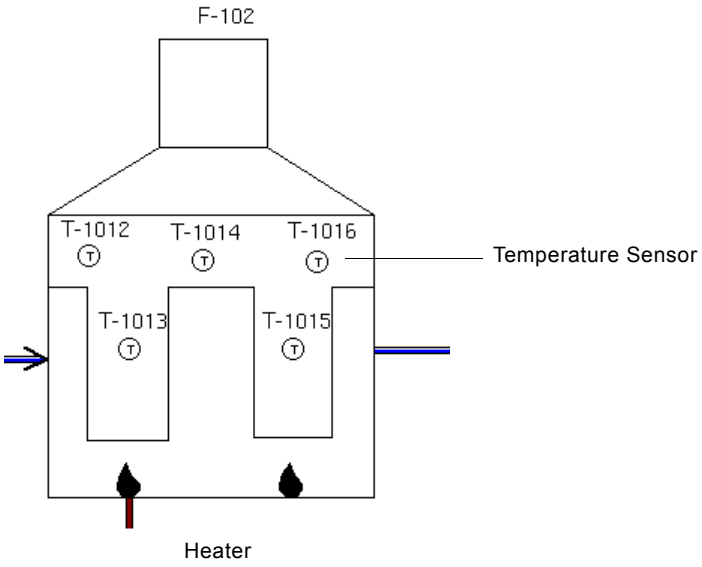
Tube Skin Temperatures

To configure the tube skin temperatures related sensor event:

1 Create two or more temperature sensors from the Instruments palette of the Process Modeling toolbox and place them on your process map near a heater.

For details, see Creating Instruments and Configuring Internal Datapoints.

For example, here is the F-102 heater with five tube skin temperature sensors:



Tip You can resize the temperature sensors and the heater by choosing Edit > Size, as needed.

2 Display the properties dialog for a heater and click the Configuration tab.

3 Select Tube Skin Temperature Sensors from the Relations list and click the Configure button.

4 Move the tube skin temperature sensors to the Related Domain Objects list.

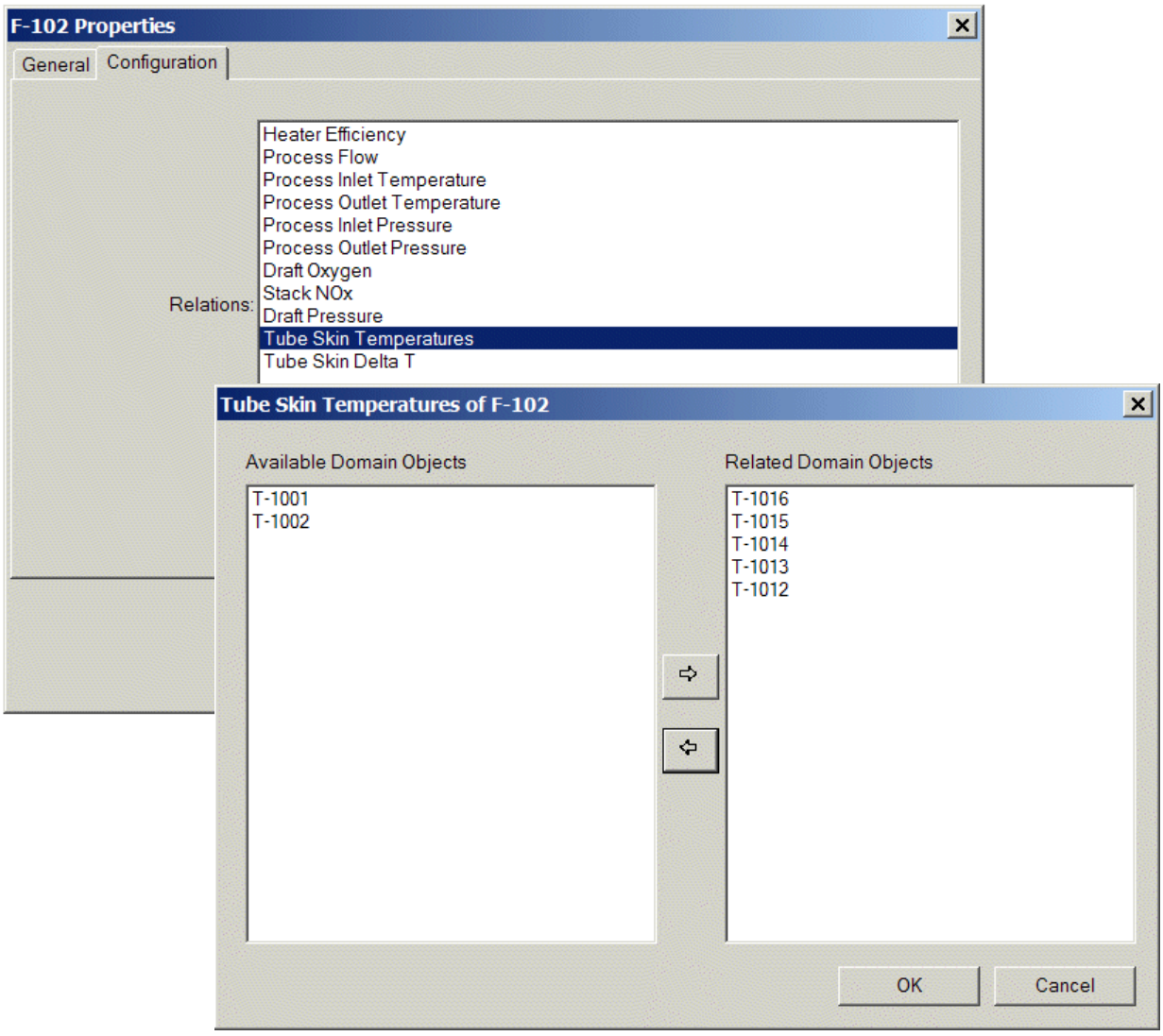
The temperature sensors are now a related sensor of the heater.

5 Configure the individual sensor events of each temperature sensors, as needed.

For details, see Built-in Event Detection for Instruments.

To monitor the difference between two tube skin temperatures, you can also configure the Tube Skin Delta T event, using a derived sensor. For details, see Tube Skin Delta T.

For example, here is how you would configure the Tube Skin Temperatures of the F-102 heater:



Tube Skin Delta T

The Tube Skin Delta T event detects the difference between two related tube skin temperature sensors of a heater. To detect the Tube Skin Delta T event, you create a Tube Skin Delta T sensor, which is a derived sensor, and relate it to a heater. You configure the PV of the derived sensor to be the difference between two related tube skin temperature sensors of the heater. You then configure the High Limit of the Tube Skin Delta T event detection diagram for a particular heater.

The event generates an operator message when the event is true and generates a SymCure High event on the derived sensor.

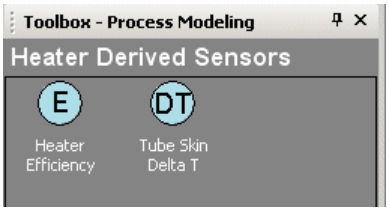
To configure the Tube Skin Delta T event:

1. Create and configure two or more tube skin temperature sensors of a heater.

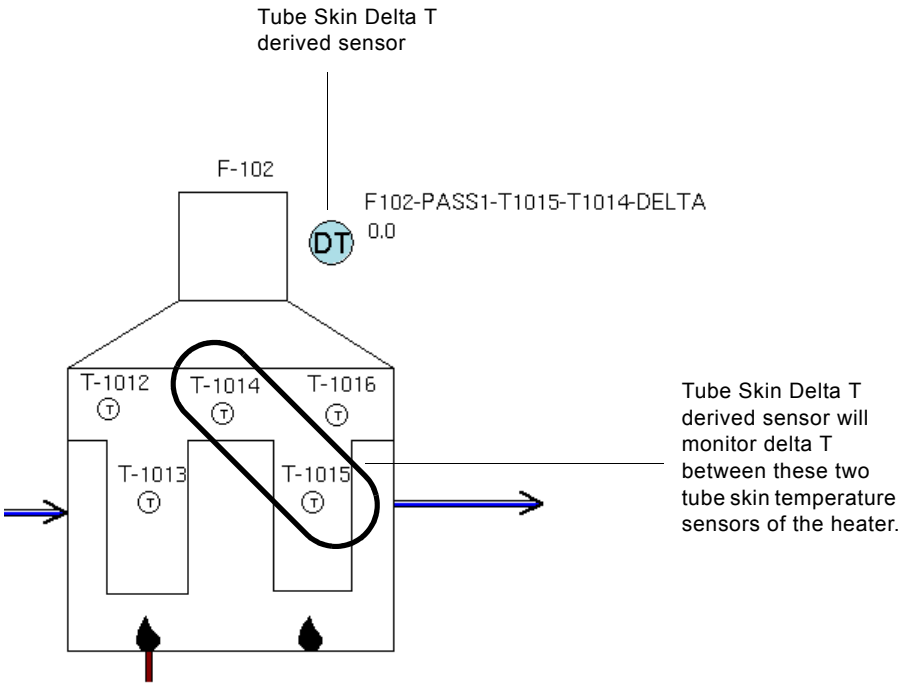
   For details, see Tube Skin Temperatures.

2. Display the Heater Derived Sensors palette of the Process Modeling toolbox, create a Tube Skin Delta T derived sensor, and place it on your process map near a heater.

Here is the Heater Derived Sensors palette of the Process Modeling toolbox:

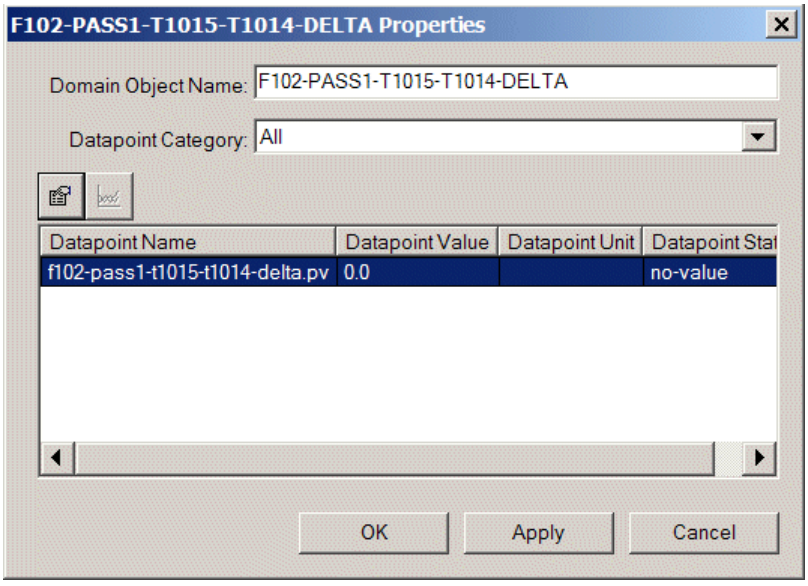


For example, here is the F-102 heater with a Tube Skin Delta T derived sensor named F102-PASS1-T1015-T1014-DELTA, which will monitor the difference between the T-1014 and T-1015 tube skin temperature sensors of the heater:



3 Display the properties dialog of the Tube Skin Delta T derived sensor and click the PV internal datapoint.

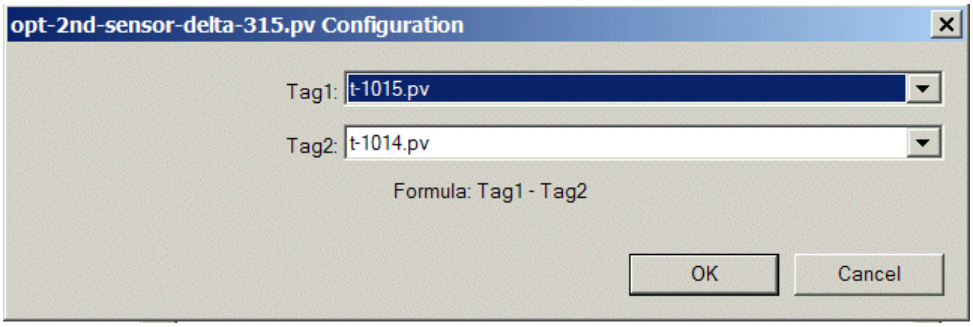
Here is the properties dialog for the F102-PASS1-T1015-T1014-DELTA Tube Skin Delta T derived sensor with the PV internal datapoint selected:



4 Click the Properties button and configure Tag1 and Tag2 to be the tube skin temperature sensors for which to compute the difference.

The derived sensor computes the delta by using the formula Tag1 minus Tag2, so the higher temperature sensor should be Tag1.

For example, here is how you would configure the Tube Skin Delta T derived sensor to calculate the temperature difference between the T-1015 and T-1014 tube skin temperature sensors:

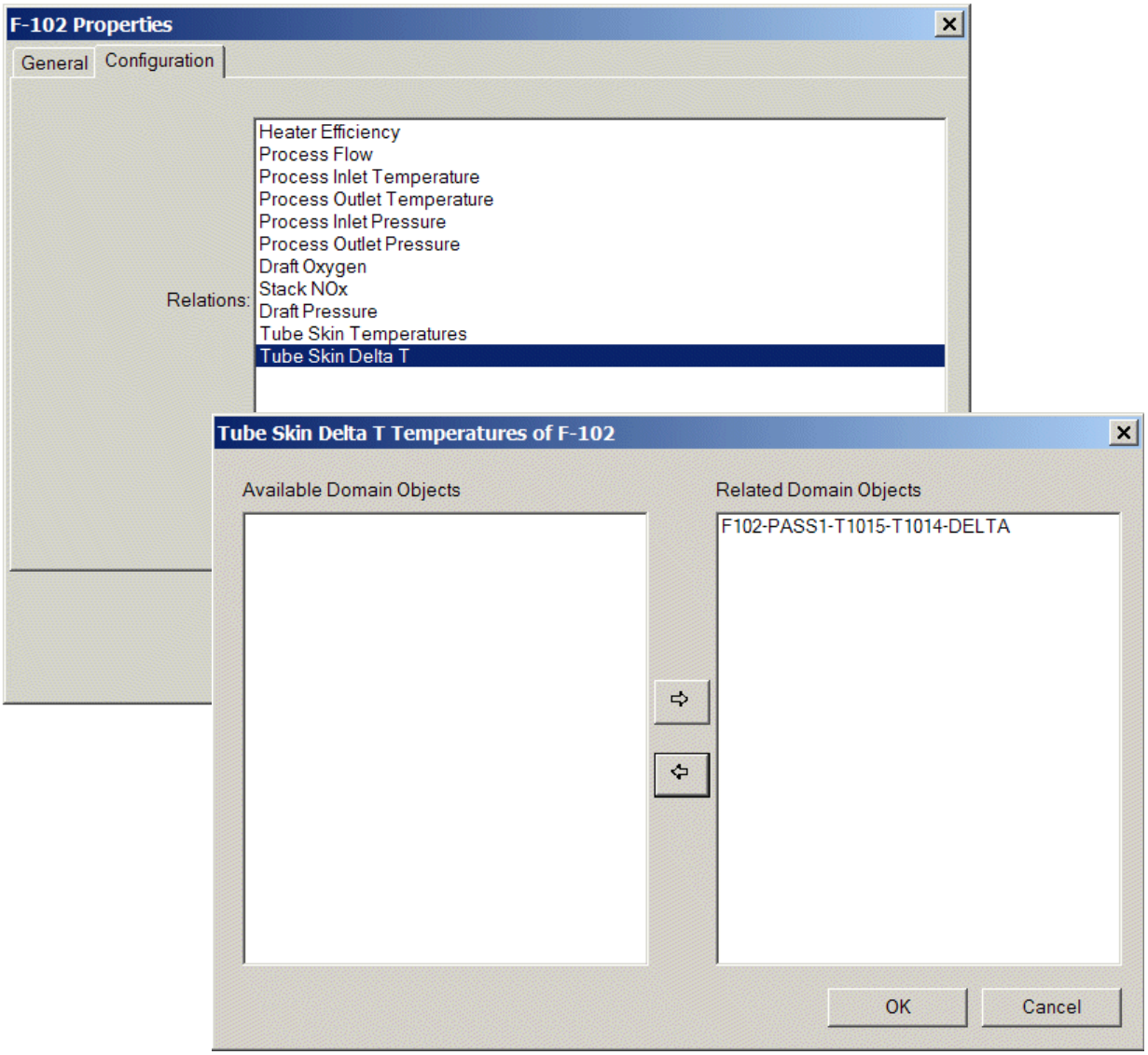


5 Display the properties dialog for the heater whose Tube Skin Delta T event you want to detect and click the Configuration tab.

6 Move the Tube Skin Delta T derived sensor to the Related Domain Objects list.

The Tube Skin Delta T sensor is now a related sensor of a heater.

For example, here is how you would configure the Tube Skin Delta T of the F-102 heater to be the F102-PASS1-T1015-T1014-DELTA Tube Skin Delta T derived sensor:

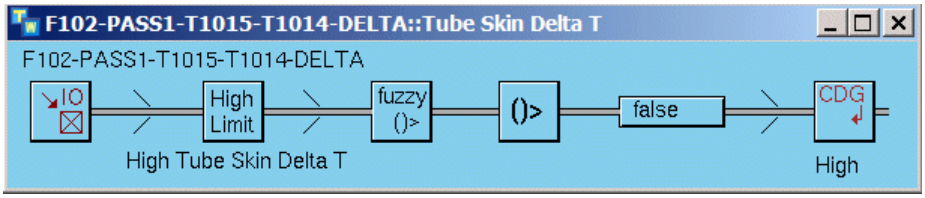


7 Initialize the process map.

For details, see Initializing Process Maps.

8 Choose Show Logic on the Tube Skin Delta T derived sensor to display the specific Tube Skin Delta T event detection diagram.

Here is the specific Tube Skin Delta T event detection diagram for the F102-PASS1-T1015-T1014-DELTA Tube Skin Delta T derived sensor:



This event detection diagram monitors the PV of the Tube Skin Delta T derived sensor, which is the difference between the two tube skin temperatures specified in Tag1 and Tag2. If the difference exceeds the specified high limit, the diagram creates a fuzzy truth-value, filters out unchanged truth values, and, when true, generates a SymCure High event.

9 Display the properties dialog for the High Limit block in the specific event detection diagram and configure the High Limit and High Limit Deadband, as needed.

For information on configuring these values, see PV High.

The heater is now configured to detect the Tube Skin Delta T event for the specified tube skin temperatures.

Efficiency Severe Change

The Efficiency Severe Change event detects a severe change in the calculated efficiency value of the Heater Efficiency derived sensor of a heater. The efficiency calculation of a heater requires these sensors:

- Process Flow related sensor.
- Process Inlet Temperature related sensor.
- Process Outlet Temperature related sensor.
- Heater Efficiency related sensor, which is a derived sensor for the heater.
- Flow Sensor, which is connected to an input fuel line.
- Heating Value Analyzer, which is connected to an input fuel line.

In addition, you must configure internal datapoints of the Heater Efficiency derived sensor to provide these values:

- Averaging time
- Process fluid heat capacity

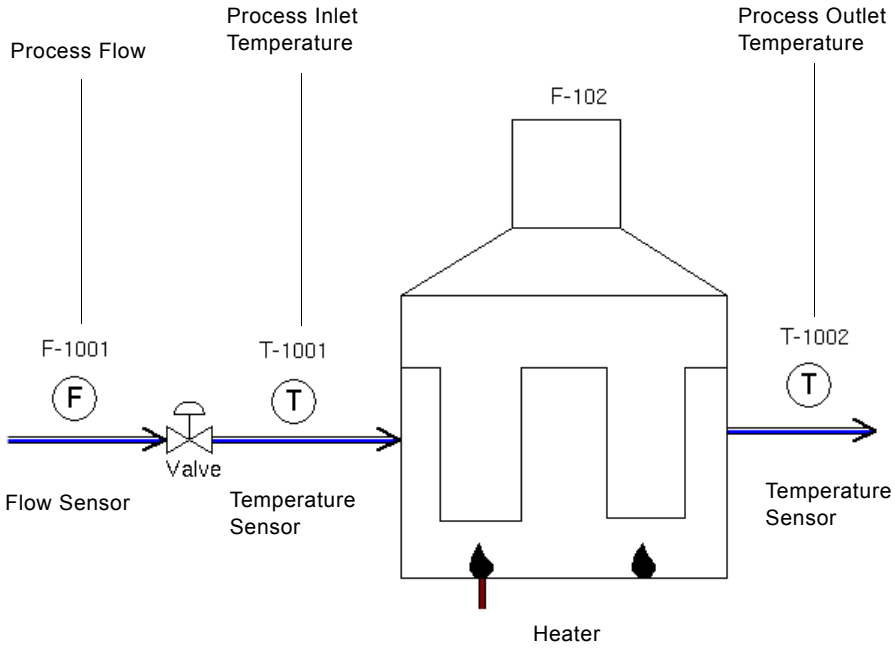
The Heater Efficiency derived sensor calculates the process heat gain, fuel heat release, efficiency, and average efficiency. The Efficiency Severe Change event for the heater detects changes in the efficiency calculation that exceed the specified limit within the response time. The event generates an operator message when the event is true.

To configure the Efficiency Severe Change event:

1 Create and configure the Process Flow, Process Inlet Temperature, and Process Outlet Temperature related sensors of a heater.

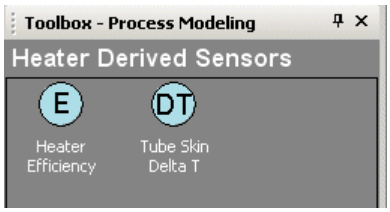
For details, see Configuring Domain Objects.

For example, here is the F-102 heater with the required related sensors for process flow, inlet temperature, and outlet temperature:

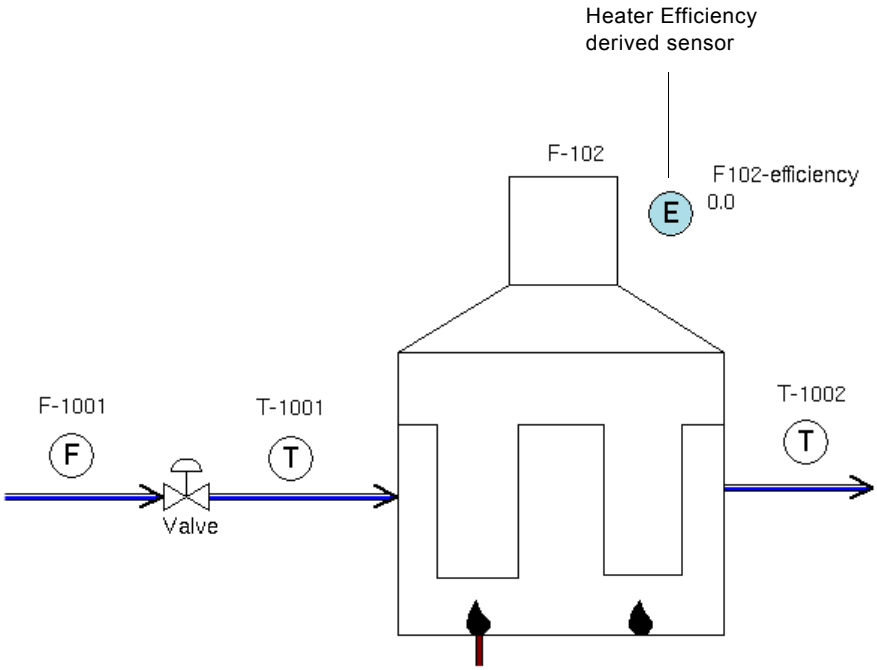


2 Display the Heater Derived Sensors palette of the Process Modeling toolbox, create a Heater Efficiency derived sensor, and place it on your process map near a heater.

Here is the Heater Derived Sensors palette of the Process Modeling toolbox:

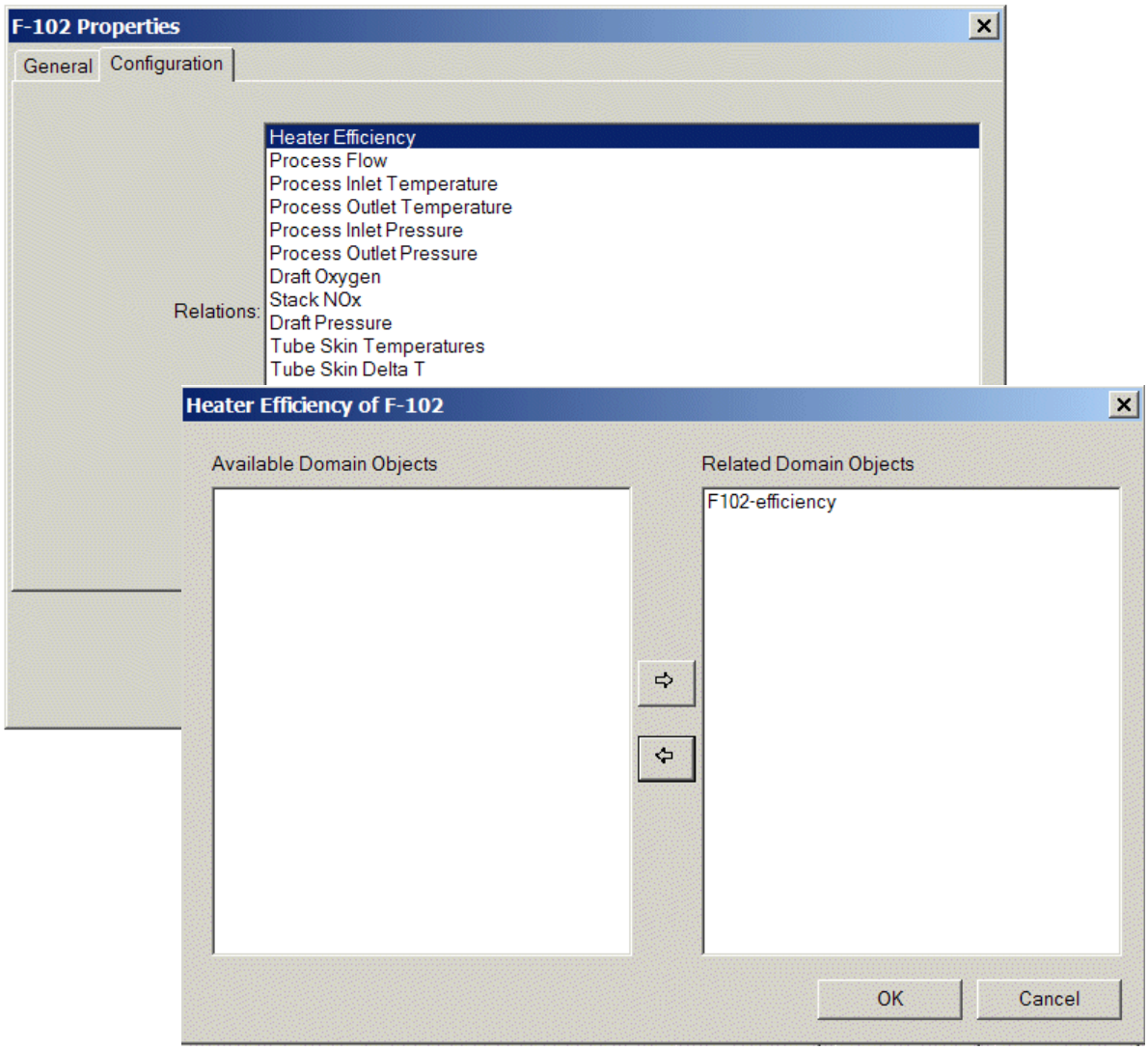


For example, here is the F-102 heater with a Heater Efficiency derived sensor named F102-Efficiency, which will calculate the heater efficiency:



3 Display the properties dialog of the heater, click the Configuration tab, and configure the Heater Efficiency related sensor.

Here is how you would configure the Heater Efficiency of the F-102 heater to be the F102-efficiency derived sensor:



The Heater Efficiency derived sensor defines internal datapoints for the process flow, inlet temperature, and outlet temperature, which are obtained from the related sensors of the heater. To automatically configure these internal datapoints, you must initialize the process map.

4 Initialize the process map.

For details, see Initializing Process Maps.

Now you must configure additional internal datapoints of the Heater Efficiency derived sensor.

5 Configure the internal datapoints of the Heater Efficiency derived sensor.

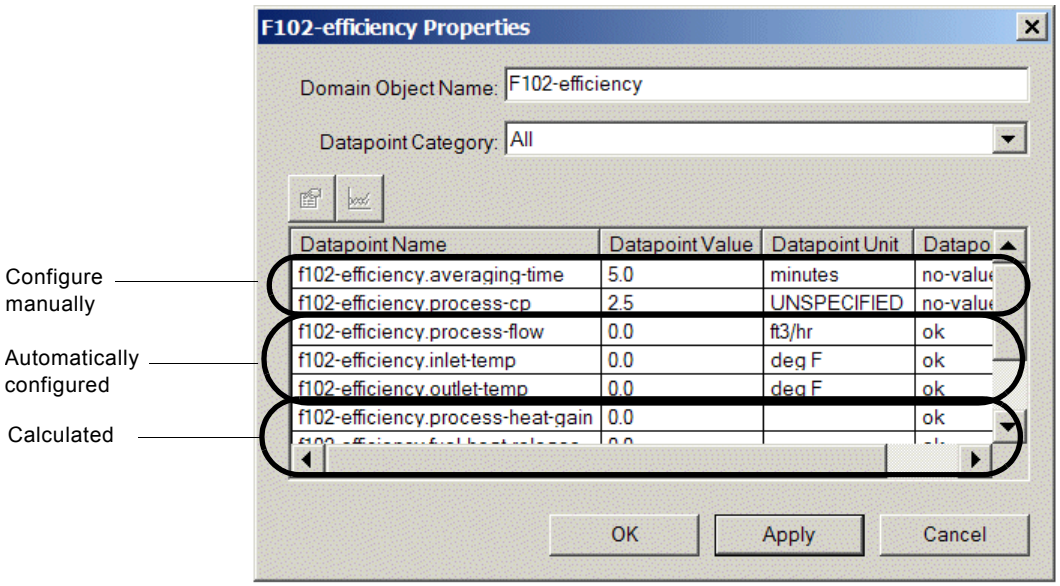
a Display the properties dialog for the Heater Efficiency derived sensor.

Once you have configured the related sensors of the heater and initialized the process map, the process-flow, inlet-temp, outlet-temp internal datapoints of the derived sensor are automatically configured to use the process values of the related sensors.

In addition, you must manually configure the averaging-time and process-cp internal datapoints. The derived sensor uses these internal datapoint values to calculate the heater efficiency.

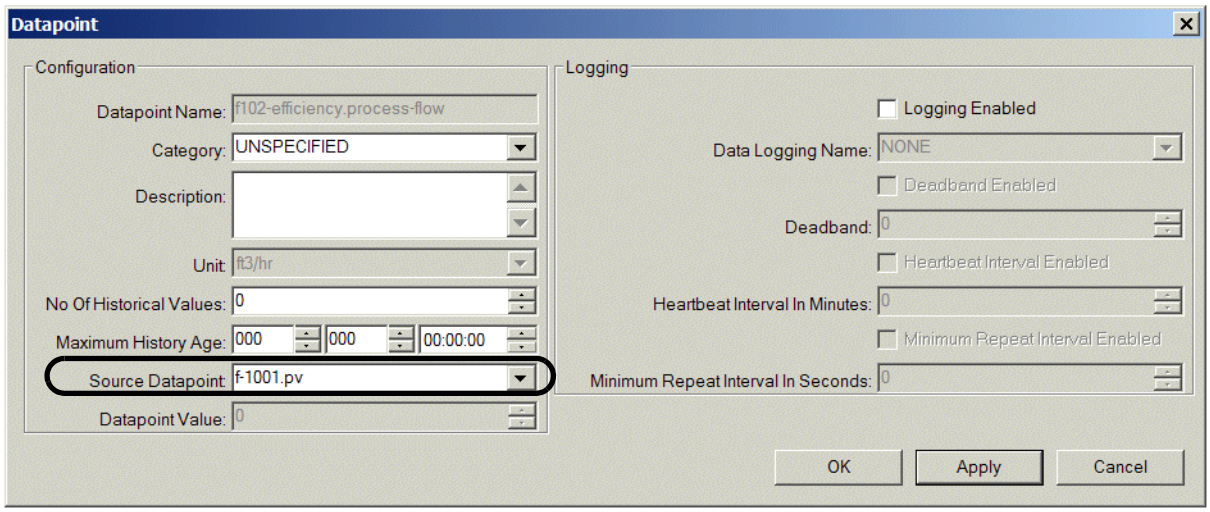
The remaining internal datapoints—process-heat-gain, fuel-heat-release, efficiency, and avg-efficiency—are calculated values of the derived sensor. You must scroll down to see all the calculated values.

Here is the dialog for the F102-Efficiency derived sensor:



b Select the process-flow internal datapoint and click the Properties button.

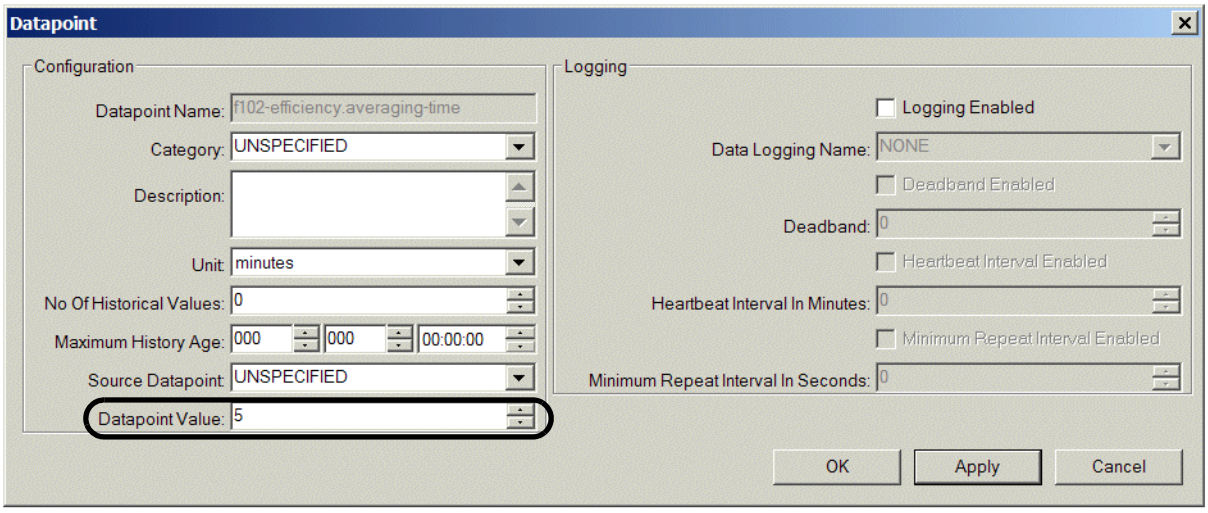
Notice that the Source Datapoint is automatically configured to be f-1001.pv, which is the PV of the Process Flow related sensor of the F-102 heater:



c Verify the Source Datapoint for the inlet-temperature and outlet-temperature of the Heater Efficiency.

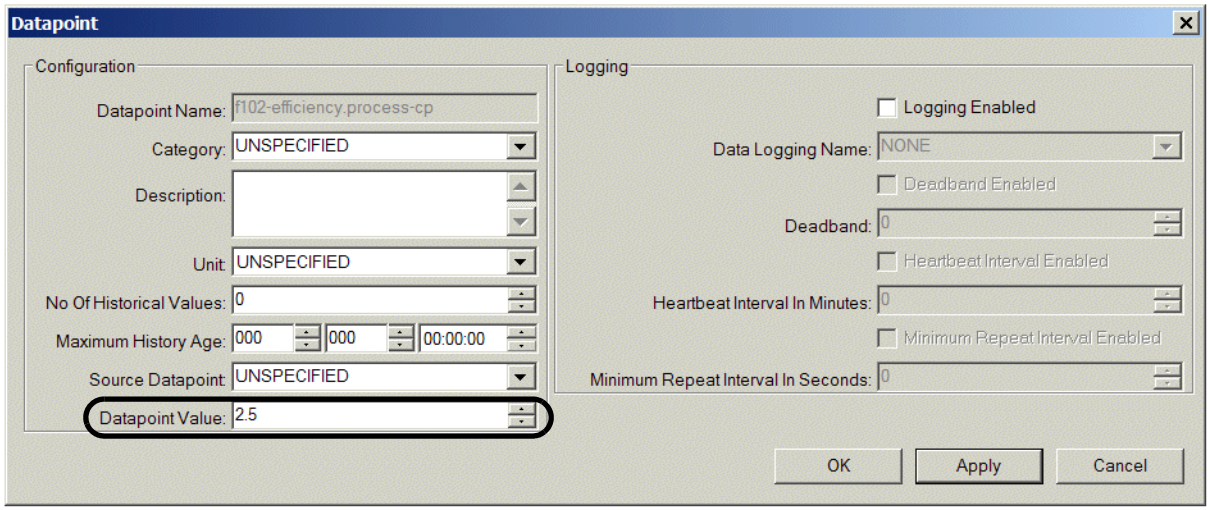
d Display the properties of the averaging-time internal datapoint and configure the Datapoint Value to be the number of minutes over which to calculate the average efficiency.

Here is the properties dialog for the f102-efficiency.averaging-time internal datapoint, which is configured to be 5 minutes:



e Display the properties of the process-cp internal datapoint and configure the Datapoint Value to be the heat capacity of the heater process fluid.

Here is the properties dialog for the f102-efficiency.process-cp internal datapoint, which is configured to be 2.5:



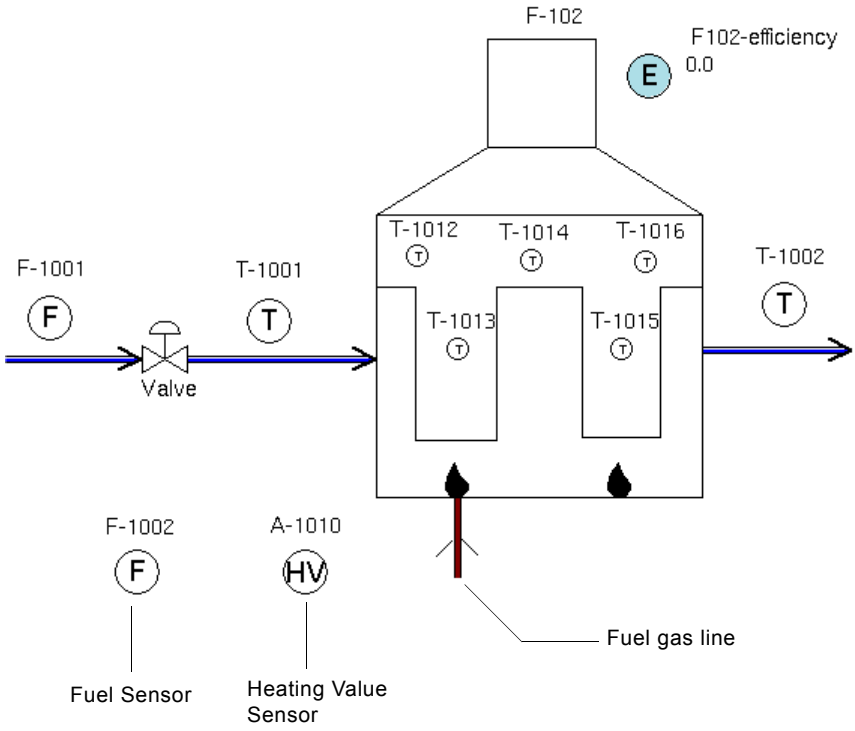
In addition to configuring the related sensors of the heater and the Heater Efficiency derived sensor, you must also create and connect a Flow Sensor and Heating Value Analyzer to the fuel gas line of the heater. The Heater Efficiency derived sensor uses these values to calculate the fuel-heat-release, which is used in the efficiency calculation.

You can also detect a projected high value for the fuel heat release. For details, see Heat Release Projected High.

Unlike the process flow, inlet temperature, and outlet temperature, which are related sensors of the heater, these sensors must be physically connected to the fuel gas line of the heater.

6 Create a Flow Sensor and Heating Value Analyzer from the Instruments palette of the Process Modeling toolbox and place them near the fuel gas line of the heater.

Here is the F-102 heater with the two sensors:

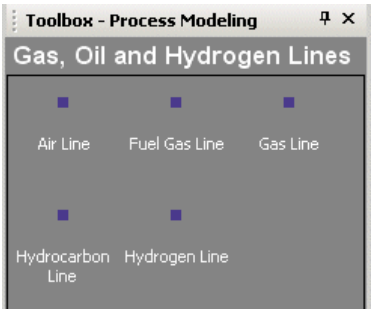


Now you must connect these sensors to the fuel gas line of the heater. The easiest way to do this is by creating two fuel gas connections and connecting the sensors through these connections.

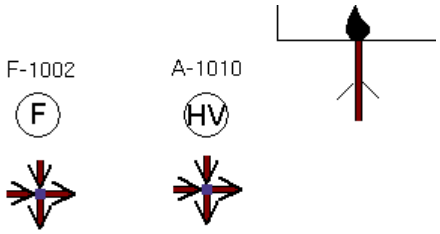
You can also use the directed or undirected connections on the Connections palette; however, note that the connections must be connected through a gas fuel line connection stub shown in the following figures.

7 Create and connect the Flow Sensor and Heating Value Analyzer to the fuel gas line of the heater, as follows:

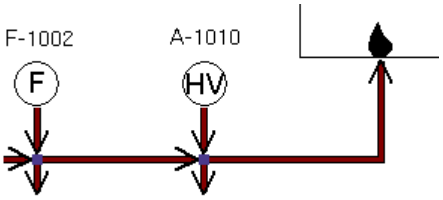
a Display the Gas, Oil, and Hydrogen Lines connection palette of the Process Modeling toolbox:



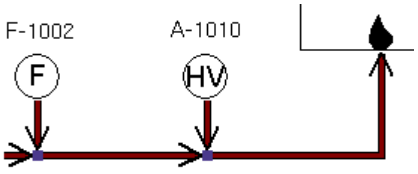
b Create two Fuel Gas Line connections and place them below the two sensors:



c Drag the top connection stubs into each sensor, connect the two connection stubs together, and connect the fuel gas line of the heater to the right-most connection stub:



d Drag the unused connection stubs into the connection stub tools:



The sensors are now connected to the input fuel gas line of the heater. The Heater Efficiency derived sensor can now use these values in its calculation.

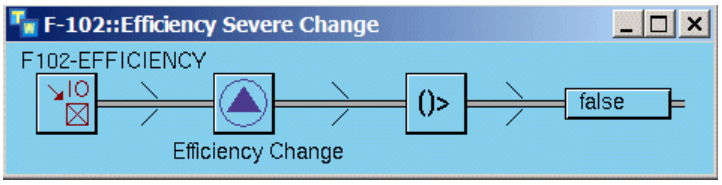
If there is more than one fuel source for the heater, you can connect separate fuel sources to the heater, each with their own flow and heating value sensors. The Heater Efficiency derived sensor will take into account all fuel sources connected to the heater in the efficiency calculation.

The last step in configuring the Efficiency Severe Change event of a heater is to configure the change limit for the efficiency calculation, which you do in the specific event detection diagram for a particular heater.

8 Choose Show Logic on the heater and select the Efficiency Severe Change event for the heater.

The process map must be initialized to configure the specific event detection diagram.

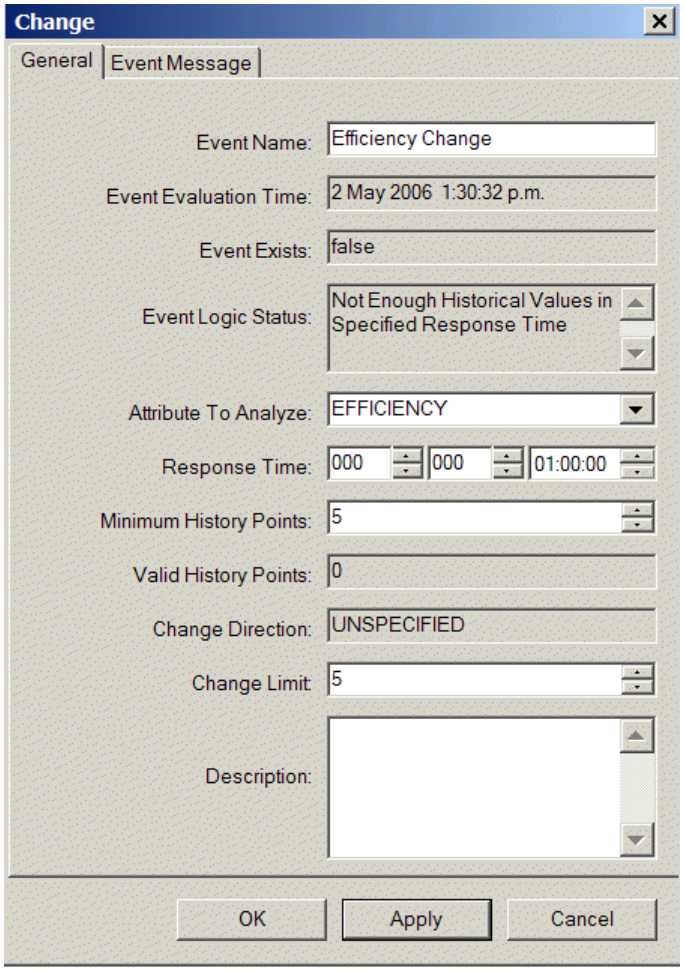
For example, here is the Efficiency Severe Change event detection diagram for the F-102 heater:



9 Display the properties dialog for the Change block and configure the Change Limit to be the limit for the efficiency value change.

You can also configure the Response Time, and the Minimum History Points, as needed.

Here is the properties dialog for the Change block that causes the event to trigger when the calculated efficiency of the heater changes by more than 5. Notice that the Attribute to Analyze is efficiency, which is calculated in the Heater Efficiency derived sensor of the heater.



The heater is now fully configured to detect the Efficiency Severe Change event.

Low Efficiency

The Low Efficiency event detects low average heater efficiency, a calculated value of the Heater Efficiency derived sensor of a heater. If the draft oxygen of the heater is, at the same time, not high, the Low Efficiency event generates a SymCure Excess Coking event.

The Low Efficiency event requires all the same sensors as the Efficiency Severe Change event, as well as the Draft Oxygen related sensor.

To configure the Low Efficiency event:

1 Create and configure a heater with the same related and connected sensors that the Efficiency Severe Change event requires.

For details, see Efficiency Severe Change.

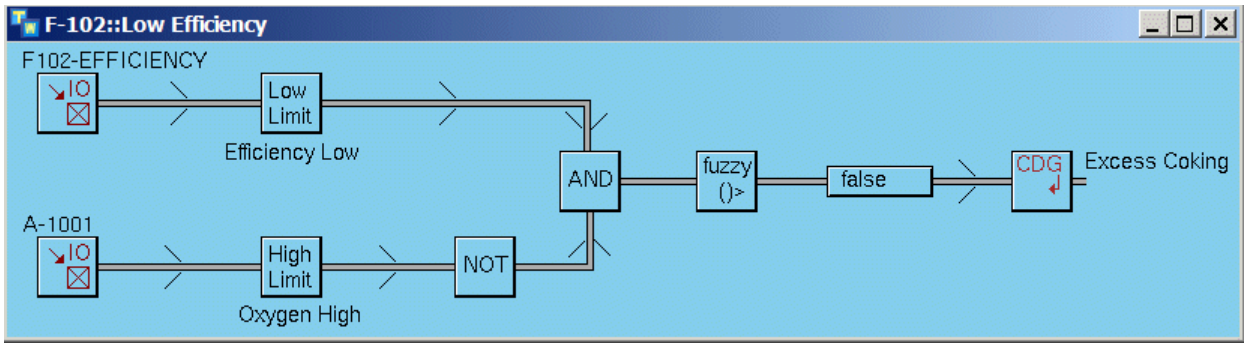
2 Create and configure the Draft Oxygen related sensor of the heater.

For details, see Draft Oxygen.

3 Choose Show Logic on the heater and select the Low Efficiency event for the heater.

The process map must be initialized to configure the specific event detection diagram.

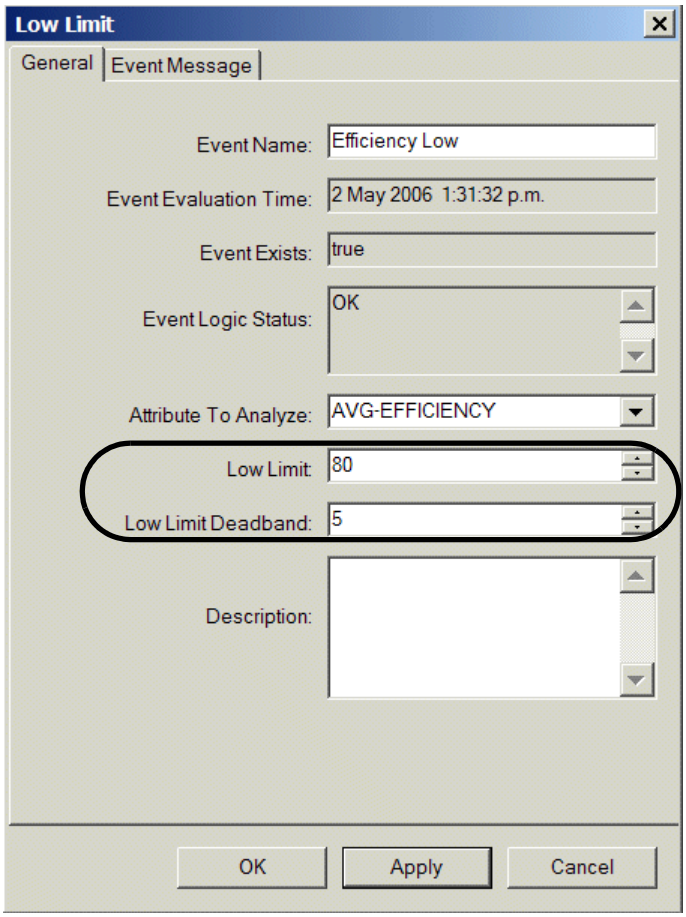
Here is the Low Efficiency event detection diagram for the F-102 heater. The diagram detects low heater efficiency and high draft oxygen. If the efficiency is low, a message is generated. If the draft oxygen is also not high, the event converts the combined values to a fuzzy truth value and generates a SymCure Excess Coking event.



4 Show the properties dialog for the Low Limit block and configure the Low Limit and Low Limit Deadband for low average heater efficiency.

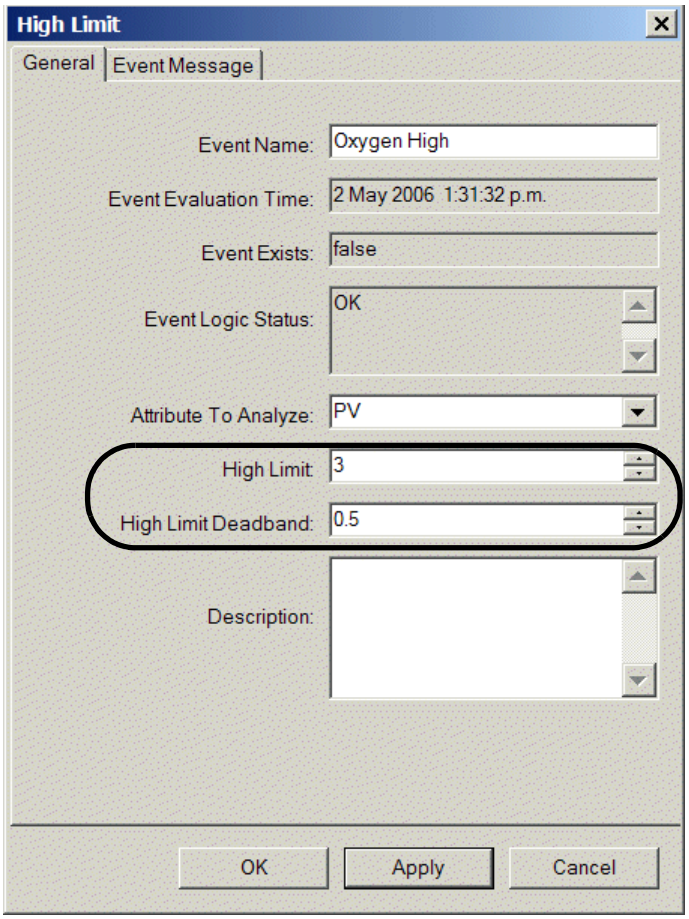
For information on how to configure the deadband, see PV Low.

By default, the Low Limit block is configured to detect low average heater efficiency between 80.0 and 85.0. Notice that the Attribute to Analyze is avg-efficiency, which is calculated in the Heater Efficiency related sensor of the heater.



5 Show the properties dialog for the High Limit block and configure the High Limit and High Limit Deadband for high draft oxygen.

By default, the High Limit block is configured to detect high draft oxygen between 2.5 and 3.0:



The heater is now configured to detect the Low Efficiency event.

Heat Release Projected High

The Heat Release Projected High event detects a projected high value for the fuel heat release calculated value of the Heater Efficiency derived sensor of a heater. The event generates an operator message when the event is true.

The Heat Release Projected High event requires all the same sensors as the Efficiency Severe Change event.

To configure the Heat Release Projected High event:

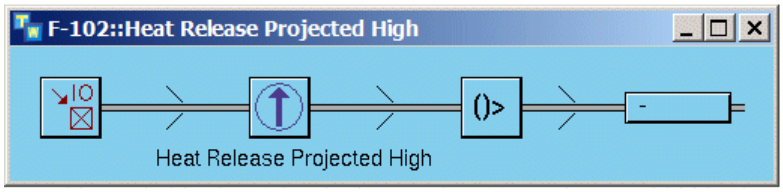
1 Create and configure a heater with the same related and connected sensors that the Efficiency Severe Change event requires.

For details, see Efficiency Severe Change.

2 Choose Show Logic on the heater and select the Heat Release Projected High event for the heater.

The process map must be initialized to configure the specific event detection diagram.

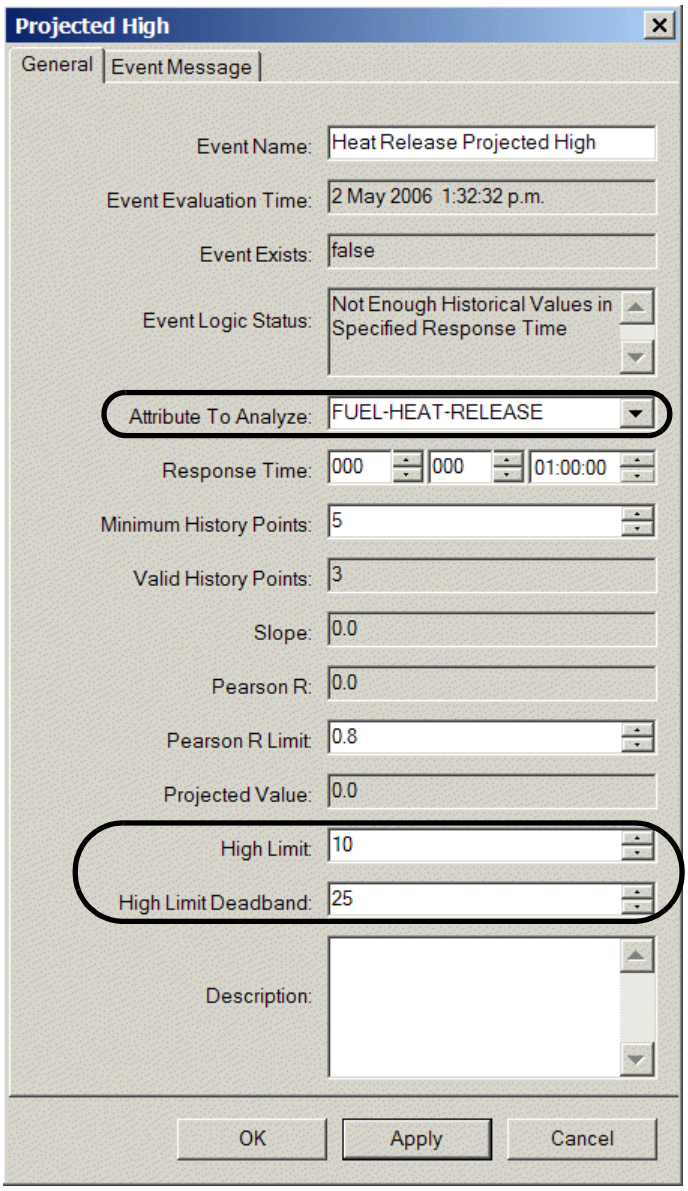
Here is the Heat Release Projected High event detection diagram for the F-102 heater. The diagram detects a projected high value for the fuel-heat-release internal datapoint of the Heater Efficiency related sensor of the heater.



3 Show the properties dialog for the Projected High block and configure the High Limit and High Limit Deadband for a projected high value for the fuel heat release.

You can also configure the Response Time, Minimum History Points, and Pearson R Limit, as needed. For information on how to configure these values, see PV Projected High.

Here is the Projected High block configured to detect high fuel heat release between 9.75 and 10.0. Notice that the Attribute to Analyze is fuel-heat-release, which is calculated in the Heater Efficiency related sensor of the heater.



The heater is now configured to detect the Heat Release Projected High event.

Built-in Event Detection for Compressors

Compressors provide a generic methodology for monitoring and diagnosing common problems with compressors. The compressor performs these functions by analyzing current and historical process values from process sensors and derived sensors that are related to the compressor.

Compressors can detect these events:

- Compression Ratio Decrease
- Power Projected High
- Polytropic Head Change

All events generate operator messages when the event occurs. The events also generate a SymCure event when the event is true, which triggers diagnostic reasoning to determine root causes.

For information about the generated SymCure events, see Built-in Generic Fault Models for Compressors.

In the description of configuring each of the following events, you must first create and configure a compressor from the Compressors palette of the Process Modeling toolbox, then you must initialize the process map. For details, see:

- Connecting Process Equipment.
- Initializing Process Maps.

Compression Ratio Decrease

The Compression Ratio Decrease event detects when the ratio of the discharge and suction pressures of a compressor decreases enough to violate a specified limit. The event generates an operator message when the event is true and generates a SymCure Compression Ratio Decrease event.

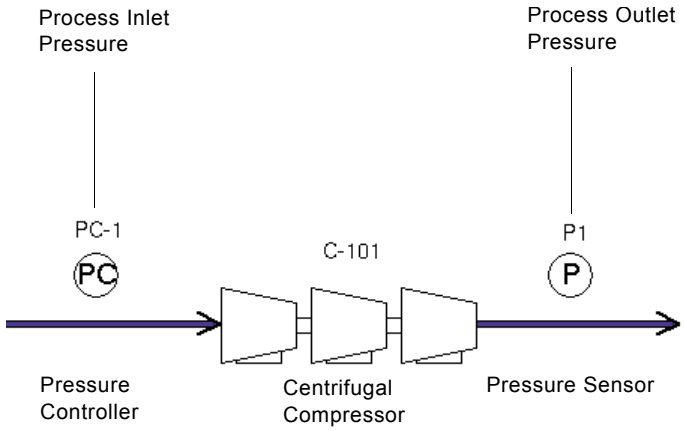
The event requires the inlet and outlet pressure related sensors of a compressor.

To configure the Compression Ratio Decrease event:

1. Create a pressure sensor from the Instruments palette, create a pressure controller from the Controllers palette, and place them near a compressor in a process map.
2. Configure the Process Inlet Pressure and Process Outlet Pressure related sensors of the compressor.

For details, see Configuring Domain Objects.

For example, here is the C-101 compressor with the required instruments:



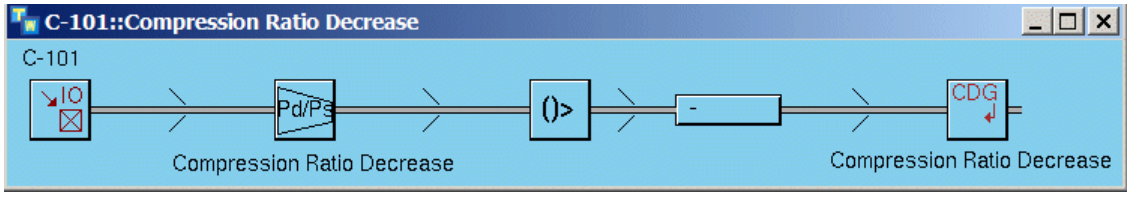
3 Initialize the process map.

For details, see Initializing Process Maps.

4 Choose Show Logic on the compressor and select the Compression Ratio Decrease event for the compressor.

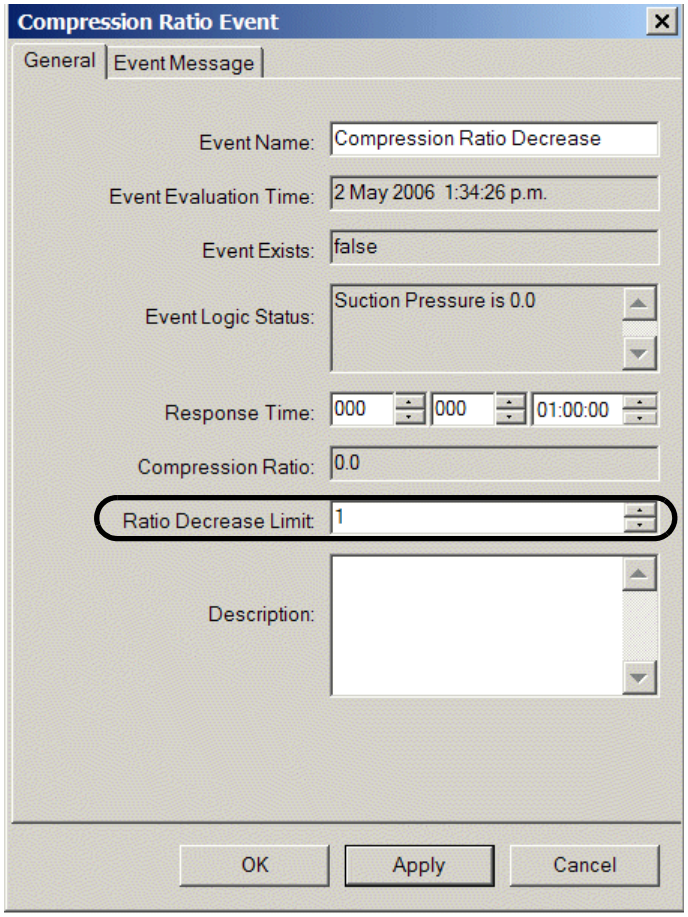
The process map must be initialized to configure the specific event detection diagram.

Here is the Compression Ratio Decrease event detection diagram for the C-101 compressor. The diagram detects a decrease in the compression ratio, filters out unchanged values, and generates a SymCure Compression Ratio Decrease event when true.



5 Show the properties dialog for the Compression Ratio Event block and configure the Ratio Decrease Limit for a decrease in compression ratio during the response time.

Here is the Compression Ratio Event block configured to detect a decrease in the compression ratio of more than 1.0 during the last 60 minutes:



The Compression Ratio is calculated by dividing the discharge pressure by the suction pressure of the compressor. The calculation is based on the Process Inlet Pressure and the Process Outlet Pressure related sensors of the compressor.

The compressor is now configured to detect the Compression Ratio Decrease event.

Power Projected High

The Power Projected High event detects a projected high value for the calculated power of the Compressor Power derived sensor of a compressor. The power calculation of a compressor requires these instruments:

- Process Flow related sensor.
- Process Inlet Pressure related sensor, which is a pressure controller used as the suction pressure for the compressor.
- Process Outlet Pressure related sensor, which is the discharge pressure.
- Compressor Power related sensor, which is a derived sensor for the compressor.

In addition, you must configure internal datapoints of the Compressor Power derived sensor to provide these values:

- Gas molecular weight
- Compressor stages
- Specific heat ratio

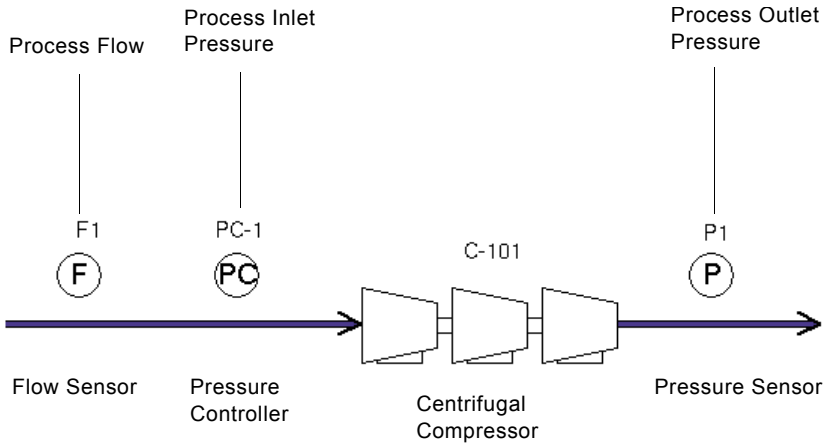
The Compressor Power derived sensor calculates the power and polytropic head for the compressor. The Power Projected High event for the compressor detects a projected high value for the calculated power within a response time. The event generates an operator message when the event is true and generates a SymCure Power Projected high event.

To configure the Power Projected High event:

1. Create a flow and pressure sensor from the Instruments palette, create a pressure controller from the Controllers palette, and place them near a compressor in a process map.
2. Configure the Process Flow, Process Inlet Pressure, and Process Outlet Pressure related sensors of the compressor.

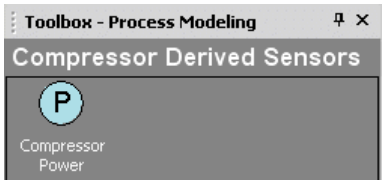
For details, see Configuring Domain Objects.

For example, here is the C-101 compressor with the required instruments:

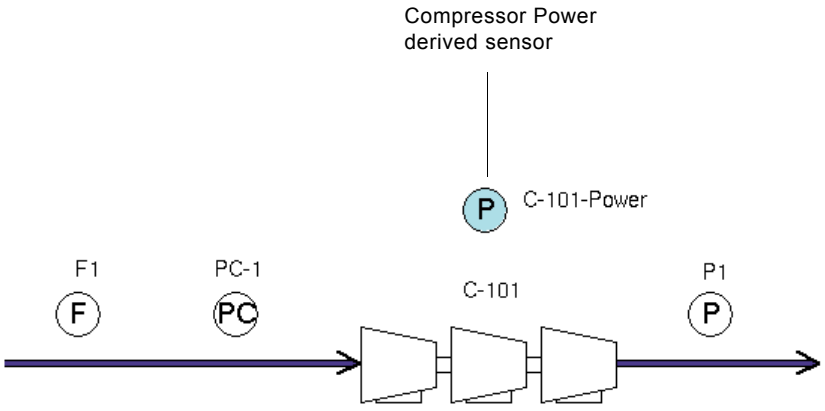


3 Create a Compressor Power derived sensor from the Compressor Derived Sensors palette of the Process Modeling toolbox and place it on your process map near the compressor.

Here is the Compressor Derived Sensors palette:

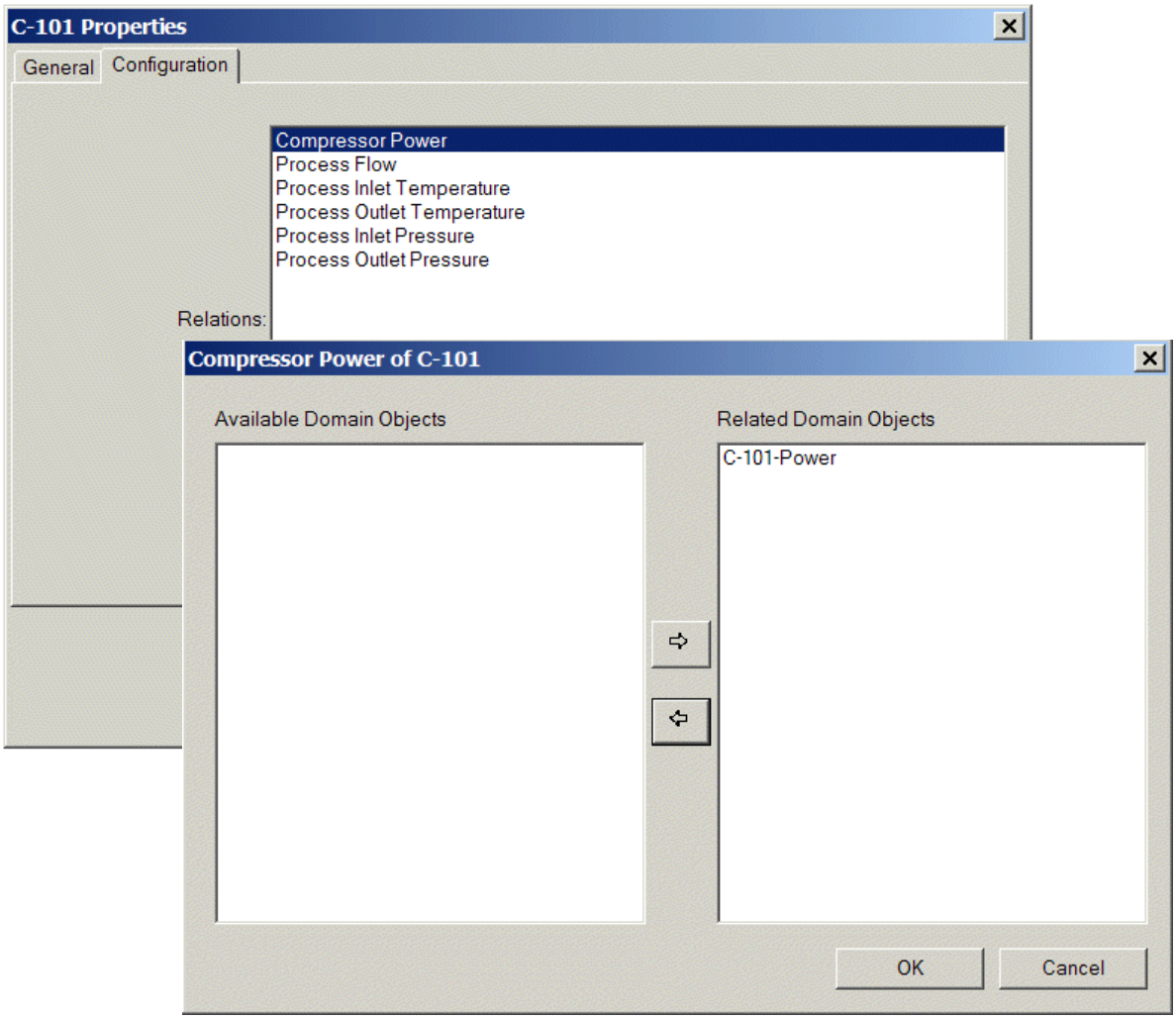


For example, here is the C-101 compressor with a Compressor Power derived sensor named C-101-Power, which will calculate the compressor power:



4 Display the properties dialog of the compressor, click the Configuration tab, and configure the Compressor Power related sensor.

Here is how you would configure the Compressor Power of the C-101 compressor to be the C-101-Power derived sensor:



The Compressor Power derived sensor defines internal datapoints for the flow, suction pressure, and discharge pressure, which it obtains from the related sensors of the compressor. To automatically configure these internal datapoints, you must initialize the process map.

5 Initialize the process map.

For details, see Initializing Process Maps.

Now you must configure additional properties of the Compressor Power derived sensor.

6 Configure the properties of the Compressor Power derived sensor.

a Display the properties dialog for the Compressor Power derived sensor.

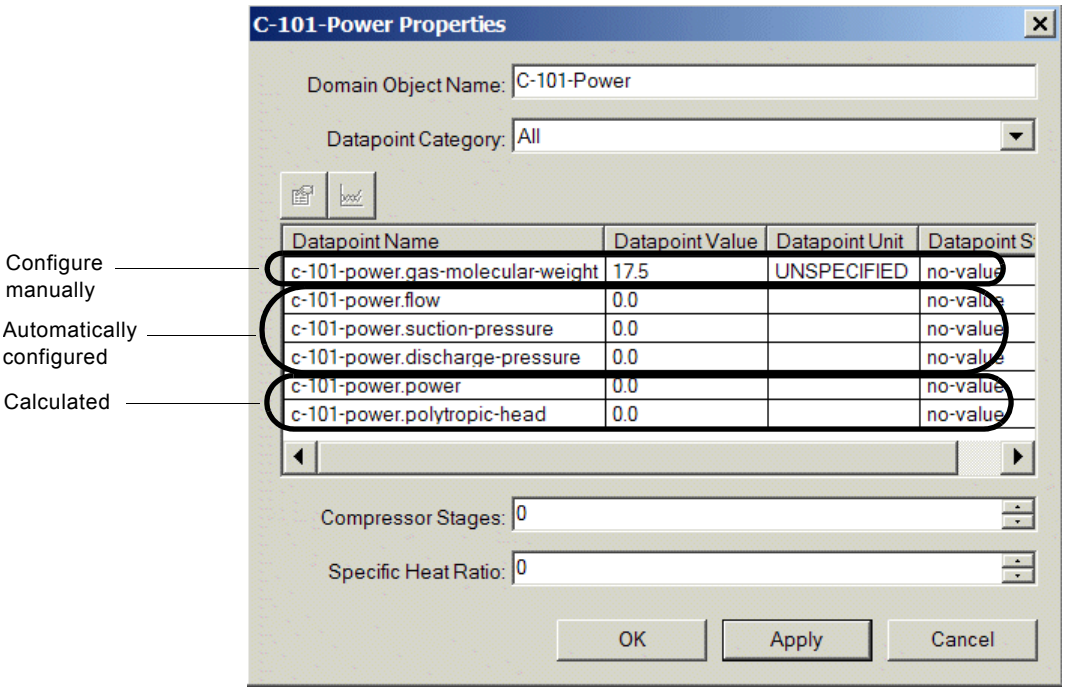
Once you have configured the related sensors of the compressor and initialized the process map, the flow, suction-pressure, and discharge-pressure internal datapoints of the derived sensor are automatically configured to use the process values of the related sensors.

In addition, you must manually configure the gas-molecular-weight internal datapoint.

The derived sensor uses these internal datapoint values to calculate the compressor power.

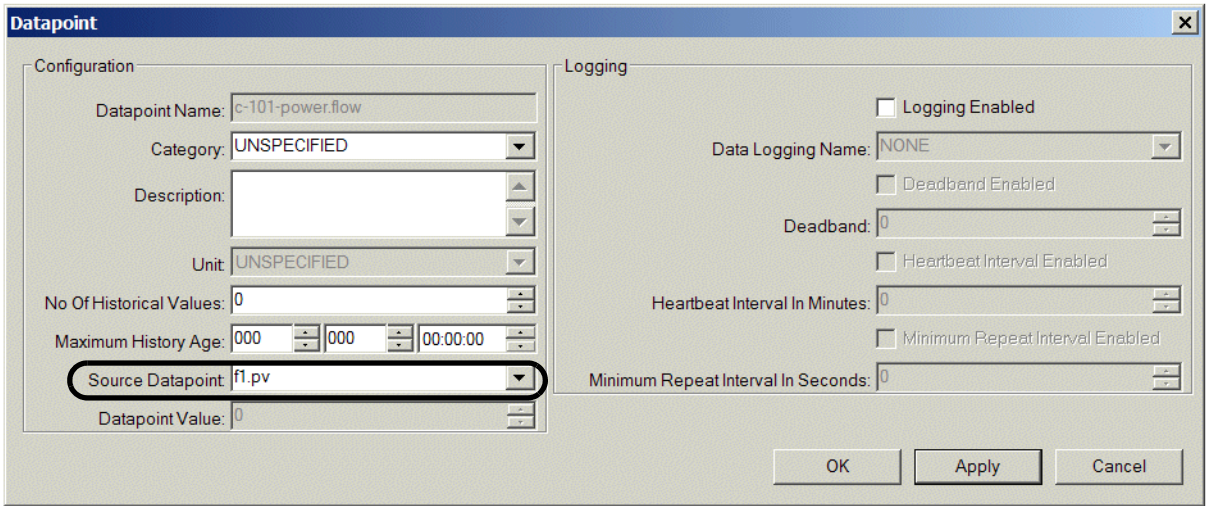
The remaining internal datapoints—power and polytropic-head—are calculated values of the derived sensor. You must scroll down to see all the calculated values.

Here is the dialog for the C-101-Power derived sensor:



b Select the flow internal datapoint and click the Properties button.

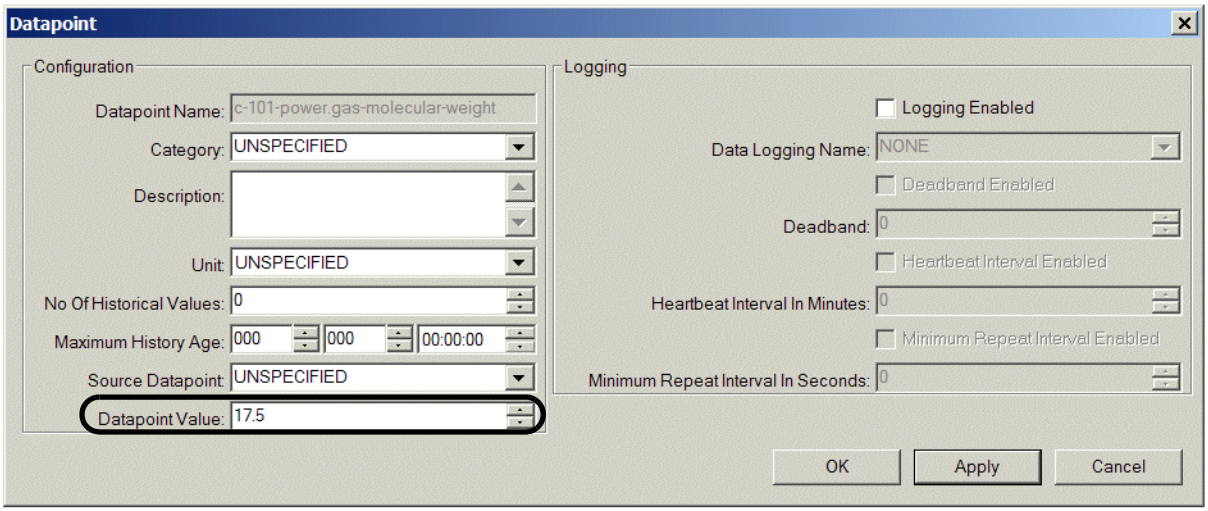
Notice that the Source Datapoint is automatically configured to be f1.pv, which is the PV of the Process Flow related sensor of the C-101 compressor:



c Verify the Source Datapoint for the suction-pressure and discharge-pressure of the Compressor Power.

d Display the properties of the gas-molecular-weight internal datapoint and configure the Datapoint Value to be the molecular weight of the process gas.

Here is the properties dialog for the c-101-power.gas-molecular-weight internal datapoint, which is configured to be 17.5:



e Finally, in the properties dialog for the Compressor Power derived sensor, configure these properties:

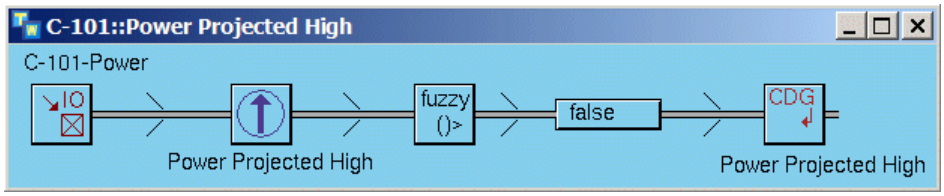
- Compressor Stages — The number of stages in the compressor.
- Specific Heat Ratio — The specific heat ratio of the process gas.

The last step in configuring the Power Projected High event of a compressor is to configure the projected high limit for the power calculation, which you do in the specific event detection diagram for the compressor.

7 Choose Show Logic on the compressor and select the Power Projected High event for the compressor.

The process map must be initialized to configure the specific event detection diagram.

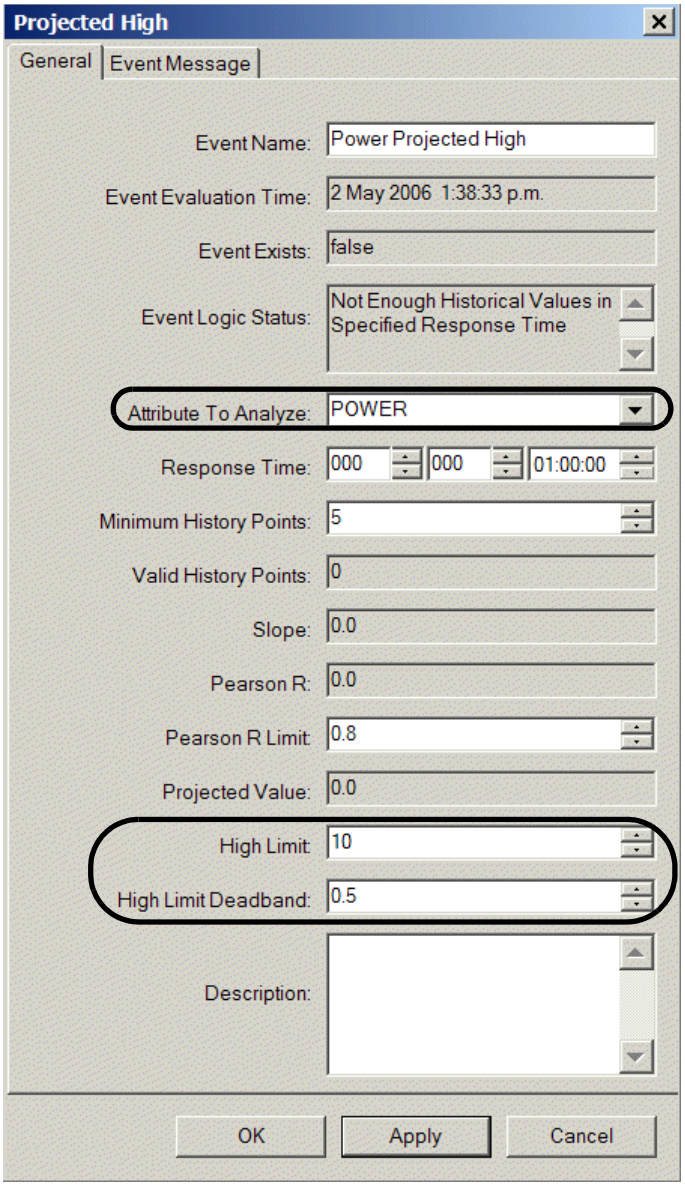
Here is the Power Projected High event detection diagram for the C-101 compressor. The diagram detects a decrease in the power, filters out unchanged values, and generates a SymCure Power Projected High event.



8 Display the properties dialog for the Projected High block and configure the High Limit and High Limit Deadband to be the limit for the compressor power.

You can also configure the Response Time, Minimum History Points, and Pearson R Limit, as needed. For information on how to configure these values, see PV Projected High.

Here is the properties dialog for the Projected High block that causes the event to trigger when the calculated power of the compressor is projected to be between 9.5 and 10.0 during an hour. Notice that the Attribute to Analyze is power, which the Compressor Power derived sensor calculates.



The compressor is now fully configured to detect the Power Projected High event.

Polytropic Head Change

The Polytropic Head Change event detects a change in the calculated polytropic head of the Compressor Power derived sensor of a compressor. The event generates an operator message when the event is true and generates a SymCure Polytropic Head Change event.

The Polytropic Head Change event requires all the same sensors as the Power Projected High Event.

To configure the Polytropic Head Change event:

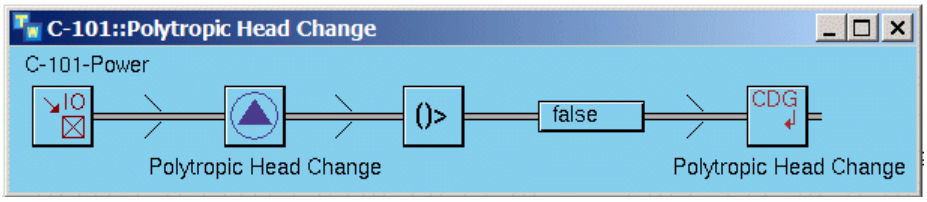
1 Create and configure a compressor with the same related sensors that the Power Projected High event requires.

For details, see Power Projected High.

2 Choose Show Logic on the compressor and select the Polytropic Head Change event for the compressor.

The process map must be initialized to create the specific event detection diagrams.

Here is the Polytropic Head Change event detection diagram for the C-101 compressor. The diagram detects a change in compressor polytropic head, filters out unchanged values, and generates a SymCure Polytropic Head Change event.

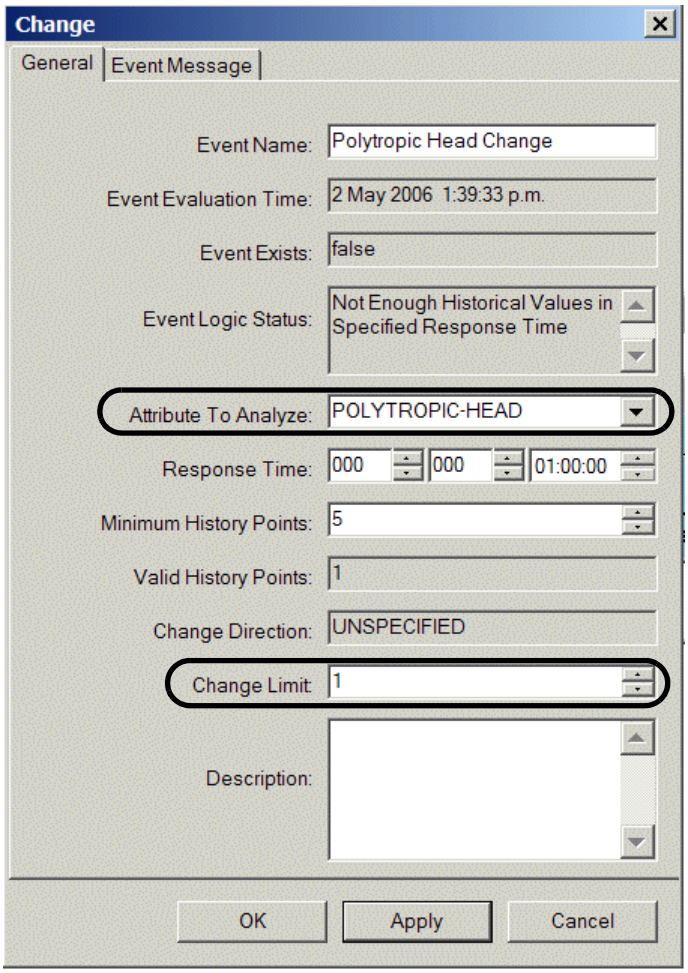


3 Show the properties dialog for the Change block and configure the Change Limit for a change in compressor polytropic head.

You can also configure the Response Time and Minimum History Points, as needed.

For information on how to configure these values, see PV Change.

Here is the Change block configured to detect a change in compressor polytropic head greater than 0.1. Notice that the Attribute to Analyze is polytropic-head, which is calculated in the Compressor Power related sensor of the compressor.



The compressor is now configured to detect the Polytropic Head event.

Built-in Event Detection for Equipment Drivers

Equipment drivers provide a generic methodology for monitoring and diagnosing common problems with motor and steam turbine equipment drivers. The equipment drivers perform these functions by analyzing current and historical process values from process sensors and derived sensors that are related to the equipment driver.

Equipment drivers can detect these events:

- Motor Power Projected High
- Turbine Power Projected High

All events generate operator messages when the event occurs.

Motor Power Projected High

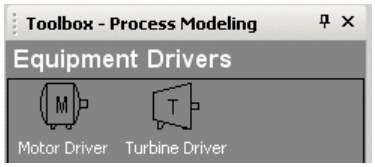
The Power Projected High event for a motor driver detects a projected high value for the calculated power of the Motor Power derived sensor of a motor driver. The power calculation of a motor driver requires these instruments:

- Motor Amps related sensor.
- Motor Voltage related sensor.
- Motor Power related sensor, which is a derived sensor for the motor driver.

The Motor Power derived sensor calculates the power for the motor driver. The Power Projected High event for the motor driver detects a projected high value for the calculated power within a response time. The event generates an operator message when the event is true.

To configure the Power Projected High event for a motor driver:

1 Create a motor driver from the Equipment Drivers palette of the Process Modeling toolbox:

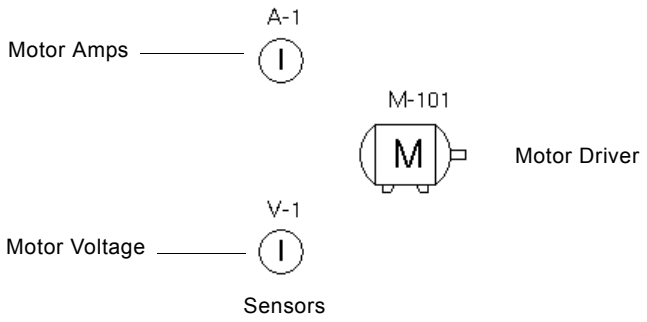


2 Create two general sensors from the Instruments palette, one for motor amps and the other for motor voltage, and place them near the motor driver.

3 Configure the Motor Amps and Motor Voltage related sensors of the motor driver by using the two general instruments.

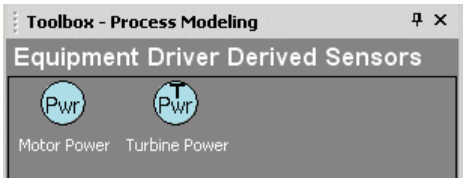
For details, see Configuring Domain Objects.

For example, here is the M-101 motor driver with the required sensors:

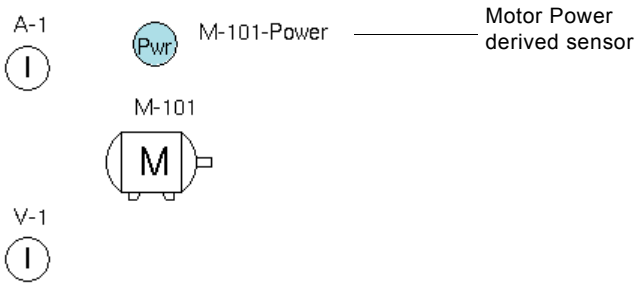


4 Create a Motor Power derived sensor from the Equipment Driver Derived Sensors palette of the Process Modeling toolbox and place it on your process map near the compressor.

Here is the Equipment Driver Derived Sensors palette:

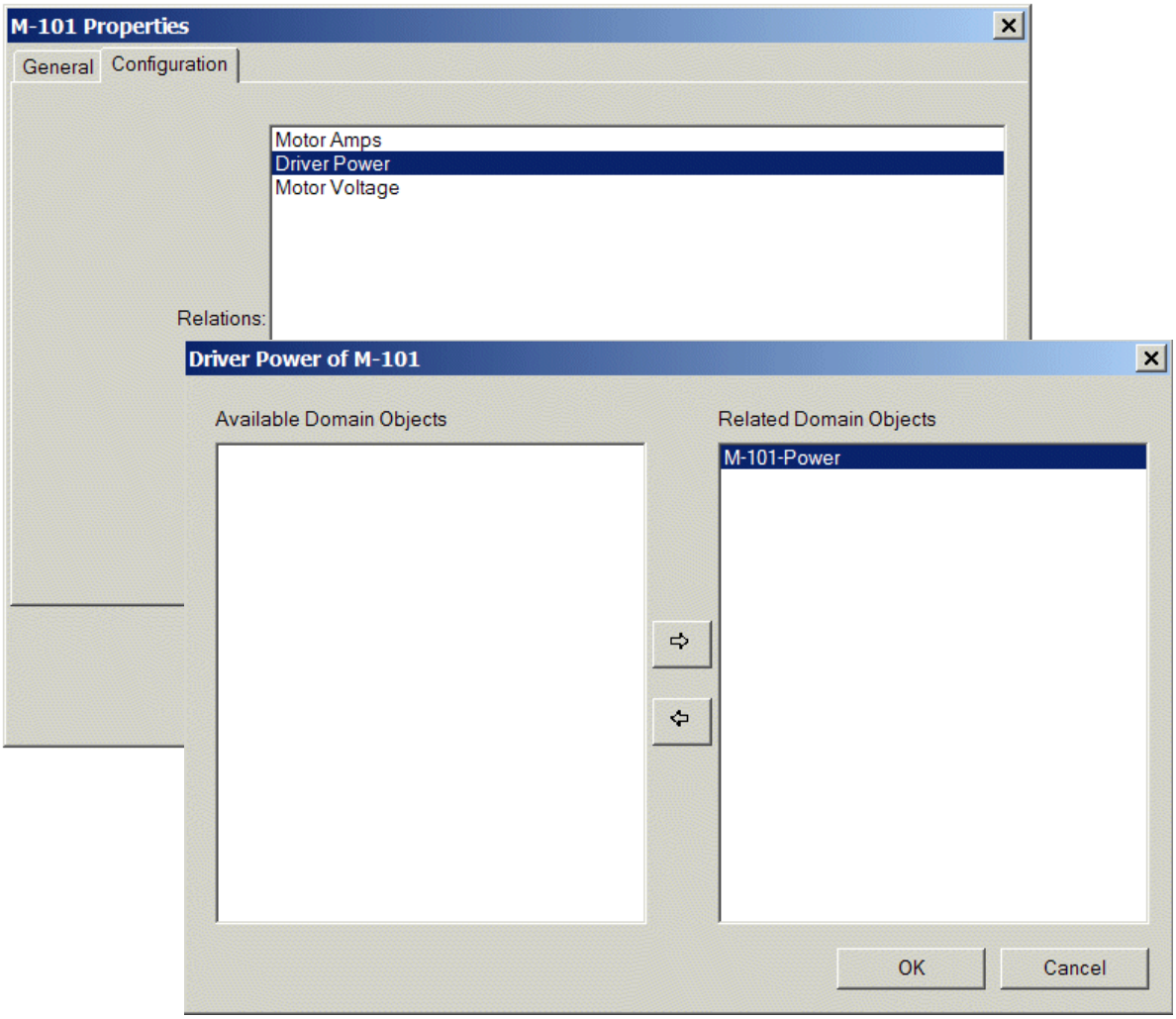


For example, here is the M-101 motor driver with a Motor Power derived sensor named M-101-Power, which will calculate the motor power:



5 Display the properties dialog of the motor driver, click the Configuration tab, and configure the Driver Power related sensor.

Here is how you would configure the Driver Power of the M-101 motor driver to be the M-101-Power derived sensor:



The Driver Power derived sensor defines an internal datapoint that calculates the motor power.

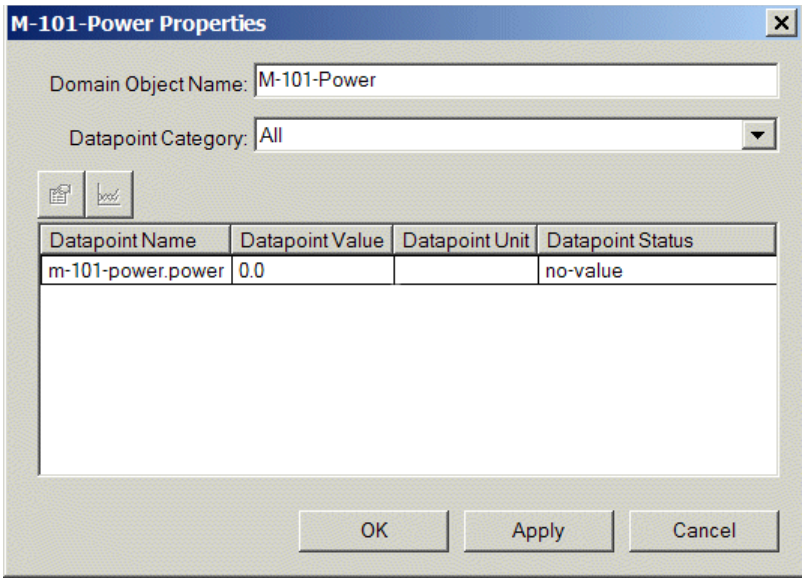
6 Initialize the process map.

For details, see Initializing Process Maps.

7 Display the properties dialog for the Motor Power derived sensor.

The derived sensor uses the related sensors of the motor driver to calculate the power internal datapoint.

Here is the dialog for the M-101-Power derived sensor:

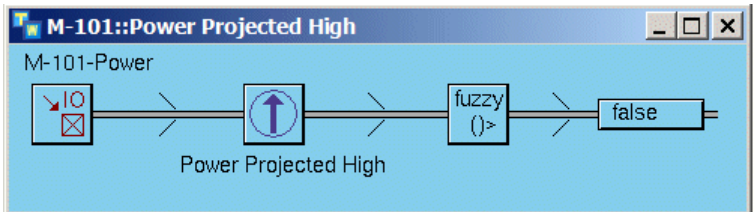


The last step in configuring the Power Projected High event of a motor driver is to configure the projected high limit for the power calculation, which you do in the specific event detection diagram for the motor driver.

8 Choose Show Logic on the motor driver and select the Power Projected High event.

The process map must be initialized to configure the specific event detection diagram.

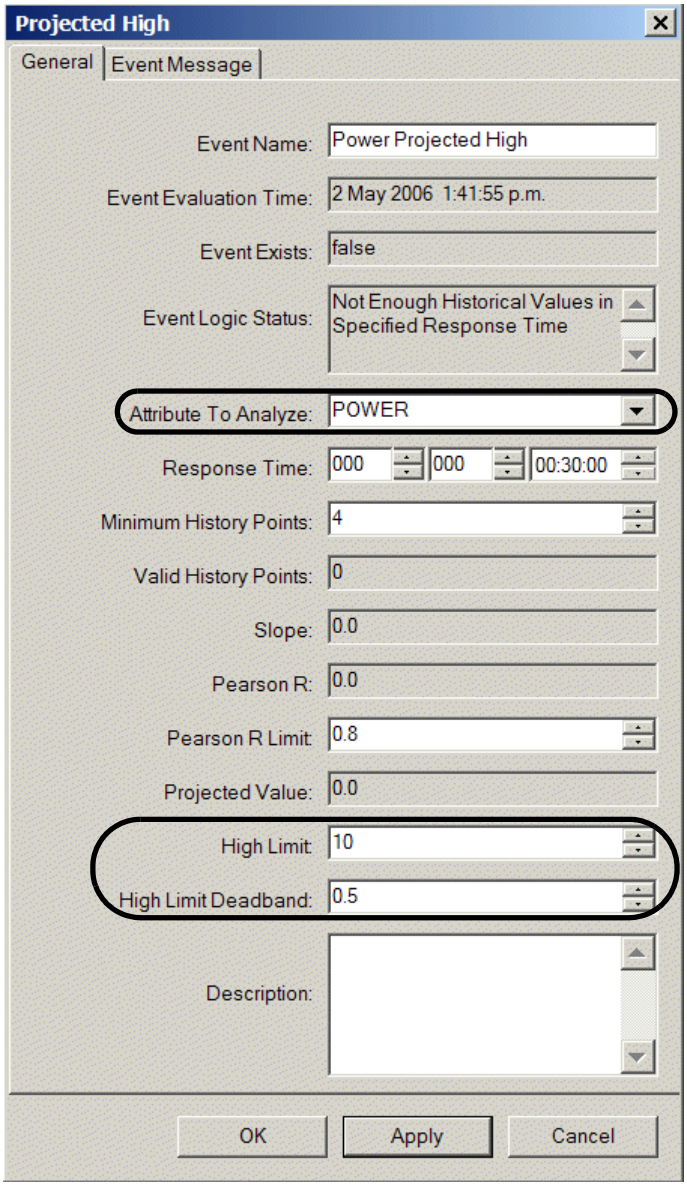
Here is the Power Projected High event detection diagram for the M-101 motor driver. The diagram detects a projected high value for the power and filters out unchanged values.



9 Display the properties dialog for the Projected High block and configure the High Limit and High Limit Deadband to be the limit for the projected high power.

You can also configure the Response Time, Minimum History Points, and Pearson R Limit, as needed. For information on how to configure these values, see PV Projected High.

Here is the properties dialog for the Projected High block that causes the event to trigger when the calculated power of the motor driver is projected to be between 9.5 and 10.0 during an hour. Notice that the Attribute to Analyze is power, which is calculated in the Motor Power derived sensor.



The motor driver is now fully configured to detect the Power Projected High event.

Turbine Power Projected High

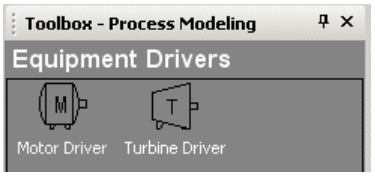
The Power Projected High event for a turbine driver detects a projected high value for the calculated power of the Turbine Power derived sensor of a turbine driver. The power calculation of a turbine driver requires these instruments:

- Steam Flow related sensor.
- Inlet Steam Pressure and Outlet Steam Pressure related sensors.
- Inlet Steam Temperature and Outlet Steam Temperature related sensors.
- Turbine Power related sensor, which is a derived sensor for the turbine driver.

The Turbine Power derived sensor calculates the power for the turbine driver. The Power Projected High event for the turbine driver detects a projected high value for the calculated power within a response time. The event generates an operator message when the event is true.

To configure the Power Projected High event for a turbine driver:

1 Create a turbine driver from the Equipment Drivers palette of the Process Modeling toolbox:

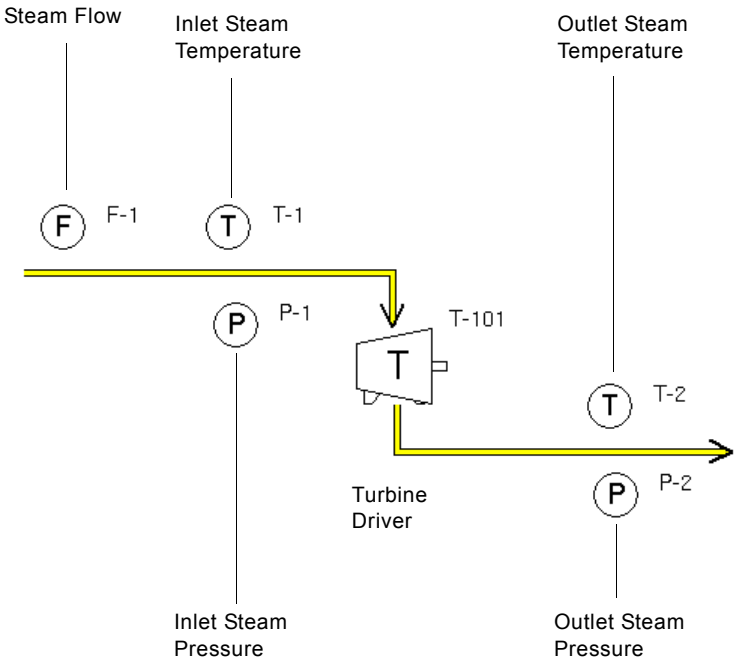


2 Create a flow sensor, two temperature sensors, and two pressure sensors from the Instruments palette, and place them near the turbine driver.

3 Configure the Steam Flow, Inlet Steam Pressure, Outlet Steam Pressure, Inlet Stream Temperature, and Outlet Stream Temperature related sensors of the turbine driver.

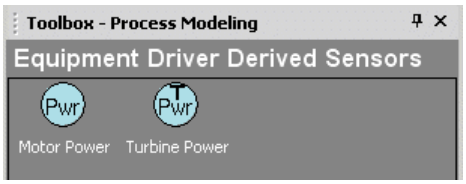
For details, see Configuring Domain Objects.

For example, here is the T-101 turbine driver with the required sensors:

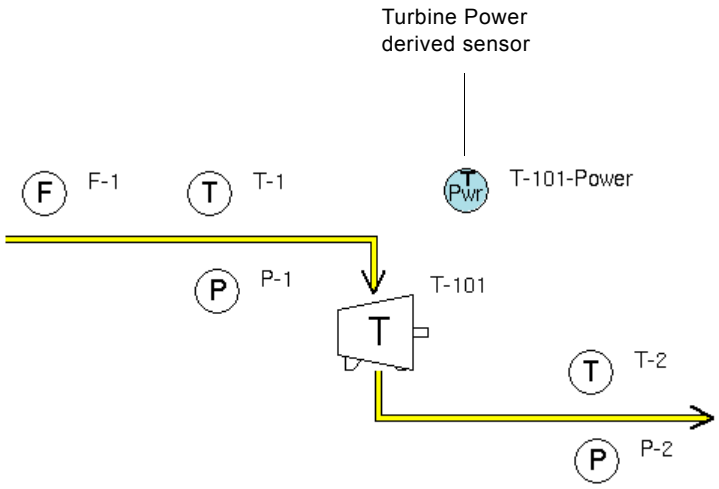


4 Create a Turbine Power derived sensor from the Equipment Driver Derived Sensors palette of the Process Modeling toolbox and place it on your process map near the turbine driver.

Here is the Equipment Driver Derived Sensors palette:

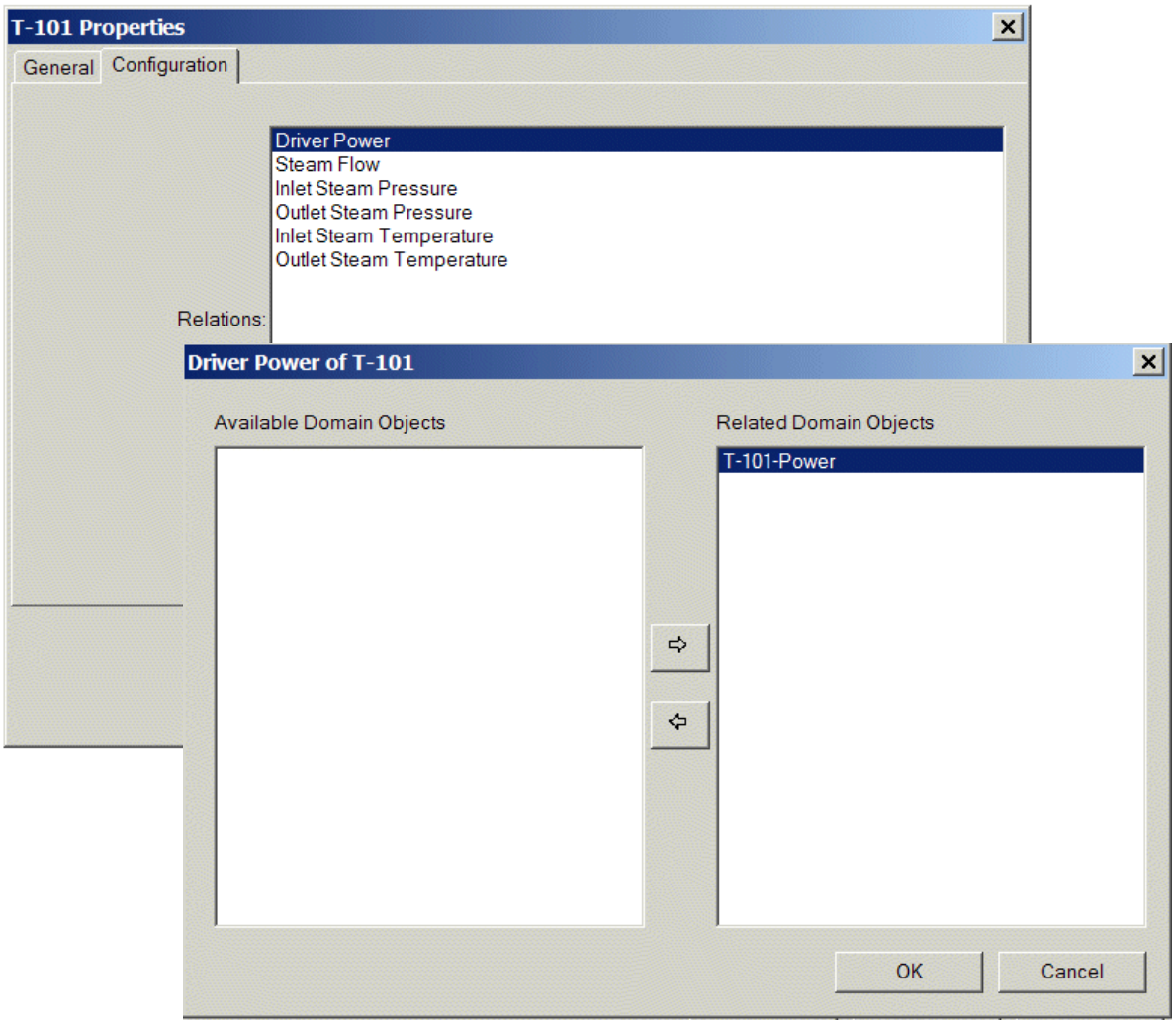


For example, here is the M-101 compressor with a Turbine Power derived sensor named T-101-Power, which will calculate the turbine driver power:



5 Display the properties dialog of the turbine driver, click the Configuration tab, and configure the Driver Power related sensor.

Here is how you would configure the Driver Power of the T-101 turbine driver to be the T-101-Power derived sensor:



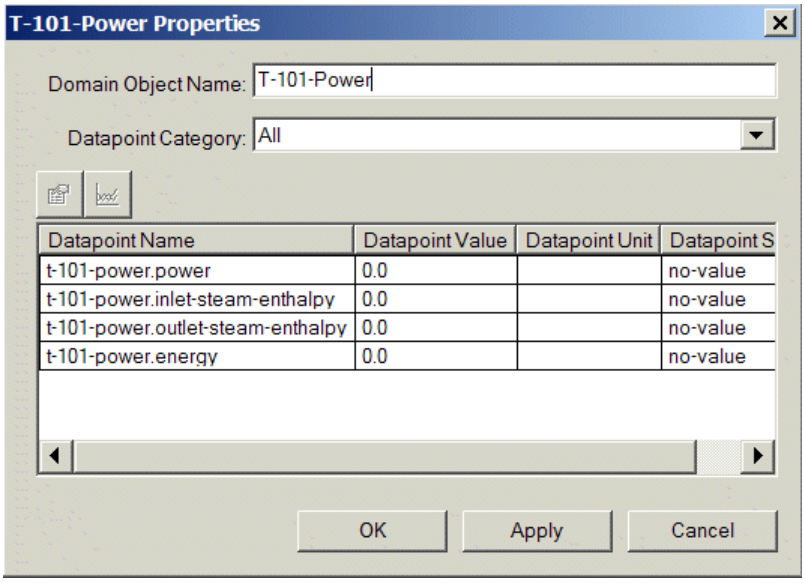
6 Initialize the process map.

For details, see Initializing Process Maps.

7 Display the properties dialog for the Turbine Power derived sensor.

The derived sensor uses the related sensors of the turbine driver to calculate the power, inlet-steam-enthalpy, outlet-steam-enthalpy, and energy internal datapoints.

Here is the dialog for the T-101-Power derived sensor:

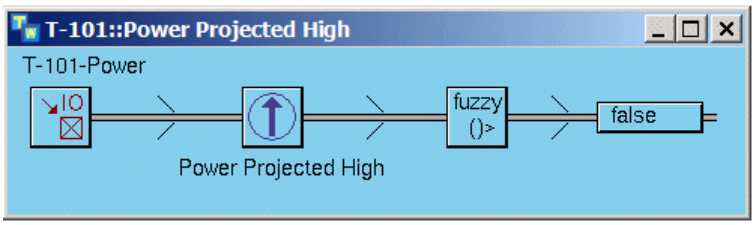


The last step in configuring the Power Projected High event of a turbine driver is to configure the projected high limit for the power calculation, which you do in the specific event detection diagram for the turbine driver.

8 Choose Show Logic on the turbine driver and select the Power Projected High event.

The process map must be initialized to configure the specific event detection diagram.

Here is the Power Projected High event detection diagram for the T-101 turbine driver. The diagram detects a projected high value for the power and filters out unchanged values.

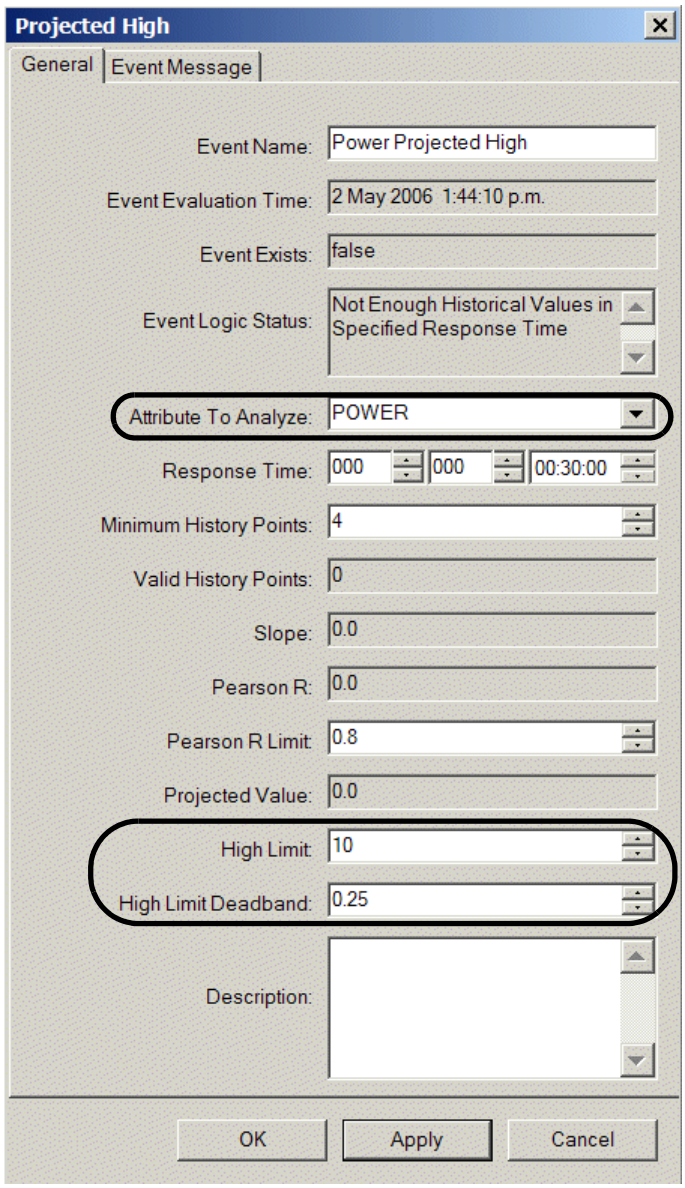


9 Display the properties dialog for the Projected High block and configure the High Limit and High Limit Deadband to be the limit for the projected high power.

You can also configure the Response Time, Minimum History Points, and Pearson R Limit, as needed.

For information on how to configure these values, see PV Projected High.

Here is the properties dialog for the Projected High block that causes the event to trigger when the calculated power of the turbine driver is projected to be between 9.5 and 10.0 during an hour. Notice that the Attribute to Analyze is power, which is calculated in the Turbine Power derived sensor.



The turbine driver is now fully configured to detect the Power Projected High event.

Built-in Generic Fault Models

When the intelligent object libraries are loaded, domain objects define a number of generic fault models. The built-in generic event detection templates generate some of the SymCure events in these generic fault models. For example, the PV High event detection template of a sensor generates a SymCure High event. Similarly, the Low Efficiency event of a heater generates a SymCure Excess Coking event.

A number of the built-in generic fault models define causal relationships between events for further root cause analysis. For example, the generic Tube Skin Temperature fault model defines three possible root causes for the High Tube Skin Temperature event, which is caused by the High event on the Tube Skin Delta T derived sensor of a heater: High Burner Pressure, Flame Impingement, and Excess Coking.

Other built-in generic fault models define simple causal relationships between process equipment events and their related sensor events. For example, in the generic flow sensor fault model, the High event on a related flow sensor is directly caused by the High Process Flow event on the process equipment.

You can customize the built-in generic fault models or you can subclass the built-in domain objects and create your own generic fault models.

For general information on generic fault models, see Part V, Diagnostic Reasoning. For detailed information, see the *SymCure User's Guide*.

Displaying Built-in Generic Fault Models

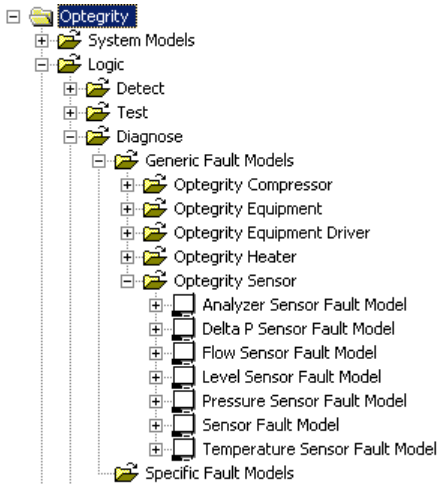
To display the built-in generic fault models:

➔ Choose Project > Logic > Diagnose > Generic Fault Models, choose a category, then choose the generic fault model to view.

or

➔ Choose View > Project, expand the Fault Models > Generic Fault Models, expand the tree to view the generic fault model in the desired category, and choose Show Details.

For example, here is a portion of the Navigator with the Generic Fault Models folder expanded to show the generic fault models for Optegrity sensors:



Built-in Generic Fault Models for Sensors

All sensor classes define these generic fault models:

- Flow Sensor Fault Model
- Level Sensor Fault Model
- Temperature Sensor Fault Model
- Pressure Sensor Fault Model
- Sensor Fault Model
- Analyzer Sensor Fault Model
- Delta P Sensor Fault Model
- Motor Driver Fault Model

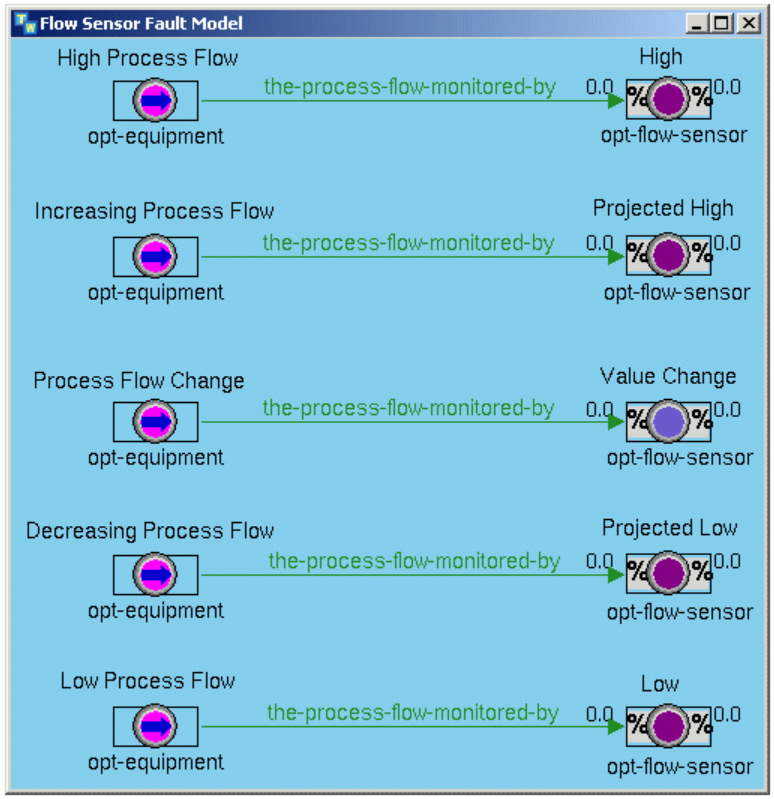
Flow Sensor Fault Model

The built-in generic flow sensor fault model defines direct causal relationships between process flow events on equipment and events on their related flow sensors.

For example, the PV High event for a flow sensor generates the SymCure High event on the flow sensor when the PV exceeds the specified limit. The generic flow sensor fault model defines a direct causal relationship between the High Process Flow event of the process equipment and the High event on its related Process Flow sensor.

Here are the causal relationships in the generic flow sensor fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV High	High	High Process Flow
PV Low	Low	Low Process Flow
PV Projected High	Projected High	Increasing Process Flow
PV Projected Low	Projected Low	Decreasing Process Flow
PV Change	Value Change	Process Flow Change



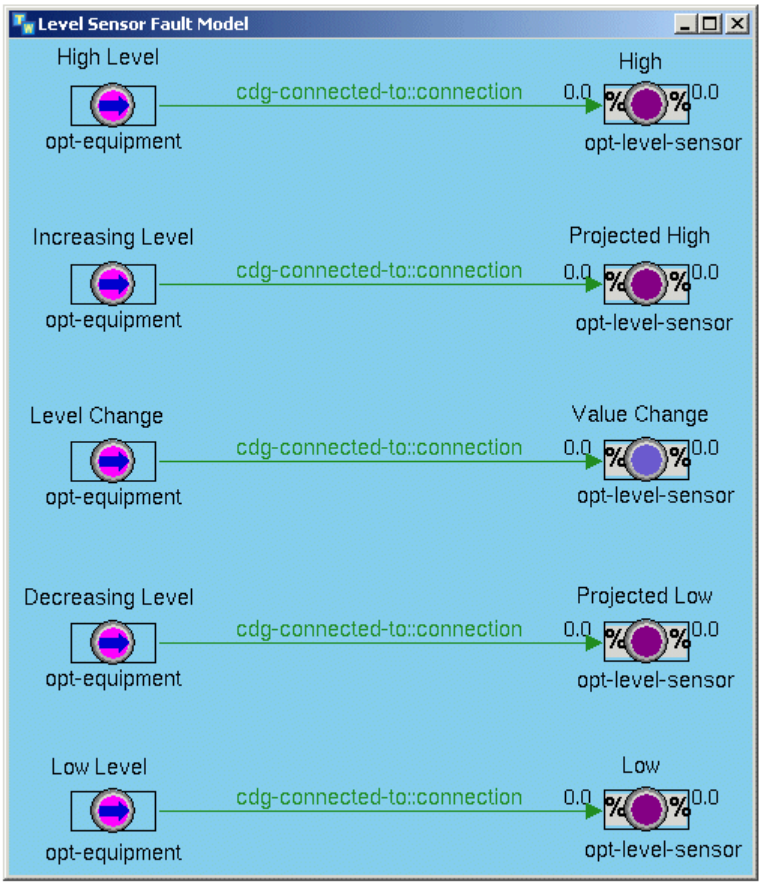
Level Sensor Fault Model

The built-in generic level sensor fault model defines direct causal relationships between events on process equipment and events on level sensors that are connected to the equipment via any type of connection.

For example, the PV High event for a level sensor generates the SymCure High event on the level sensor when the PV exceeds the specified limit. The generic level sensor fault model defines a direct causal relationship between the High Level event of the process equipment and the High event on its connected level sensor.

Here are the causal relationships in the generic level sensor fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV High	High	High Level
PV Low	Low	Low Level
PV Projected High	Projected High	Increasing Level
PV Projected Low	Projected Low	Decreasing Level
PV Change	Value Change	Level Change



Temperature Sensor Fault Model

The built-in generic temperature sensor fault model defines causal relationships between process temperature events on process equipment and events on their related temperature sensors.

For example, the PV High event for a temperature sensor generates the SymCure High event on the temperature sensor when the PV exceeds the specified limit. The generic temperature sensor fault model defines a causal relationship between the High Process Temperature event of the process equipment and the High event on either its related Process Inlet Temperature or Process Outlet Temperature related sensor.

Here are the causal relationships in the generic temperature sensor fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV High	High	High Process Temperature
PV Low	Low	Low Process Temperature
PV Projected High	Projected High	Increasing Process Temperature
PV Projected Low	Projected Low	Decreasing Process Temperature
PV Change	Value Change	Process Temperature Change

Pressure Sensor Fault Model

The built-in generic pressure sensor fault model defines causal relationships between process pressure sensor events on process equipment and events on their related pressure sensors.

For example, the PV High event for a pressure sensor generates the SymCure High event on the pressure sensor when the PV exceeds the specified limit. The generic pressure sensor fault model defines a causal relationship between the

High Process Pressure event of the process equipment and the High event on either its related Process Inlet Pressure or Process Outlet Pressure related sensor.

Here are the causal relationships in the generic pressure sensor fault model:

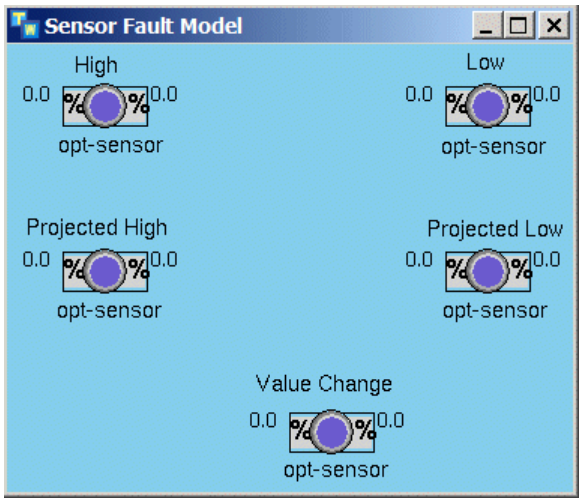
This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV High	High	High Process Pressure
PV Low	Low	Low Process Pressure
PV Projected High	Projected High	Increasing Process Pressure
PV Projected Low	Projected Low	Decreasing Process Pressure
PV Change	Value Change	Process Pressure Change

Sensor Fault Model

The built-in generic sensor fault model defines generic events for each built-in sensor event that can occur on a sensor; it defines no causal relationships.

Here are the events in the generic sensor fault model:

This sensor event...	Generates this SymCure event on the sensor...
PV High	High
PV Low	Low
PV Projected High	Projected High
PV Projected Low	Projected Low
PV Change	Value Change

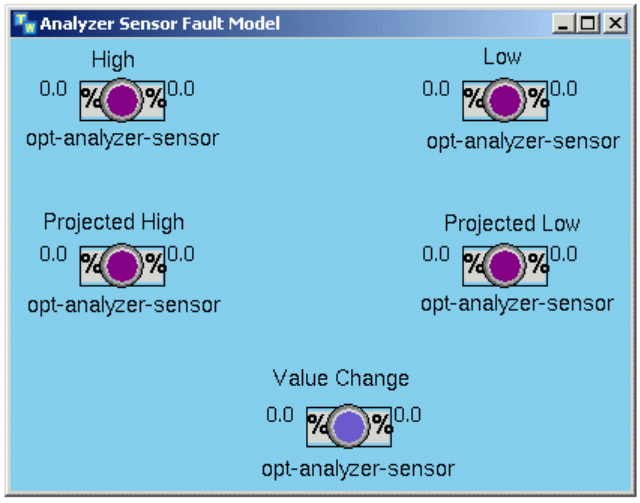


Analyzer Sensor Fault Model

The built-in generic analyzer sensor fault model defines generic events for each built-in sensor event that can occur on an analyzer.

Here are the events in the generic analyzer sensor fault model:

This analyzer event...	Generates this SymCure event on the analyzer...
PV High	High
PV Low	Low
PV Projected High	Projected High
PV Projected Low	Projected Low
PV Change	Value Change

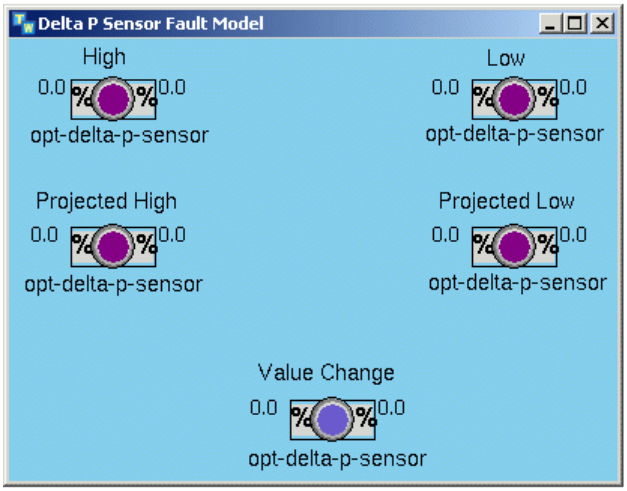


Delta P Sensor Fault Model

The built-in generic delta P sensor fault model defines generic events for each built-in sensor event that can occur on a delta P sensor.

Here are the events in the generic delta P sensor fault model:

This analyzer event...	Generates this SymCure event on the analyzer...
PV High	High
PV Low	Low
PV Projected High	Projected High
PV Projected Low	Projected Low
PV Change	Value Change



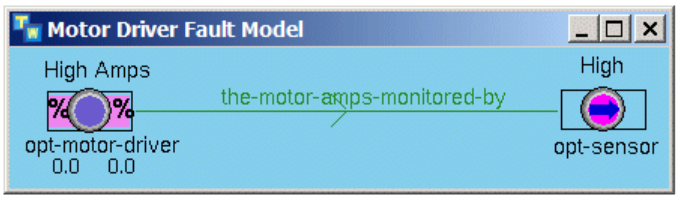
Motor Driver Fault Model

The built-in generic motor driver fault model defines a causal relationship between a high amps event on a motor driver and a high event on its related sensor.

Specifically, the High Amps event for a sensor generates the SymCure High event on the sensor when the PV exceeds the specified limit. The generic motor driver fault model defines a direct causal relationship between the High Amps event of a motor driver and the High event on its related Motor Amps sensor.

Here is the causal relationship in the generic motor driver fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV High	High	High Amps



Built-in Generic Fault Models for Process Equipment

All process equipment classes define generic fault models that describe causal relationships between events on the process equipment and events on domain objects that are connected upstream via a hydrocarbon line connection.

These generic fault models provide limited event propagation for certain types of process equipment, such as a heater, which requires two independent systems to operate—a fuel system and a hydrocarbon system.

For example, a high pressure on the fuel gas system of a heater does not propagate a high pressure to a pump connected upstream of the heater. A high pressure on the upstream pump propagates a high pressure to the heater that is connected via a hydrocarbon line; however, it does not propagate a high pressure to the fuel system, because the fuel system and hydrocarbon systems are independent.

All process equipment classes define these generic fault models:

- Process Equipment Flow Fault Model
- Process Equipment Level Fault Model
- Process Equipment Temperature Fault Model
- Process Equipment Pressure Fault Model

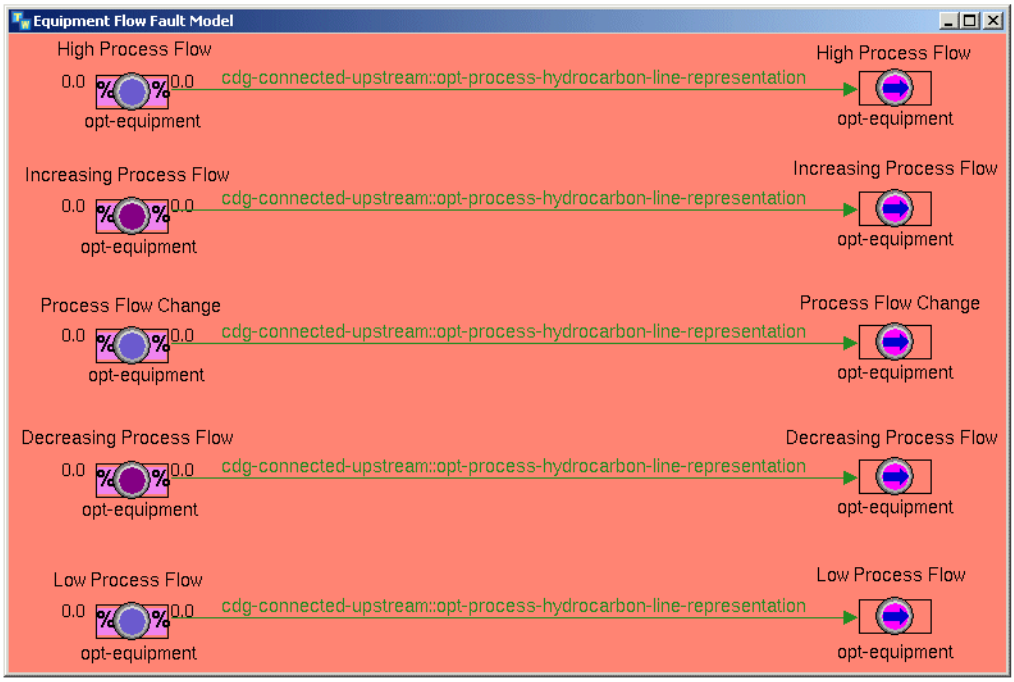
Process Equipment Flow Fault Model

The built-in generic equipment flow fault model defines causal relationships between flow events on the process equipment and flow events on the domain object that is connected upstream via a hydrocarbon line connection.

The downstream event may be caused by an event on the Process Flow related sensor of the equipment. See Flow Sensor Fault Model.

Here are the causal relationships in the generic equipment flow fault model:

This generic event on process equipment...	Causes this generic event on the domain object that is connected upstream...
High Process Flow	High Process Flow
Low Process Flow	Low Process Flow
Increasing Process Flow	Increasing Process Flow
Decreasing Process Flow	Decreasing Process Flow
Process Flow Change	Process Flow Change



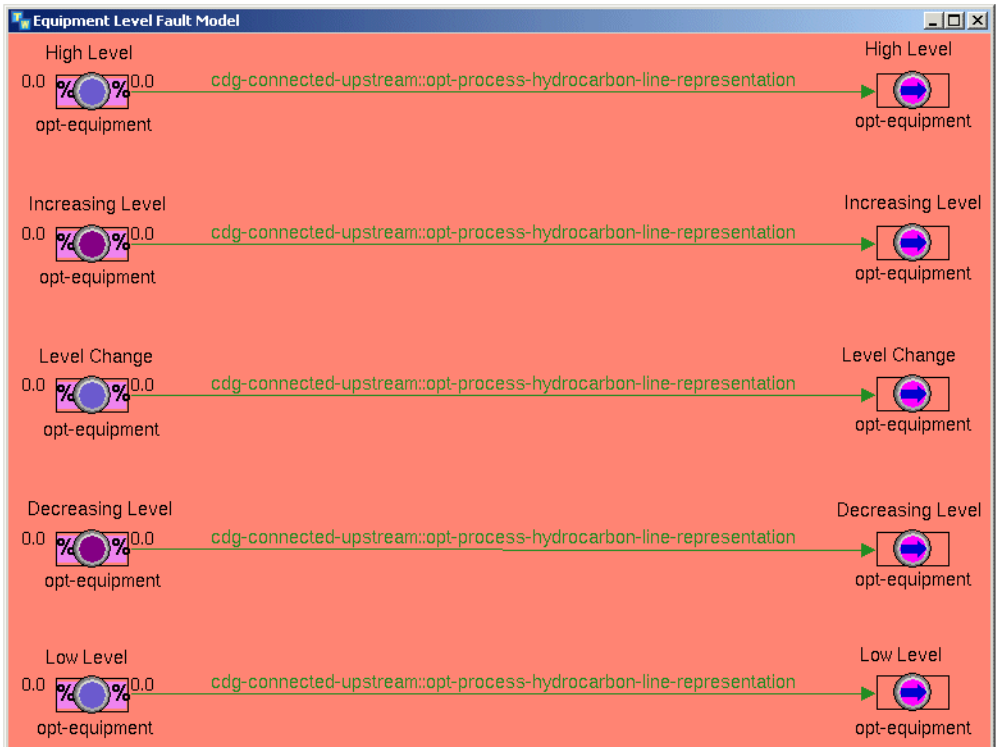
Process Equipment Level Fault Model

The built-in generic equipment level fault model defines causal relationships between level events on process equipment and level events on domain objects that are connected upstream via a hydrocarbon line connection.

The downstream event may be caused by an event on a level sensor that is connected to the downstream equipment. See Level Sensor Fault Model.

Here are the causal relationships in the generic equipment level fault model:

This generic event on process equipment...	Causes this generic event on the domain object that is connected upstream...
High Level	High Level
Low Level	Low Level
Increasing Level	Increasing Level
Decreasing Level	Decreasing Level
Level Change	Level Change



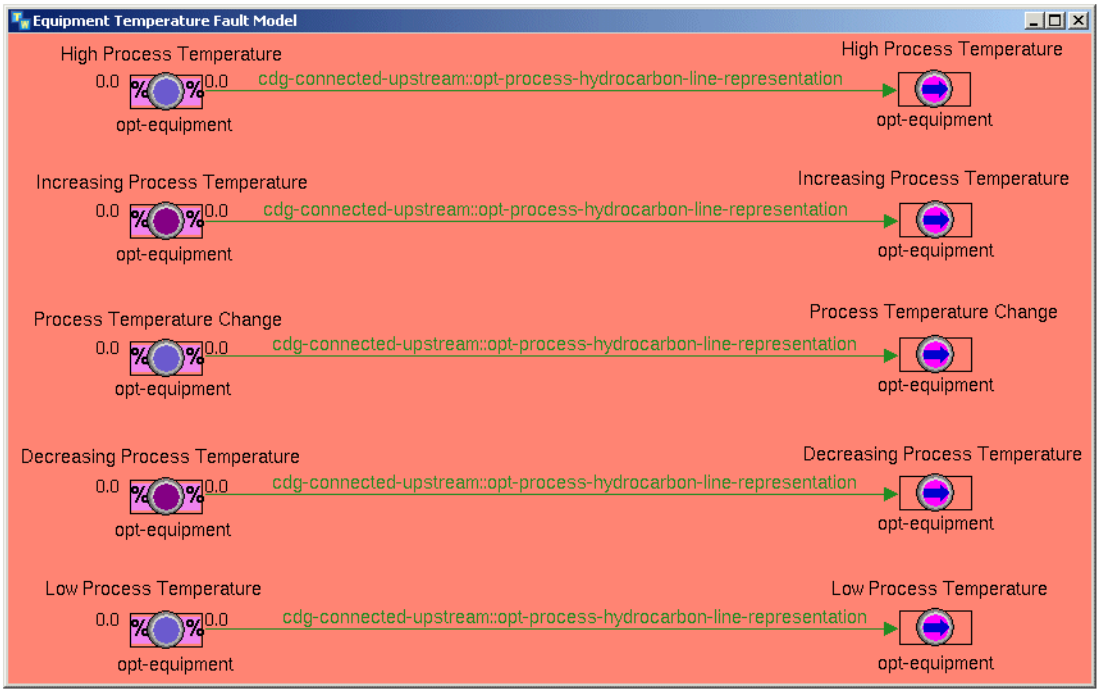
Process Equipment Temperature Fault Model

The built-in generic equipment temperature fault model defines causal relationships between temperature events on process equipment and temperature events on domain objects that are connected upstream via a hydrocarbon line connection.

The downstream event may be caused by an event on the Process Inlet Temperature or Process Outlet Temperature related sensor of the equipment. See Temperature Sensor Fault Model.

Here are the causal relationships in the generic equipment temperature fault model:

This generic event on process equipment...	Causes this generic event on the domain object that is connected upstream...
High Process Temperature	High Process Temperature
Low Process Temperature	Low Process Temperature
Increasing Process Temperature	Increasing Process Temperature
Decreasing Process Temperature	Decreasing Process Temperature
Process Temperature Change	Process Temperature Change



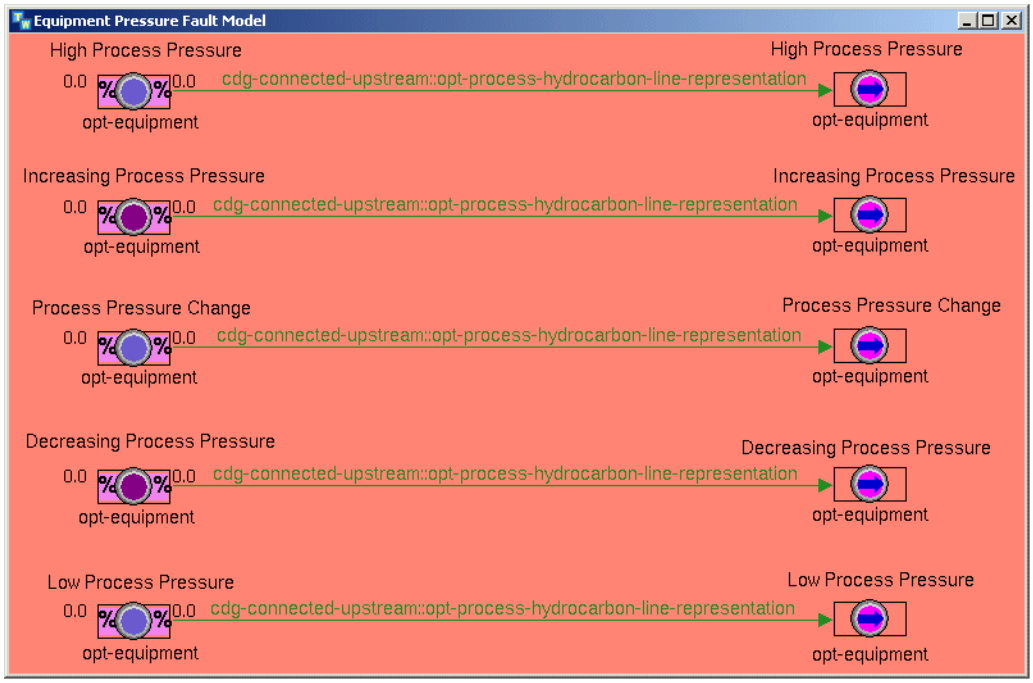
Process Equipment Pressure Fault Model

The built-in generic equipment pressure fault model defines causal relationships between pressure events on process equipment and pressure events on domain objects that is connected upstream via a hydrocarbon line connection.

The downstream event may be caused by an event on the Process Inlet Pressure or Process Outlet Pressure related sensor of the equipment. See Pressure Sensor Fault Model.

Here are the causal relationships in the generic equipment pressure fault model:

This generic event on process equipment...	Causes this generic event on the domain object that is connected upstream...
High Process Pressure	High Process Pressure
Low Process Pressure	Low Process Pressure
Increasing Process Pressure	Increasing Process Pressure
Decreasing Process Pressure	Decreasing Process Pressure
Process Pressure Change	Process Pressure Change



Built-in Generic Fault Models for Heaters

The heater class defines these generic fault models:

- Draft Pressure Fault Model
- O2 Fault Model
- Tube Skin Temperature and Derived Delta T Fault Models

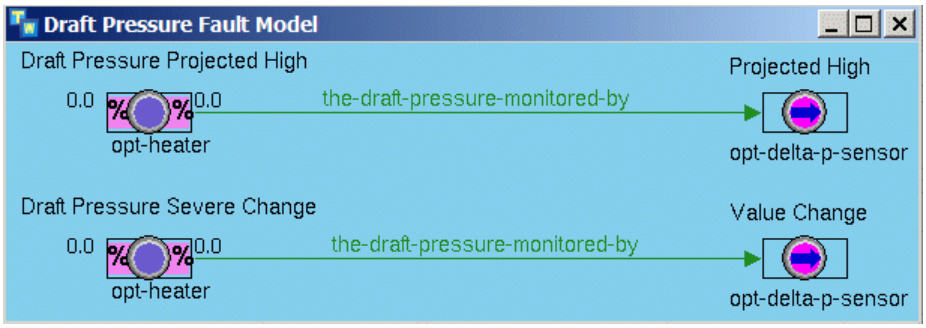
Draft Pressure Fault Model

The built-in generic delta P fault model defines causal relationships between events on a heater that monitors Draft Pressure and events on its related delta P sensor.

For example, the PV Projected High event for a sensor generates the SymCure High event on the sensor when the projected PV exceeds the specified limit. The generic delta P fault model defines a direct causal relationship between the Draft Pressure Projected High event of the heater and the Projected High event on the Draft Pressure related delta P sensor.

Here are the causal relationships in the generic draft pressure fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV Projected High	Projected High	Draft Pressure Projected High
PV Change	Value Change	Draft Pressure Severe Change



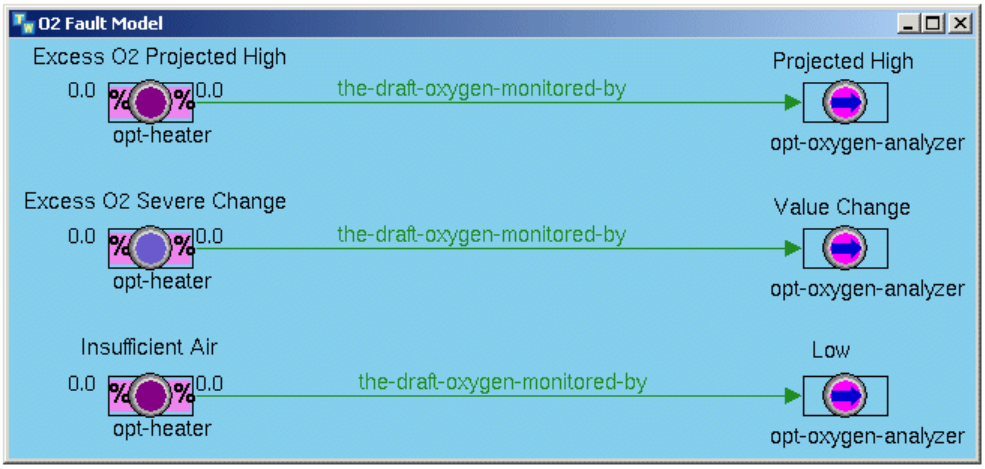
O2 Fault Model

The built-in generic O2 fault model defines causal relationships between events on a heater that monitors Draft Oxygen and events on its related oxygen analyzer.

For example, the PV Projected High event for an oxygen analyzer generates the SymCure Projected High event on the analyzer when the projected PV exceeds the specified limit. The generic O2 fault model defines a direct causal relationship between the Excess O2 Projected High event of the heater and the Projected High event on the Draft Oxygen related oxygen analyzer.

Here are the causal relationships in the generic O2 fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV Projected High	Projected High	Excess O2 Projected High
PV Change	Value Change	Excess O2 Severe Change
PV Low	Low	Insufficient Air



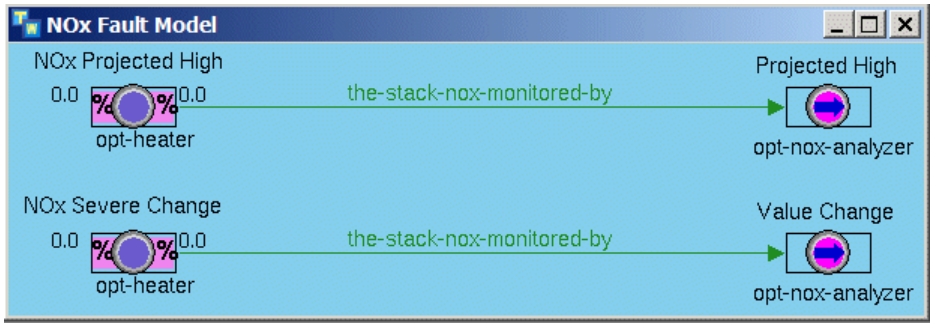
NOx Fault Model

The built-in generic NOx fault model defines causal relationships between events on a heater that monitors Draft NOx and events on its related NOx analyzer.

For example, the PV Projected High event for a NOx analyzer generates the SymCure Projected High event on the analyzer when the projected PV exceeds the specified limit. The generic NOx fault model defines a direct causal relationship between the Excess NOx Projected High event of the heater and the Projected High event on the Draft NOx related oxygen analyzer.

Here are the causal relationships in the generic NOx fault model:

This sensor event...	Generates this SymCure event on the sensor...	Which has this root cause on the related process equipment...
PV Projected High	Projected High	NOx Projected High
PV Change	Value Change	NOx Severe Change

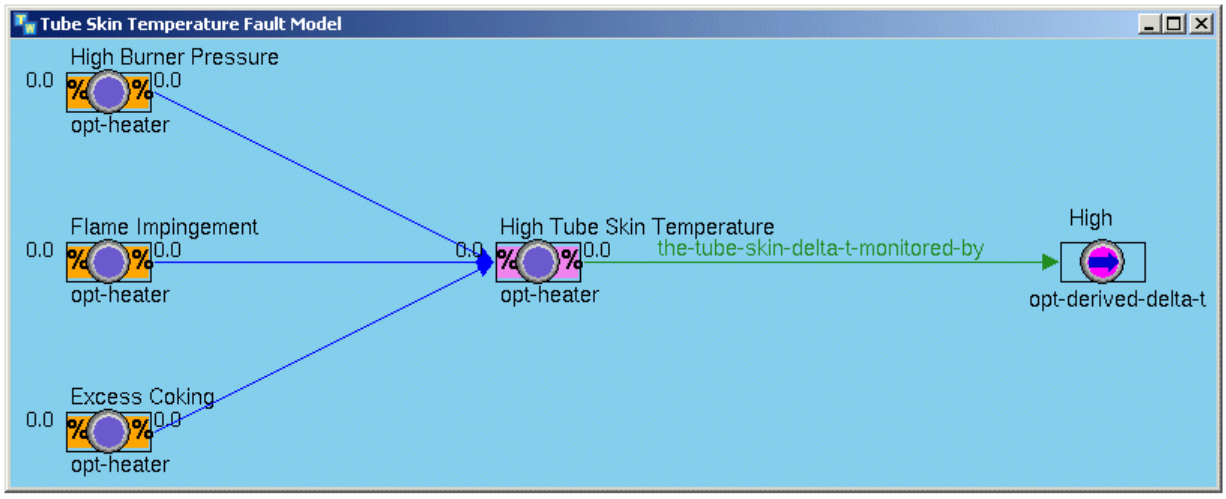


Tube Skin Temperature and Derived Delta T Fault Models

The tube skin temperature fault model defines causal relationships between the High event on the Tube Skin Delta T derived sensor of a heater, a High Tube Skin Temperature event on the heater, and three probable root causes: High Burner Pressure, Flame Impingement, or Excess Coking.

Here are the causal relationships in the generic tube skin temperature fault model:

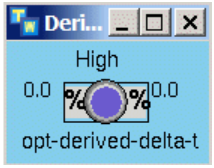
This sensor event...	Generates this SymCure event on the sensor...	Which generates this SymCure event on the heater...	Which has these possible root causes...
PV High	High	High Tube Skin Temperature	High Burner Pressure Flame Impingement Excess Coking



The built-in generic delta T sensor fault model defines a generic event for the Tube Skin Delta T derived sensor on a heater.

Here is the event in the generic delta T fault model:

This event on the Tube Skin Delta T derived sensor...	**Generates this SymCure event on the sensor...**
Tube Skin Delta T	High

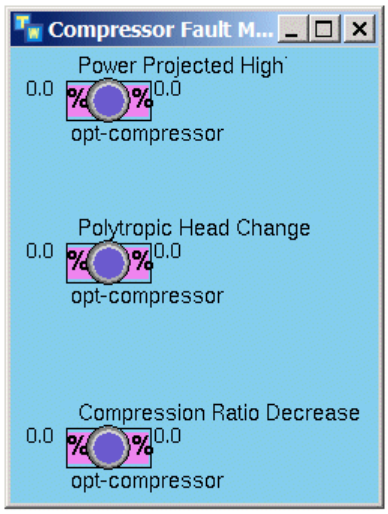


Built-in Generic Fault Models for Compressors

The compressor classes define a generic fault model that provides generic events for each built-in event that can occur on a compressor; it defines no causal relationships.

Here are the events in the generic compressor fault model:

This compressor event...	Generates this SymCure event on the compressor...
Power Projected High	Power Projected High
Polytropic Head Change	Polytropic Head Change
Compression Ratio Decrease	Compression Ratio Decrease



7

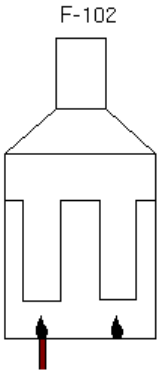
Creating Domain Object Definitions

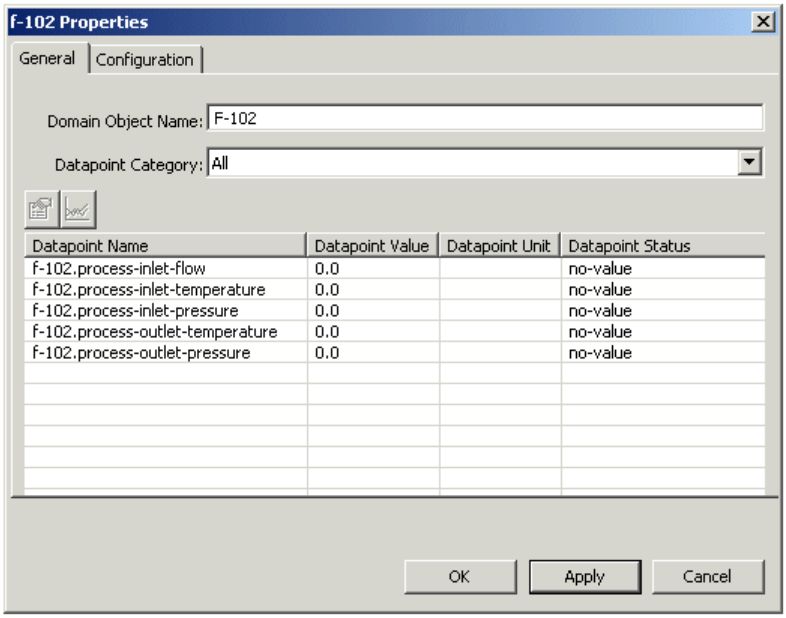
Describes how to create your own domain object definitions, which are based on the built-in Optegrity equipment and instrument definitions.

Introduction

Optegrity defines a number of built-in class definitions, which you can use to create a process map. The built-in class definitions define icons and various internal datapoints relevant for each type of object. You create instances of built-in equipment and instrument classes by using the Process Modeling toolbox.

For example, here is the properties dialog for a heater named F-102, which defines built-in internal datapoints for process inlet flow, temperature, and pressure, and process outlet temperature and pressure. You can configure the source datapoints for each internal datapoint, as needed for your application.





You can also use the built-in classes as the superior class to create your own domain object definitions. To create a domain object definition, you configure the name, superior class, internal datapoints, and icon. You can define the internal datapoints to obtain their values from external variables, or to derive their values from a formula. Internal datapoints can contain any type of data—quantity, float, integer, symbol, boolean, and text.

When you create a class definition, Optegrity allows you to place an icon for the new definition on a user-defined palette.

You access user-defined domain objects through user-defined palettes, which Optegrity creates when you create the definition.

Built-in Domain Object Foundation Classes

Optegrity provides the following foundation classes, upon which all other built-in domain object classes are built:

Class	Description
opt-equipment	Use the equipment classes to create process equipment, such as heat exchange equipment, rotating equipment, vessels, and valves.
opt-sensor	Use the sensor classes to create sensors that measure things like flow, pressure, temperature. These classes define a built-in datapoint for the process value (pv).
opt-controller	Use the controller classes to create controllers such as PID controllers and feed forward controllers.
grtl-datapoint	Use the datapoint classes to create internal datapoints in subclasses of opt-equipment, opt-sensor, and opt-controller. These datapoints obtain their values from external variables that you create from CSV files. You can create datapoints of any value type, such as float, integer, text, symbol, or boolean. You can also create derived datapoints and specify formulas to compute their values.

You can build your own domain object definitions by creating subclasses of the foundation classes. A subclass of a foundation class inherits the icon and the internal datapoints of the built-in definition.

Built-in Process Equipment and Instrument Classes

Optegrity provides a large variety of built-in equipment and instrument classes, upon which you can build your own classes.

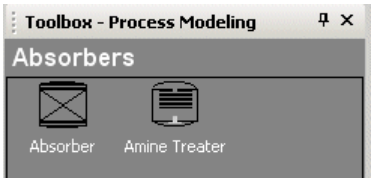
Process Equipment Classes

Here are the equipment classes that you can subclass to define your own process equipment:

- Absorbers
- Boilers
- Compressors
- Distillation Columns
- Equipment Drivers
- Evaporators
- Fin Fans
- General
- Generators
- Heat Exchangers
- Heaters
- Pumps
- Reactors
- Storage Tanks
- Turbines
- Valves
- Vessels

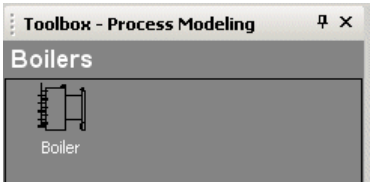
Each equipment class defines various internal datapoints relevant to that class.

Absorbers



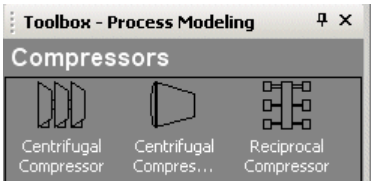
opt-absorber
opt-amine-treater

Boilers



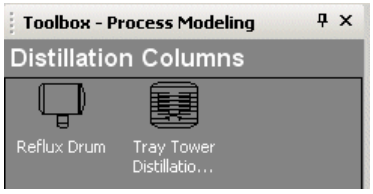
opt-boiler-class

Compressors



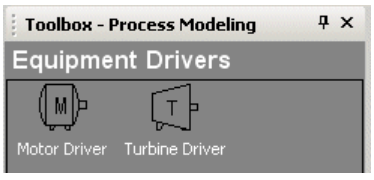
opt-centrifugal-compressor
opt-centrifugal-compressor-stage
opt-reciprocal-compressor

Distillation Columns



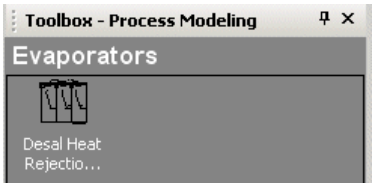
opt-reflux-drum
opt-tray-tower-distillation-column

Equipment Drivers



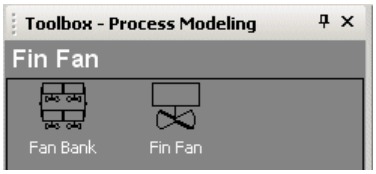
opt-motor-driver
opt-turbine-driver

Evaporators



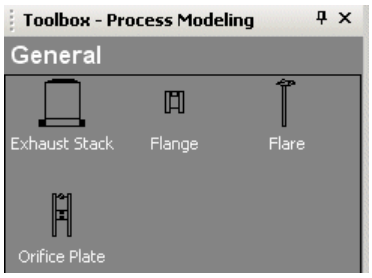
opt-desal-heat-rejection-multi-effect-evaporator

Fin Fans



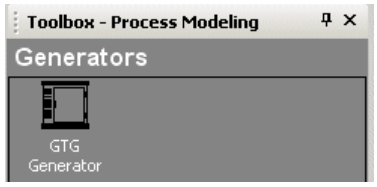
opt-fan-bank
opt-finfan

General



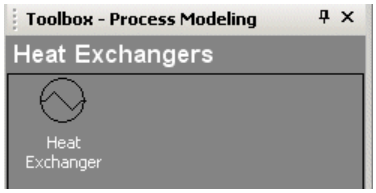
opt-exhaust-stack
opt-flange
opt-flare
opt-orifice-plate

Generators



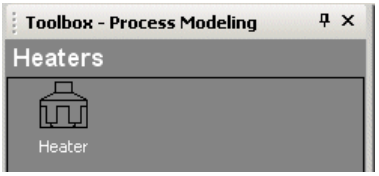
opt-gtg-generator

Heat Exchangers



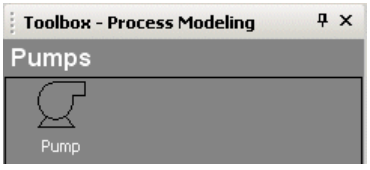
opt-heat-exchanger

Heaters



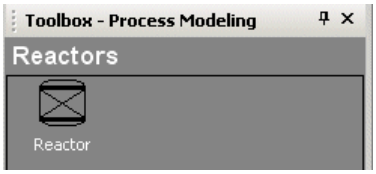
opt-heater

Pumps



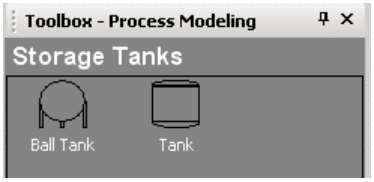
opt-pump

Reactors



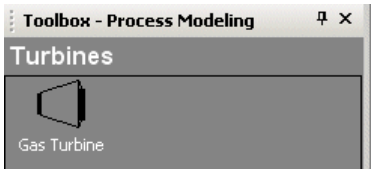
opt-reactor

Storage Tanks



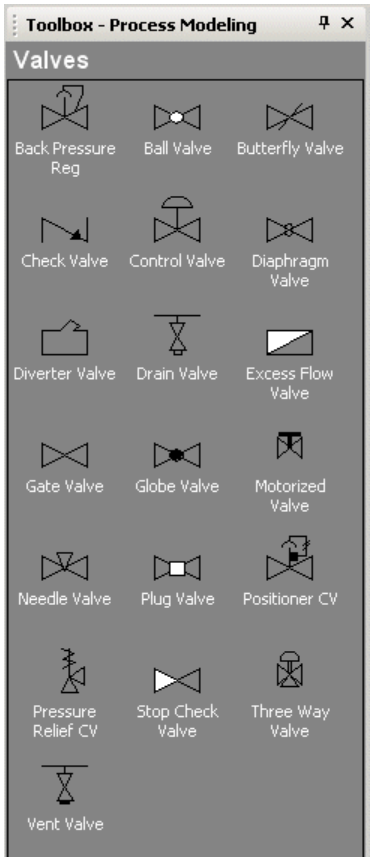
opt-ball-tank
opt-tank

Turbines



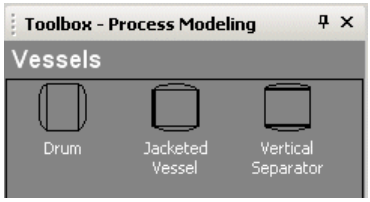
opt-gas-turbine

Valves



opt-back-press-reg-cv
opt-ball-valve
opt-butterfly-valve
opt-check-valve
opt-control-valve
opt-diaphragm-valve
opt-diverter-valve
opt-drain-valve
opt-excess-flow-valve
opt-gate-valve
opt-globe-valve
opt-motorized-valve
opt-needle-valve
opt-plug-valve
opt-positioner-cv
opt-press-relief-cv
opt-stop-check-valve
opt-three-way-valve
opt-vent-valve

Vessels



opt-drum
opt-jacketed-vessel
opt-vertical-separator

Instrument Classes

There are two categories of sensor classes that you can subclass to define your own instruments:

- Sensors and Analyzers
- Controllers

Each class defines internal datapoints appropriate to the instrument.

Internal Datapoints of Instruments

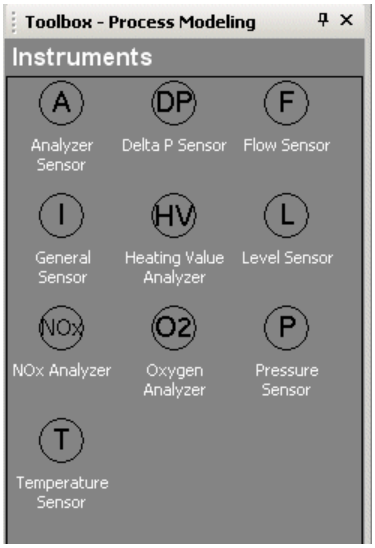
All the sensor and analyzer classes define the following internal datapoint:

Datapoint	Description
pv	A grtl-simple-quantitative-datapoint that defines the process variable value.

All the controller classes define the following internal datapoints:

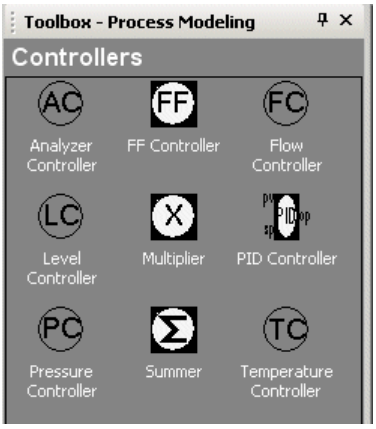
Datapoint	Description
pv	A grtl-simple-quantitative-datapoint that defines the process variable value.
sp	A grtl-simple-quantitative-datapoint that defines the controller setpoint.
op	A grtl-simple-quantitative-datapoint that defines the controller output value.
mode	A grtl-simple-quantitative-datapoint that defines the controller mode value.

Sensors and Analyzers



opt-analyzer-sensor
opt-delta-p-sensor
opt-flow-sensor
opt-heating-value-sensor
opt-level-sensor
opt-nox-analyzer
opt-oxygen-analyzer
opt-pressure-sensor
opt-sensor
opt-temperature-sensor

Controllers



opt-analyzer-controller
opt-ff-controller
opt-flow-controller
opt-level-controller
opt-multiplier
opt-pid-controller
opt-pressure-controller
opt-summer
opt-temperature-controller

Creating Domain Object Definitions

When you create a domain object definition, you must specify its name and its superior class. You may also define internal datapoints for this class, assign it to a toolbox, and edit its icon.

You can define two basic categories of datapoints:

- Simple datapoints, which get their values from external datapoint sources:
  - grtl-simple-quantity-datapoint
  - grtl-simple-float-datapoint
  - grtl-simple-float-datapoint-with-statistics
  - grtl-simple-integer-datapoint
  - grtl-simple-integer-datapoint-with-statistics
  - grtl-simple-symbolic-datapoint
  - grtl-simple-logical-datapoint
  - grtl-simple-text-datapoint
- Derived datapoints, which compute their values based on a formula:
  - grtl-derived-quantity-datapoint
  - grtl-derived-float-datapoint
  - grtl-derived-integer-datapoint
  - grtl-derived-symbolic-datapoint

- grtl-derived-logical-datapoint
- grtl-derived-text-datapoint

For information on these datapoint classes, see the *G2 Developers' Utilities Runtime Library User's Guide*.

Note If you change the name of a class definition in the Domain Object Definition dialog and that class has been configured as the Target Class of a GEDP generic diagram template, a SymCure generic fault model folder, or a SymCure generic event or action, Optegrity automatically updates the Target Class of the affected GEDP and SymCure items.

Creating the Domain Object Definition

To create a domain object definition:

1. Choose Project > Object Models > Instruments and Equipment > Manage to display the Manage dialog, then click the New button.

   The dialog for creating a new domain object definition appears.

2. Edit the Class Name to identify your domain object class definition.

   The class name must be a symbol, with no spaces. We recommend that you prefix class names with your application name, for example, myapp-heater.

Note By default, Optegrity strips the module name from the domain object label in the palette.

---

3. Choose an existing class as the Superior Class by clicking the button to the right of the type-in box.

   A tree view of built-in equipment and instrument classes appears. Expand the tree until you find the desired class.

   The new class inherits its definition from its superior class, including its icon and any internal datapoints. For example, to create a custom heater class definition, the superior class would be opt-heater.

4. Specify a Palette Name on which to place the icon for the new domain object.

   For example, to place the user-defined domain object in the Myapp palette, the Palette Name would be Myapp. You can also choose from a list of existing user-defined palettes.

   The Palette Group is always Process Modeling, which places the user-defined palette in the Process Modeling toolbox.

Note Do not use the "&" character in the palette name. Use the word "and" instead.

5 Specify any text in the Description to provide information about the class definition.

This information serves to document your class definition and is used by Optegrity's search tools to find domain object definitions.

6 To create internal datapoints that are embedded in your domain object definition, enter the name of the internal datapoint in the Datapoints column.

7 To configure the type of datapoint, click in the Type column corresponding with the datapoint and choose the datapoint type from the list.

8 To create additional datapoints, select the left-most column of an existing row, then click the Insert Before or Insert After toolbar button to add rows, then configure the datapoint name and type of each.

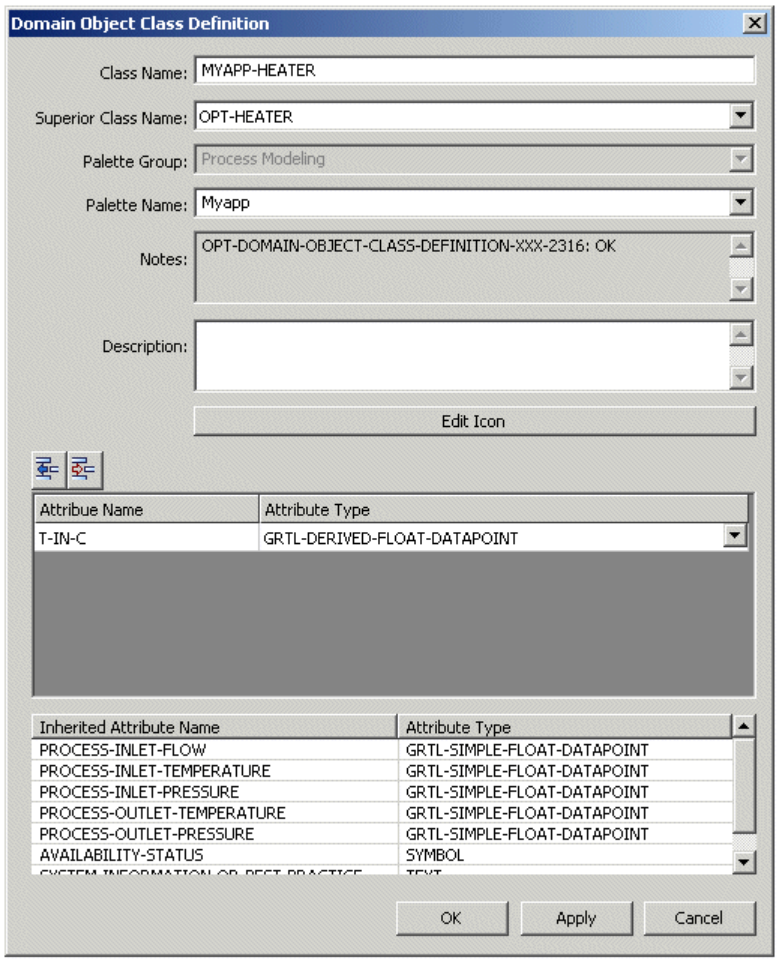
Tip Create the rows first, then configure the name and type of each datapoint.

The toolbar buttons are only enabled when a row is selected. You can also delete the selected row by using the Delete Row button.

9 Click Apply or OK to create the domain object definition.

The Notes field indicates the status of the definition. For instance, the Notes indicate if the name you specify already exists. Once you accept the dialog and open it again, if your newly created class definition is free of errors, the Notes indicate that the domain object definition is OK.

Here is the domain object definition and properties dialog for myapp-heater, which inherits its definition from opt-heater. The domain object appears in the Process Modeling palette group, which puts it in the Process Modeling toolbox, and in the Myapp palette. The class defines an internal datapoint named t-in-c, which is a type of grtl-derived-float-datapoint. The domain object definition also has a number of inherited datapoints, which are a type of grtl-simple-derived-datapoint, by default.



Editing the Domain Object Definition Icon

You can edit the icon associated with a domain object by using the Icon Editor. For more information, see Chapter 39 "The Icon Editor and Icon Management" in the *G2 Reference Manual*.

To edit the domain object definition icon:

1. Choose Project > Object Models > Instruments and Equipment > Manage, select the definition whose icon you want to change, and click the Properties button.

   The properties dialog for the class definition appears.

2. Click the Edit Icon button to display the Icon Editor.

Tip You can also show the class definition object and choose Edit Icon to display the Icon Editor.

Configuring Derived Internal Datapoints

Configuring derived internal datapoints is the same as configuring simple internal datapoints except that instead of configuring the source datapoint, you configure a formula.

To configure a derived internal datapoint:

1. Follow the steps under Configuring Internal Datapoints, except do not configure the Source Datapoint.
2. Configure the Formula to compute the value.

To refer to an internal datapoint in the formula, use dot notation to refer to the internal datapoint, and surround the name with vertical bars (| |), for example, (| F1.PROCESS-INLET-TEMPERATURE| - 32 /1.8).

The formula for derived datapoints supports arithmetic operators only, not functions.

Note Do not refer to the current internal datapoint in the formula; otherwise, an error will occur due to a cyclical reference.

Accessing User-Defined Domain Objects

You access user-defined domain objects from palettes. Optegrity creates toolbox entries automatically when a domain object definition is created. You create and configure a domain object and its internal datapoints when you create the process map.

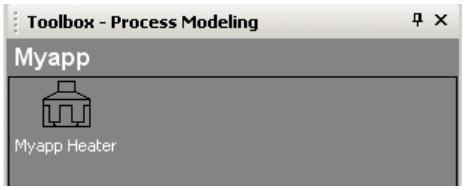
Note To delete a user-defined palette, go into Administrator mode, choose Tools > Inspect, enter show on a workspace-every grtl-palette-group containing *PaletteName*, choose go to original on the object representation, then choose delete on palette group object.

To access user-defined domain objects:

➔ Choose View > Toolbox - Process Modeling and click the button associated with the user-defined palette.

For example, if the Palette Name is Myapp, you would access the user-defined domain object by choosing View > Toolbox - Process Modeling and displaying the Myapp palette.

Here is the Myapp palette in the Process Modeling toolbox, which contains the myapp-heater domain object. Notice that Optegrity automatically strips the module prefix from the label.



To customize the label:

➔ Add an entry such as the following to the *resources-english.txt* file in the *g2i\kbs* directory:

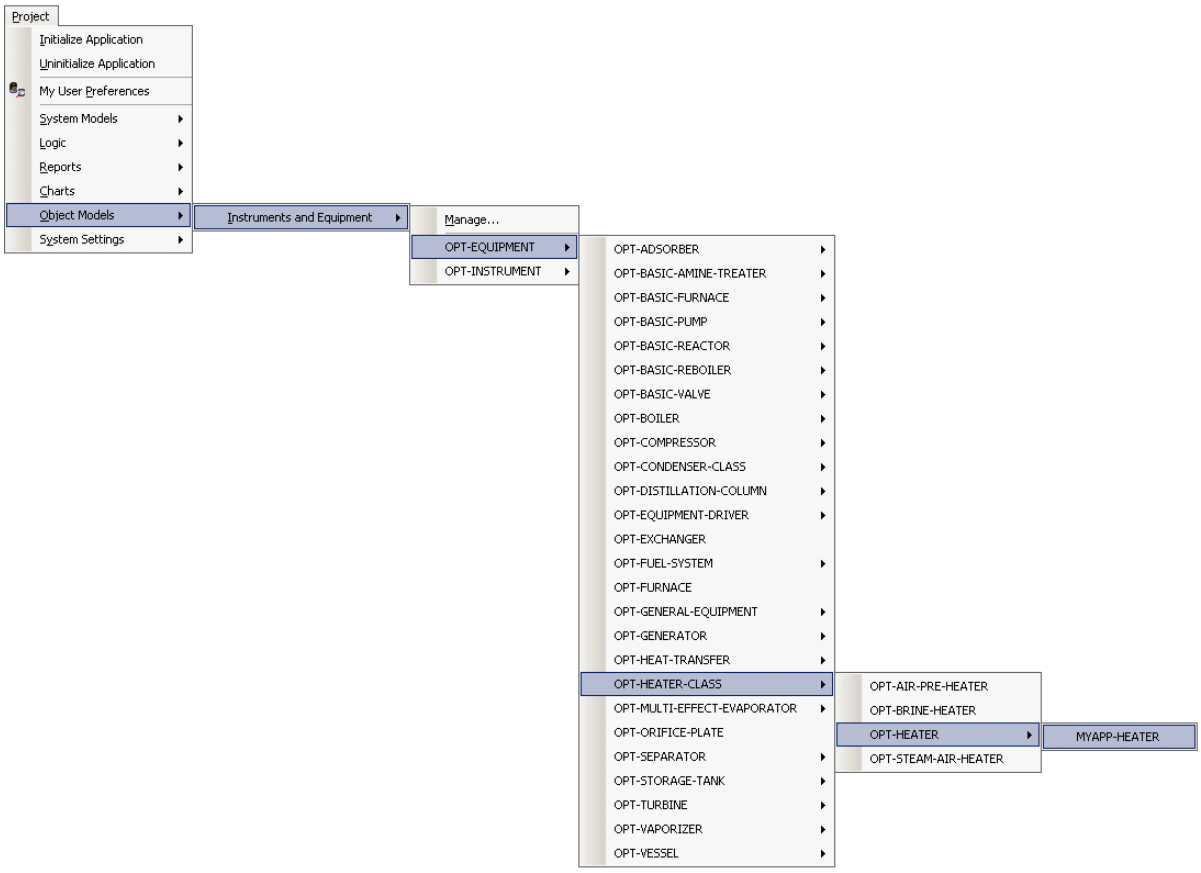
```
PALETTE.ITEM.MYAPP-HEATER.LABEL, "Custom Heater"
```

Managing Domain Object Definitions

To manage domain object definitions:

1 Choose Project > Object Models > Equipment and Instruments > Manage.

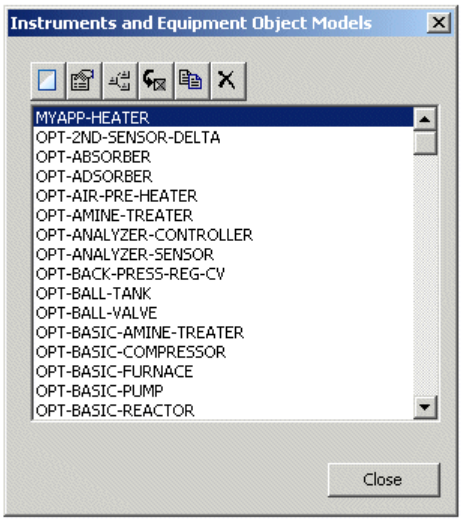
All user-defined domain object classes appear in the submenu, for example:



2 To configure the properties of a domain object definition, choose one from the appropriate Object Models submenu.

3 To display a dialog for managing all domain objects, choose Manage.

Here is the Object Models Manage dialog:



For information on using this dialog and the Project menu to manage domain objects, see Using the Project Menu.

Part III

Data Sources

Chapter 8: Configuring Network Interfaces

Describes how to create and configure network interfaces for communicating with external systems through a bridge.

Chapter 9: Configuring External Datapoints

Describes how to configure external datapoints using a CSV file.

Chapter 10: Converting Engineering Units

Describes how to create and configure engineering unit conversions.

Chapter 11: Configuring Logging

Describes how to configure logging for internal and external datapoints.

Chapter 12: Replaying Data

Describes how to replay internal and external datapoint values from continuous and differential CSV files.

Chapter 13: Simulating Datapoint Values

Describes how to simulate values for internal and external datapoints.

8

Configuring Network Interfaces

Describes how to create and configure network interfaces for communicating with external systems through a bridge.

Introduction

To communicate with external DCS systems, you must create and configure a DCS interface object. Optegrity provides two types of interface objects for communicating with DCS systems over a network:

Name	Description
OPC Interface	Communicates with external OPC systems, using the G2 OPCLink bridge.
PI Interface	Communicates with external PI systems, using the G2-PI Bridge.

You associate a DCS interface object with an External Datapoints container, which defines the external datapoints whose values you want to monitor and manage.

The interface is responsible for communicating with the external system and providing data to each external datapoint.

Optegrity also provides three additional types of interface objects for communicating with these external systems:

Name	Description
Database Interface	Communicates with external Oracle, Sybase, and ODBC databases, using the G2-Oracle Bridge, G2-Sybase Bridge, and G2-ODBC Bridge. You connect to databases for message logging. Optegrity provides built-in support for logging messages to databases.
JMail Interface	Communicates with external Java Mail systems, using the G2 JMail Bridge. You connect to JMail systems to send and receive e-mail messages.
JMS Interface	Communicates with external Java Messaging Service systems, using G2 JMSLink. You connect to JMS systems to send and receive text and XML messages.

You can interact with network interfaces through the Interfaces submenu of the System Settings menu.

Note To create and configure network interfaces, you must be in Developer mode.

Once you configure the network interface, you must establish a connection to the bridge process. To do this, you must first start the bridge process. If the interface connection goes down, Optegrity automatically attempts to restart the connection.

If you have a large number of external DCS datapoints, you can create multiple interfaces and interface pools to distribute processing across multiple connections. One reason you might want to distribute processing is if you need to poll external datapoints at different intervals.

The G2 Data Source Management (GDSM) module is responsible for communicating with external DCS systems.

For information about associating interface objects with external datapoints, see Configuring External Datapoints.

For information about message logging to a database, see Logging Messages to a Database.

For information on creating and interacting with interfaces, using the Project menu, see Using the Project Menu.

Creating and Connecting Network Interfaces

To create a network interface, you configure the interface name, and the host and the port of the machine that is running your bridge process. The default host is localhost and the default port is 22041.

Depending on the type of network interface, you must also configure additional information for connecting to the bridge process. For example, when connecting to a database, you must configure the type of database, and the user name and password for logging into the database.

Once you have configured the interface, you can connect it to the bridge process.

You determine the status of the connection in the properties dialog for the interface. The color of the arrow in the network interface object also indicates its connection status. The status can be any of these colors and values:

This color...	Has this connection status...	Which indicates...
Red	-2	An error has occurred.
Yellow	-1	A timeout has occurred.
Khaki	0	The connection is inactive.
Blue	1	The connection is initializing.
Green	2	The connection is active.

For information on configuring additional features such as time outs, see Advanced Features.

To create and connect the network interface:

1. Choose Start > Programs > Gensym G2 2011 > Bridges, then choose the bridge to which you want to connect, for example, G2 OPC Client Bridge or G2 PI Bridge.

   The bridge process starts in a command window, and an icon for the bridge process appears in the system tray. Note the host and port of the bridge process.

2. Switch to Developer mode.

   For details, see Switching User Modes.

3. Choose Project > System Settings > Interfaces, then choose the type of interface you want to create, depending on the external system, and choose Manage to display the Manage dialog for the specified type of interface.

4. Click the New button to display the properties dialog for configuring the network interface.

5 Configure the Interface Name to be a unique name, which is system-generated, by default.

The name must be a symbol without spaces.

Tip We recommend that you prefix the name with your application name, for example, myapp-interface.

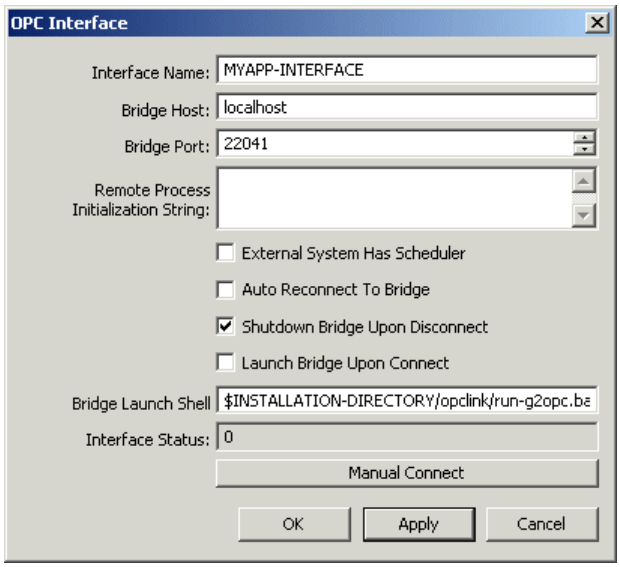
---

6 Configure the Bridge Host and Bridge Port of your bridge process.

7 Connect the interface to the bridge process by using one of these techniques:

- Click Manual Connect in the properties dialog.
- Click the Connect button in the Manage dialog for the particular type of network interface.
- Choose Connect to Bridge on the interface object.

Here is the properties dialog for an OPC Interface named myapp-interface:



For information on configuring the other options, see Advanced Features.

For information on configuring the interface-specific properties for Database (SQL), JMail (SMTP), and JMS Interfaces, see the relevant guides for the associated bridges as described in Advanced Features.

To disconnect from the bridge process:

➔ Click the Disconnect button in the Manage dialog for the particular type of interface.

or

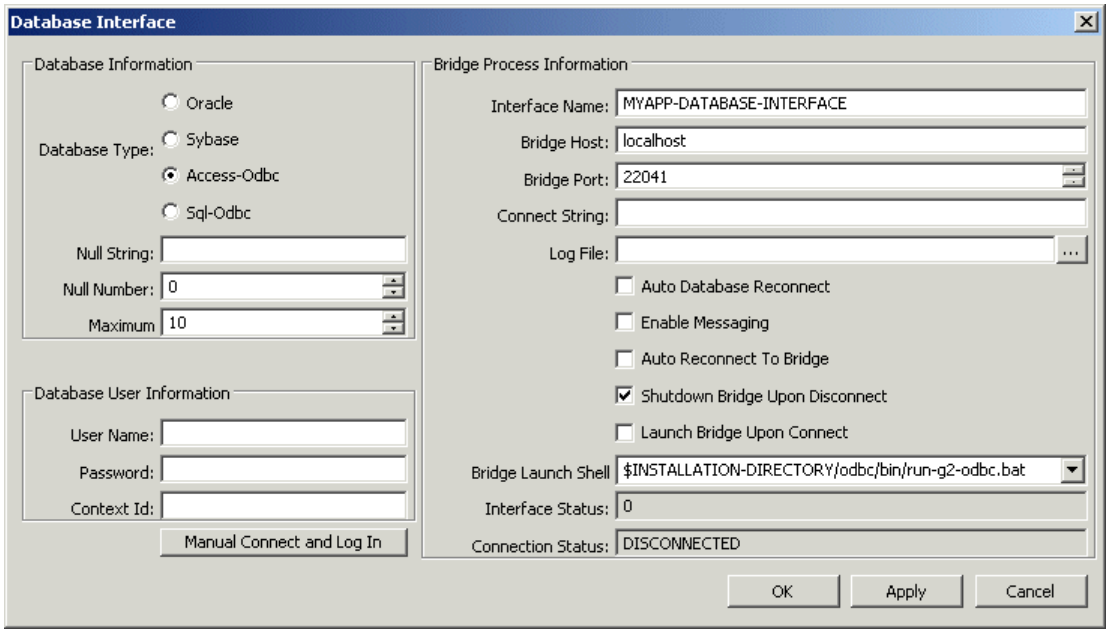
➔ Choose Disconnect from Bridge on the interface object.

Advanced Features

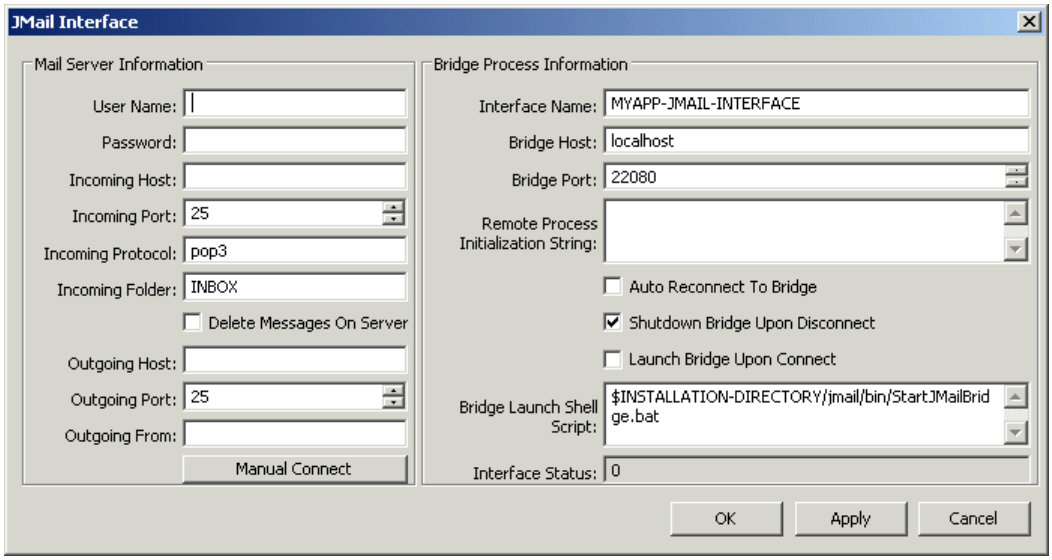
Network interfaces communicate with external systems, using various bridges. These bridges support additional features for communicating with external systems, such as configuring the remote process initialization string, polling the external system, rather than receiving data when it changes, and creating multiple interface object connections.

You must also configure interface-specific properties defined by the Database (SQL), JMail (SMTP), JMS, and HTTP interfaces.

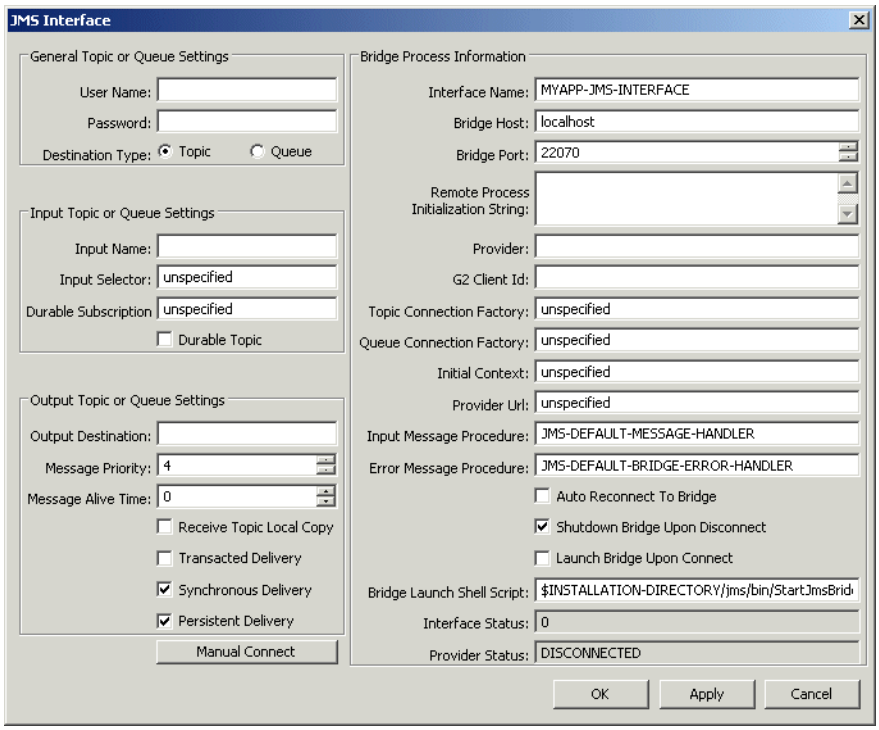
Here is the properties dialog for a database interface:



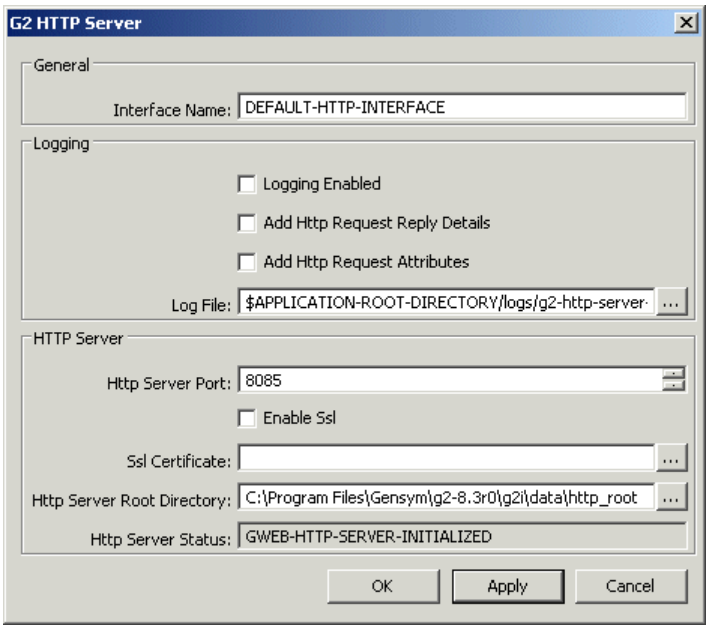
Here is the properties dialog for the JMail interface:



Here is the properties dialog for the JMS interface:



Here is the properties dialog for the HTTP interface:



For information on configuring user preferences for sending messages by email, see Delivering Messages by Email.

For information on configuring the advanced features, as well as the Database, JMail, JMS, and HTTP Interfaces, see the appropriate guide, depending on the type of network interface:

This interface...	Uses this bridge process...	Described in...
OPC Interface	G2 OPC Client	*G2-OPC Client Bridge User's Guide*
PI Interface	G2-PI Bridge	*G2-PI Bridge User's Guide*
Database Interface	G2-Oracle Bridge G2-Sybase Bridge G2-ODBC Bridge	*G2 Database Bridge User's Guide* *G2-Oracle Bridge Release Notes* *G2-Sybase Bridge Release Notes* *G2-ODBC Bridge Release Notes*
JMail Interface	G2 JMail Bridge	*G2 JMail Bridge User's Guide*
JMS Interface	G2 JMSLink	*G2 JMSLink User's Guide*
HTTP Interface	N/A	*G2 Web User's Guide*

Using Interface Pools

For scalability, Optegrity supports pools of interfaces to communicate with databases or other remote programs. The pool uses parallel bridges and communication channels to interact with external databases and improve performance. This means that Optegrity can perform multiple database queries or updates in parallel and use multiple instances of remote programs performing processing-intensive tasks. For each request, Optegrity selects an available or the least-used network object from the pool at run time.

Optegrity provides several types of interface pools, which are configured to create network objects that connect to a database, JMail bridge, JMS Message server, or another Optegrity process.

To configure multiple communication channels, you configure the initial interface count to be the desired number of connections. When the model resets, Optegrity automatically creates and configures the network objects to connect to the remote process or bridge. Alternatively, you can configure the network communication pool manually, and when the model resets, Optegrity launches the remote programs or bridges.

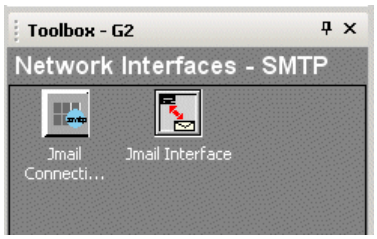
If you use the default procedures to launch the remote process or bridge, the command line to launch it must accept one argument, which is the TCP/IP port name to use for listening for connections.

To configure an interface pool:

1 Choose View > Toolbox - G2 and choose the desired Network Interfaces palette.

Alternatively, choose Project > System Settings > Interface Pools, choose the desired type of interface, and choose Manage to create a new interface pool.

For example, here is the Network Interfaces - SMTP palette:



2 Create a network interface pool from the palette, display its properties dialog, the interface pool and configure these attribute on the General tab:

Attribute	Description
Label	Any label for the object.
Comments	Any comment for the object.
User Name	User name to connect to interfaces.
Password	Password used to connect to interfaces.
Initial Network Interface Count	The default number of interfaces created in the pool at reset time.
Network Connection Timeout	The timeout of the network connection.
Enable Initialization During Reset	Enables and disables the initialization of the pool at reset time of the model.

3 Click the Network Interface tab and configure these attributes:

Attribute	Description
Interface Default Host	The host where the interface to connect to is running.
Interface Base Port Number	The TCP/IP port to which the remote interfaces connect. For every additional interface automatically added at reset time, this port number is incremented by 1, except for G2-to-G2 interfaces.
Network Interface Timeout	The timeout for the network interface.
Network Interface Initialization String	The initialization string for the network interface.
Remote Process Launch Arguments	Arguments to the Bridge Process Launch Command.

Attribute	Description
Bridge Process Launch Command	The command line to launch the bridge or process. Optegrity expects the command line to take one argument, which is the TCP/IP port number the bridge should use to listen for G2 connections. Standard G2 bridges such as database bridges offered by Gensym comply with this requirement.
Bridge Process Launch Procedure	The name of a procedure to launch a bridge or process. This procedure can be a executable (*.exe*) or a batch file. The procedure should return the process ID of the process it launched or –1 if it failed. The default procedure is gdsm-launch-bridge-process.
Bridge Process Kill Procedure	The name of a procedure to kill the bridge to which an interface is connected. The default procedure is gdsm-kill-bridge-process.
Auto Connect to Bridge	Whether to automatically connect to the bridge when required.
Shutdown Bridge on Disconnect	Whether to automatically shutdown the bridge when disconnecting.
Launch Bridge Upon Connect	Whether to automatically launch the bridge when attempting a connection.

4 For Database Interface Pools, click the Database tab and configure these attributes:

Attribute	Description
Database Connect String	The database name to connect to or the ODBC DSN to use to connect to the database.
Maximum Number of Cursors	The maximum number of cursors to manage in each database bridge. The default value is 100.
Bind Variable Prefix	The prefix used by the database to tag arguments as being bind variables. The default value is colon (:).

For details, see the *G2 Database Bridge User's Guide*.

5 For JMail Interface Pools, click the JMail tab and configure these attributes:

Attribute	Description
Incoming Host	The name of the host computer used for incoming email.
Incoming Port	The port number of the host used for incomign email.
Incoming Protocol	The protocol used for incoming email.
Incoming Folder	The name of the folder for incoming email.
Delete Messages on Server	Whether to delete messages on the server when sent.
Outgoing Host	The name of the host computer used for outgoing email.
Outgoing Port	The port number of the outgoing host.
Outgoing From	The email address to use as the From address when the email message is sent.

For details, see the *G2 JMail Bridge User's Guide*.

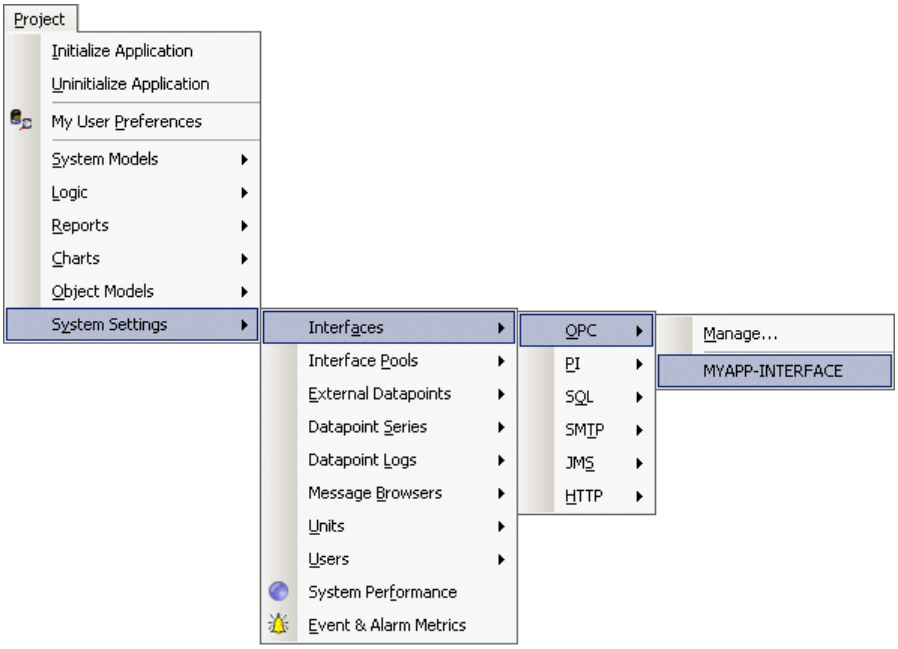
6 For JMS Interface Pools, click the JMS tab and configure the attribute.

For details, see the *G2 JMSLink User's Guide*.

Managing Network Interfaces

To manage network interfaces:

1 Choose Project > System Settings > Interfaces, then choose the network interface type to display its submenu, for example:

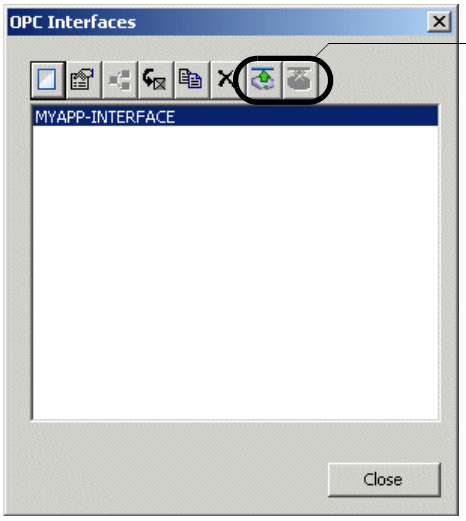


2 To configure the properties of an interface object, choose one from the appropriate submenu in the Interfaces submenu.

3 To display a dialog for managing all interface objects, choose Manage.

Note To enable the buttons for interacting with network interfaces through the Manage dialog, you must be in Developer mode.

Here is the OPC Interfaces Manage dialog in Developer mode:



For information on using this dialog and the Project menu to manage process maps, see Using the Project Menu.

For information on using the buttons specific to network interfaces, see Performing Specific Operations.

9

Configuring External Datapoints

Describes how to configure external datapoints using a CSV file.

Introduction

One of the most tedious tasks in building an intelligent fault management application is creating the hundreds and maybe thousands of G2 variables corresponding to the monitored tag variables in your external DCS system.

An important feature of Optegrity is the ability to create external variables for each DCS tag variable—automatically—using CSV files. You simply export to a file the OPC or PI configuration data for the tag variables you want to manage, specify a name for each external variable, and specify the interface type and datapoint type. Optegrity reads the file and automatically creates variables of the correct type.

When creating external datapoints from a CSV file, you also have the option of automatically linking the external datapoints to internal datapoints in a process map. To do this, for each external datapoint in the CSV file, you provide the name of an internal datapoint that will get its data from the external datapoint. You

express the internal datapoint name, using **dot notation**, which concatenates the domain object name with its internal datapoint, for example, f-1001.pv where f-1001 is an object and pv is an attribute of f-1001. When Optegrity reads the file, it automatically links the external and internal datapoints.

When creating the datapoints from a file, you can configure limits, targets, and deviations to perform data validation of external datapoint values. When a datapoint value is out of range, a data validation message occurs in the Messages browser.

You can configure engineering units for external datapoints in the CSV file. For details, see Configuring External Datapoint Units in the CSV File.

Once the datapoints have been created from the CSV file, you can change them through the external datapoints properties dialog, as needed.

In addition to creating external datapoints from a CSV file, you can create external datapoints from a palette.

You can also configure logging for external datapoints. For details, see Configuring Datapoints for Logging.

You create external datapoints in an External Datapoints container:

Container	Description
External Datapoints	Allows you to create external datapoints from DCS tag variables, using a CSV file.

Creating External Datapoint Configuration Files

The CSV file that configures external datapoints contains the following information. The order of the information is preconfigured. Only some of the information is required; the rest is optional.

- Datapoint Name — A unique name for the external datapoint.
- Datapoint Server Information — Information about how the external datapoint is configured in the server, which includes:
  - Default Update Interval — How often to update data from the DCS system.
  - Datapoint Tag Type — The type of DCS tag variable the external datapoint represents.

  - Datapoint Type — The value type for the external datapoint. The options depend on the type of external system.
  - Datapoint Units — The engineering unit that the external datapoint uses. You can specify any of the built-in engineering unit or a user-defined synonym.
  - Related Process Map Datapoint Names — The name of an internal datapoint that gets its data from the external datapoint. If you do not want to associate external datapoints with internal datapoints when you create the external datapoints, you can leave this slot empty.
- Data validation limits, targets, and rates, which perform data validation on the external datapoint values. Data validation is optional.
- DCS configuration information for the external OPC or PI system that provides the external data.

You can configure the optional information in the external datapoints configuration CSV file, or you can configure it manually for individual internal datapoints in the process map. For more information, see Configuring Domain Objects.

Configuring the External Datapoint Name

You use the external datapoint name as the source datapoint of an internal datapoint in a process map. The external datapoint name must be a symbol, with no spaces.

To configure the external datapoint name:

➔ In the CSV file that configures external datapoints, configure the Datapoint Name column, the first column, to be a unique symbol.

Configuring the Default Update Interval

By default, external datapoints update their data whenever the value in the DCS system changes. To update external datapoints at regular intervals, you can configure the default update interval. The value is any time interval, such as 1 hour and 30 minutes or 1 minute.

Configuring the Datapoint Tag Type

Each external datapoint represents a DCS tag variable for a sensor or controller. For sensors, the datapoint tag type is pv, which represents the process value. Controllers can have the following datapoint tag types: pv, sp, which represents the controller setpoint, op, which represents the controller output, and mode, which represents the controller mode.

By configuring the type of datapoint tag type, you see only external datapoints of the required type when manually configuring the source datapoint of an internal datapoint.

Configuring the Datapoint Type

The external datapoint must define a data type, which depends on the external DCS system you are using:

- For OPC variables, the options are:
  - float
  - integer
  - logical
  - text
- For PI variables, the options are:
  - real
  - integer
  - digital

When you create the external datapoints, Optegrity creates a datapoint of the proper class, based on the specified data type.

To configure the external datapoint type:

➔ In the CSV file that configures external datapoints, configure the Datapoint Type column to be one of the types listed above, depending on your DCS system.

Configuring the Datapoint Units

For information on configuring the datapoint units, see Configuring External Datapoint Units in the CSV File.

Configuring the Related Internal Datapoint

If you know ahead of time which internal datapoints should obtain their values from external datapoints, you can configure this information in the external datapoints configuration file. This step requires that you have already created and configured your process map.

If you do not have this information at the time that you create the external datapoints configuration file, you can configure each internal datapoint manually to refer to a particular external datapoint as the source datapoint.

Of course, if you have this information ahead of time, you will save time configuring this information from the CSV file. On the other hand, you might be creating your application in stages, adding domain objects to the process map incrementally as you manage more and more of your external process. You must decide which approach works best for your application. You can also use a combination of both approaches by automatically configuring some internal datapoints and adding others later.

To relate external datapoints to internal datapoints:

➔ In the CSV file that configures external datapoints, configure the Related Process Map Datapoint Names column to be the name of the internal datapoint that gets its data from the particular external datapoint.

When referring to the internal datapoint name, you must use dot notation, which concatenates the domain object name and its internal datapoint, using a period as a separator. For example, to relate the external datapoint named f-1001-external to the pv datapoint for the f-1001 flow sensor, you would use f-1001.pv.

Configuring Data Validation

In the CSV file that configures external OPC float and PI real datapoints, you can set up limits, targets, and rates for validating external data as it arrives. You can perform three types of data validation:

This type of data validation...	Validates external datapoint values based on...
Detected limit	Minimum and maximum limits.
Detected target	Minimum and maximum targets, based on a value.
Detected rate	Minimum and maximum rates, based on a time interval.

When an external data value does not conform to the specified data validation limits, targets, and/or rates, Optegrity generates an operator message, which indicates the reason the value is invalid. You can view these messages in the

Message Browser. For more information, see Interacting with Operator Messages.

Just as you can relate external and internal datapoints manually for individual datapoints in a process map, you can configure data validation for internal datapoints manually.

You cannot configure data validation for integer, logical, text, or digital datapoint types.

To configure data validation:

➔ In the CSV file that configures external datapoints, configure some or all of these columns for a particular external datapoint, where the Enabled options are true or false:

Column	Value
Min-Max Enabled	true or false
Min-Max High-High	High-high limit value
Min-Max High	High limit value
Min-Max Low-Low	Low-low limit value
Min-Max Low	Low limit value
Target Enabled	true or false
Target High-High	High-high target value
Target High	High target value
Target Low-Low	Low-low target value
Target Low	Low target value
Target Value	The target value to which the external datapoint value is compared, relative to the high and low targets.
Rate Enabled	true or false
Rate High-High	High-high rate value
Rate High	High rate value
Rate Low-Low	Low-low rate value
Rate Low	Low rate value
Rate Interval	The rate at which the time interval between external datapoints is compared.

Configuring the DCS Datapoint Data

Depending on the type of DCS system you are using, you need to identify the actual OPC or PI tag variable for which you are creating an external variable. The configuration information varies, depending on your DCS system.

To configure the DCS datapoint data:

➔ Use the last columns in the CSV file to configure the datapoint data for your DCS system.

For example, to configure an OPC tag variable, you would configure these values:

- OPC Item ID
- OPC Access Path

For both OPC and PI tag variables, you can also configure these values in the second-to-last and last columns of the CSV file, respectively:

- High Process Limit — The high limit for external datapoint values.
- Low Process Limit — The low limit for external datapoint values.

Summary of the CSV File Format

The CSV file that configures external datapoints defines each external datapoint in its own row and the configuration information in each column. The following table defines the order of columns, from left to right, where the separators indicate groups of related configuration information.

Column	Description
Datapoint	
Datapoint Name	The name of the external datapoint.
Datapoint Server Configuration	
Default Update Interval	How often to update data from the DCS system. The value is any time interval, such as 1 minute or 1 hour and 30 minutes.

Column	Description
Datapoint Tag Type	The type of datapoint the external datapoint represents. The options are: pv, sp, op, or mode. Pv represents the process value for a sensor or controller; sp represents the setpoint of a controller; op represents the controller output; and mode represents the controller mode. By configuring the type of datapoint tag, Optegrity shows you only datapoints of the required type when manually configuring the source datapoint of an internal datapoint.
Datapoint Type	The value type for the external datapoint. The options depend on the type of external system. For OPC variables, the options are: • float • integer • logical • text For PI variables, the options are: • real • integer • digital
Datapoint Units	The engineering unit that the external datapoint uses, for example, C or deg C.
Related Process Map Datapoint Names	The name of the internal datapoint that gets its value from the external datapoint.
Data Validation	
Min-Max Enabled High-High High Low-Low Low	Detected target settings for data validation.

Column	Description
Target Enabled High-High High Low-Low Low Value	Detected target settings for data validation.
Rate Enabled High-High High Low-Low Low Interval	Detected deviation settings for data validation.
Animation	
Animation Enabled	Animation Enabled is currently not supported.
Animation Object Name	Animation Object Name is currently not supported.
DCS Configuration	
OPC Item ID	The ID of the OPC tag variable from which the external variable gets its data.
OPC Access Path	The access path to the OPC tag variable in the external OPC system.
High Process Limit	The high limit for external data. Values above this limit are rejected.
Low Process Limit	The low limit for external data. Values below this limit are rejected.

Using an Existing CSV File as a Template

Optegrity provides sample CSV files that you can use as templates for configuring external datapoints. These files provide configuration data for the heater tutorial that ships with Optegrity.

The data file that the tutorial uses provides configuration data for external OPC tag variables.

For information on how the sample KB uses these files, see the *Optegrity Furnace Tutorial*.

To use an existing CSV file as a template:

➔ Open the following file, located in the `\optegrity\data` directory of your Optegrity installation directory:

`f102-external-datapoints-configuration.csv`

Creating External Datapoints from a CSV File

Once you have created the CSV file for configuring external datapoints, you create an External Datapoints container that refers to this file. The container also refers to the OPC or PI Interface that it will use to obtain data through the bridge.

You can configure the external datapoints to propagate data when their values change or based on a schedule.

Once you have created the external datapoint configuration, you can create the external variables.

Note In the sections that follow, the menu choices assume that enable-menus-and-toolbars-upon-startup is enabled in the grtl-module-settings object. For more information, see the *G2 Run-Time Library User's Guide*.

Creating the External Datapoints Container

By default, the value of an external datapoint is propagated to its associated internal datapoint whenever the value changes. In some cases, the value of an external datapoint must be propagated to its internal datapoint at regular time intervals, based on a schedule. For example, this situation arises when using a bridge in synchronous mode.

You can configure External Datapoints containers to update, based on event-driven evaluation or based on a schedule. By default, external datapoints update their values based on event-driven evaluation, that is, when the external datapoint value changes. When configuring datapoints to update based on a schedule, you configure the update interval.

To create the External Datapoints container:

1 Choose Project > System Settings > External Datapoints > Manage and click the New button.

The properties dialog for configuring external datapoints appears.

2 Configure a unique Name, which is system-generated, by default.

Tip We recommend that you prefix the name with your application name, for example, Myapp External Datapoints.

---

3 Click the arrow to configure the Interface Name to refer to an existing network interface to use for obtaining external data.

4 Configure Value Propagation to determine how you want datapoint values to propagate, based on event-driven evaluation or based on a schedule.

The default value is Event Driven.

5 If you choose Schedule Driven, configure Update Period to be the number of seconds between updates.

6 Accept the dialog.

To configure the CSV filename, you must be in developer mode.

To configure the CSV filename:

1 Switch to Developer mode.

For details, see Switching User Modes.

2 Display the properties dialog for the External Datapoints container you want to configure.

3 Configure the Filename to be a complete path name to the CSV file that contains the external datapoints configuration data.

We recommend that you use the *data* subdirectory of the *optegrity* directory for CSV files.

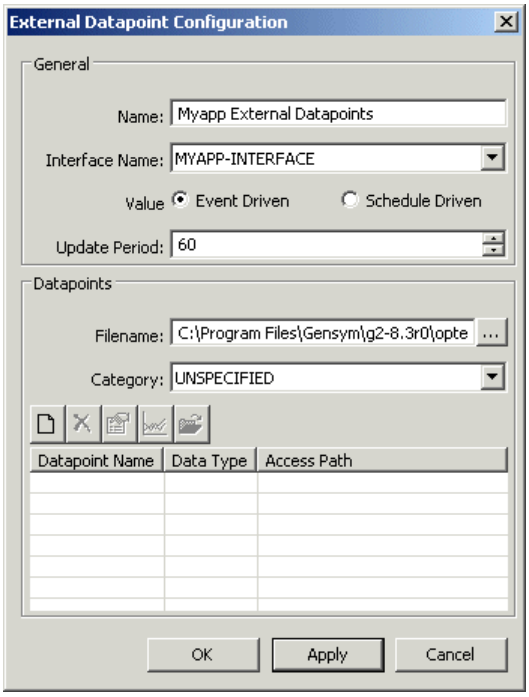
For information on the format of the CSV file, see Creating External Datapoint Configuration Files.

4 Click OK.

You can create the external datapoints directly from the properties dialog by clicking the Create External Datapoints button. For details, see Creating and Configuring External Datapoints.

Once you have created the external datapoints, the properties dialog displays them in a spreadsheet. You can filter the external datapoints, based on the category.

Here is the properties dialog and associated External Datapoints container named Myapp External Datapoints, which obtains it data through the network interface named myapp-interface and from the specified CSV file. It propagates datapoint values once every 30 seconds, based on a schedule. This figure shows the dialog in Developer mode.



Creating and Configuring External Datapoints

You create the external datapoints from the External Datapoints container properties dialog.

Optegrity reads the data from the CSV file and creates one external datapoint for each row in the file. The external datapoints appear on the External Datapoints container detail. The external datapoints are automatically related to the specified internal datapoint as specified in the CSV file.

Each external datapoint has an associated dialog, which contains all the information obtained from the CSV file. OPC float and PI real datapoints have two tabs, the General tab and the Alarm Limit Detection tab. The other datapoint types have only the General tab.

Once the external datapoints have been created, you can configure various additional information, such as the category, description, value translation

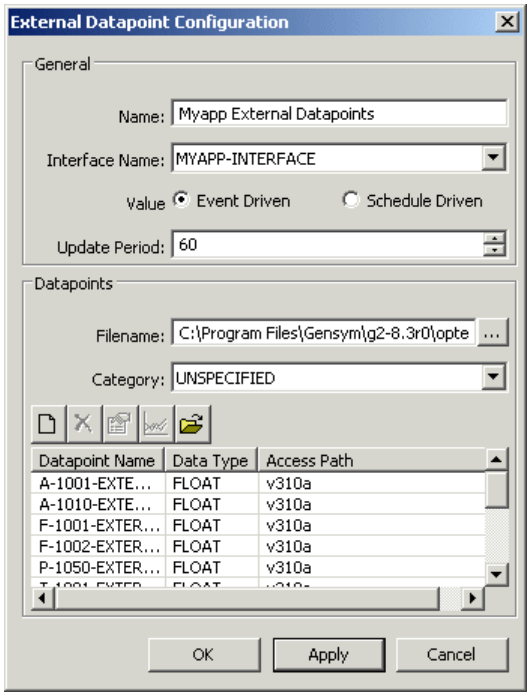
procedure, persistence, and data logging. You can also override values in the properties dialog.

Note Overriding values in the external datapoint properties dialog does not update the corresponding CSV file.

To create and configure the external datapoints:

1 Display the properties dialog for the External Datapoints container and click the Create from File button: 

All external datapoints specified in the CSV file appear in the spreadsheet at the bottom of the properties dialog. For example, here is the result of creating external datapoints from the *f102-external-datapoints-configuration.csv* file:



2 Select a row in the spreadsheet and click the Properties button to display the properties dialog for the external datapoint.

3 Configure the Category to filter external datapoints in the external datapoints configuration dialog.

You can define a category for external datapoints, then filter them in the external datapoint configuration properties dialog, based on the category. For example, you might want to specify the equipment subsystem as a category to show only those datapoints related to that subsystem, such as the fuel system.

You can also categorize the datapoints, based on the type of sensor, such as flow or temperature.

4 Configure the Description to provide a textual description of the external variable.

5 To translate the value, configure the Value Translation to be the name of an existing procedure.

For details, see Translating External Datapoint Values.

6 To filter values from the external system, configure the High Limit and Low Limit, and enable or disable the High Limit Check or Low Limit Check options, as needed.

You can also configure these values in the CSV file. When process limits are enabled, Optegrity rejects data values that are above the high limit or below the low limit.

7 Configure the Min Persistence Value and Min Persistence Units to determine how long to keep operator messages in the Message Browser when an alarm limit is detected.

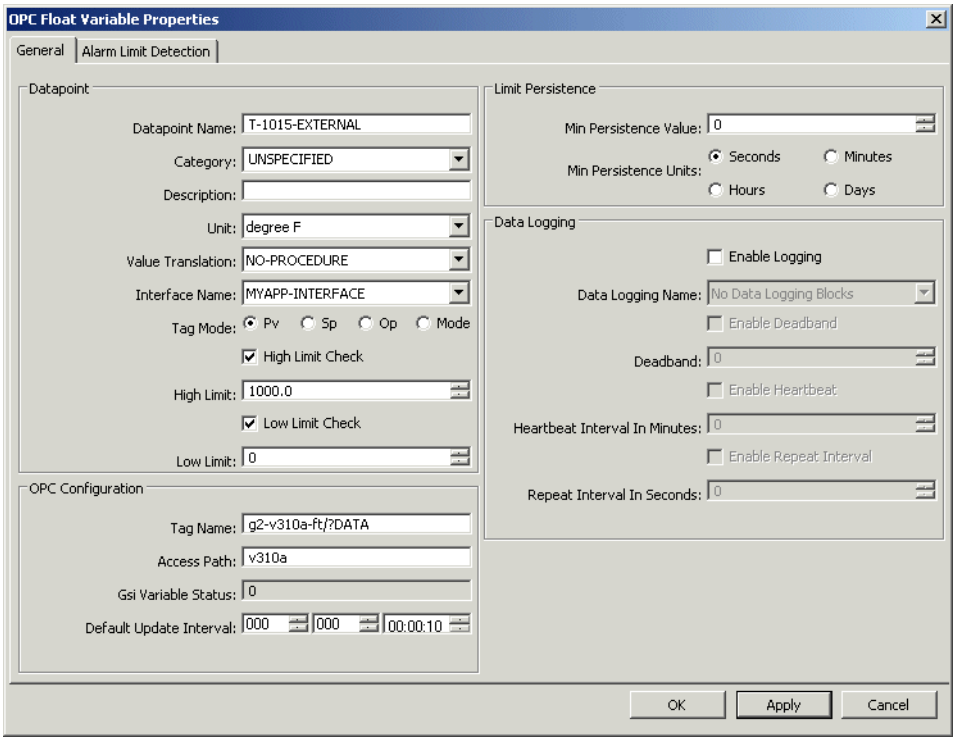
The default value is 0 seconds, which keeps the alarm limit detection message until the operator deletes it.

8 To log external datapoint values, configure Data Logging, as needed.

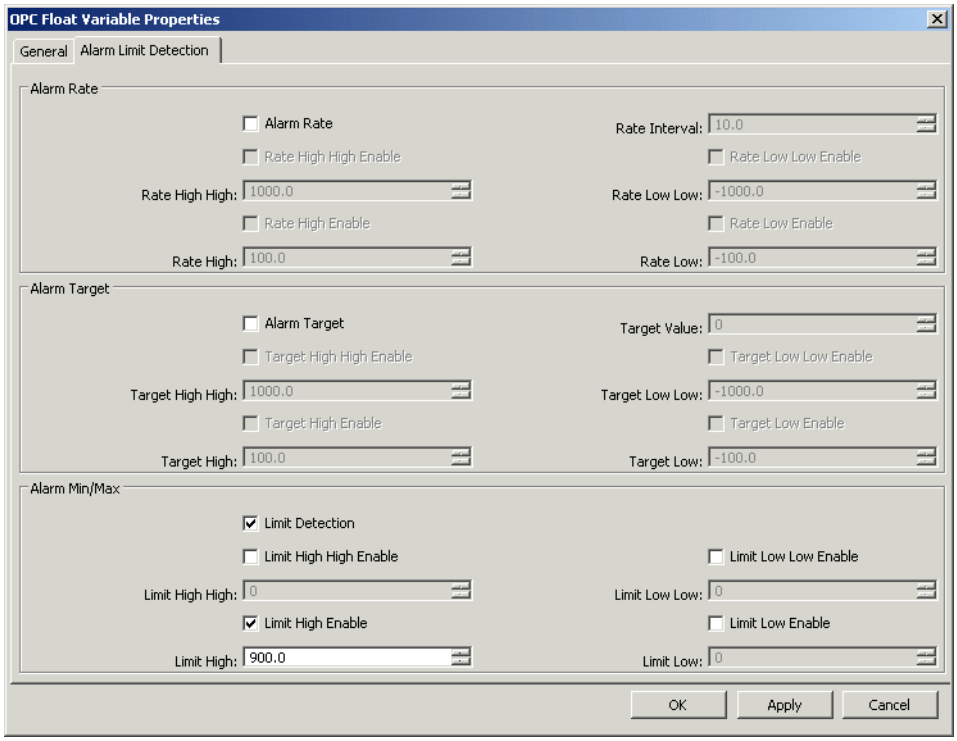
For details, see Configuring Datapoints for Logging.

9 For OPC float and PI real datapoints, click the Alarm Limit Detection tab and override the rate, target, and min-max limits for data validation, as needed.

Here is the General tab in the properties dialog for the external datapoint named t-1015-external, which is a type of OPC float. The Category has been specified as temperature. The OPC configuration identifies the tag variable in the external OPC system. The GSI Variable Status indicates the connection status of the external datapoint to the network interface, where 0 means inactive, 1 means in transition, 2 means ok, -1 means timeout, and -2 means error. The datapoint sets high and low process limits, and it keeps history for one day.



Here is the Alarm Limit Detection tab, which configures Alarm Limit Detection for data validation.



For an example of limit detection, see Viewing Data Validation Alarms.

Manually Relating External Datapoints

When you initialize process maps, Optegrity automatically relates all internal datapoints to their source external datapoints, where an external datapoint can be the source datapoint for one or more internal datapoints. When you uninitialize process maps, Optegrity de-links all internal and external datapoints.

You might want to manually relate an individual external datapoint to its associated internal datapoint(s), either to re-link the datapoints after they have already been linked or to link them after they have been de-linked. Manually relating external datapoints re-links the external datapoint with all of its associated internal datapoints.

To manually relate external datapoints to their internal datapoints:

1 Display the Navigator.

For details, see Navigating Applications.

2 Choose Show Detail on an External Datapoints container in the Navigator to show its detail.

The detail contains the individual external datapoints in the container.

3 Choose Relate Sensors and Controllers on an individual external datapoint to relate the external datapoint to all internal datapoints that specify it as the source datapoint.

Creating Individual External Datapoints

In addition to creating external datapoints automatically from a CSV file, you can create individual external datapoints from a palette. You might want to do this if you add a datapoint after you have created them automatically. Alternatively, you might build your application by starting with the domain objects and adding the external datapoints individually later.

Note When creating individual external datapoints manually, be sure to configure the Source Datapoint of the internal datapoint that gets its values from the external datapoint you create.

For information about configuring the source datapoint of an internal datapoint, see Configuring Domain Objects.

To create individual external datapoints:

1 Create an External Datapoints container.

2 Display the properties dialog for the container and configure its properties.

For details, see Creating the External Datapoints Container.

3 Click the New button in the Datapoints area.

A dialog for choosing the external datapoint class appears.

4 Choose a class and click OK.

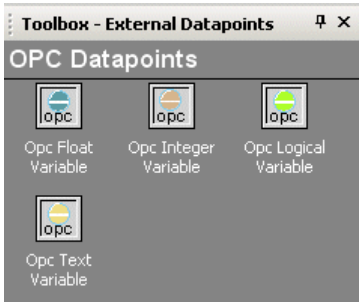
The properties dialog of the external datapoint appears.

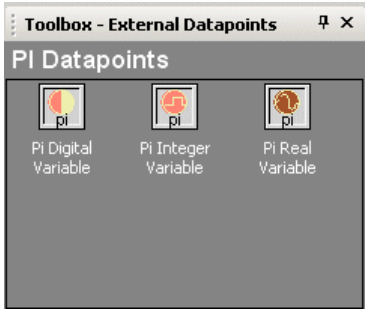
5 Configure the properties of the external datapoint manually.

For details, see Creating and Configuring External Datapoints.

You can also create individual external datapoints and manually add them to the External Datapoints container detail. To do this, show the detail of the External Datapoint container through the Navigator, create external datapoints from the

OPC Datapoints and PI Datapoints palettes in the External Datapoints toolbox, and place them on the detail. Here are the external datapoints palettes:





Translating External Datapoint Values

You can translate values for external datapoints by writing a custom procedure that takes a value, translating the value, then assigning the resulting value to the corresponding internal datapoints. The original external datapoint value is unchanged.

The signature for the translation procedure is:

my-translate-value-procedure
(*item*: class grtl-external-datapoint, *val*: value, *collection-time*: quantity)
-> *new-value*: value, *new-collection-time*: quantity

where:

- *item* is the external datapoint that received the value.
- *val* is the new raw value from the external datapoint.
- *collection-time* is the collection time of the new value.

The procedure returns these values:

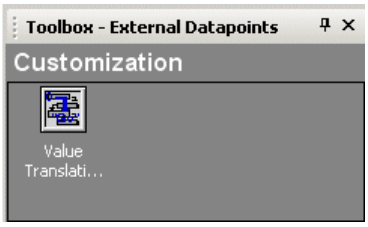
- *new-value* is the new translated value.
- *new-collection-time* is the collection time, if updated by this procedure.

Once the procedure has been defined, you can specify the translation procedure for any external datapoint by configuring the Value Translation Procedure in the properties dialog for the external datapoint. The dropdown list contains all the procedures that are a subclass of grtl-value-translation-procedure. Only procedures that are cloned from the palette appear in the dropdown list.

Translating external datapoint values requires knowledge of G2 and is, therefore, a developer task.

To translate external datapoint values:

1 Choose View > Toolbox - External Datapoints and display the Customization palette:



2 Create a Value Translation Procedure from the palette and place it within your application module.

3 Modify the procedure to perform the type of translation you want to occur.

Defining a procedure require knowledge of G2 and is an advanced feature. See the *G2 Reference Manual*.

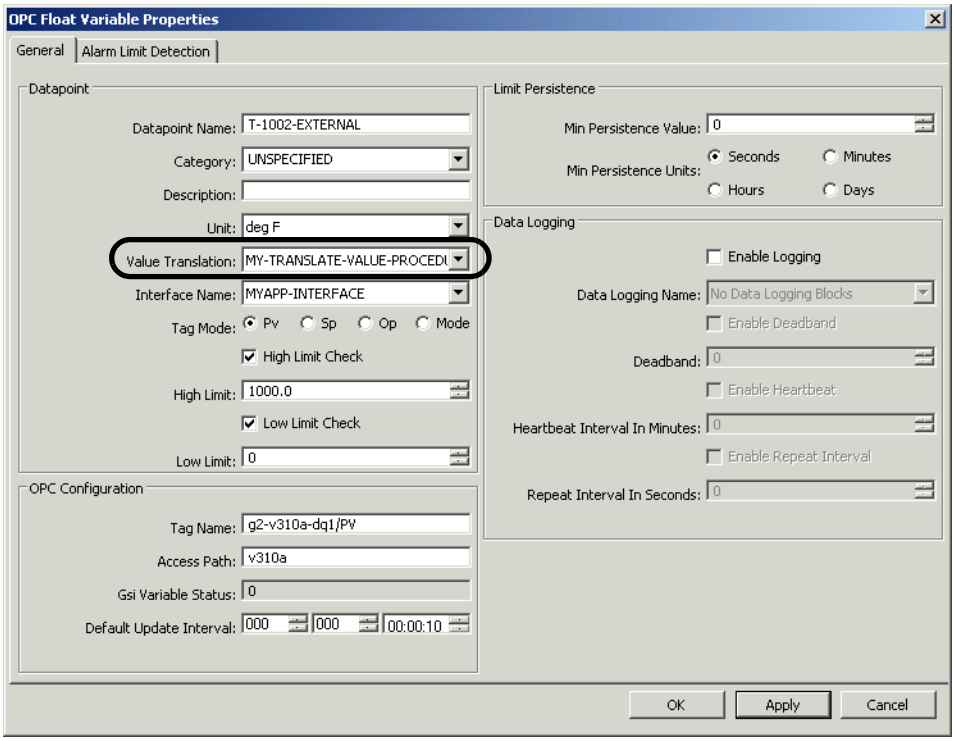
Be sure to change the name of the procedure to a unique name.

4 Display the properties dialog for the external datapoint whose value you want to translate and configure the Value Translation Procedure.

If you do not want any value translation to occur, choose no-procedure, which is the default for all external datapoints.

This dialog shows how to configure the Value Translation Procedure procedure for an OPC Float datapoint:

MY-TRANSLATE-VALUE-PROCEDURE

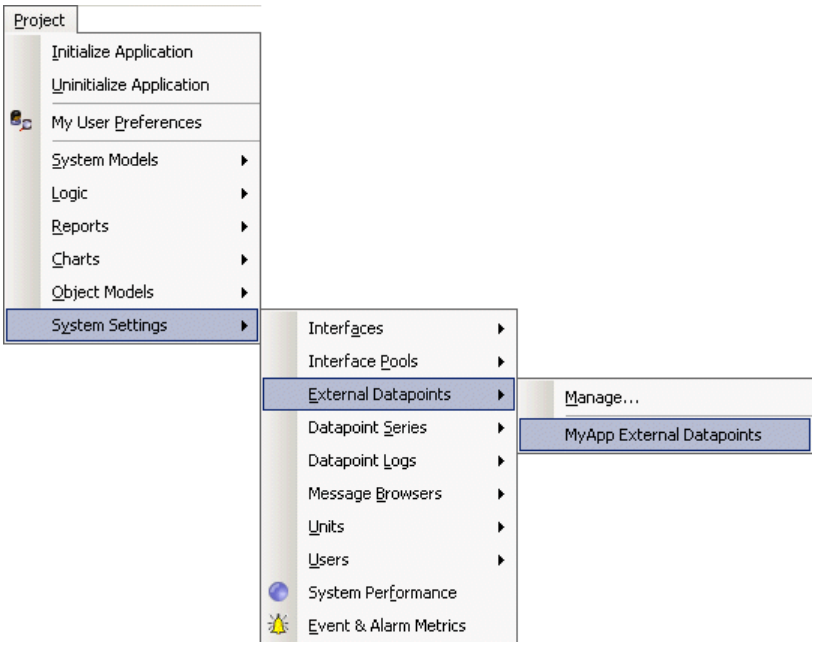


Managing External Datapoints

To manage external datapoints:

1 Choose Project > System Settings > External Datapoints.

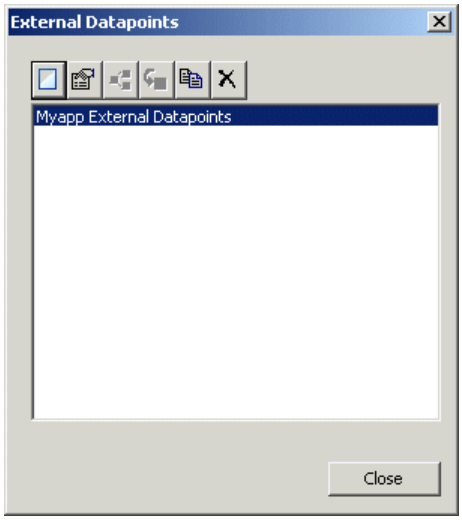
All External Datapoints containers appear in the submenu, for example:



2 To configure the properties of an external datapoint container, choose one from the External Datapoints submenu.

3 To display a dialog for managing all external datapoint configuration objects, including displaying the detail, choose Manage.

Here is the External Datapoints Manage dialog:



For information on using this dialog and the Project menu to manage external datapoints, see Using the Project Menu.

10

Converting Engineering Units

Describes how to create and configure engineering unit conversions.

Introduction

Optegrity provides a way of specifying the engineering units for entering and displaying values, as well as a large number of synonyms for those conversions in both the English and metric systems. Optegrity defines a large set of built-in engineering unit conversions and synonyms for dimensions such as pressure, length, volume, volumetric flow, mass, density, temperature, power, heat transfer, and time. It also provides a mechanism for defining custom dimensions, engineering units, and synonyms.

When you create external datapoints from a CSV file, you specify the engineering units that the DCS system uses for datapoints, which are known as the **field units**. You can configure the units that Optegrity uses for entering property values and displaying computed metrics, which are known as the **user units** or **external**

units. Optegrity converts all field units and external units to a set of common units that it uses for its internal calculations, which are known as **internal units**.

In addition, Optegrity provides API procedures and functions that you can call to work with engineering unit conversions programmatically. For details, see Working with Engineering Unit Conversions.

Working with Engineering Unit Conversions

You work with engineering unit conversions in various ways in an Optegrity application.

Configuring External Datapoint Units in the CSV File

You can import units from the CSV file used for configuring external datapoints. You configure the Datapoint Units in the column to the right of the Datapoint Type column in the CSV file. The datapoint units that you configure are equivalent to the field units that domain objects use as sensor values.

You can specify any of the built-in engineering units or a synonym. For information on determining the built-in engineering units, see Viewing Built-in Engineering Unit Conversion Definitions.

If you specify an engineering unit that does not exist, Optegrity automatically creates a unit synonym definition and places it in the Undefined-Dimension category in the Unit Synonyms submenu. For more information, see Creating New Engineering Units and Synonyms.

Note Sometimes units provided in the DCS system are inaccurate, thus Optegrity requires you to enter the units explicitly rather than obtaining them directly from the DCS system.

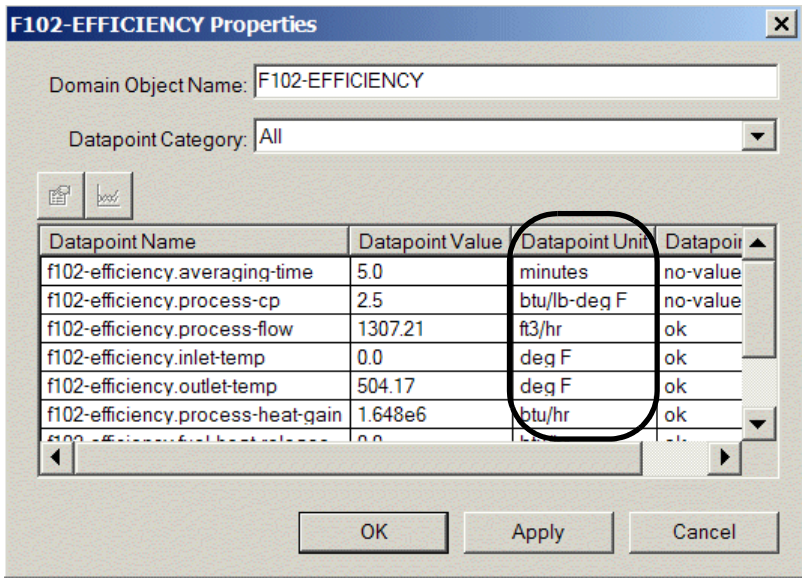
Here is part of an external datapoint configuration file, which defines field units for several datapoints in the metric system:

Datapoint	Datapoint Server Configuration				
Datapoint	Default	Datapoint	Datapoint	Datapoint	Related Process Map
Name	Update Interval	Tag Type	Type	Units	Datapoint Names
t1-dp	10 minutes	pv	float	deg C	t1.pv
t2-dp	10 minutes	pv	float	deg C	t2.pv
p1-dp	10 minutes	pv	float	kg/cm2	p1.pv
f1-dp	10 minutes	pv	float	m3/hr	f1.pv
f2-dp	10 minutes	pv	float	m3/hr	f2.pv
f3-dp	10 minutes	pv	float	m3/hr	f3.pv
a1-dp	10 minutes	pv	float	kj/m3	a1.pv

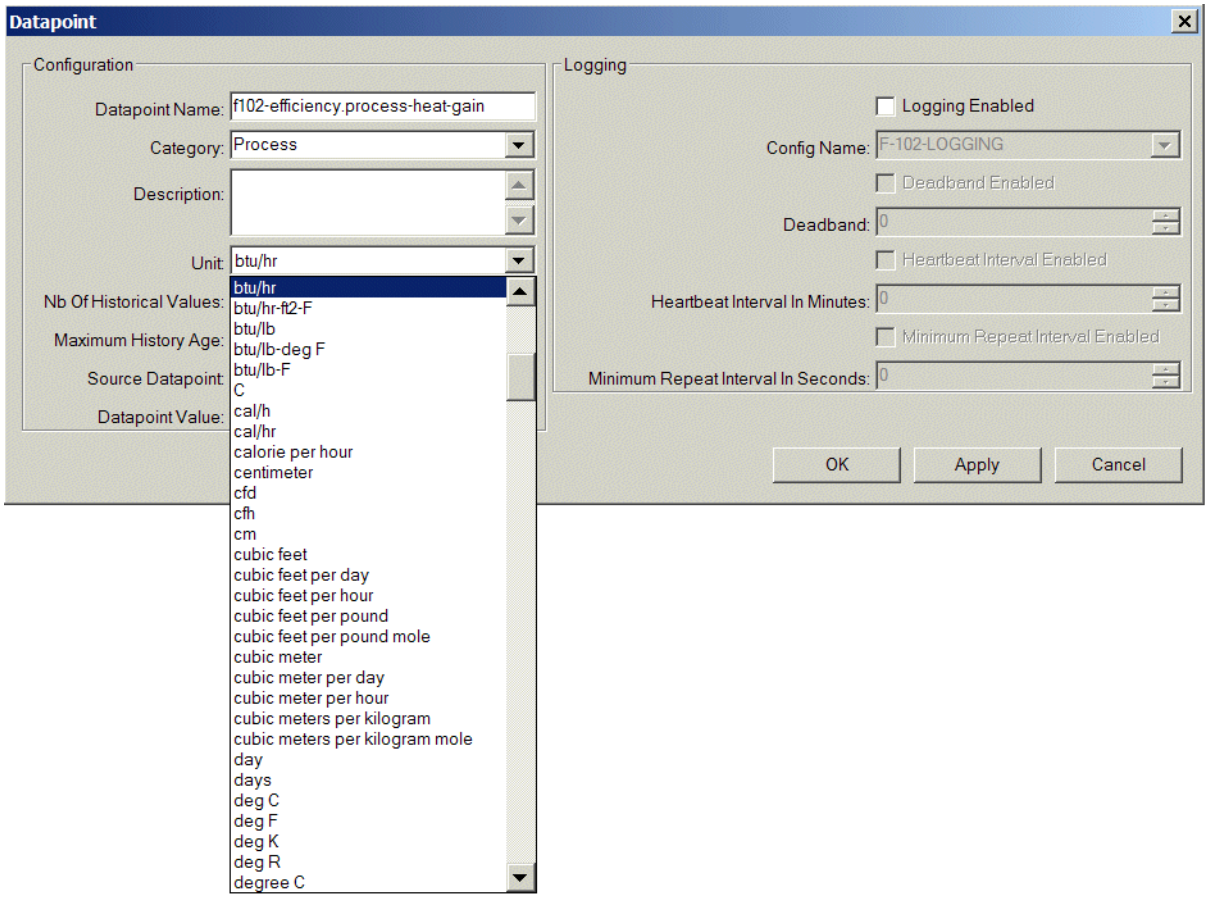
Caution When upgrading older versions of Optegrity, you must add the Units column to the CSV file before creating external datapoints. Creating external datapoints from a CSV file that does not include the Units column generates an error.

Configuring Engineering Units for Domain Objects

You can configure engineering units for entering and displaying various event parameters and metrics. For example, here is the properties dialog for the Heater Efficiency derived sensor of a heater, which specifies units for the various internal datapoints of the sensor:



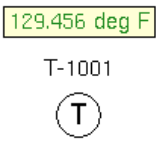
Here is the properties dialog for the calculated heat-gain internal parameter, which shows a dropdown list with some of the available engineering units:



Displaying Engineering Units for Datapoints

Units appear in the datapoint displays for internal and external datapoints.

For example, here is a display that shows the units of a flow sensor:



Configuring the Internal Units

You configure the unit system that Optegrity uses for its internal units in the configuration file by setting this parameter:

IOC-INTERNAL-UNIT-SYSTEM=english

The engineering unit system that all domain object parameters and metrics use for internal calculations. The default value is english. The other option is metric.

For more information on configuring parameters, see Customizing Optegrity.

Viewing Built-in Engineering Unit Conversion Definitions

When configuring engineering units for a given dimension, you choose from a list of engineering units in the given system, either English or metric. Similarly, when configuring the units for external datapoints, you specify the engineering units in a CSV file.

You can view the built-in engineering unit conversions for each dimension to see how they are defined. Each conversion definition specifies the following information:

- The dimension type, such as area, pressure, or temperature.
- The input and output units for the conversion.
- Whether the conversion defines a multiplier and/or offset.
- A multiplier and offset for the conversion.
- Input and output synonyms.

You might want to view the built-in engineering unit conversion definitions to see which synonyms are defined for the input and output units and whether you need to define additional synonyms.

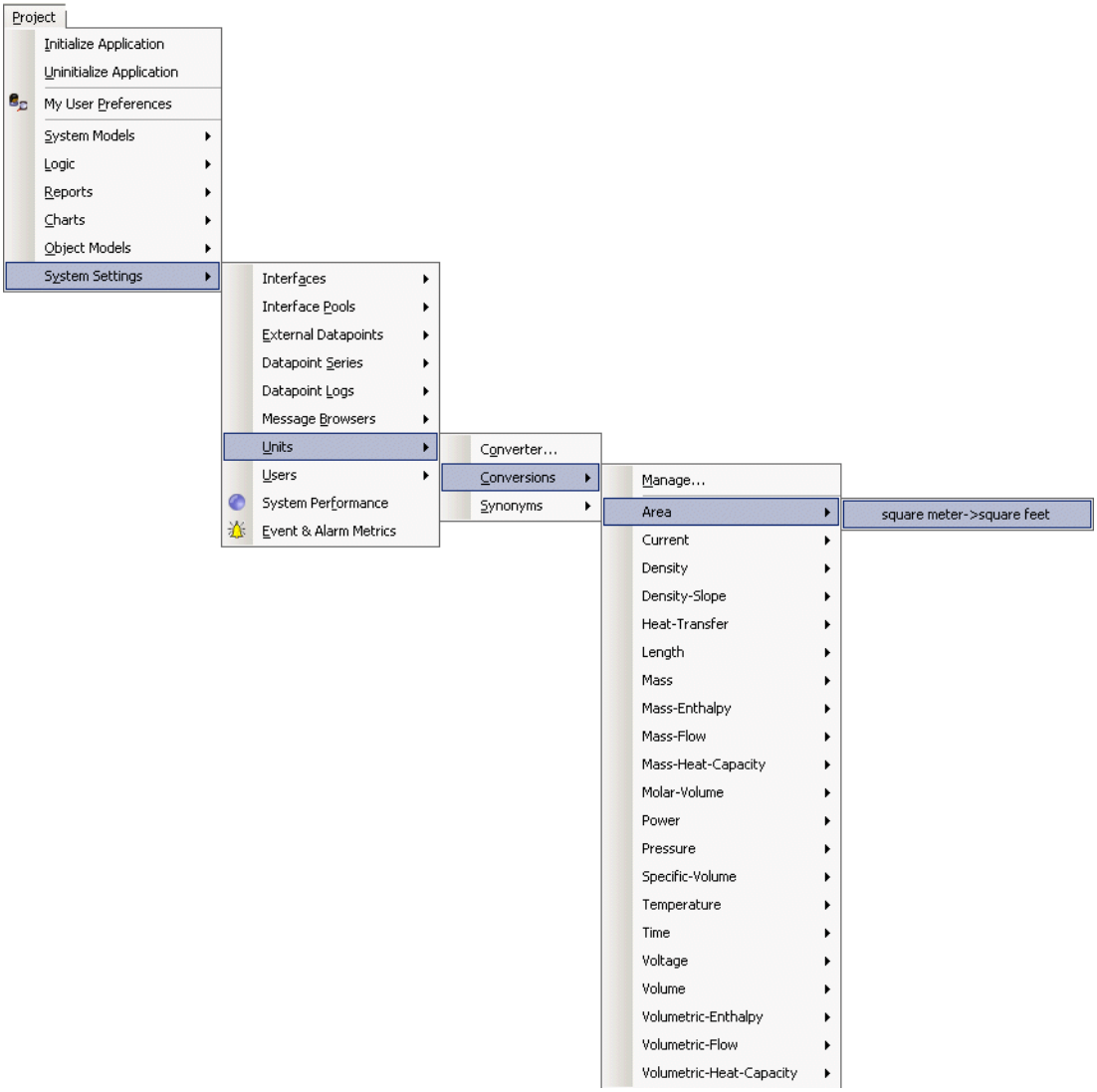
For example, the area dimension defines an engineering unit conversion called square meter->square feet, which converts square meters to square feet. The input and output units define these synonyms:

Engineering Unit	Synonyms		
square meter	m2	meter2	
square feet	ft2	foot2	feet2

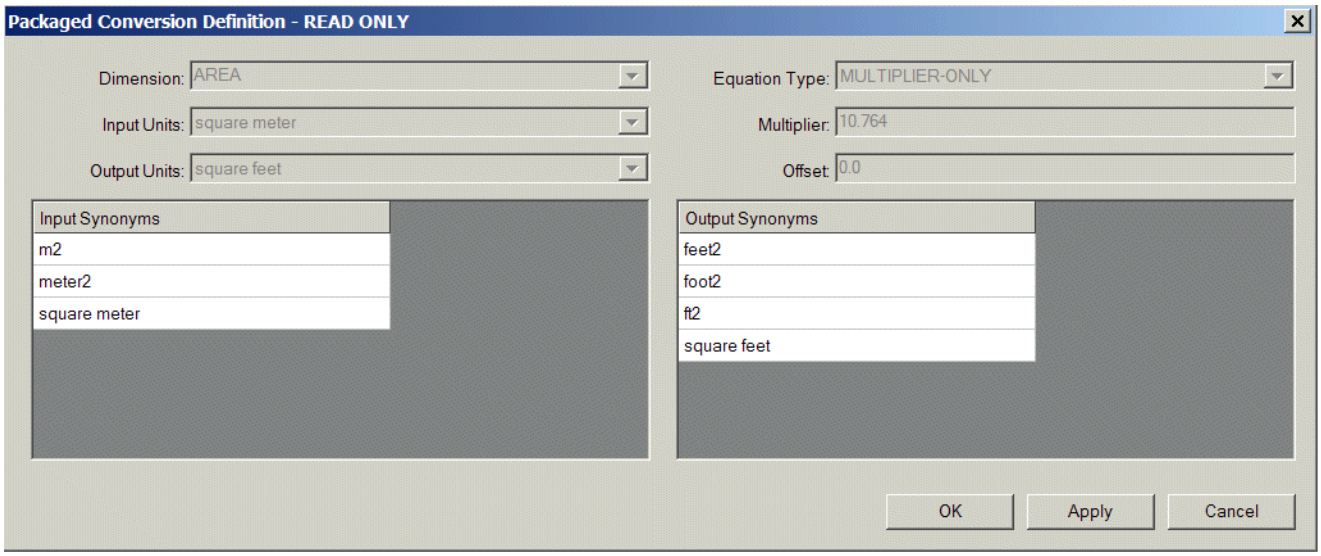
To view built-in engineering unit conversion definitions:

➔ Choose Project > System Settings > Units > Conversions, then choose the dimension and conversion definition you want to view.

For example, to view the conversion definition for the area dimension that converts square meters to square feet, you would choose:



Here is the Engineering Unit Conversion dialog for the square meter->square feet conversion definition. Notice that the conversion has three synonyms for square meter, and four synonyms for square feet. The conversion definition multiplies the input value by the specified multiplier to calculate the output value.



Defining Engineering Unit Conversion Synonyms

You can configure additional synonyms for any of the built-in engineering unit conversion definitions. You can also define new engineering units and synonyms to create new engineering conversion unit definitions.

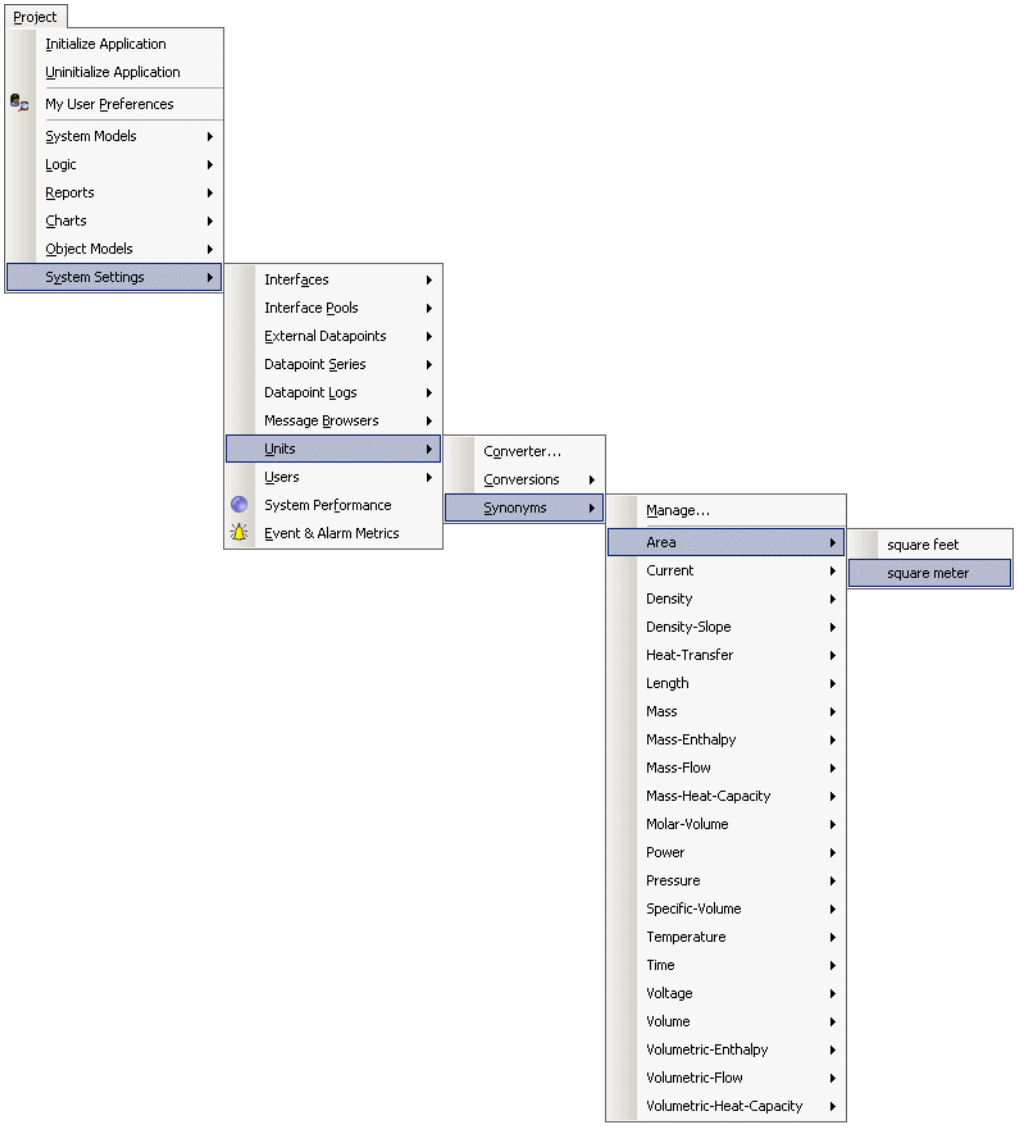
Adding New Synonyms to Existing Engineering Unit Conversion Definitions

Suppose you want to add a new synonym called meters2 for the square meter engineering unit.

To add a new synonym to an existing engineering unit conversion definition:

1 Choose Project > System Settings > Units > Synonyms, then choose the dimension and corresponding conversion unit for which you want to define a new synonym.

For example, to add meters2 as a synonym for square meter, choose:



The Synonym Definition dialog appears with the packaged synonyms for the specified unit in the list on the left.

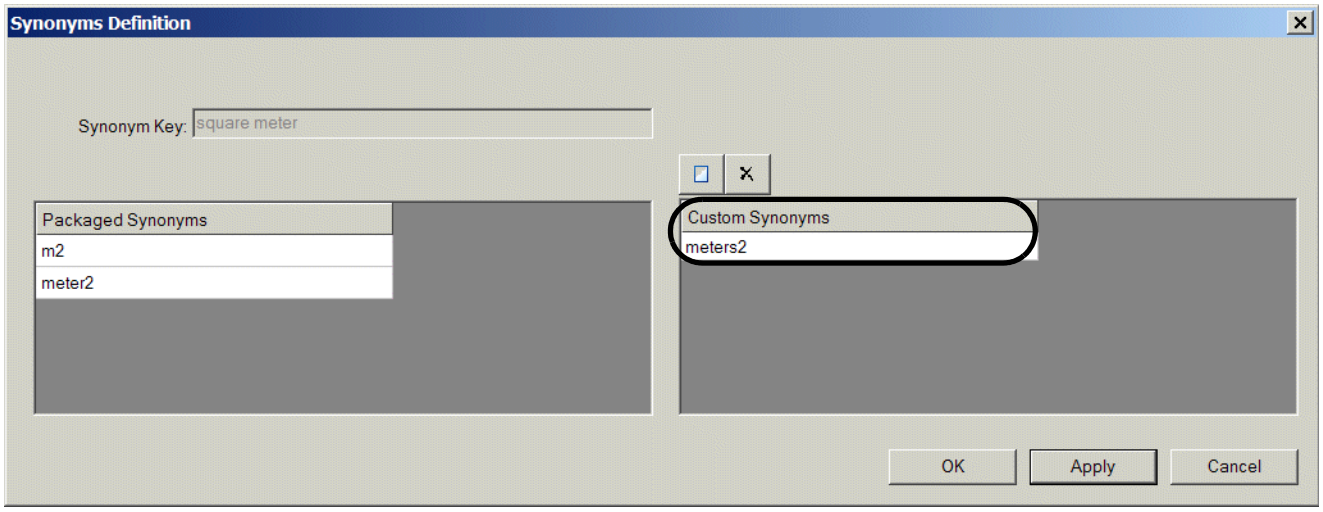
2 To create a new synonym, click the New button in the Custom Synonyms list and enter a synonym.

3 To enter additional synonyms, click in the number column of the row where you want to insert a new synonym, and click the Insert Before or Insert After toolbar button to insert a new row before or after the selected row.

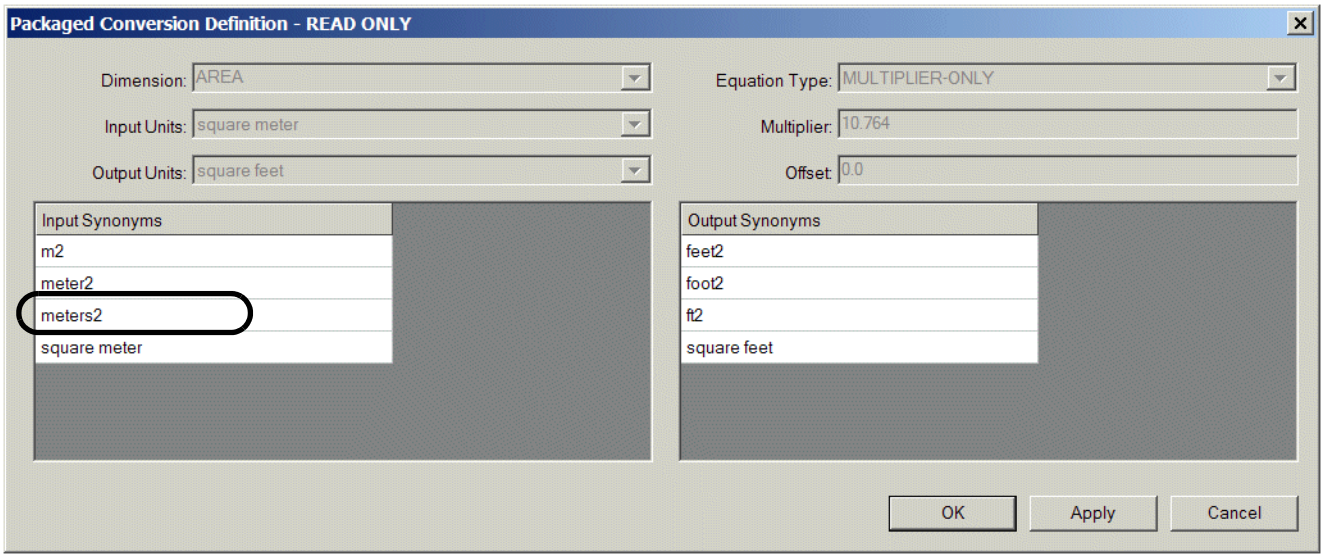
4 Enter a new synonym in the new row.

5 Click OK in the dialog, then click OK in the confirmation dialog to save the synonym to the custom synonyms file.

Here is the Synonym Definition dialog that defines meters2 as a new synonym for the square meter unit:



The new synonym now appears in the Engineering Unit Conversion dialog for the conversion definition:



Creating New Engineering Units and Synonyms

Suppose you want to create a new engineering unit conversion definition for the area dimension that converts square centimeters to square inches. You would create two new engineering units called square centimeter and square inch, each of which might define several synonyms.

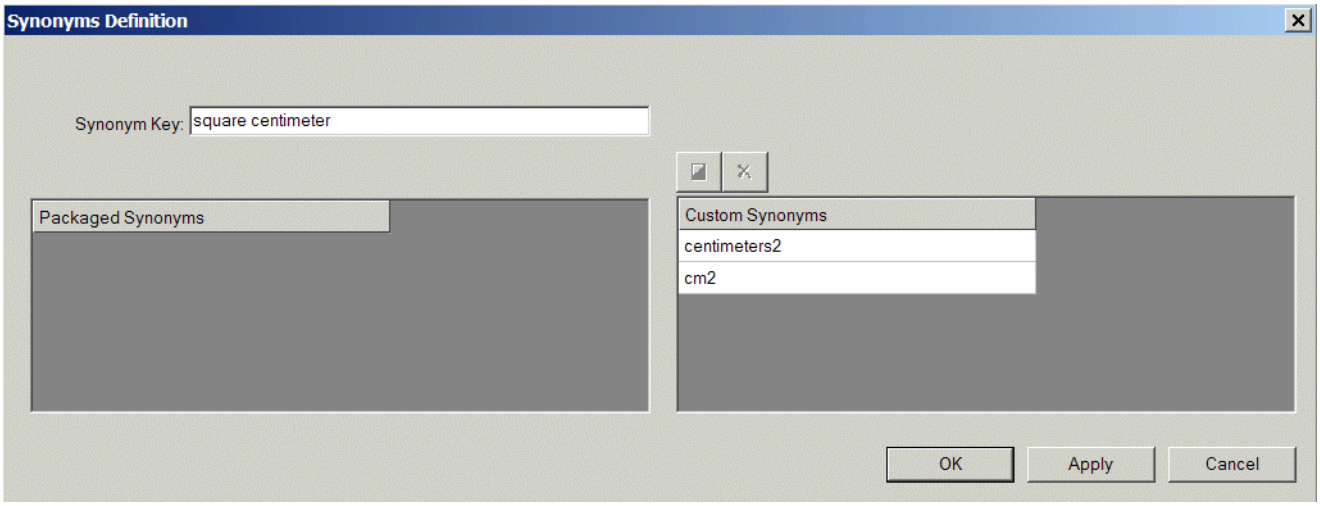
When you initially create new engineering units, they appear in the Undefined-Dimension category in the Unit Synonyms submenu. As soon as you create a new engineering unit conversion definition that uses the new conversion units, they appear in the appropriate dimension category in the submenu.

Note Any existing unit synonyms that have not yet been used as part of an engineering unit conversion definition appear in the Undefined-Dimensions category.

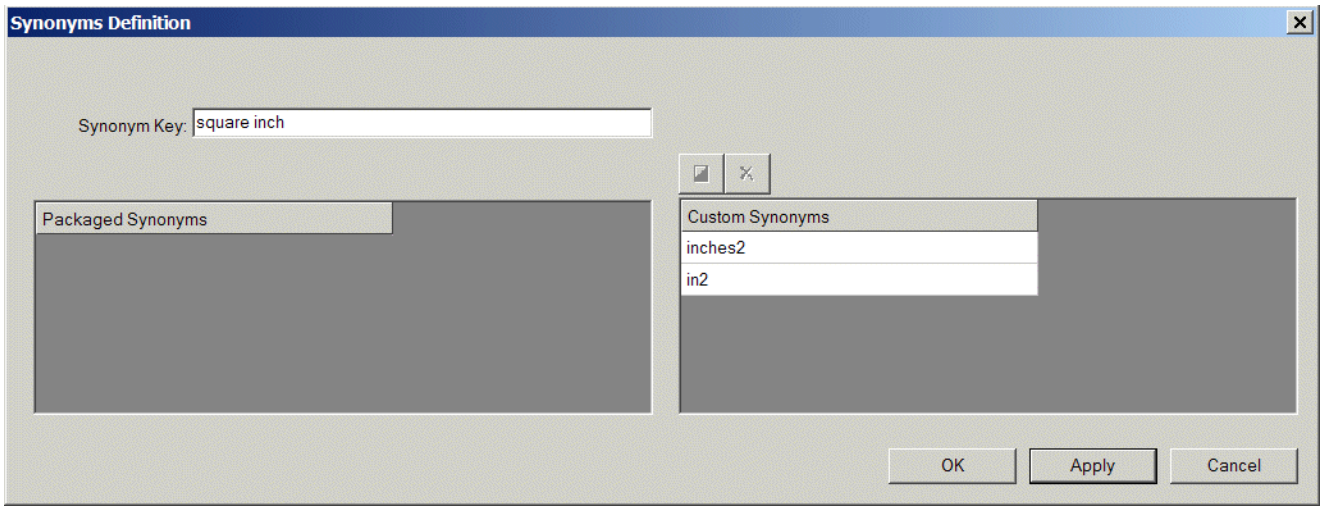
To create a new engineering unit and synonyms:

1. Choose Project > System Settings > Units > Synonyms > Manage to display the Manage dialog for all the unit synonyms.
2. Click the New button to create a new engineering unit definition.
3. Configure the Synonym Key to be most generic form of the engineering unit.
4. To create a synonym, click the New button in the Custom Synonyms list and enter a synonym.
5. Click OK in the dialog, then click OK in the confirmation dialog to save the engineering unit and its synonyms to the custom synonyms file.
6. Repeat for the input and output engineering units required for the new engineering unit conversion definition.

Here is the Synonym Definition dialog that defines the square centimeter engineering unit and two synonyms:



Here is the Synonym Definition dialog that defines the square inch engineering unit and two synonyms:



Initially, the new engineering units appear in the Undefined-Dimension category in the Unit Synonyms menu, along with other engineering units that are not yet used in any engineering unit conversion definitions.

Defining Engineering Unit Conversion Definitions

Once you have defined the input and output units and their synonyms, you can define a new engineering unit conversion definition that uses those engineering units. For example, you might want to create a new engineering unit conversion definition for the area dimension that converts square centimeters to square inches.

To do this, first, you must create the engineering units and synonyms for square centimeter and square inch, as described in Defining Engineering Unit Conversion Synonyms.

To create a new engineering unit conversion definition:

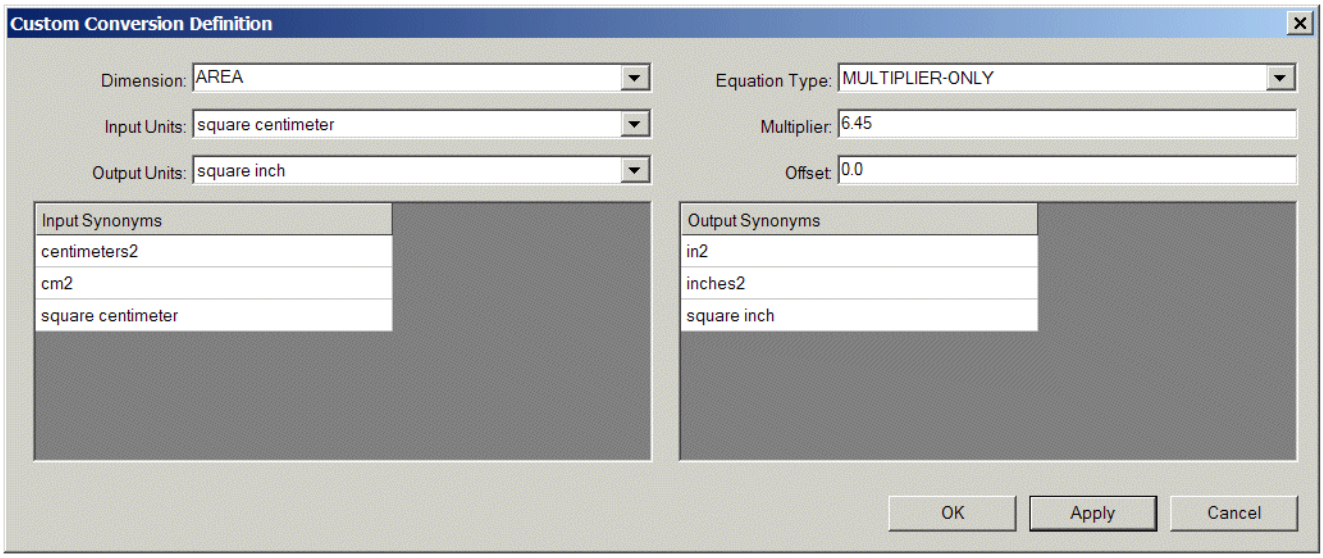
1. Choose Project > System Settings > Units > Conversions > Manage to display the Manage dialog for all unit conversions.
2. Click the New button to create a new engineering unit conversion definition.
3. Configure the Dimension Class for the engineering unit conversion definition.

   You can configure an existing dimension, such as area, or you can create a new dimension. For example, to create an engineering unit conversion called square centimeter->square inch, you would configure the dimension to be area.
4. Configure the Input Units and Output Units for the dimension.

   In the example, the Input Units would be square centimeter and the Output Units would be square inch.
5. Configure the Equation Type to be one of the following:
   - multiplier-only — Specifies a Multiplier only.
   - offset-only — Specifies an Offset only.
   - multiply-first — Specifies both a Multiplier and Offset, where the multiplication operation happens before the offset.
   - offset-first — Specifies both an Offset and a Multiplier, where the offset operation happens before the multiplication.
6. Depending on the value of Equation Type, configure the Multiplier and Offset to be the values to use for multiplying and offsetting the input value to calculate the output value.

7 Click OK in the dialog, then click OK in the confirmation dialog to save the engineering unit conversion to the custom conversions file.

Here is the engineering unit conversion definition for square centimeter->square inch, which converts square centimeter to square inch, using a multiplier. Once you initially accept the dialog, the custom synonyms for the input and output units all appear in the dialog.



The engineering unit conversion definition appears in the menus under the area dimension along with the built-in conversion definition.

Now that the custom engineering units have been used in an engineering unit conversion definition, they appear under the appropriate dimension in the menus, in this case, area. They no longer appear under the Undefined-Dimension category.

Converting Engineering Units on Demand

You can perform engineering unit conversions on demand through the Engineering Unit Converter dialog. You might want to do this to verify a unit conversion before choosing the units.

To convert engineering units on demand:

1 Choose Project > System Settings > Units > Converter.

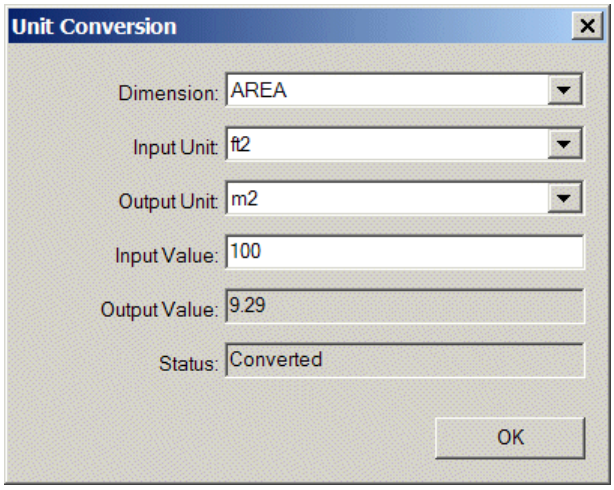
2 Choose the Dimension type.

3 Configure the Input Units and the Output Units.

4 Enter an Input Value in the input units and press Return.

The converted value in the output units appears with a status value of converted with a status of converted. The converted value also updates automatically if you change the Input Units and Output Units for a given Input Value.

Here is the Engineering Unit Converter dialog that shows the unit conversion from ft2 to m2:



Managing Engineering Units

You manage engineering unit conversions and synonyms separately.

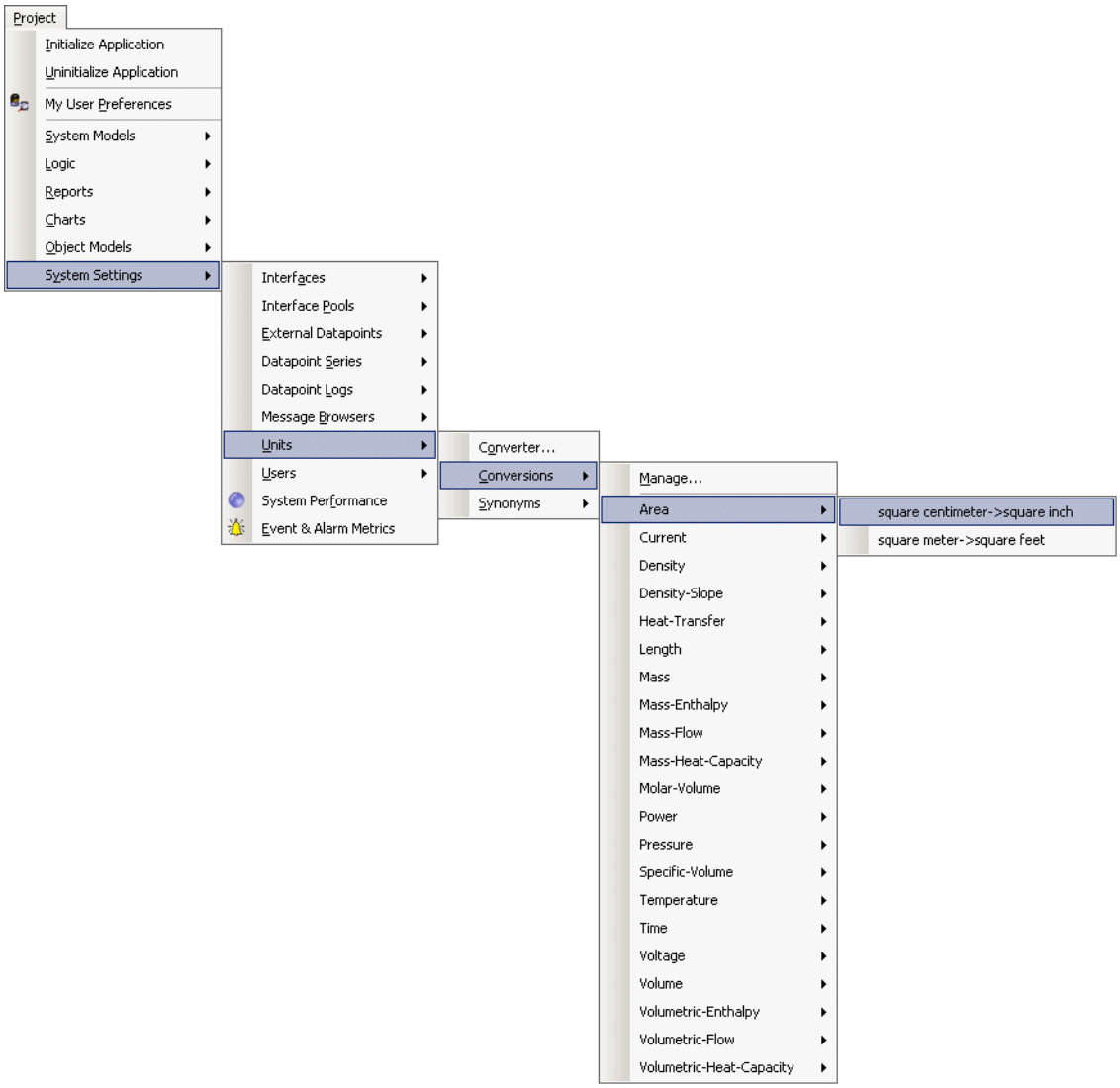
Managing Engineering Unit Conversions

To manage engineering unit conversions:

1 Choose Project > System Settings > Unit > Conversions > Manage.

All engineering unit conversion classes appear in the submenu, and all built-in and custom engineering unit conversions appear in the submenus for each conversion class. For example, here is the submenu for the Area unit conversion, which includes the built-in square meter->square foot unit

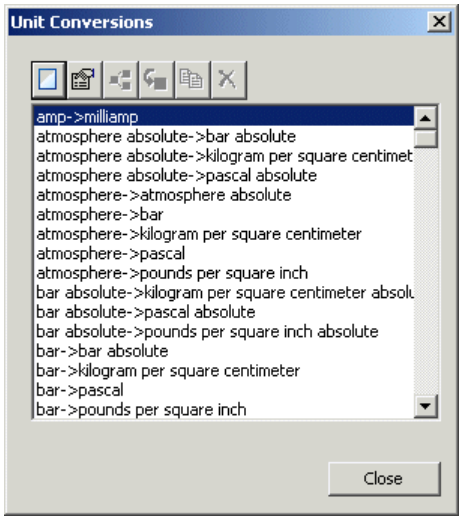
conversion and the user-defined square centimeter->square inch unit conversion:



2 To configure the properties of a unit conversion, choose one from the appropriate category in the Unit Conversions submenu.

3 To display a dialog for managing all unit conversions, choose Manage.

Here is the Unit Conversions Manage dialog:



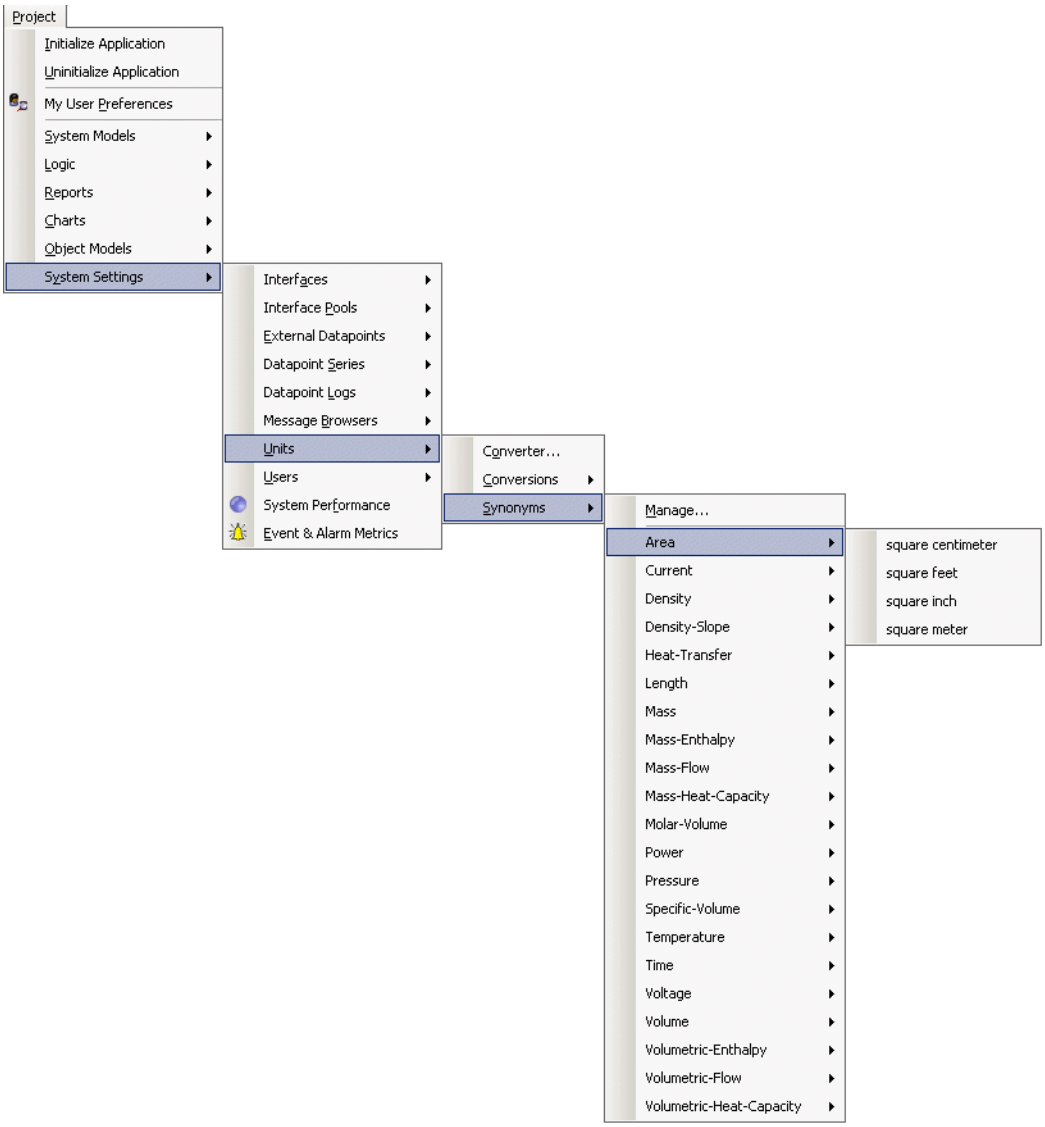
Managing Engineering Unit Synonyms

To manage engineering unit conversions:

1 Choose Project > System Settings > Units > Synonyms > Manage.

All engineering unit conversion classes appear in the submenu, and all built-in and custom engineering unit synonyms appear in the submenus for each conversion class. For example, here is the submenu for the Area unit

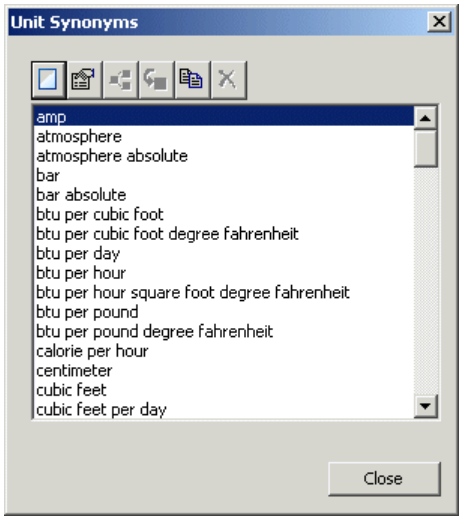
synonym, which includes the built-in square feet and square meter synonyms, as well as the user-defined square centimeter and square inch synonyms:



2 To configure the properties of a unit synonym, choose one from the appropriate category in the Unit Synonyms submenu.

3 To display a dialog for managing all unit synonyms, choose Manage.

Here is the Unit Synonyms Manage dialog:



For information on using this dialog and the Project menu to manage process maps, see Using the Project Menu.

11

Configuring Logging

Describes how to configure logging for internal and external datapoints.

Introduction

You can configure your application to log internal or external datapoints as the application runs. By default, Optegrity logs data values as they arrive. You can also log data on a schedule. You can filter logged data, based on the rate at which it arrives and the variation of data values.

To log internal datapoints, you create a Data Logging configuration object:

Configuration Object	Description
Data Logging	Logs internal datapoints in a process map to a CSV file.

To configure data logging, you must specify the name of a CSV where data is to be logged. You also specify the name of a process map container object, whose internal datapoints you want to log. When logging is enabled, Optegrity logs all internal datapoints in the process map, by default. You can selectively enable and disable logging for individual datapoints, through a spreadsheet. You can also configure logging for individual datapoints.

You can also log internal and external datapoint values to the default Message Browser.

One use of a log file is for data replay. For more information, see Replaying Data.

For information on logging messages to a file, database, or JMS provider, see Logging Messages.

Configuring Datapoints for Logging

You can configure logging for internal and external datapoints to:

- Post log messages to the Message Browser.
- Post log messages to a CSV file.
- Log data values each time they change, the default, or on a schedule, which you might do when the values change infrequently.

To configure datapoints for logging, you can assign all internal datapoints in a process map and/or all external datapoints in one or more External Datapoints containers. Once you assign internal and external datapoints for logging, you can configure logging parameters for individual datapoints.

By default, Optegrity does not perform data logging. You must explicitly enable data logging to begin logging.

You can also configure datapoints to filter logged values by enabling and configuring these logging parameters:

- Heartbeat Interval, which logs data according to a schedule, rather than each time the value changes. The value cannot exceed the Maximum Heartbeat given in the Data Logging configuration object. The default value is the Default Heartbeat in Minutes given in the Data Logging configuration object.
- Repeat Interval, which allows you to reduce the number of logged datapoints, based on the rate of incoming data. The datapoint logs value is logged each time it changes or once every Repeat Interval, whichever is longer. The data is not logged if the elapsed time from the last logged value is less than the Repeat Interval. You configure this attribute when the rate of incoming data is greater than the rate at which you want to log data.
- Deadband, which only logs a new value if that value differs from the last logged value by more than the deadband.

You can configure logging parameters for all assigned datapoints through a spreadsheet, or you can configure logging for individual datapoints through the individual properties dialogs.

For information on configuring the properties of internal datapoints, see Configuring Internal Datapoints.

For information on configuring the properties of external datapoints, see Creating and Configuring External Datapoints.

To configure datapoints for logging:

1 Choose Project > System Settings > Datapoint Logs > Manage and click the New button.

The properties dialog for configuring data logging appears.

2 Configure a unique Name, which is system-generated, by default.

The name must be a symbol, without spaces.

Tip We recommend that you prefix the name with your application name, for example, myapp-logging-configuration.

---

3 To send log messages to the Message Browser, enable the Enable Message Browser Logging option.

For information on accessing the Message Browser, see Interacting with Operator Messages.

4 To send log messages to a log file, enable the Enable CSV Logging option and click the browse button to configure the Filename to be a complete path name to a CSV file to use for logging.

The Filename can refer to an existing file, or you can specify a new file, in which case Optegrity creates the file. The *logs* subdirectory of the *optegrity* directory is available for you to place log files.

5 To log data on a schedule, enable the Enable Heartbeat Manager option, and configure the Maximum Heartbeat and Default Heartbeat in Minutes.

Maximum Heartbeat is the maximum value that any internal or external datapoint can use for the Default Heartbeat.

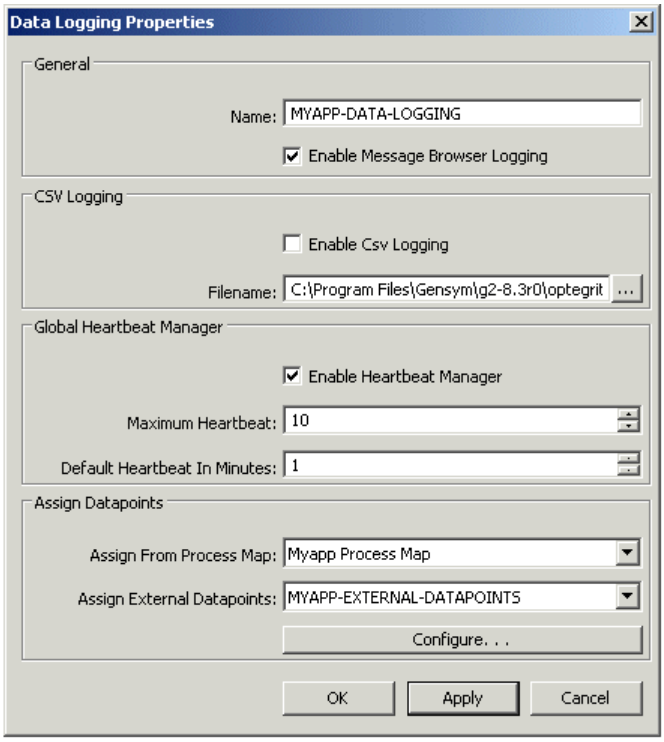
Default Heartbeat in Minutes is the default heartbeat for each internal or external datapoint, in minutes.

6 To assign internal datapoints for logging, configure the Assign from Process Map to be a process map whose internal datapoints you want to assign.

You can assign internal datapoints from as many process maps as you need.

7 To assign external datapoints for logging, configure the Assign External Datapoints to be the External Datapoints container whose external datapoints you want to assign.

Here is the properties dialog for the Data Logging configuration object named myapp-data-logging, which logs all internal datapoints in the Myapp Process Map and Myapp External Datapoints to the specified CSV file and to the Message Browser:



8 To assign the internal and external datapoints, click Apply.

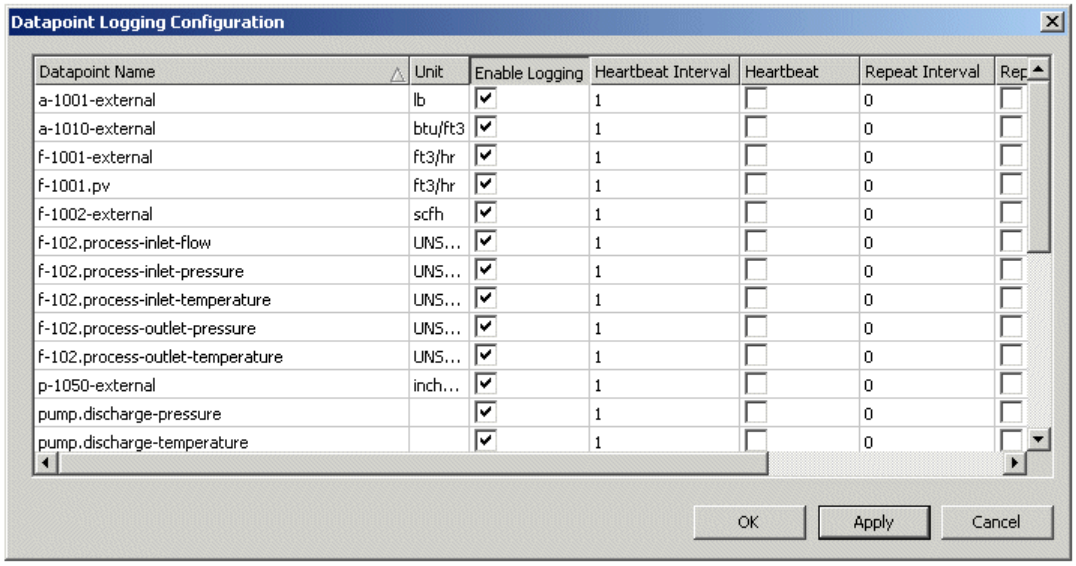
Now that you have assigned internal and external datapoints for logging, you can configure logging parameters for each assigned datapoint through a spreadsheet.

9 Click the Configure button to display a spreadsheet of all assigned internal and external datapoints, then configure the individual logging parameters for each datapoint.

You must specifically enable logging for each datapoint. You can also enable, disable, and configure the Heartbeat Interval, Repeat Interval, and Deadband for each datapoint.

To enable a logging parameter, click the spreadsheet cell labeled Disabled to toggle it to Enabled. To disable it, click the Enabled label to toggle it back to Disabled.

For example, this spreadsheet enables logging for all internal datapoints in the Myapp Process Map and all associated external datapoints:



Configuring logging parameters in this spreadsheet automatically updates the logging specification in the properties dialogs for the individual internal and external datapoints.

Log File Format

The first column in the log file is the time at which the value was logged. Each subsequent column in the log file corresponds with an external or internal datapoint whose value is being logging. Each row represents the logged value for that datapoint.

By default, Optegrity appends data to the log file.

Here is a sample log file for the Myapp Process Map application, which logs both external and internal datapoints:

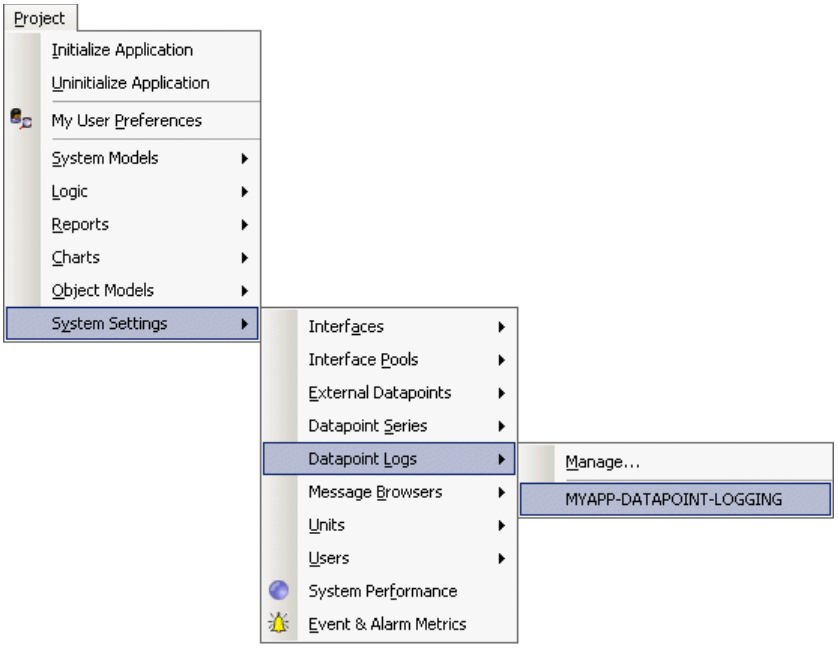
	A	B	C	D	E	F	G	H	I	J
1		FO2-IO-O	FO2-IO-TI	FO2-IO-F	FO2-IO-TI	fo2.T-IN-(	fo2.T	f_101.PV	t_102.PV	fo2.O2-SE
2	0	2.403	550.26	74.949	497.673		550.26	74.949	497.673	2.403
3	0.017	1.976	506.835	71.982	538.293		506.835	71.982	538.293	1.976
4	0.033	1.038	557.284	69.999	539.081		557.284	69.999	539.081	1.038
5	0.05	2.503	580.281	68.685	509.207		580.281	68.685	509.207	2.503
6	0.067	1.073	511.767	67.955	501.302		511.767	67.955	501.302	1.073
7	0.083	1.656	538.692	69.323	513.968		538.692	69.323	513.968	1.656
8	0.1	1.365	542.9	67.153	520.527		542.9	67.153	520.527	1.365
9	0.117	2.327	506.091	69.916	451.553		506.091	69.916	451.553	2.327
10	0.133	1.146	529.965	69.912	516.274		529.965	69.912	516.274	1.146
11	0.15	2.921	512.78	70.465	478.23		512.78	70.465	478.23	2.921
12	0.167	1.562	599.378	73.545	522.112		599.378	73.545	522.112	1.562
13	0.183	1.184	519.731	66.85	477.222		519.731	66.85	477.222	1.184
14	0.2	1.088	512.066	65.537	450.092		512.066	65.537	450.092	1.088
15	0.217	1.95	548.395	72.037	457.349		548.395	72.037	457.349	1.95
16	0.233	1.729	561.626	69.981	455.609		561.626	69.981	455.609	1.729
17	0.25	2.777	597.235	65.945	533.391		597.235	65.945	533.391	2.777
18	0.267	2.541	526.451	71.125	547.95		526.451	71.125	547.95	2.541
19	0.283	1.445	574.82	74.116	471.94		574.82	74.116	471.94	1.445
20	0.3	2.987	524.158	73.742	521.48		524.158	73.742	521.48	2.987
21	0.317	1.108	568.792	73.451	538		568.792	73.451	538	1.108
22	0.333	1.42	530.216	69.27	491.927		530.216	69.27	491.927	1.42
23	0.35	2.51	555.321	70.716	469.735		555.321	70.716	469.735	2.51

Managing Data Logging

To manage data logging:

1 Choose Project > System Settings > Datapoint Logs.

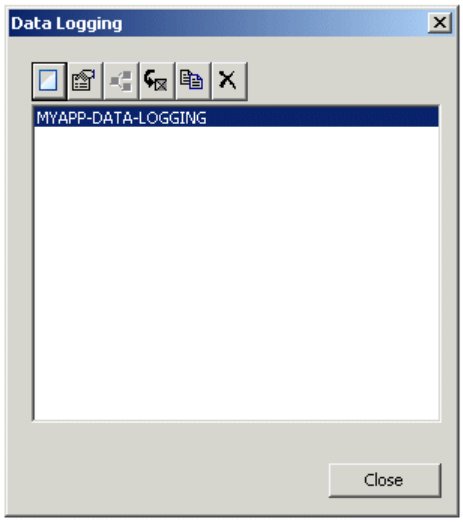
All Data Logging configuration objects appear in the submenu, for example:



2 To configure the properties of a data logging configuration object, choose one from Data Logging submenu.

3 To display a dialog for managing data logging, choose Manage.

Here is the Data Logging Manage dialog:



For information on using this dialog and the Project menu to manage data logging, see Using the Project Menu.

12

Replaying Data

Describes how to replay internal and external datapoint values from continuous and differential CSV files.

Introduction

You replay data to:

- Test an application in offline mode.
- Validate control algorithms prior to placing a system online.

You can simulate data for internal datapoints or external datapoints.

You use these configuration objects for replaying data from CSV files:

Configuration Object	Description
Data Replay	Configures the data file(s) to use for replaying data, and controls when to start and stop replaying data.
Data File	Configures various information about the CSV file to use for data replay, depending on the type.

You must configure a Data Replay object to specify the data file. You can replay data from one or more CSV files at once.

You can use datapoint displays to see how internal and external data values are updated. Datapoint displays are available in the Optegrity toolbox.

When simulating data or when the application is online, Optegrity performs data validation, depending on how the external datapoints are configured.

For information on configuring external datapoints for data validation, see Configuring Data Validation.

Creating Data Series

You can create one of two types of data series for data replay, depending on the type of process you want to simulate:

To simulate...	Use this type of data series...	Which sends new values...
A continuous process	Continuous	At regular time intervals, 5 seconds, by default.
A batch process	Differential	At a specified timestamp, based on an acceleration factor.

Creating a Continuous Data Series

To create a continuous data series:

1 Choose Project > System Settings > Datapoint Series > Continuous Data Series > Manage and click the New button.

The properties dialog for configuring the data series appears.

2 Configure a unique Name, which is system-generated, by default.

The name must be a symbol without spaces.

3 Configure the File Name to be a complete path to a CSV file to use for data simulation.

By default, the first row of the data replay file contains the name of the internal or external datapoint whose values should be replayed, and the second row of the file contains the data.

4 If necessary, configure the Tagname Row and First Data Row to refer to different rows.

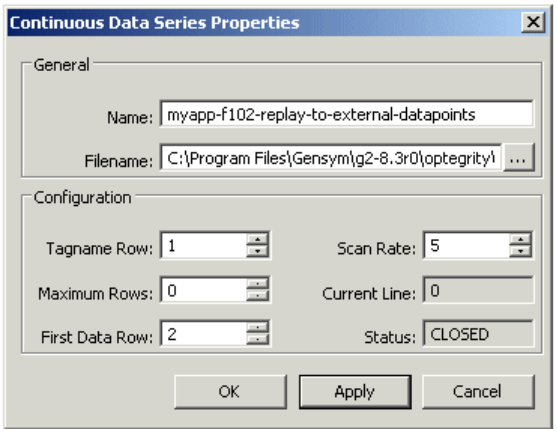
By default, the data replay file sends a value once every 5 seconds.

5 To increase or decrease the rate at which the data file sends values, configure the Scan Rate, in seconds.

By default, the data replay file sends all the values in the file.

6 To limit the number of rows of data to send, configure the Maximum Rows.

Here is the properties dialog for a continuous data file named myapp-f102-replay-data-to-external-datapoints:



Creating a Differential Data Series

For a differential data series, you can configure the following attributes, in addition to those you configure for a continuous data series:

- Start Time allows you to skip the beginning portion of the data series up to the first row greater than or equal to the specified time. The start time must use the same time format as the timestamp in the first column of the CSV file.
- Acceleration Factor specifies the speed at which to send data to the specified datapoints, which is 1.0, by default. For example, an acceleration of 20 means that each timestamp specified in the CSV file is divided by 20 to determine the actual time at which the data is sent. For example, if the CSV file specifies a timestamp of 2 minutes (120 seconds), the data is actually sent at 6 seconds (120/20 seconds).
- Time Format indicates how to interpret time values specified in the CSV file. By default, the data series assumes time values are in decimal hours. For example, 90 minutes is expressed as 1.5 hours. You can specify time values, using one of the other formats, including a custom format. If you use a custom time format, you must provide a Time Conversion Procedure that determines how to interpret time values in the CSV file.
- Time Conversion Procedure is the name of a G2 procedure that interprets the time value in the differential data series.

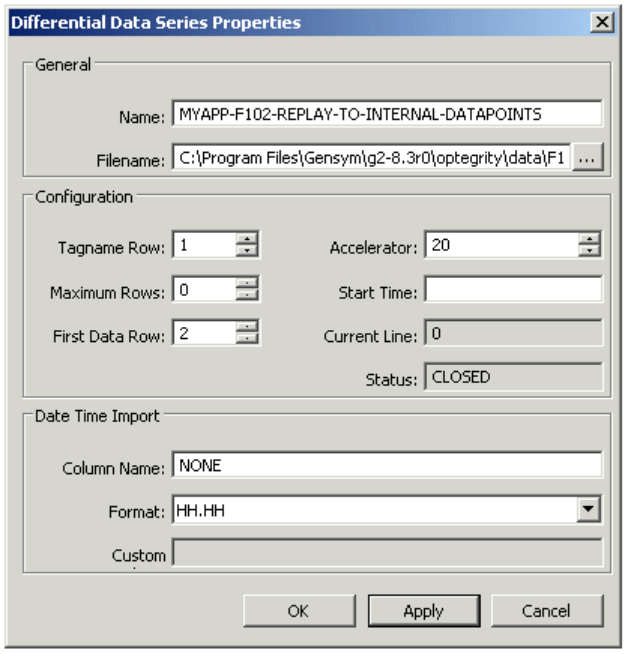
To create a differential data series:

1. Choose Project > System Settings > Datapoint Series > Differential Data Series > Manage and click the New button.

   The properties dialog for configuring the data file appears.

2. Follow steps 2 through 6 under Creating a Continuous Data Series.
3. Configure the additional attributes, described above, as needed.

Here is the properties dialog for a differential data file named myapp-f102-replay-data-to-internal-datapoints. The data file assumes timestamps are expressed in decimal hours, with an acceleration factor of 20.0.



Creating Data Replay Files

The format of a data replay file is similar to the format of a log file, where each column represents data values for an internal or external datapoint, and each row represents a new value. By default, the first row contains the name of the internal or external datapoint, and the second row contains the first row of data.

For differential data series, the first column of the data file provides a timestamp that determines the rate at which to replay the data, expressed in decimal hours. The rate is determined by calculating the difference in time between consecutive rows. The time interval between rows does not need to be evenly spaced. For continuous data files, the first column is ignored.

Tip You can use a log file as a data replay file. For details, see Configuring Logging.

To create a data replay file:

1 Create a CSV file in which the first row contains the names of each internal or external variable whose value you want to replay, and the subsequent rows contain the data values to replay.

   To replay external datapoint values, use the names of external datapoints in the External Datapoints container, for example, t-1001-external.

   To replay internal datapoint values, use dot notation to refer to the name of the internal datapoint and its corresponding domain object, for example, t-1001.pv.

2 For differential data files, the first column must be a timestamp.

Here is a partial CSV file for replaying external datapoint values:

	A	B	C	D	E	F
1	T-1015-EXTERNAL	T-1014-EXTERNAL	A-1010-EXTERNAL	F-1002-EXTERNAL	T-1016-EXTERNAL	P-1050-EXTERNAL
2	880.559	867.319	924.36	1513.582	910.896	-185.823
3	884.124	867.128	921.363	1514.142	909.295	-196.09
4	883.222	869.856	927.397	1527.227	906.567	-196.309
5	881.445	869.339	930.449	1521.695	910.683	-180.239
6	879.705	870.293	928.661	1518.973	908.261	-196.178
7	882.302	867.933	920.235	1515.675	909.133	-182.598
8	880.012	868.06	938.024	1515.178	906.819	-183.601
9	881.591	867.024	939.798	1515.43	908.614	-193.986
10	880.036	867.042	925.199	1527.177	907.133	-198.784
11	884.382	868.551	928.61	1516.293	909.524	-193.921
12	883.023	870.725	929.824	1525.676	907.284	-181.704
13	884.181	867.846	922.269	1525.748	910.635	-188.431
14	883.859	869.679	925.713	1521.503	908.859	-180.553
15	880.727	870.73	922.924	1513.038	906.346	-188.007

Configuring Data Replay

To configure data replay, you create a Data Replay configuration object to determine which data series to use for data replay.

To configure data replay:

1 Choose Project > Logic > Datapoint Replay > Manage and click the New button.

The properties dialog for configuring data replay appears.

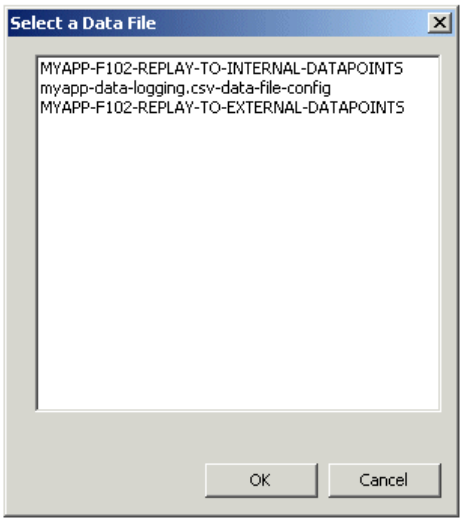
2 Configure a unique Name, which is system-generated, by default.

Tip We recommend that you prefix the name with your application name, for example, Myapp Data Replay.

---

3 Click the Add File button ( ) to display a list of available data files to replay.

The list includes all continuous and differential data files, and all log files that have been created. For example:



4 Select a data file to use for data replay.

You can add as many data series as you want to the dialog. For continuous data series, when data replay finishes replaying from the first data series, it begins replaying data from the next data series, and so on. For differential data series, it simulates data from both data series simultaneously according to the specified time differentials. You can use this feature to create more complex simulations.

5 To remove a data series from the simulation, select a file and click the Remove File button ( ) in the dialog.

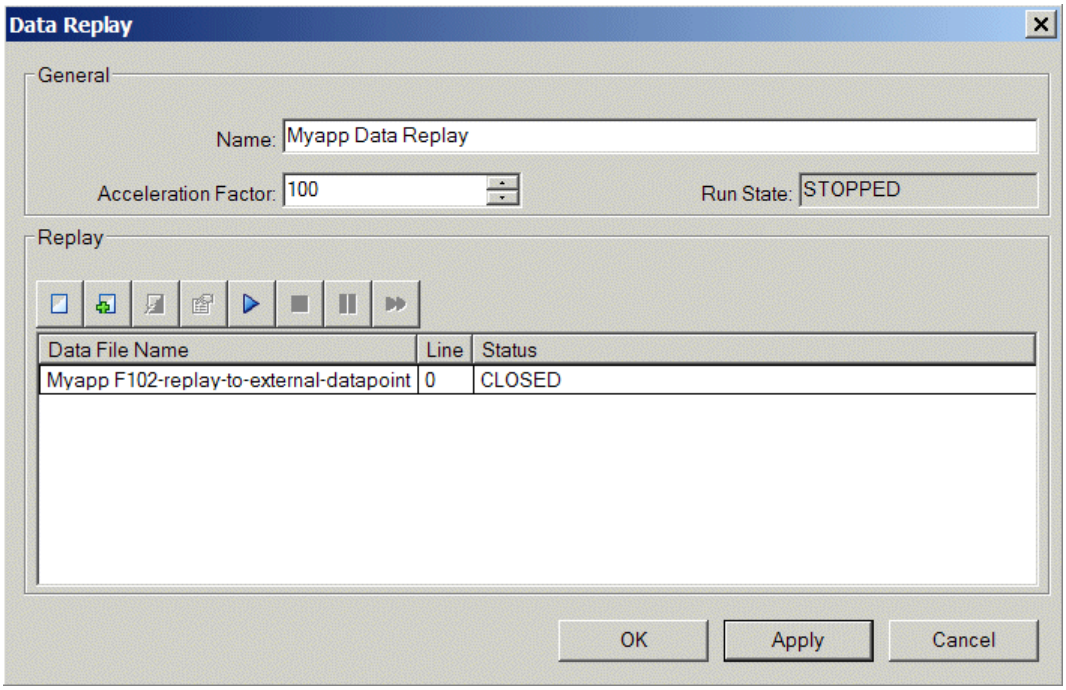
6 To create a new data series, click the New button ( ), choose the type of data series to create, and click OK.

7 Configure the properties of the new data series and add it to the list in the Data Replay dialog.

8 Configure the Acceleration Factor to speed up the rate at which data is sent.

For details, see Creating Data Series.

Here is the properties dialog for the Data Replay configuration object named Myapp Data Replay configured to replay data from a single differential data file that simulates internal datapoint values. The Acceleration Factor is set to 100.0, which, combined with an Acceleration Factor of 20.0 in the data file, means the timestamps in the CSV file are divided by 200.0.



Replaying Data from CSV Files

You control data replay from the Data Replay properties dialog or manage dialog. You can start, pause, and stop the replaying of data at any time.

Before you can replay data, you must create and configure at least one data series, create the associated CSV file, and create and configure Data Replay to use a specific data series.

Note Before you replay data from a CSV file, you must initialize the process map. For more information, see Initializing Process Maps.

When replaying data, Optegrity detects errors in the associated CSV data series, for example, a bad file name or bad data. For information about the error, see the Message Browser.

To replay data from a CSV file:

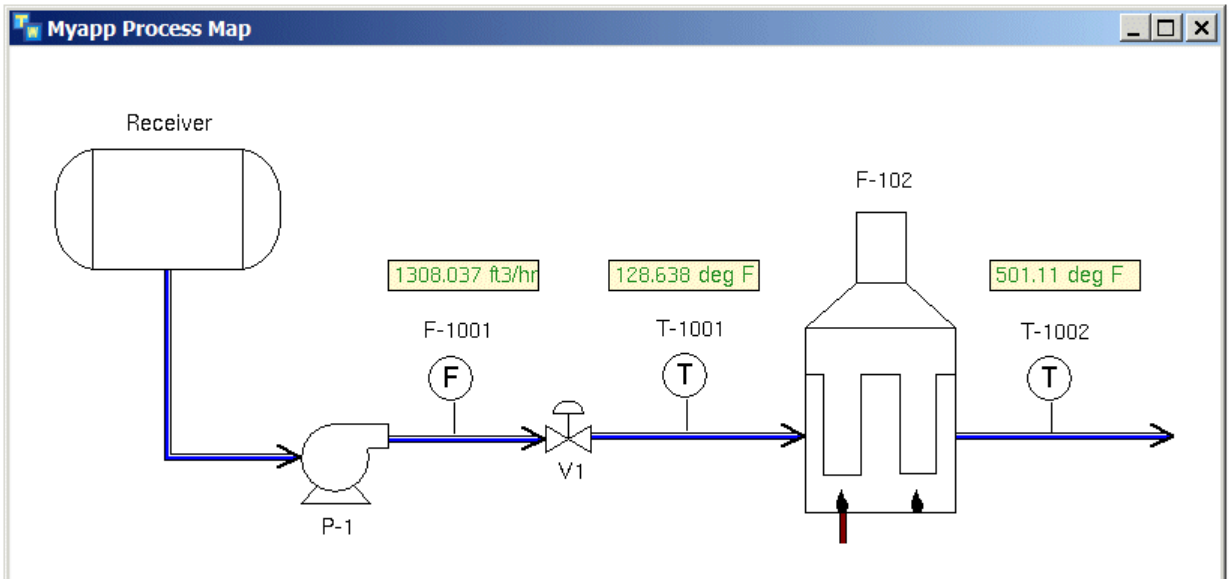
1. After initializing the process map, click the Start button () in the data replay properties dialog.

   You can also click the Start button in the manage dialog. For details, see Managing Data Replay.

   The Data Replay object sends values to the specified internal or external datapoints at the specified time interval and acceleration, depending on the type of data file. Datapoint values update.

2. To pause the simulation, click the Pause button ().
3. To continue the simulation, click the Resume button ().
4. To stop the simulation, click the Stop button ().

Here is the Myapp process map with datapoint displays that show the values of internal datapoints in the process map after running a simulation:



Displaying Trend Charts of Datapoint Values

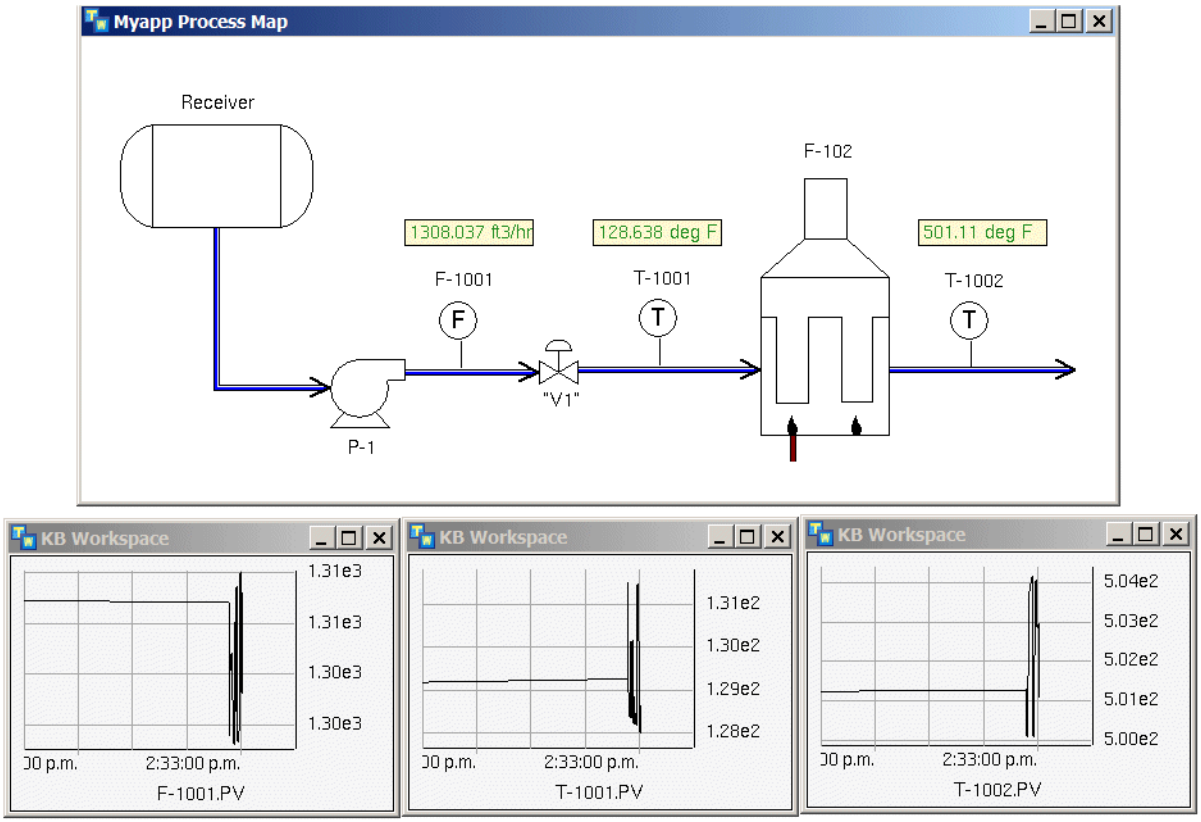
In addition to viewing datapoint values in displays, you can display trend charts of internal datapoint values so you can see a visual representation of the data as it arrives.

To display trend charts of datapoint values:

1. Display the properties dialog for the domain object whose internal datapoint you want to view in a trend chart.
2. Select a datapoint and click the Show Trend button.

Note If you create multiple trend charts, the charts are placed on top of each other. You must move the new chart to uncover the exiting charts.

When the internal datapoint receives values, the trend chart is automatically updated. For example, here is the result of running the simulation for Myapp Process Map with trend charts:



Viewing Data Validation Alarms

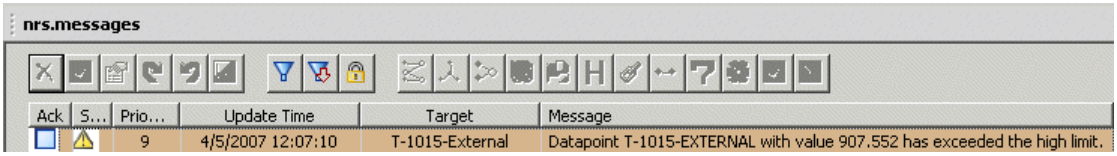
When simulating data, Optegrity performs data validation on all external datapoints that configure validation limits and targets. If the external datapoint value violates these limits, Optegrity generates an operator message, which you can view in the Message Browser.

For information about interacting with the Message Browser, see Interacting with Operator Messages.

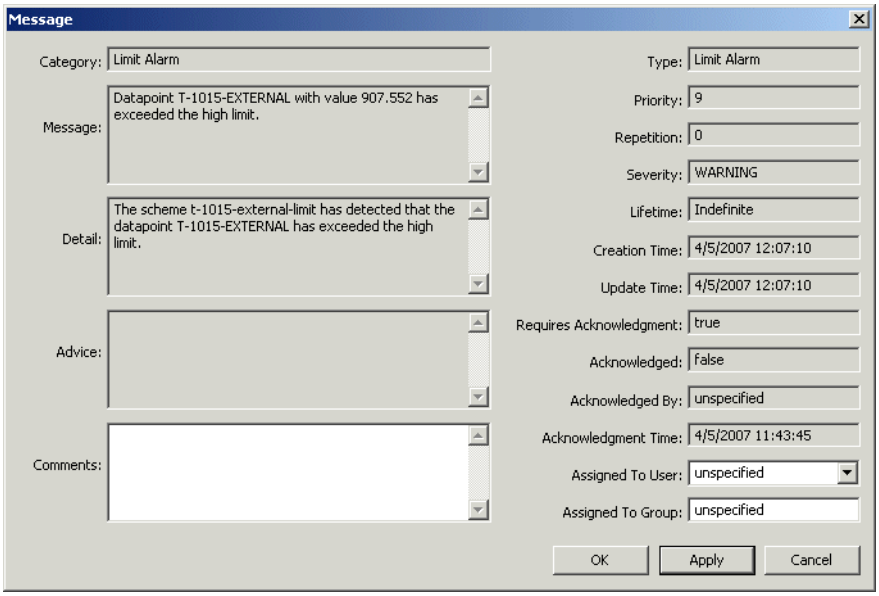
To view data validation alarms:

1. Run a simulation in which the simulated value for an external datapoint violates a data validation limit.
2. Choose View > Message Browser to view the data validation alarm.

Here is the Message Browser with a data validation message for the F-102 demo:



Here is the alarm properties:

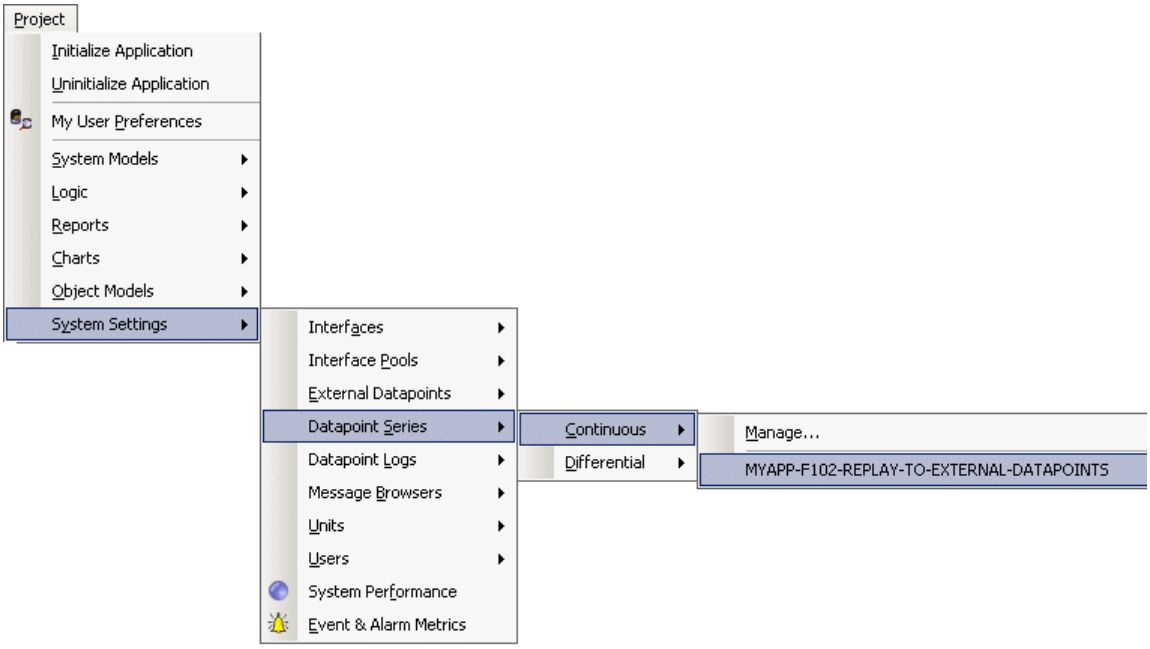


Managing Data Series

To manage data series:

1 Choose Project > System Settings > Datapoint Series > Continuous Data Series or Differential Data Series.

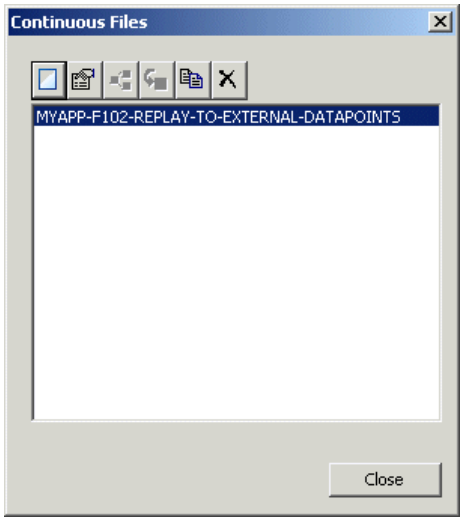
All data series appear in the submenu, for example:



2 To go to a data series, choose one from appropriate submenu of the Data Series menu.

3 To display a dialog for managing data files, choose Manage.

Here is the dialog for managing data series:

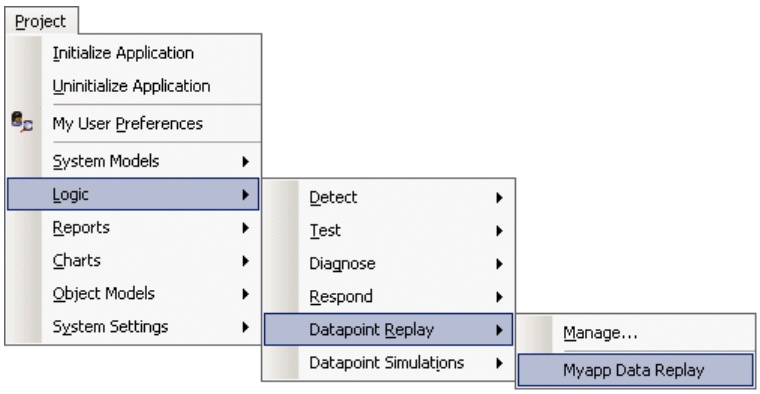


Managing Data Replay

To manage data replay:

1 Choose Project > Logic > Datapoint Replay.

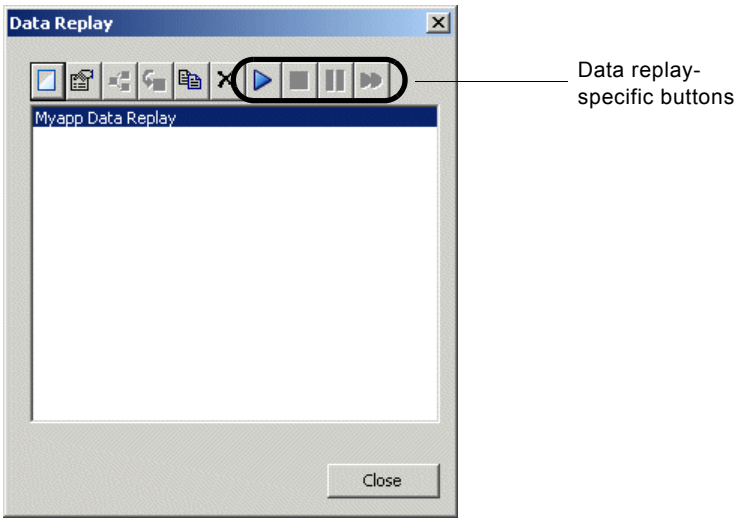
All Data Replay configuration objects appear in the submenu, for example:



2 To configure the properties of a data replay configuration object, choose one from the Data Replay submenu.

3 To display a dialog for managing data replay, choose Manage.

Here is the Data Replay Manage dialog:



For information on using this dialog and the Project menu to manage data series and data replay, see Using the Project Menu.

For information on using the buttons specific to data replay, see Performing Specific Operations.

13

Simulating Datapoint Values

Describes how to simulate values for internal and external datapoints.

Introduction

You can use data drivers to simulate internal or external datapoint data. This feature provides an alternative to using data replay to simulate datapoint data from a CSV file.

The data driver provides the ability to:

- Simulate external or internal datapoint data.
- Specify the time interval for updating datapoint values.
- Specify the average value and noise.
- Simulate state transitions.
- Selectively activate and deactivate the simulation.

You can create one or more data drivers for individual datapoints.

Creating a Simple Data Simulation

Before you create a data simulation, you must first create either:

- A sensor or controller whose process value (pv), set point (sp), or controller output (op) you want to simulate.
- An external datapoint that is the source datapoint for the internal datapoint of any type of domain object.

The simplest way to simulate data is to create a single data driver that simulates datapoint values around an average, with noise. The data driver configures:

- The name of the sensor object and its associated datapoint (pv, sp, or op), or the name of the external datapoint whose values you want to simulate.
- The average value and the signal noise.

For information on creating sensors and controllers, see Building a Process Map.

For information on creating external datapoints, see Configuring External Datapoints.

To create a simple data simulation:

1. Create a sensor or controller whose datapoint you want to simulate, or create an external datapoint that is the source datapoint for an internal datapoint of any type of domain object.
2. Choose Project > Logic > Datapoint Simulations > Manage and click the New button.

   The properties dialog for configuring the Data Driver appears.
3. Configure the Organizer Name.
4. Click the New button ( ) to create a new data driver.

   The properties dialog for configuring the data driver appears.
5. Configure these attributes, depending on the type of data driver:

Attribute	Description
Datapoint Name	The name of an internal or external datapoint whose data should be simulated.
Active	Whether the simulation is currently active.
Average Value	The average value for the specified datapoint.

Attribute	Description
Noise	The maximum value above and below the average that represents noise in the current datapoint value.
Interval Seconds	The time interval, in seconds, for updating datapoint values.

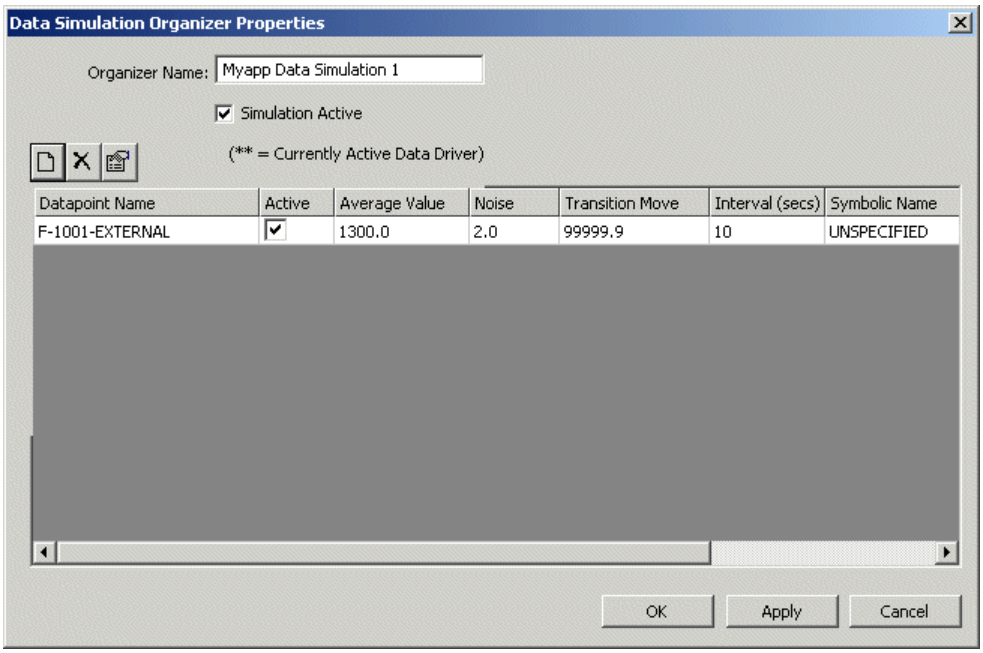
6 Create as many data drivers as you need.

The simulation is disabled, by default. You must explicitly enable it to simulate data.

7 Enable the Simulation Active toggle button and click the Apply button.

Tip You can also activate and deactivate the simulation from the Manage dialog by selecting a Data Simulation and clicking the Activate and Deactivate buttons.

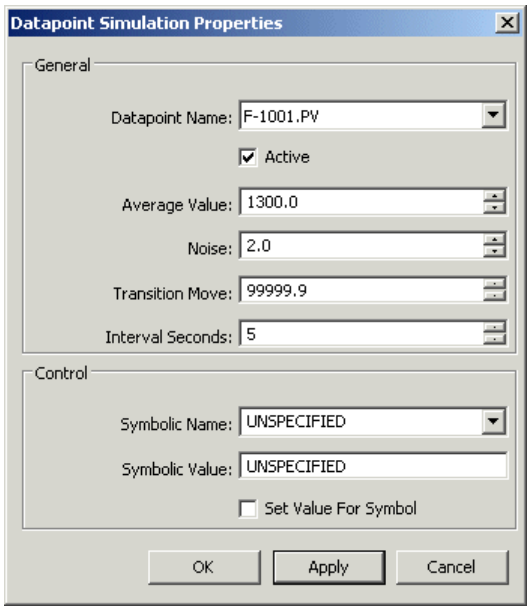
The properties dialog for the data simulation shows all associated data drivers. You can also delete a data driver from the data simulation and display its properties, using the Delete and Properties toolbar buttons. For example:



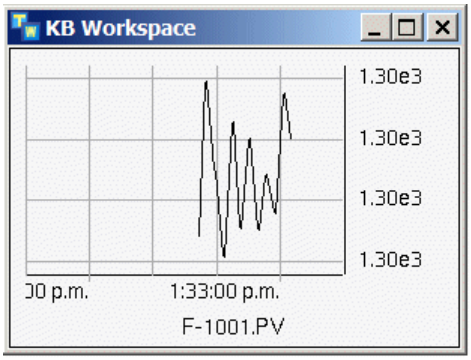
For information on configuring the other data driver attributes and on using the other toolbar buttons, see Creating a Data Simulation with Transitions.

Example: Internal Datapoint Simulation for a Sensor

This example shows a simple data simulation that is configured to simulate data for the pv of a flow sensor named f-1001. The data values update once every five seconds, and the simulated values range between 1300 +/- 2.

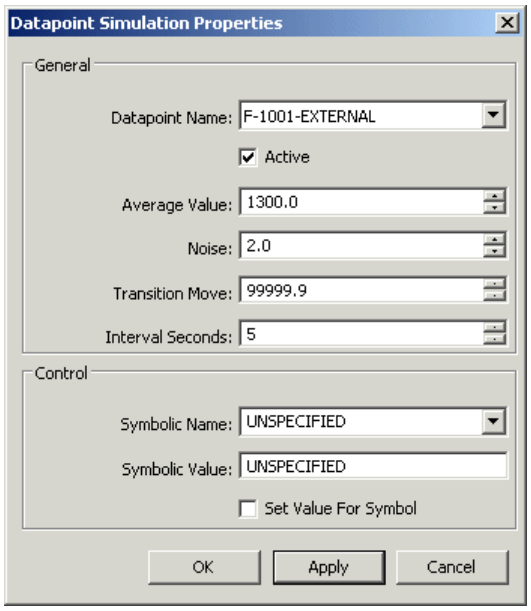


Here is the trend chart for the f-1001 sensor, which shows the history of the pv datapoint:

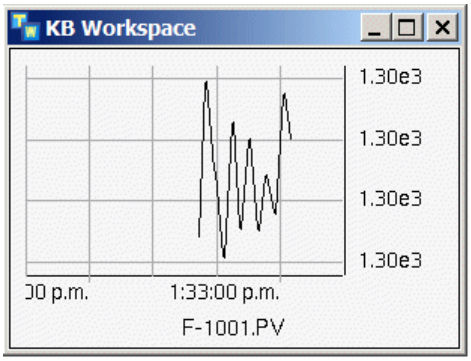


Example: External Datapoint Simulation for a Sensor

This example shows a simple data simulation for an external datapoint named f-1001-external, which is the source datapoint of the flow sensor named f-1001. The pv of the f-1001 sensor gets its data from the external datapoint. The data driver is configured to simulate data for the f-1001-external, using the same update interval, average, and noise as the internal data simulation.



Here is the properties dialog for the flow-sensor, which shows the history of the pv datapoint, which gets its data from the external datapoint named flow-dp:



Creating a Data Simulation with Transitions

Rather than creating a simple data simulation with a single data driver, you might want to create multiple data drivers for a single datapoint, each of which represents a transition from one state to another. For example, you might create a data driver that represents the normal state, another that represents a high state, another that represents a severe change, another that represents a noisy signal, and another that represents a flat-line.

You create data simulations with transitions by creating a data driver for each state, a symbolic parameter that represents the current state, and buttons for choosing each state. In addition to the basic configuration information, you must configure each data driver with the following information:

- The name of a symbolic parameter that determines the current state.
- The symbolic state associated with the data driver.
- The transition value to use when switching to the specified symbolic state.

Before you create a data simulation with transitions, you must first create the internal or external datapoint whose data you want to simulate. For details, see Creating a Simple Data Simulation.

To create a data simulation with transitions:

1. Create and configure a data driver for an internal or external datapoint that represents a normal state.

   For details, see Creating a Simple Data Simulation.

2. Configure these additional attributes for the data driver:

Attribute	Description
Transition Move	A value by which the datapoint should increment, either up or down, when making a state transition.
Symbolic Name	The name of a symbolic parameter that provides state transition values for the specified datapoint.
Symbolic Value	The value of the specified symbolic parameter associated with the specified state.

Note If the Symbolic Name does not exist, Optegrity creates a symbolic parameter of that name.

For example, you might specify Symbolic Name of flow-status, whose Symbolic Value is normal to represent the normal operating state of the datapoint.

The Transition Move is the value to increment the datapoint when transitioning back to a normal state from some other state. Typically, you specify a large transition, relative to the average values of each transition state, so as to return to the normal average relatively quickly.

For example, if the normal average is 100 and the high average is 120, a Transition Move of 10 would cause the datapoint value to return to the normal average in approximately two transitions, depending on the specified noise. On the other hand, a Transition Move of 2 would cause the datapoint value to return to normal in approximately ten transitions.

3 Create and configure a data driver for the same datapoint to represent the transition to another state.

For example, to simulate a high state, the Average Value might be 120, and the Symbolic Value might be high. The Transition Move determines how quickly the simulation moves from the normal state to the high state. To simulate a projected high, you would configure the transition to be a small increment, relative to the average values of each state. Thus, a Transition Move of 1.0 would cause the datapoint value to rise to the high average in approximately 20 steps.

4 Create and configure additional data drivers for the same datapoint to represent transitions to additional states, as needed.

For example, to simulate a severe change, configure the Transition Move to be a large number, relative to the average values of each state, such as 20, which would cause the datapoint value to spike to the specified average in one transition.

To simulate a noisy signal, configure the Noise to be a large number relative to the average, and to simulate a flat-line, configure the Noise to be zero. In both cases, you would configure the Average Value and Transition Move to be the same as the normal signal, 100.0 and 10.0, so the noisy and flat-line signal are based on a normal signal.

Tip In general, you should set the Noise to be less than the Transition Move; otherwise, the noise counteracts the transition and causes the datapoint value to take a long time to transition to the new state.

---

5 To set the current state, display the properties dialog for a particular data driver and enable the Set Value for Symbol option.

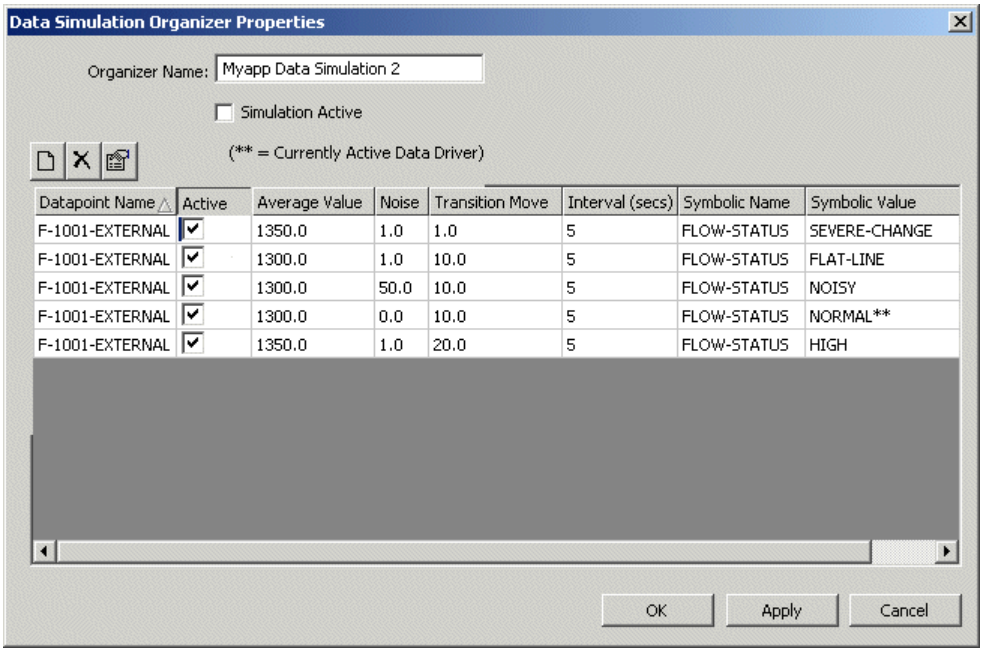
In the properties dialog for the Data Simulation, the Symbolic Value has ** next to the value to indicate that it is the currently active data driver.

6 Enable the Simulation Active toggle button in the Data Simulation properties dialog to activate the data drivers, and click OK or Apply.

The data driver with the default state determines the datapoint values to use for the simulation.

Example: External Datapoint Simulation with Transitions

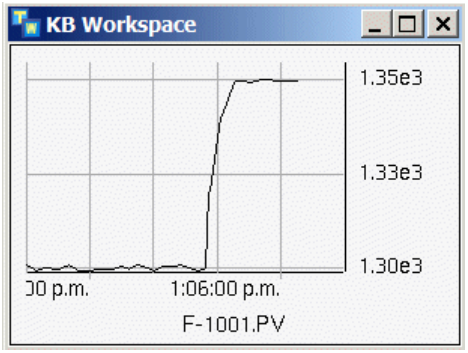
This example shows a data simulation for an external datapoint with five states for normal, high, severe change, noisy, and flat-line, where the current state is normal:



This table summarizes the values of the relevant attributes for each data driver:

	Normal	High	Severe Change	Noisy	Flat-Line
Symbolic Parameter Value	normal	high	severe-change	noisy	flat-line
Average Value	1300.0	1350.0	1350.0	1300.0	1300.0
Noise	1.0	1.0	1.0	50.0	0.0
Transition Move	10.0	1.0	20.0	10.0	10.0

Here is a trend chart for the f-1001 sensor, which shows the pv history for a normal state that transitioned to a severe change state:

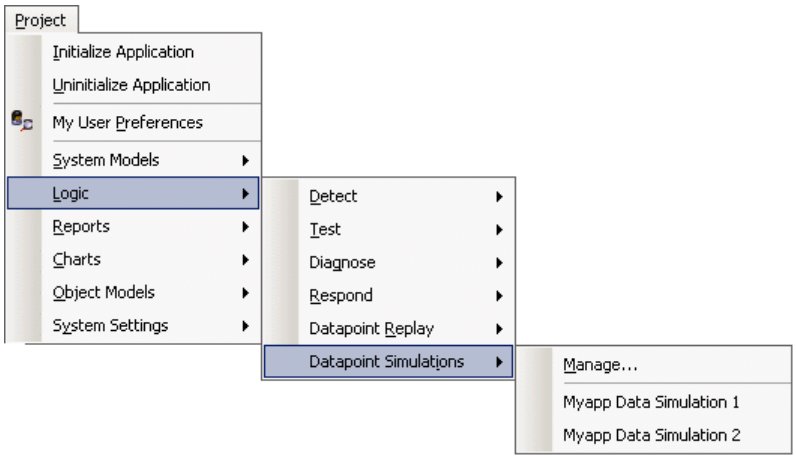


Managing Data Simulations

To manage data simulations:

1 Choose Project > Logic > Datapoint Simulations.

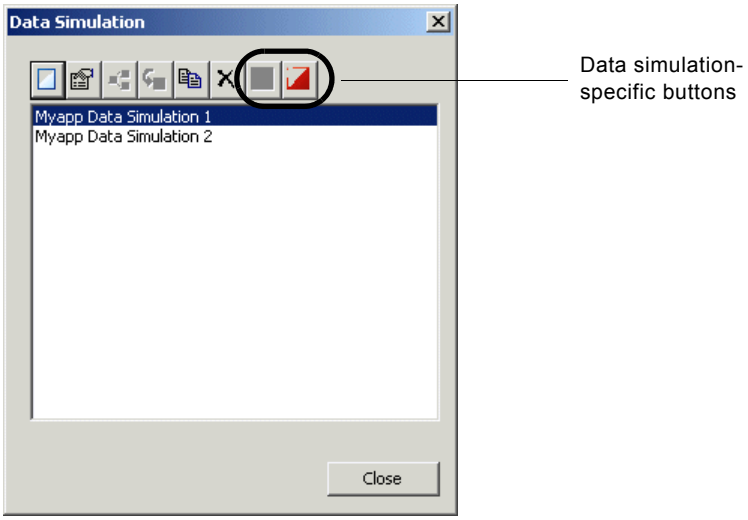
All data simulations appear in the submenu, for example:



2 To configure the properties of a data simulation, choose one from the Data Simulations submenu.

3 To display a dialog for managing all data simulations, choose Manage.

Here is the Data Simulation Manage dialog:



For information on using this dialog and the Project menu to manage data simulations, see Using the Project Menu.

For information on using the buttons specific to data simulation, see Performing Specific Operations.

Part IV

Event Detection

Chapter 14: Creating Generic Dataflow Diagrams

Describes how to create generic dataflow diagrams for domain object classes that detect, test, and respond to application events.

Chapter 15: Initializing Process Maps

Describes how to initialize process maps to create specific event detection diagrams for each domain object.

Chapter 16: Reporting and Charting

Describes how to generate event metrics reports and system performance reports, as well as charts.

14

Creating Generic Dataflow Diagrams

Describes how to create generic dataflow diagrams for domain object classes that detect, test, and respond to application events.

gensym

Introduction

When the intelligent object libraries are loaded, Optegrity provides a variety of built-in generic dataflow diagrams, which detect common events on sensors and process equipment. These dataflow diagrams monitor internal datapoints for domain objects in a process map, test the values against some kind of criteria, and generate some kind of event when the criteria is met.

The event detection diagrams are defined generically for classes of domain objects. When you initialize the domain map, Optegrity creates specific event detection diagrams for each instance of the target class.

You create also create your own generic dataflow diagrams that detect, test for, and respond to other types of events that are relevant in your particular process.

You create dataflow diagrams, using blocks in the various palettes in the Event Detection toolbox. The diagrams flow from left to right. The first blocks are entry points, which can:

- Subscribe to datapoints in an Optegrity process map to obtain data values from domain objects.
- Generate continuous data, for example, a real-time clock signal.

A dataflow diagram can trigger SymCure diagnostics by sending SymCure events. A dataflow diagram can get and set values of domain objects. Dataflow diagrams can also post low-level notifications and operator messages.

You can also create custom domain object events, as described in Creating Custom Event Detection.

For detailed information about using the G2 Event and Data Processing (GEDP) module to create dataflow diagrams, see the *G2 Event and Data Processing User's Guide*.

Creating Generic Dataflow Template Folders

You create generic dataflow templates on the detail of a generic detection template folder. The template applies to a domain object definition, as specified by its target class.

You can create generic dataflow templates for event detection, testing, or response. By creating dataflow templates in one of these categories, you both organize application logic, as well as provide the ability to run diagrams in each category associated individual domain objects.

To create a generic dataflow template folder:

1. Choose Project > Logic > Detect > Dataflow Templates > Manage and click the New button.

   Similarly, to create a generic diagram template for testing or response, choose Project > Logic > Test or Respond, respectively.

   A properties dialog for configuring the generic template appears.

2. Configure the Template Name to be any text value.

   The name can include spaces, for example, O2 Monitoring.

3. Configure the Target Class to refer to a domain object definition to which the generic diagram applies.

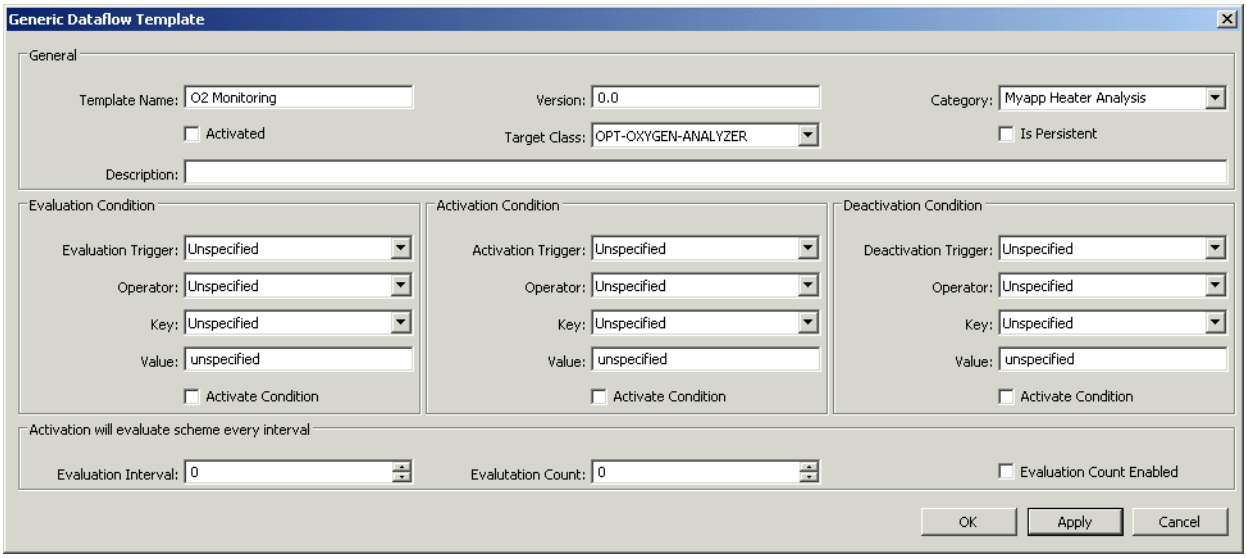
   For example, to obtain datapoint values and generate events for the Draft Oxygen related sensor of a heater, the target class would be opt-oxygen-analyzer.

4 Optionally, configure the Version to be any user-defined value that indicates the diagram revision, for example, 1.1, 1.2, 1.3.

5 Configure the Category to be a system-defined category or enter any user-defined text to define a new category.

The event detection diagram appears in the specified category in the Project > Logic > Detect, Test, or Respond submenu.

Here is the properties dialog for the generic dataflow template named O2 Monitoring, which is defined for the opt-oxygen-analyzer target class in the Myapp Heater Analysis category:

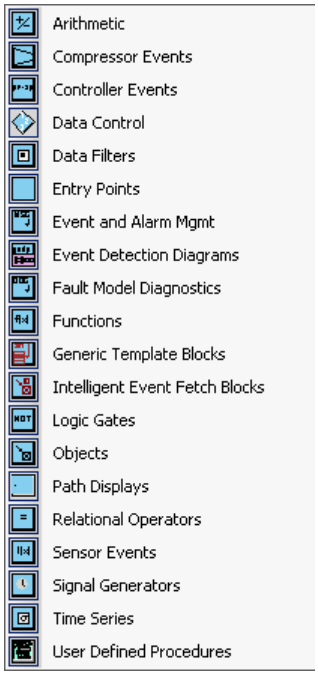


Creating Generic Dataflow Templates

A generic dataflow template:

- Obtains data and events from domain objects in a process map in form of entry points.
- Optionally filters data, performs calculations, and reasons over time.
- Makes logical inferences about the data.
- Generates operator messages and SymCure events, based on the inferences.

You create generic dataflow templates by using these palettes in the Event Detection toolbox:



Palette	Description
Arithmetic	Perform arithmetic operations on data values.
Compressor Events	Built-in event detection blocks for compressors. This palette is only available when the intelligent compressor library is loaded.
Controller Events	Built-in event detection blocks for controllers. This palette is only available when the intelligent controller library is loaded.

Palette	Description
Data Control	Control how data flows through a diagram.
Data Filters	Filter out noise and find data trends.
Entry Points	Provide float, integer, boolean, text, and symbolic values to the diagram. You can specify the value directly, or you can specify a source datapoint from which the entry point obtains its data.
Event and Alarm Mgmt	Generate, manage, and listen for low-level notifications and operator messages related to specific domain objects.
Fault Model Diagnostics	Trigger SymCure events that apply to specific object instances.
Functions	Perform statistical operations on data values, including user-defined functions.
Generic Template Blocks	Listen for property changes, set property values, and generate low-level notifications, messages, and SymCure events for domain objects that are the target class of a generic event detection template.
Intelligent Event Fetch Blocks	Get domain objects. This palette is available when any intelligent object library is loaded.
Logic Gates	Perform logical operations on boolean values.
Objects	Get specific domain objects, and get and set property values of those objects.
Path Displays	Display the content of any path.
Relational Operators	Perform comparisons on incoming data values and pass boolean values, based on the result of the test.
Sensor Events	Built-in events for sensors. This palette is only available when the intelligent sensor library is loaded.
Signal Generators	Provide sample data for testing GEDP diagrams.
Time Series	Perform operations on data histories.

To create a generic event detection template:

1. Go to the detail of a generic dataflow template folder.

   For information on how to do this, see Managing Dataflow Templates and Diagrams.

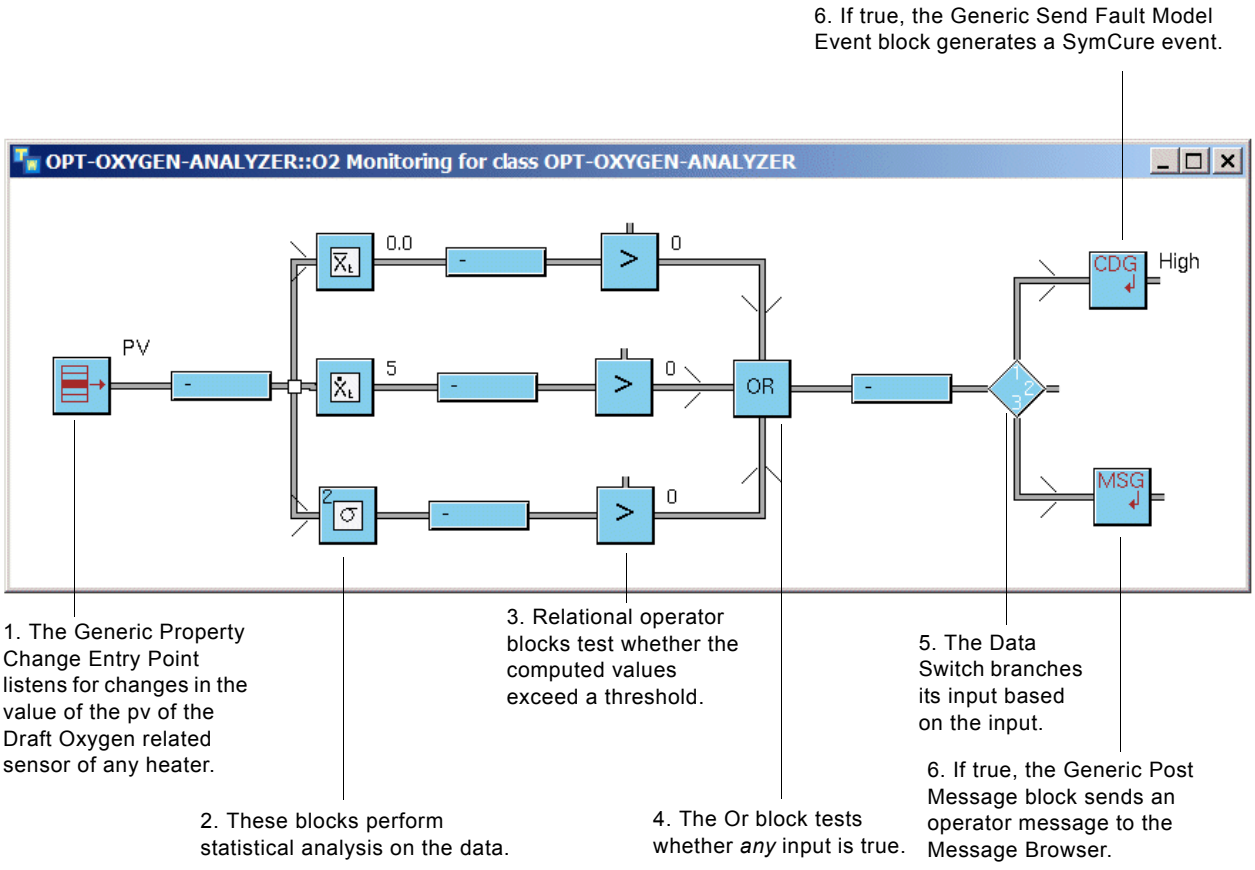
2. Choose View > Toolbox - Event Detection and choose a palette.

3. For each block required in the template, clone it from its palette and place it on the generic template.

4. Connect the blocks together.

Note When you first connect blocks from the Generic Template Blocks palette, the icon turns partially red, indicating that the block is not yet initialized. After you have configured the model, you can initialize the blocks.

Here is a generic dataflow template for event detection for the opt-oxygen-analyzer class:



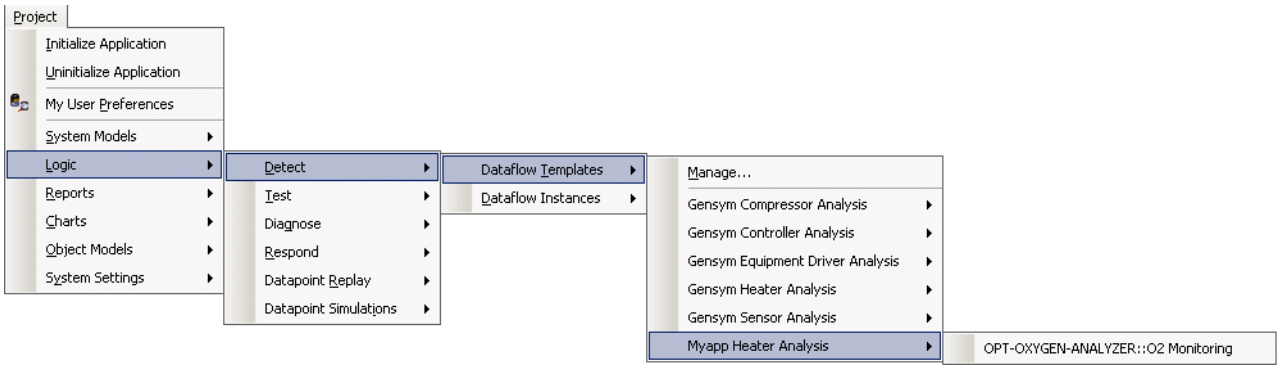
For a description of the blocks in this diagram, see the *G2 Event and Data Processing User's Guide*.

Managing Dataflow Templates and Diagrams

To manage dataflow diagrams and templates:

1 Choose Project > Logic > Detect/Test/Respond > Dataflow Templates or Dataflow Instances, then choose a category.

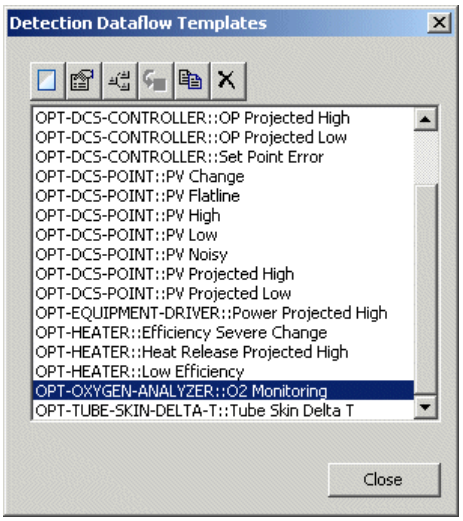
All generic dataflow templates or specific dataflow instances appear in the submenu under the specified category or in the Unspecified submenu if no category is assigned. Here is the Project > Logic > Detect > Dataflow Templates submenu with the O2 monitoring generic dataflow template defined in the Myapp Heater Analysis category:



2 To display the detail of an event detection template or diagram, choose one from the appropriate category of the Dataflow Templates or Dataflow Instances submenu of the Detect menu.

3 To display a dialog for managing all event detection diagrams and templates, choose Manage.

Here is the Detection Dataflow Manage dialog, which includes the built-in generic dataflow templates:



For information on using this dialog and the Project menu to manage event detection templates and diagrams, see Using the Project Menu.

15

Initializing Process Maps

Describes how to initialize process maps to create specific event detection diagrams for each domain object.

Introduction

Before you can run an Optegrity application or replay data for simulation purposes, you must initialize the process map. Initializing process maps performs these tasks:

- Creates specific GEDP diagrams for each domain object with an associated generic diagram template.
- Creates and initializes external datapoints and relates them to internal datapoints.
- Resets datapoint histories.
- Compiles all SymCure fault model folders.
- Clears all diagnoses and their messages from the various message browsers.

Note Each time you restart your application, you must initialize the process map.

Note If you change the name of a domain object after your application has been deployed, Optegrity automatically updates messages and SymCure events to reflect this change. However, we recommend that you initialize process maps after any domain object name change.

Initializing Process Maps

You can initialize each domain object individually through their menu options, or you can initialize all domain objects simultaneously using the Project menu. You can also initialize all GEDP blocks in a diagram or individual GEDP blocks.

Note If the generic event detection template is configured to be persistent, initializing does not create new specific event detection diagrams. Instead, the specific event detection diagrams persist when the process map is reinitialized to allow persistent configuration of events for specific domain objects. For details, see the *G2 Event and Data Processing User's Guide*.

To initialize all domain objects in all process maps:

➔ Choose Project > Initialize Application.

To initialize all domain objects in a process map:

➔ Choose Initialize Domain Objects on the process map detail.

To initialize individual domain objects:

➔ Choose Initialize Domain Object on an individual domain object in a process map.

To initialize all GEDP blocks in a diagram:

➔ Choose Initialize Blocks on the generic diagram template detail that contains GEDP blocks.

To initialize individual GEDP blocks in a diagram:

➔ Choose Initialize on a GEDP block in an diagram folder or generic diagram template folder.

Showing Specific Dataflow Diagrams

When a domain object has been initialized, you can show its specific dataflow diagrams.

To show specific dataflow diagrams:

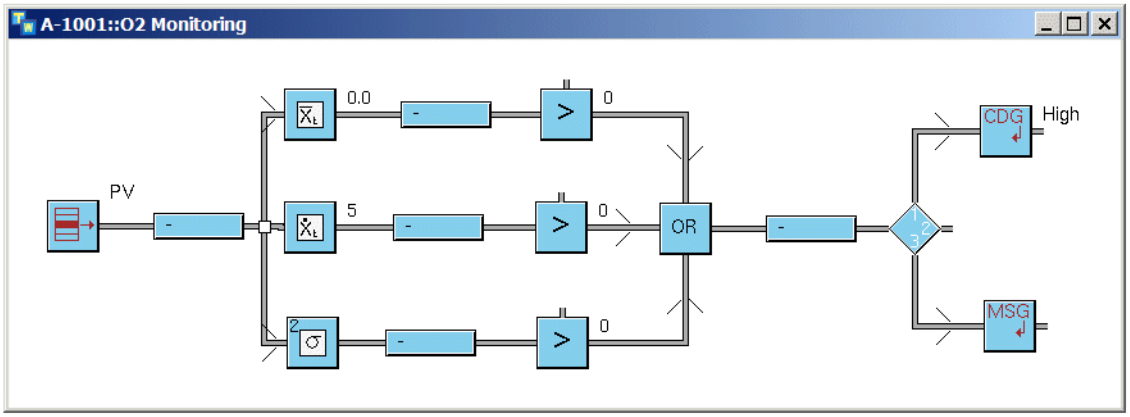
1 Choose Show Logic on a domain object in a process map.

A dialog with all associated specific dataflow diagrams for the domain object appears.

This menu choice only appears when the domain object has been initialized and has any associated specific dataflow diagrams.

2 Select a specific dataflow diagram from the dialog and click OK to go to its detail.

Here is the specific event detection diagram associated with the A-1001 oxygen analyzer, which looks identical to the generic event detection template except for the title, which refers to the specific oxygen analyzer:



Uninitializing Process Maps

Uninitializing process maps deletes specific event detection diagrams for domain objects in all process maps. You can also uninitialize individual domain objects. You might need to uninitialize a process map if you change the GEDP diagram for a domain object.

Note If the generic event detection template is configured to be persistent, uninitializing does not delete specific event detection diagrams. Instead, the specific event detection diagrams persist when the process map is uninitialized and reinitialized to allow persistent configuration of events for specific domain

objects. You can explicitly delete persistent specific event detection diagrams, if necessary. For details, see the *G2 Event and Data Processing User's Guide*.

To uninitialize all domain objects in all process maps:

➔ Choose Project > Uninitialize Application.

To uninitialize all domain objects in a process map:

➔ Choose Uninitialize Domain Objects on the process map detail.

To uninitialize individual domain objects:

➔ Choose Uninitialize Domain Object on an individual domain object in a process map.

16

Reporting and Charting

Describes how to generate event metrics reports and system performance reports, as well as charts.

Introduction

Optegrity provides two types of built-in reports:

- Event metrics — Calculates frequency and duration statistics about events, based on the event type, event target, and event category. You can view event metrics on an hourly, daily, or monthly basis.
- System performance — Calculates various statistics related to system performance, including the count of various types of objects such as events, messages, and errors, memory and scheduler statistics.

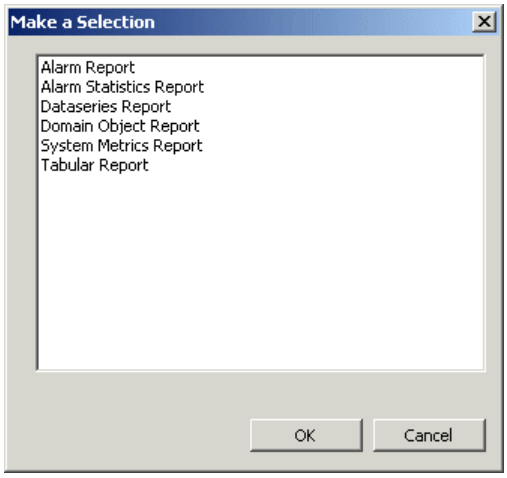
You can save event metrics to a file or to a database.

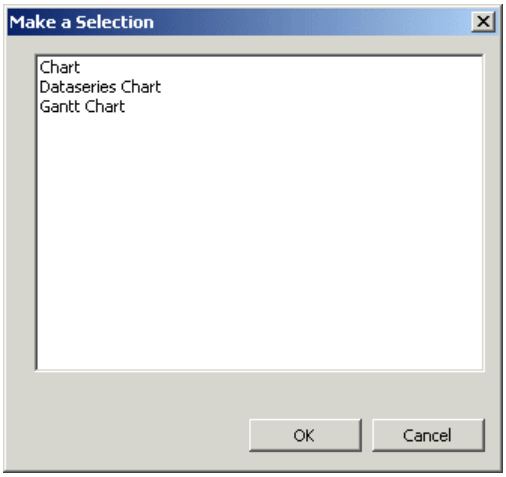
By default, reporting is disabled; you must explicitly enable both types of reports.

You can also create and configure various types of standard reports and charts, using the G2 Reporting Engine (GRPE).

Creating GRPE Reports and Charts

You can create the following standard reports and charts through the Project menu or Navigator:





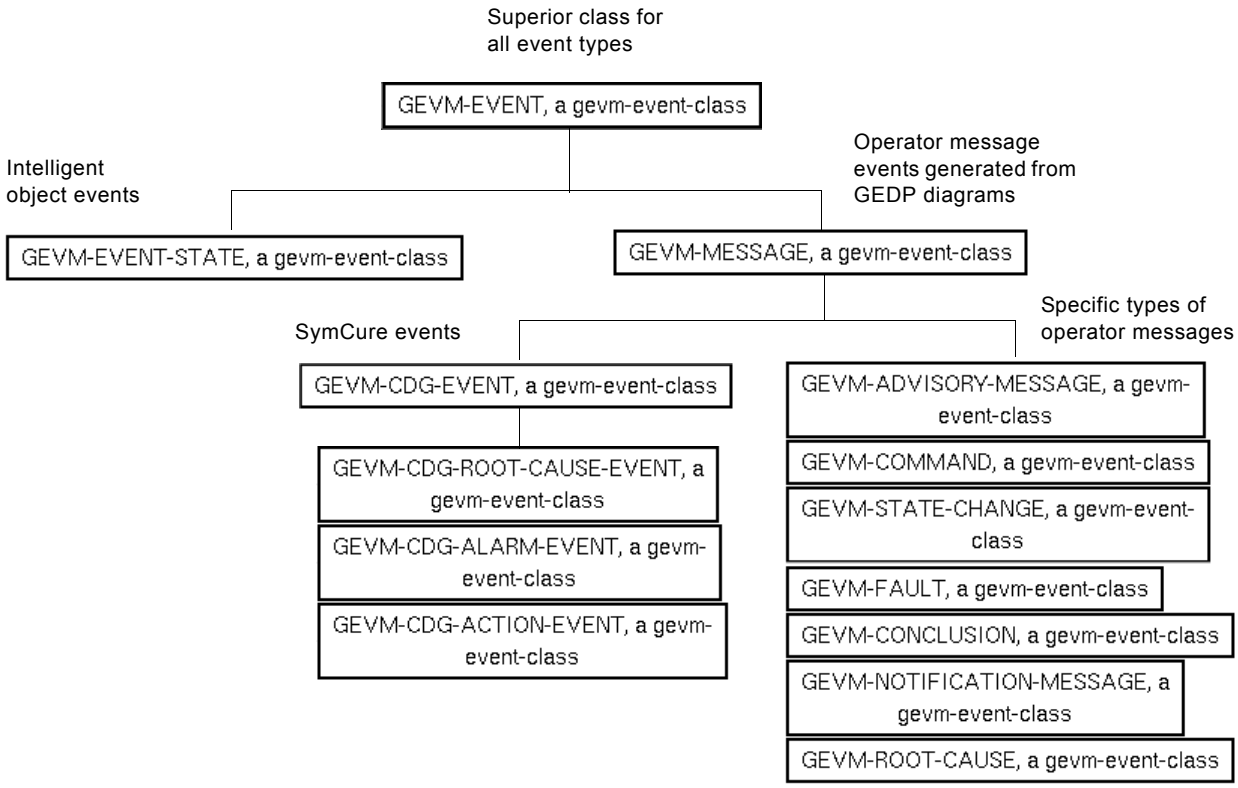
For information on creating and configuring these types of reports and charts, see the *G2 Reporting Engine User's Guide*.

Configuring Event Metrics Reports

You can configure event metrics reports to generate metrics for various event types, including domain object events and SymCure events, low-level notifications and operator messages, and general errors, for example, messages from bridge failures.

Note Optegrity computes metrics for events of the specified types only, not subclasses of those types.

Here is a partial class hierarchy of gevm-event, the superior class of all event types:



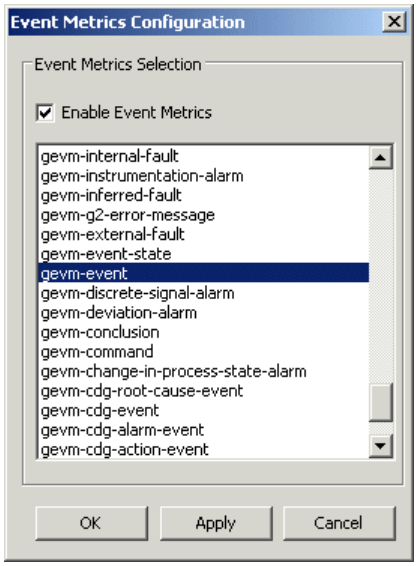
To configure event metrics reports:

1. Choose Project > System Settings > Event & Alarm Metrics.
2. Choose one or more event types, whose metrics you want to compute.

   Use the Shift key to select multiple event types.
3. Enable the Enable Event Metrics option.

When events of the specified type occur, Optegrity generates the event metrics in a report.

Here is the properties dialog for configuring event metrics, which computes statistics for all subclasses of gevm-event:



In System-Administrator mode, you can also log event metrics to a file and/or to a database. The database interface must exist before you can log events to a database. In most cases, the database table must also exist. See note below.

To log event metrics to a file and/or database:

1. Switch to System-Administrator mode.

   For details, see Switching User Modes.

2. Create and configure a database interface for message logging.

   For details, see Creating and Connecting Network Interfaces.

3. Choose Project > System Settings > Event & Alarm Metrics.

4. Choose one or more event types, whose metrics you want to compute.

5. Enable the Enable Event Metrics option.

6. To log metrics to a file, enable the Log to File option and specify the Log Directory to be the directory name in which to create the log files.

   Optegrity creates log files named *hourly-metrics.csv*, *daily-metrics.csv*, and *monthly-metrics.csv*. Each report contains columns for the event target, event type, and event category, as well as frequency and duration columns for 24 hours, 31 days, and 12 months, respectively, based on the type of report.

7. To log metrics to a database, enable the Log to Database option and select a database interface object to use as a database bridge.

By default, Optegrity writes hourly, daily, and monthly metrics to the specified database tables.

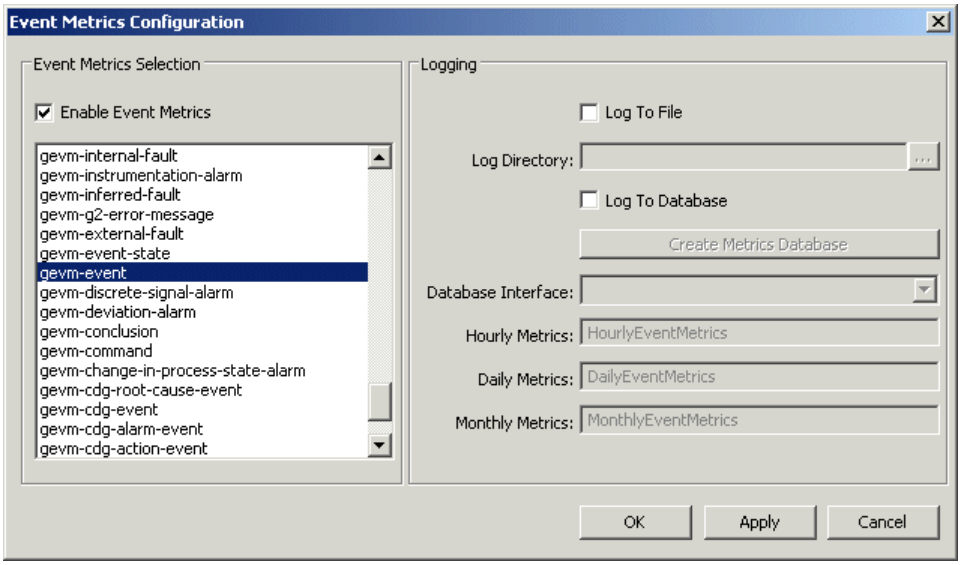
8 Configure Hourly Metrics, Daily Metrics and Monthly Metrics to specify the name of a database table in which to write the event metrics, or use the defaults.

The database table must exist before you can log metrics to the database.

9 If the database table does not already exist, click Create Metrics Database to create the specified database table.

Note The Create Metrics Database button only works for the SQL Server database when using the G2-ODBC Bridge. If the G2-ODBC Bridge is connecting to any other database, such as Access or Oracle, then you must create the database table manually, because the database data types of the fields in the table are not common to all databases. You can also customize the procedure that creates the database table for your particular database. For more information, see the *G2 Database Bridge User's Guide*.

Here is the properties dialog for configuring event metrics to log to a file or database:



For information on...	See...
Generating built-in events on domain objects	Configuring Domain Objects
Generating SymCure events	*SymCure User's Guide*
Creating dataflow diagrams and templates for generating low-level notifications and operator messages	*G2 Event and Data Processing User's Guide*

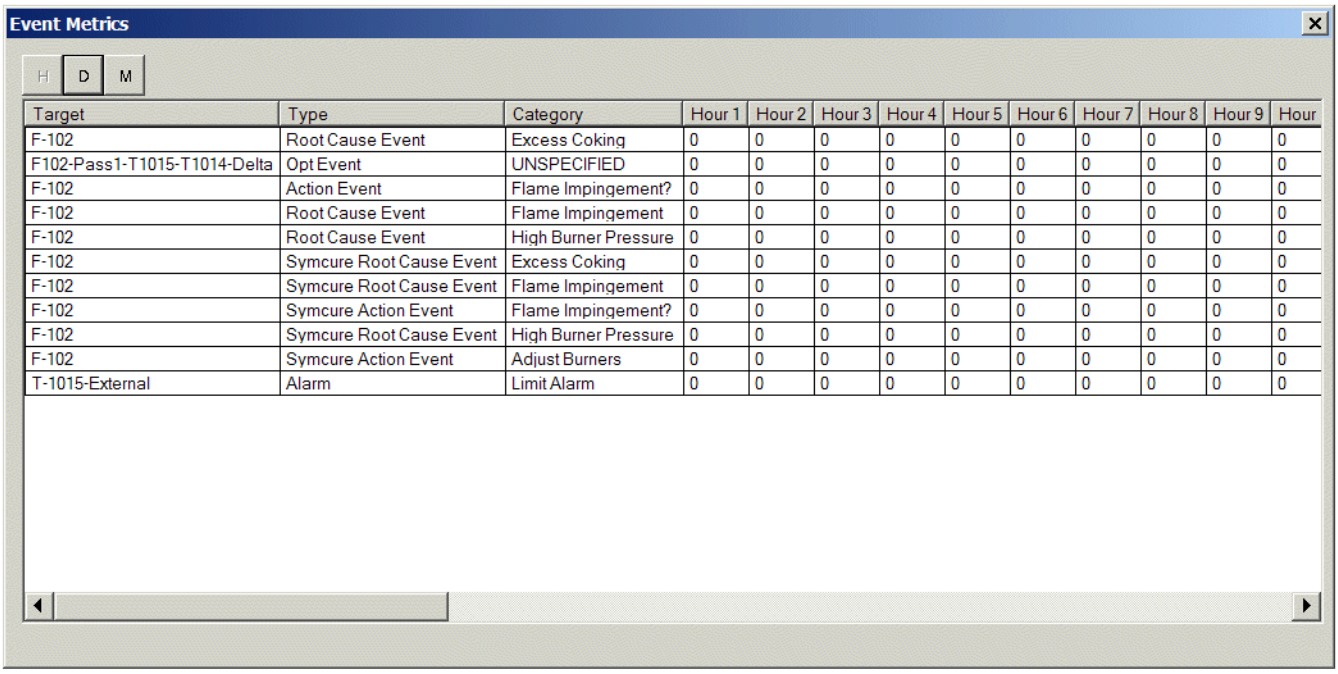
Viewing Event Metrics Reports

You can view event metrics reports in modeler mode or operator mode.

To view event metrics reports in modeler mode:

1 Choose Project > Message Queues > Metrics > Hourly Event Metrics, Daily Event Metrics, or Monthly Event Metrics.

Here is an hourly metrics report for the `f102demo.kb` application after running the simulation:



Target	Type	Category	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6	Hour 7	Hour 8	Hour 9	Hour
F-102	Root Cause Event	Excess Coking	0	0	0	0	0	0	0	0	0	0
F102-Pass1-T1015-T1014-Delta	Opt Event	UNSPECIFIED	0	0	0	0	0	0	0	0	0	0
F-102	Action Event	Flame Impingement?	0	0	0	0	0	0	0	0	0	0
F-102	Root Cause Event	Flame Impingement	0	0	0	0	0	0	0	0	0	0
F-102	Root Cause Event	High Burner Pressure	0	0	0	0	0	0	0	0	0	0
F-102	Symcure Root Cause Event	Excess Coking	0	0	0	0	0	0	0	0	0	0
F-102	Symcure Root Cause Event	Flame Impingement	0	0	0	0	0	0	0	0	0	0
F-102	Symcure Action Event	Flame Impingement?	0	0	0	0	0	0	0	0	0	0
F-102	Symcure Root Cause Event	High Burner Pressure	0	0	0	0	0	0	0	0	0	0
F-102	Symcure Action Event	Adjust Burners	0	0	0	0	0	0	0	0	0	0
T-1015-External	Alarm	Limit Alarm	0	0	0	0	0	0	0	0	0	0

For Hour 13, the report looks like this, which indicates that events occurred:

Target	Type	Category	Hour 13
F-102	Root Cause Event	Excess Coking	1
F102-Pass1-T1015-T1014-Delta	Opt Event	UNSPECIFIED	1
F-102	Action Event	Flame Impingement?	1
F-102	Root Cause Event	Flame Impingement	1
F-102	Root Cause Event	High Burner Pressure	1
F-102	Symcure Root Cause Event	Excess Coking	1
F-102	Symcure Root Cause Event	Flame Impingement	1
F-102	Symcure Action Event	Flame Impingement?	1
F-102	Symcure Root Cause Event	High Burner Pressure	1
F-102	Symcure Action Event	Adjust Burners	1
T-1015-External	Alarm	Limit Alarm	1

2 To compute daily or monthly metrics, click the buttons at the top of the report.

To view event metrics reports in operator mode:

3 In Operator mode, display the process map view.

For information on displaying the process map view in Operator mode, see Using the Operator Interface.

4 Click the Metrics button ( ) to show the Hourly Metrics Report, by default.

5 Click the H (Hourly), D (Daily), or M (Monthly) button at the top of the report to view the metrics on an hourly, daily, or monthly basis.

Configuring and Viewing System Performance Reports

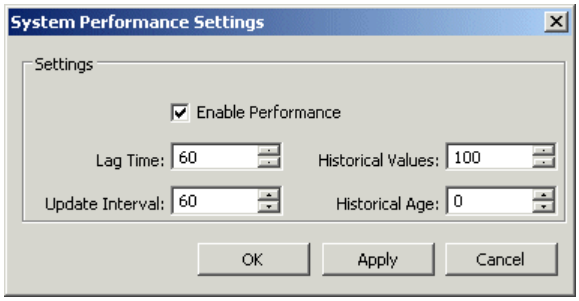
You can configure how often to update system performance reports and how much information to keep. You configure and view system performance in the same dialog. Here are the attributes you can configure:

Attribute	Description
Lag Time (sec)	The degree to which system performance metrics smooth data. For details, see "Specifying the Meter Lag Time" in Chapter 46 "G2-Meters" in the *G2 Reference Manual*.
Update Interval (sec)	How often to update the system performance metrics. The default value is 30 seconds.

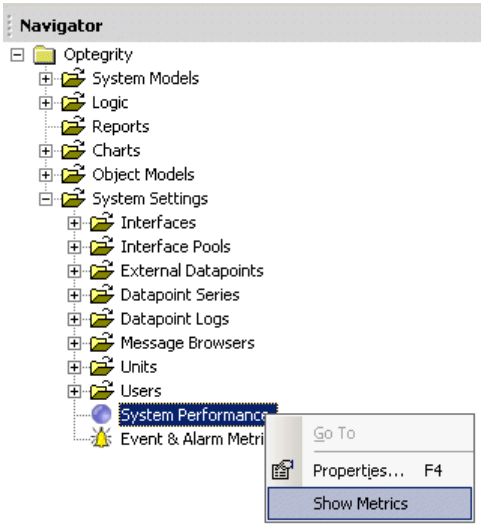
Attribute	Description
Historical Values	The number of history values to keep. The default is 120 points. Use this setting to limit the system performance history, based on the number of datapoints.
Historical Age	The maximum length of the history of the performance counter, minus one hour. Use this setting to limit the system performance history, based on time. The default value is 0 seconds, which does not limit the history.

To configure and view system performance reports:

1. Choose Project > System Settings > System Performance and configure the attributes, as needed.
2. Enable the Enable Performance option:



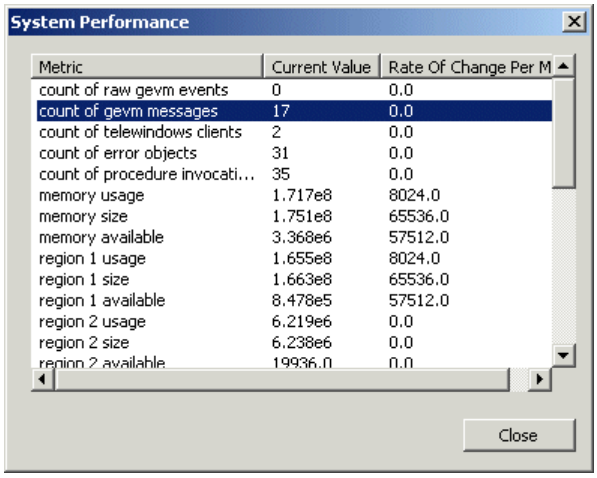
3 Display the Navigator, expand the System Settings node, and choose Show Metrics on the System Performance node:

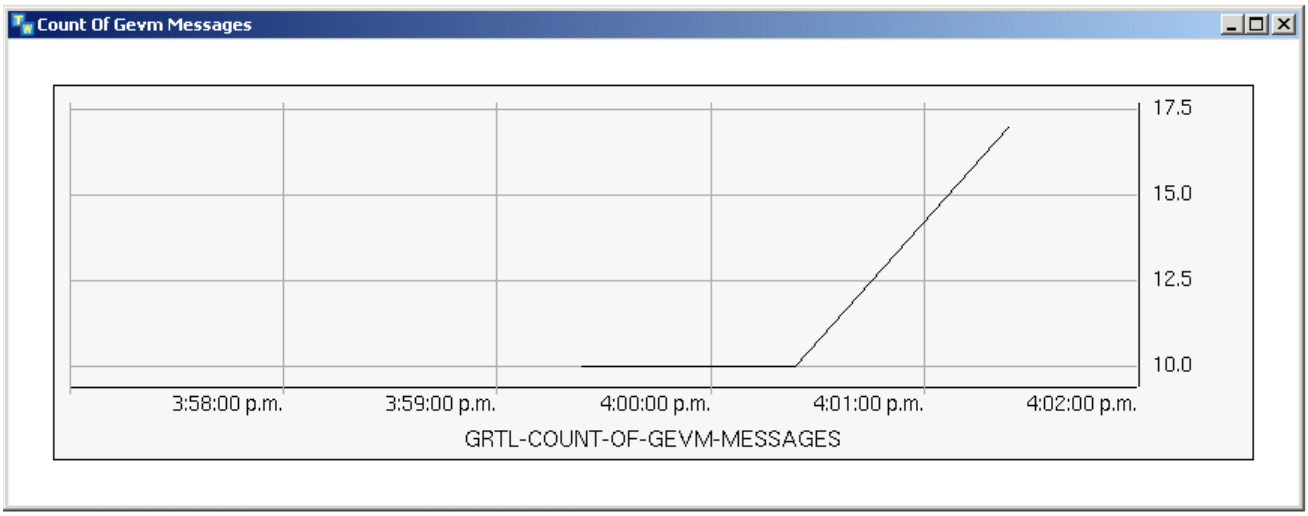


A dialog appears with all the system metrics.

4 Select a system performance metric that you want to view in the trend chart.

Here is a system performance report and the trend chart associated with the count of gevm messages:





This table describes each of the system performance metrics. Many of the system performance metrics are based on G2 meters, which are fully described in the *G2 Reference Manual*.

Metric	Description
clock time length	The number of seconds a G2 clock tick lasts. See the description of the clock-tick-length meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
count of error objects	The number of instances of the class error or any subclass. This metric helps to find memory leaks, in cases where error objects are created and not deleted.
count of GEVM messages	The number of instances of the class gevm-message or any subclass. For more information on this class, see Chapter 3 "GEVM Event Types and API" in *G2 Developers' Utilities Runtime Library User's Guide*.
count of procedure invocations	The number of G2 procedure invocations. See the discussion of procedure invocations in Chapter 22 "Procedures" and in Chapter 51 "Memory Management" in the *G2 Reference Manual*.
count of raw GEVM events	The number of instances of the class gevm-event or any subclass. For more information on this class, see Chapter 3 "GEVM Event Types and API" in *G2 Developers' Utilities Runtime Library User's Guide*.
count of Telewindows clients	The number of Optegrity clients currently connected to the server.
instance creation count	The number of instances. See the description of the instance-creation-count-as-float meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
maximum clock tick length	The duration in seconds of the longest clock tick that G2 has experienced since the knowledge base started running. See the description of the maximum-clock-tick-length meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
memory available	The total amount of memory currently allocated by the operating system but not used by G2. See the description of the memory-available meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.

Metric	Description
memory size	The total memory allocated to G2 by the operating system for holding data. See the description of the memory-size meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
memory usage	The total amount of memory that G2 currently uses. See the description of the memory-usage meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
percent runtime	The processing time G2 is using, as a percent of the processing time available for it to use. See the description of the percent-run-time meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
priority 1-10 scheduler time lag	The number of seconds behind current system time the scheduler is for a given priority. See the description of the priority-*n*-scheduler-time-lag meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*.
region 1, 2, and 3 size	The memory in the G2 region specified by *n*. The figure includes both used and available memory. See the description of the region-*n*-memory-size meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*. For a description of region 1, 2, and 3 memory, see Chapter 51 "Memory Management" in the *G2 Reference Manual*.
region 1, 2, and 3 usage	The amount of memory that G2 currently uses in the G2 region specified by *n*. See the description of the region-*n*-memory-usage meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*. For a description of region 1, 2, and 3 memory, see Chapter 51 "Memory Management" in the *G2 Reference Manual*.
region 1, 2, and 3 available	The total amount of memory currently available to G2 but not used by it in the G2 region specified by *n*. See the description of the region-*n*-memory-available meter in Chapter 50 "G2-Meters" in the *G2 Reference Manual*. For a description of region 1, 2, and 3 memory, see Chapter 51 "Memory Management" in the *G2 Reference Manual*.

Part V

Diagnostic Reasoning

Chapter 17: Creating Generic Fault Models

Describes how to create generic fault models to perform diagnostic reasoning.

Chapter 18: Running SymCure Fault Models

Describes how to run SymCure fault models for diagnosis.

17

Creating Generic Fault Models

Describes how to create generic fault models to perform diagnostic reasoning.

Introduction

You create generic fault models to perform diagnostic reasoning on events that are generated by specific event detection diagrams for domain objects in a domain map. You define these models generically for domain object classes. When an event is generated for a particular domain object, SymCure creates a specific fault model that includes the specific events that are either observed to be true or are predicted to be true, based on event propagation.

The specific fault model includes alarms, which appear in the Alarms Browser, and root causes, which appear in the Root Causes browser. As part of diagnosis, SymCure can also execute external tests to help diagnose root causes, and external repair actions that execute when a root cause is known to be true.

For detailed information about building generic fault models and interacting with specific fault models, see the *SymCure User's Guide*.

Creating Generic Fault Model Folders

You create generic fault models on a generic fault model folder detail. The folder is associated with a target class, which is the domain object class to which the fault model applies.

To create a generic fault model folder:

1 Create a generic fault model folder and display its properties dialog, using one of these two techniques:

➔ Choose Project > Logic > Diagnose > Generic Fault Models > Manage and click the New button to create a new folder and display its properties dialog.

or

➔ Display the Fault Models toolbox and clone a Fault Model Folder from the Fault Model Folder palette, and choose Properties on the folder.

2 Configure the Folder Name to be any text value.

The name can include spaces, for example, Myapp Fault Models.

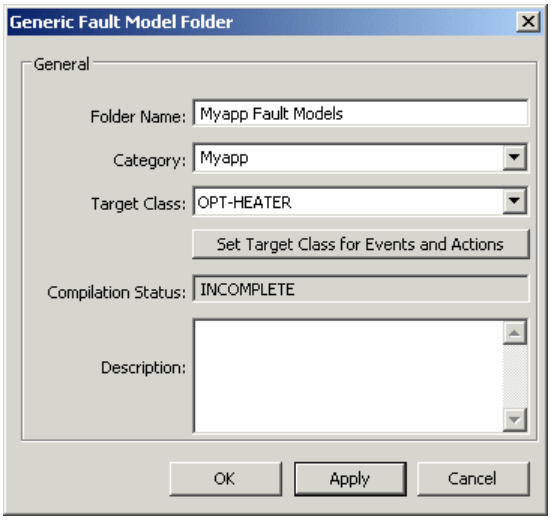
3 Configure the Category to be a user-defined text or one of the built-in categories that is used to organize folders in the Project menu.

4 Configure the Target Class to refer to a domain object class to which the generic fault model applies.

For example, to perform diagnostics on the heater, the target class would be opt-heater.

5 Optionally, configure the Description to be any user-defined text that describes the generic fault model.

Here is the properties dialog for the generic fault model folder named Myapp Fault Models:



Creating Generic Fault Models

A generic fault model describes the causal relationships between events for classes of domain objects.

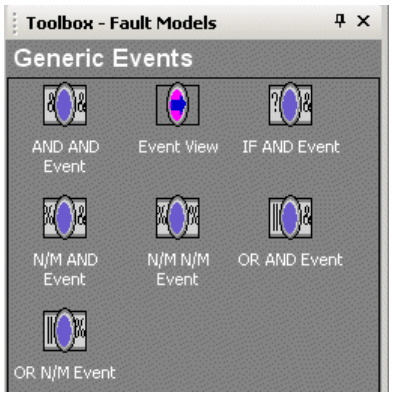
For detailed information about building generic fault models, see Chapter 4 "Creating Generic Fault Models" in the *SymCure User's Guide*.

To create a generic fault model:

1 Go to the detail of a generic fault model folder.

For information on how to do this, see Managing Generic Fault Models.

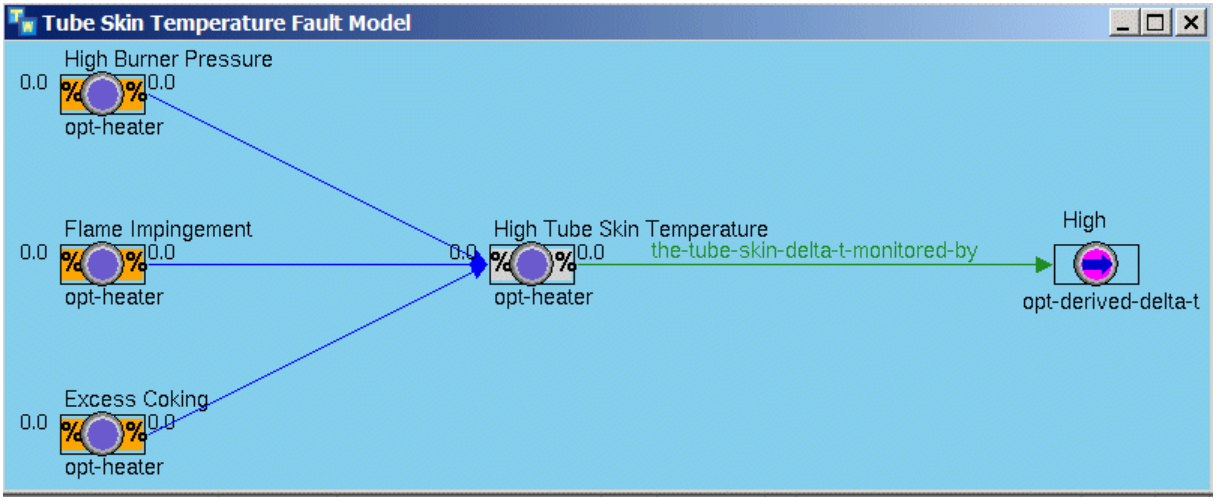
2 Choose View > Toolbox - Fault Models and display the Generic Events palette:



3 Select generic events from the palette and place them on the detail of the generic fault model folder.

4 Connect and configure the generic events to create a generic fault model.

Here is a generic fault model that describes the causal relationships between events for the F102 heater application described in the *Optegrity Heater Tutorial*:



For a description of how this generic fault model is used in the F102 heater application, see the *Optegrity Heater Tutorial*.

Creating Generic Actions

SymCure defines external actions in the form of tests and repair actions to isolate and recover from root causes. You can associate tests and repair actions with each generic event in a fault model. During diagnosis, SymCure creates and activates specific actions for specific events when the value of the underlying event changes, depending on the type of generic action. By default, a generic test action actives when the value of the underlying event becomes suspect, and a generic repair action activates when the value of the underlying event becomes true.

For detailed information about creating and configuring generic actions, see Chapter 5 "Creating Generic Fault Models" in the *SymCure User's Guide*.

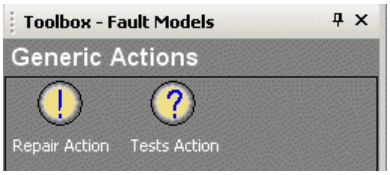
To create generic actions:

1 Create a generic fault model folder and go to its detail.

You can create generic fault models and generic actions in separate folders or even place them in separate modules.

For information on how to do this, see Managing Generic Fault Models.

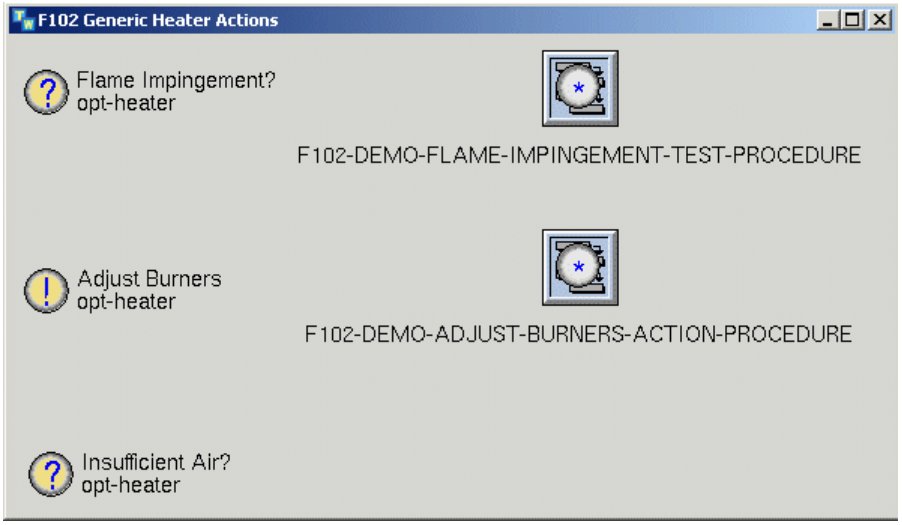
2 Choose View > Toolbox - Fault Models and display the Generic Actions palette.



3 Select generic actions from the palette and place them on the detail of the generic fault model folder.

4 Configure the generic actions.

Here are the generic actions for the F102 heater application described in the *Optegrity Heater Tutorial*:



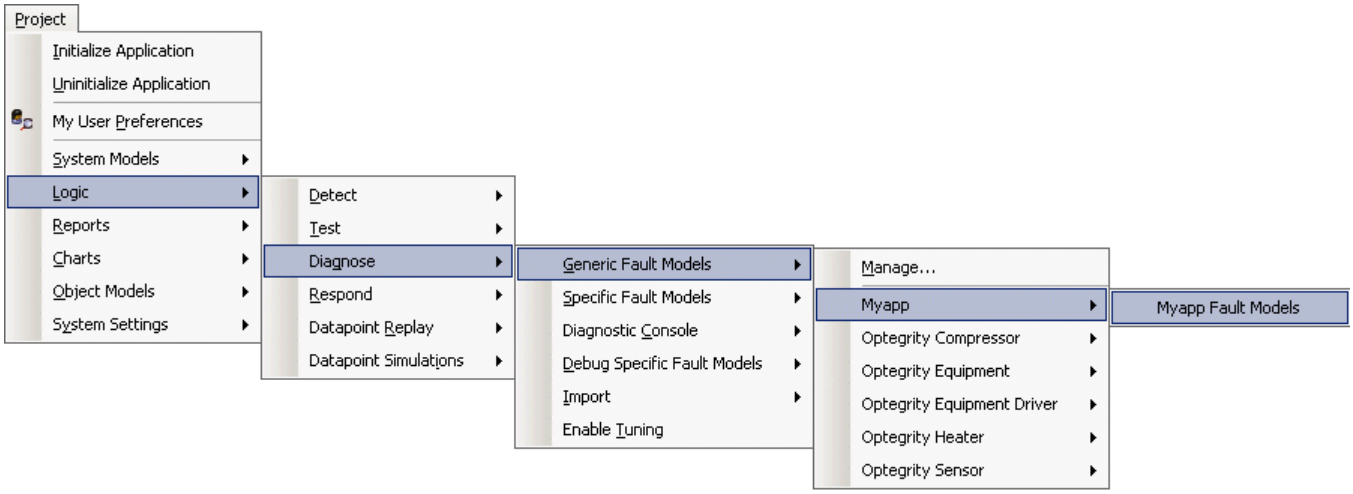
For a description of how these actions are used in the heater application, see the *Optegrity Heater Tutorial*.

Managing Generic Fault Models

To manage generic fault models:

1 Choose Project > Logic > Diagnose > Generic Fault Models, then choose a category.

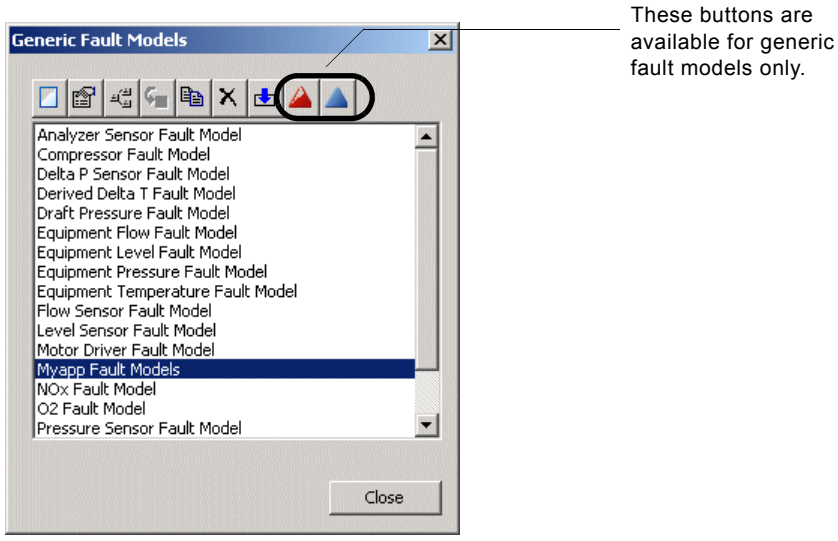
All generic folders appear in the submenu under the specified category or in the Unspecified submenu if no category is assigned, for example:



2 To display the detail of a generic fault model, choose one from the appropriate category in the Generic Fault Models submenu.

3 To display a dialog for managing all generic fault models, including displaying the properties dialog, choose Manage.

Here is the Generic Fault Models Manage dialog, which includes the built-in generic fault models:



For information on using this dialog and the Project menu to manage generic fault models, see Using the Project Menu.

For information on using the buttons specific to generic fault models, see Performing Specific Operations.

18

Running SymCure Fault Models

Describes how to run SymCure fault models for diagnosis.

Introduction

Before SymCure can diagnose events, the generic fault models must be compiled. Compiling generic fault models verifies that the fault models are syntactically correct for building specific fault models.

When you start your application, SymCure automatically compiles all existing generic fault models. You must compile generic folders after you make any changes to them.

When a specific event occurs on a domain object, SymCure creates a specific fault model with specific events and actions. You view specific alarm and root cause events through the Alarms and Root Causes Browsers, and you view specific actions through the Test Actions Browser and the Repair Actions Browser.

You can simulate fault model events for domain objects.

For a description of the built-in SymCure browsers, see Interacting with SymCure Diagnostic Console Browsers.

For detailed information about running SymCure applications, see Chapter 5, "Running SymCure Applications" in the *SymCure User's Guide*.

Compiling Generic Fault Models

When you compile a single fault model folder, SymCure compiles all other generic fault models as well. The color of the folder indicates whether it compiled correctly.

For more information, see Chapter 5 "Running SymCure Applications" in the *SymCure User's Guide*.

To compile all generic fault models:

1 Choose Project > Logic > Diagnose > Generic Fault Models > Manage.

2 Select the folder and click the Compile button.

Tip You can also compile the folder by choosing the Compile menu choice on the generic fault model folder in the Navigator. For details, see Navigating Applications.

---

3 To check compilation status, show the properties dialog for the folder.

If compilation is successful, the Compilation Status should be COMPLETE.

Checking for Errors and Warnings

SymCure creates errors and warnings for a generic fault model if it does not compile correctly.

For information on what these errors and warnings mean, see Chapter 5 "Running SymCure Applications" in the *SymCure User's Guide*.

To check for errors and warnings:

1 Choose Project > Logic > Diagnose > Generic Fault Models > Manage.

2 Select a folder and click the Errors or Warnings button.

or

➔ Choose View Errors or View Warnings on a fault model folder in the Navigator.

For details, see Navigating Applications.

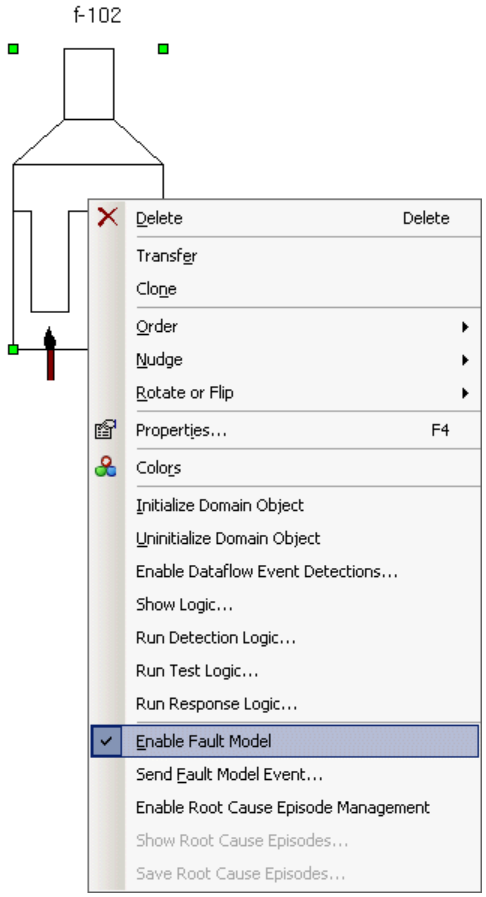
Enabling Fault Models

Before you can send SymCure events for a domain object in a process map, you must check the Enable Fault Model option for the domain object. By default, this option is enabled for all domain objects that you create from the Process Modeling toolbox. This means that when the intelligent object libraries are loaded, all built-in fault models are automatically enabled.

To enable fault models for a domain object:

➔ In the popup menu for a domain object, check the Enable Fault Model option.

Here is the popup menu for a heater with the Enable Fault Model option enabled:



Sending Fault Model Events

To simulate events for domain objects in a process map, you choose an event and specify the event value as true, false, or suspect. The list of available events includes all generic events defined for the domain object class.

To send a fault model event for a domain object, the fault model must be enabled.

For more information, see Chapter 5 "Running SymCure Applications" in the *SymCure User's Guide*.

To send a fault model event:

1 Choose Send Fault Model Event on a domain object in a process map.

Note This menu choice is only available if the Enable Fault Model menu choice is enabled. For details, see Enabling Fault Models.

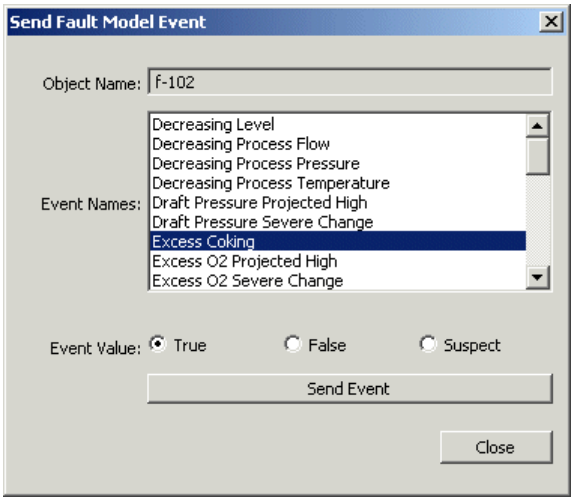
---

2 Choose an event from the list of available events.

3 Specify the event value as true, false, or suspect.

4 Click the OK button to send the event.

Here is the Send Fault Model Event dialog for a heater, which is configured to send a value of true for the Excess Coking built-in event:



Part VI

Alarm and Message Management

Chapter 19: Interacting with Operator Messages

Describes how to interact with operator messages through the operator interface.

Chapter 20: Interacting with SymCure Diagnostic Console Browsers

Describes how to interact with the SymCure Alarms Browser, Root Causes Browser, Test Actions Browser, and Repair Actions Browser.

Chapter 21: Using Message Queues

Describes how to manage the message queues associated with the various types of browsers.

19

Interacting with Operator Messages

Describes how to interact with operator messages through the operator interface.

Introduction

Optegrity provides an end user interface for interacting with operator messages and their associated process maps. The operator interface has two views:

- The message browser view shows all operator messages in a single browser, by default. Individual operators can subscribe to different types of messages, based on their user preference.
- The process map view provides a default process map for viewing and interacting different aspects of the model.

By default, the message browser view includes all types of messages, including messages associated with:

- Event detection diagrams.
- SymCure alarms, root causes, test actions, and repair actions.
- General operator information, for example, related to network interface connection status and logging.

For information on configuring the operator interface for individual users, see Configuring User Preferences.

For information on creating custom message browsers, see Custom Messaging.

Using the Operator Interface

You display the operator interface by switching to operator mode. You can configure user preferences such that operators go directly to the operator interface when they start the client.

By default, the operator interface displays the message browser view, which shows the message text, creation/update time, and acknowledgement status for all operator messages to which the current user subscribes. Operators can choose to display a mini browser view instead of the message browser view. The most recent message also appears in the status bar.

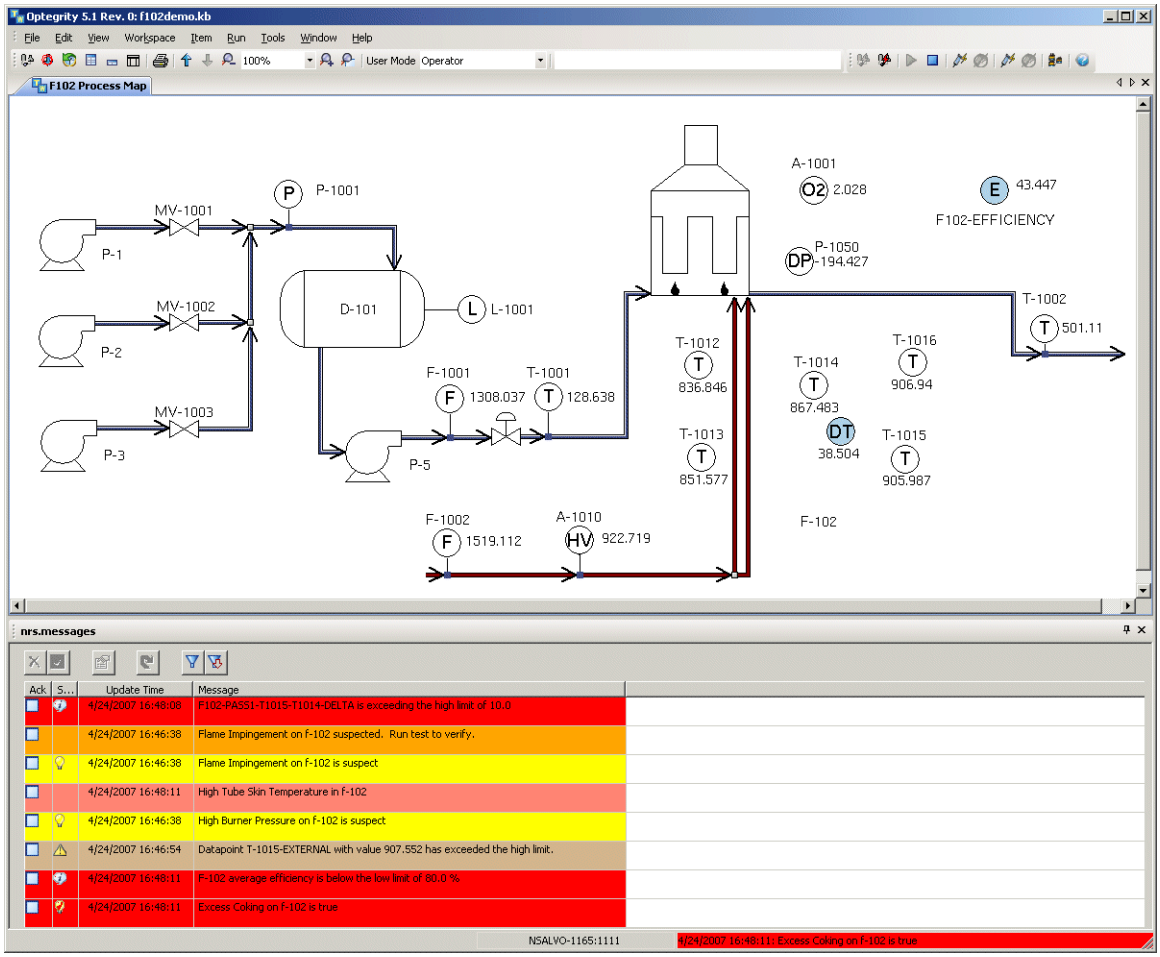
Operators can also display a default process map in the operator interface. You can configure the user preferences to choose the default process map to display in the operator interface. For details, see Configuring User Preferences.

You can configure the use-basic-message-browser parameter to determine the type of message browser to display, by default. For details, see Message Browser.

To display the operator interface:

➔ Choose Tools > User Mode > Operator.

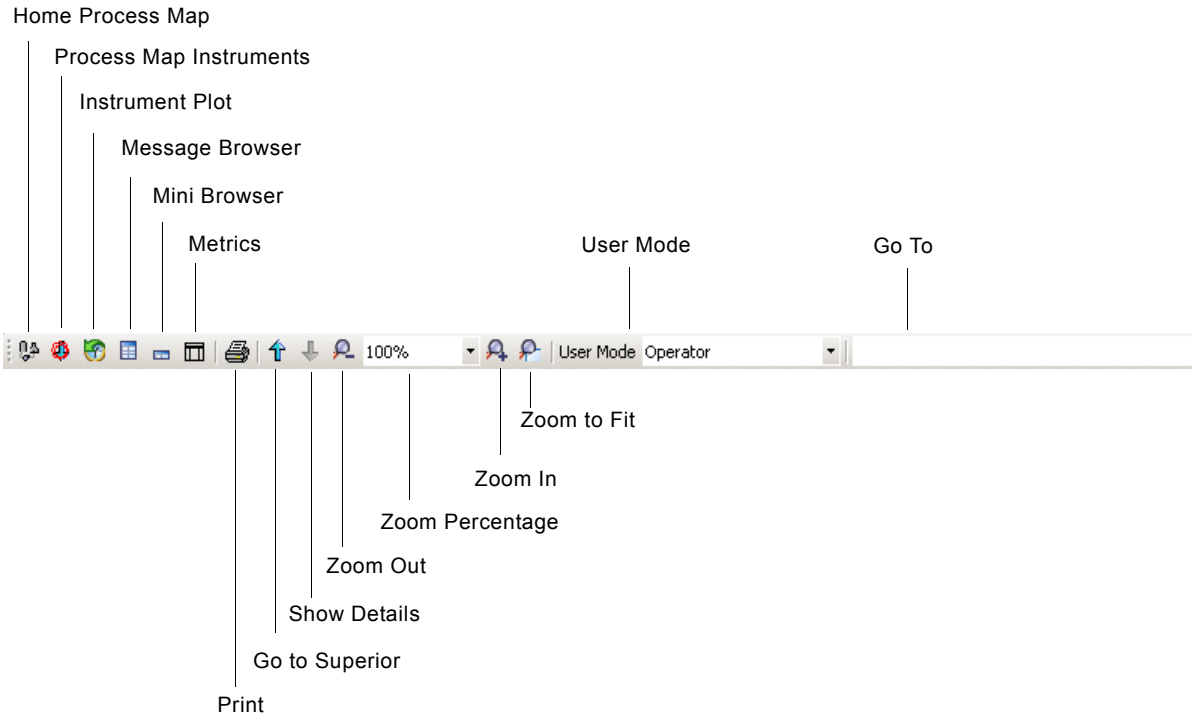
Here is the operator interface for the *f102demo.kb* after running the simulation. The process map view includes a toolbar for interacting with various aspects of the model, and the message browser view includes a toolbar for interacting with individual messages.



For information on configuring the default operator interface, see the description of Home Process Map and Show Browser in Operator Mode in Configuring User Preferences.

Interacting with the Process Model

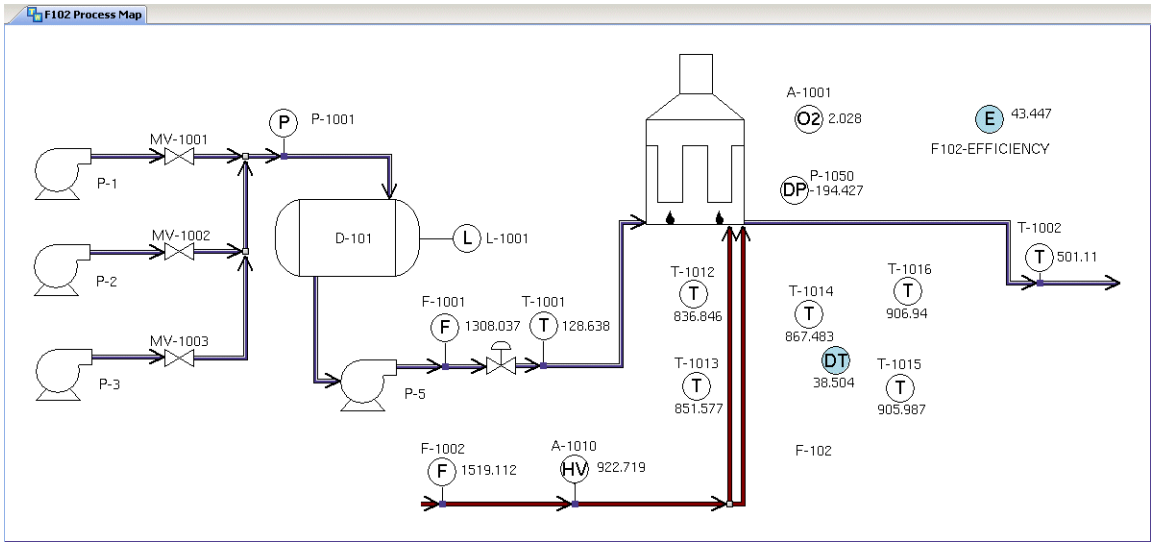
Operators can interact with various aspects of the process model in the operator interface, using the following toolbar buttons:



To interact with the process model:

1 To show the default process map, click the Home Process Map button.

Here is the home process map for the *f102demo.kb* application:

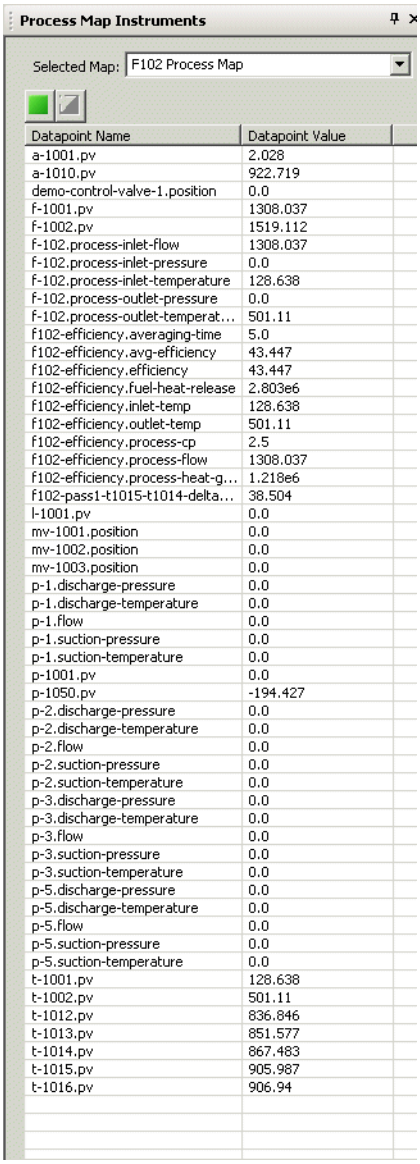


The process map appears in its own tab page. If other tab pages are showing, click the process map tab page to show the map. You can hide the process map by clicking the close button.

For information on configuring the default process map, see the description of Home Process map in Configuring User Preferences.

2 To show a list of all the internal datapoints in the process map, click the Process Map Instruments button.

Here are the process map instruments for the *f102demo.kb* application:



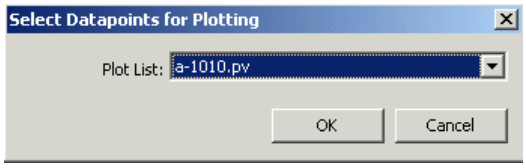
Datapoint Name	Datapoint Value
a-1001.pv	2.028
a-1010.pv	922.719
demo-control-valve-1.position	0.0
f-1001.pv	1308.037
f-1002.pv	1519.112
f-102.process-inlet-flow	1308.037
f-102.process-inlet-pressure	0.0
f-102.process-inlet-temperature	128.638
f-102.process-outlet-pressure	0.0
f-102.process-outlet-temperat...	501.11
f102-efficiency.averaging-time	5.0
f102-efficiency.avg-efficiency	43.447
f102-efficiency.efficiency	43.447
f102-efficiency.fuel-heat-release	2.803e6
f102-efficiency.inlet-temp	128.638
f102-efficiency.outlet-temp	501.11
f102-efficiency.process-cp	2.5
f102-efficiency.process-flow	1308.037
f102-efficiency.process-heat-g...	1.218e6
f102-pass1-t1015-t1014-delta...	38.504
l-1001.pv	0.0
mv-1001.position	0.0
mv-1002.position	0.0
mv-1003.position	0.0
p-1.discharge-pressure	0.0
p-1.discharge-temperature	0.0
p-1.flow	0.0
p-1.suction-pressure	0.0
p-1.suction-temperature	0.0
p-1001.pv	0.0
p-1050.pv	-194.427
p-2.discharge-pressure	0.0
p-2.discharge-temperature	0.0
p-2.flow	0.0
p-2.suction-pressure	0.0
p-2.suction-temperature	0.0
p-3.discharge-pressure	0.0
p-3.discharge-temperature	0.0
p-3.flow	0.0
p-3.suction-pressure	0.0
p-3.suction-temperature	0.0
p-5.discharge-pressure	0.0
p-5.discharge-temperature	0.0
p-5.flow	0.0
p-5.suction-pressure	0.0
p-5.suction-temperature	0.0
t-1001.pv	128.638
t-1002.pv	501.11
t-1012.pv	836.846
t-1013.pv	851.577
t-1014.pv	867.483
t-1015.pv	905.987
t-1016.pv	906.94

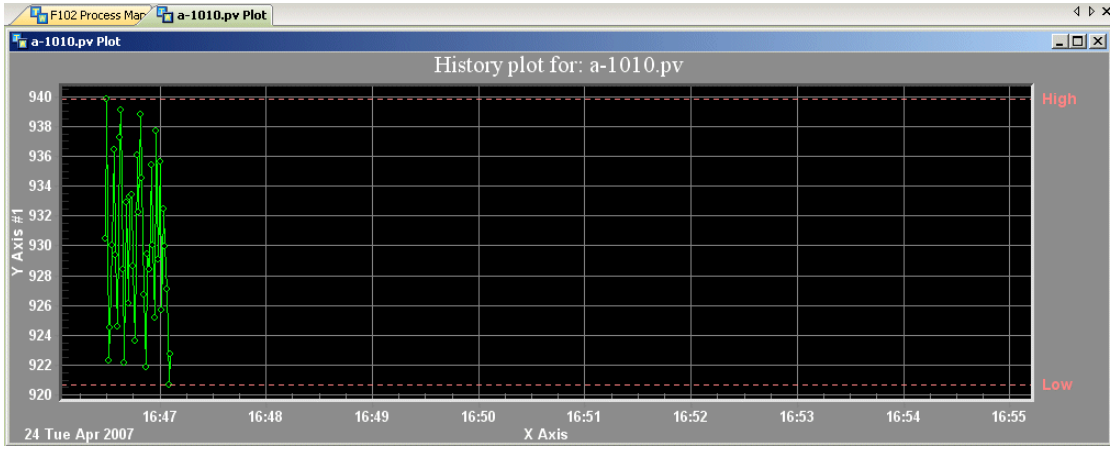
3 To update the datapoint values, click the Activate Update button ( ).

4 To stop updating the datapoint values, click the Deactivate Update button ( ).

5 To display a plot of a datapoint value, click the Instrument Plot button and select an internal datapoint in the process map to plot.

Here is how you would plot the A-1001.pv internal datapoint:

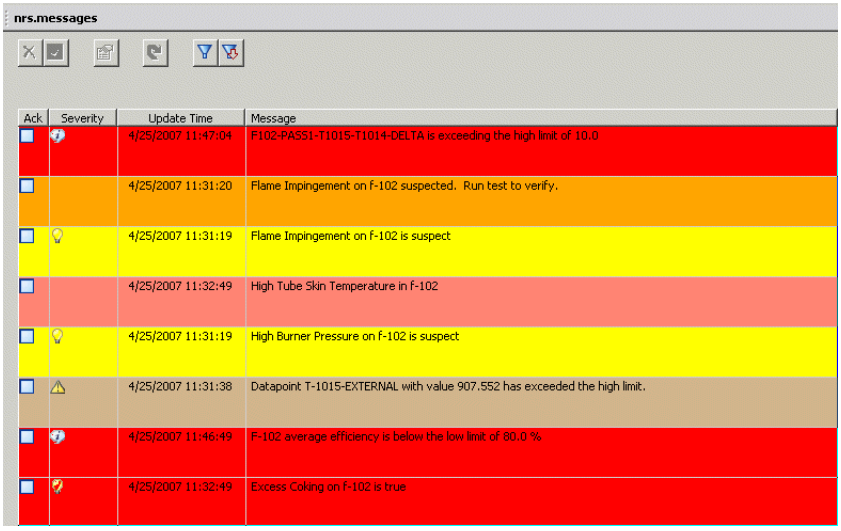




You can add multiple plots, each of which appears in its own tab page.

6 Click the Message Browser button.

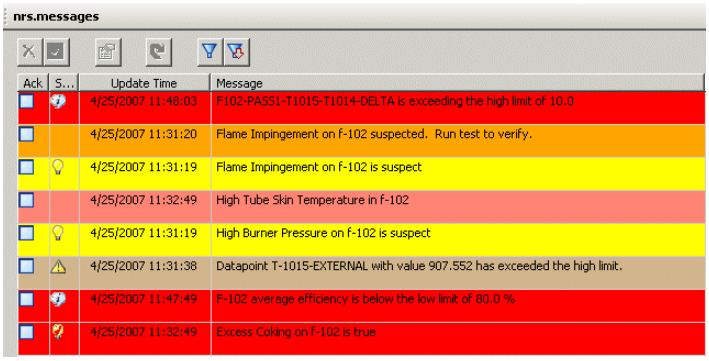
Here is the message browser for the *f102demo.kb* application after running the simulation:



For details, see Interacting with Operator Messages.

7 To show a mini message browser view, click the Mini Browser button.

Here is the mini browser:



8 To show a report of metrics related to messages, click the Metrics button.

For more information and an example, see Reporting and Charting.

9 To go to the superior process map or show the detail of the selected object, click the Go to Superior button or Show Details button.

You use these options when configuring hierarchical process maps. For more information and an example, see Creating a Process Map.

10 To zoom the process map, click the Zoom In, Zoom Out, Zoom Percentage, and Zoom to Fit buttons.

11 To configure the application, choose Modeler mode.

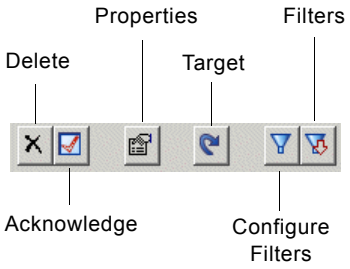
For information about configuring the operator interface to disallow configuring the model, see the description of Configuration Permission in Configuring User Preferences.

12 To search for objects in the process map, enter the exact name in the Search field.

Interacting with Operator Messages

You view and interact with operator messages in the message browser or mini browser view by selecting a message and clicking the various toolbar buttons. You can acknowledge and delete messages, show properties, filter messages, go to the message target, and sort messages.

Here is the toolbar in the Message Browser in the operator interface:



To view and interact with operator messages:

1 Click one or more messages in the Message Browser to select it.

To select multiple contiguous messages, select the starting message, then hold down the Shift key and click the ending message or drag the cursor to select multiple messages. To select multiple non-contiguous messages, hold down the Ctrl key and click a message to add to the selection.

2 Click a toolbar button to operate on the selected message or messages.

For details on the behavior of these buttons, see the *Optegrity Heater Tutorial*.

Interacting with Operator Messages in Modeler Mode

In Modeler mode, you interact with operator messages through the Message Browser. In Modeler mode, the Message Browser provides additional buttons, including buttons for interacting with SymCure messages.

Note Messages about SymCure events are not representations of the actual events; they are merely messages about the underlying events. For such messages, deleting the message does not delete the underlying event.

You can select a range of messages by selecting the starting message, then holding down the SHIFT key and selecting the ending message. You can also drag the cursor across a range of messages to select multiple contiguous messages. You can select multiple non-contiguous messages by using the Ctrl key.

Note By default, Optegrity sorts messages chronologically by creation time, based on the current subsecond time. If multiple messages are created within the same subsecond, the sort order is not predictable.

For information on the SymCure specific buttons, see Interacting with SymCure Diagnostic Console Browsers.

To display the message browser in Modeler mode:

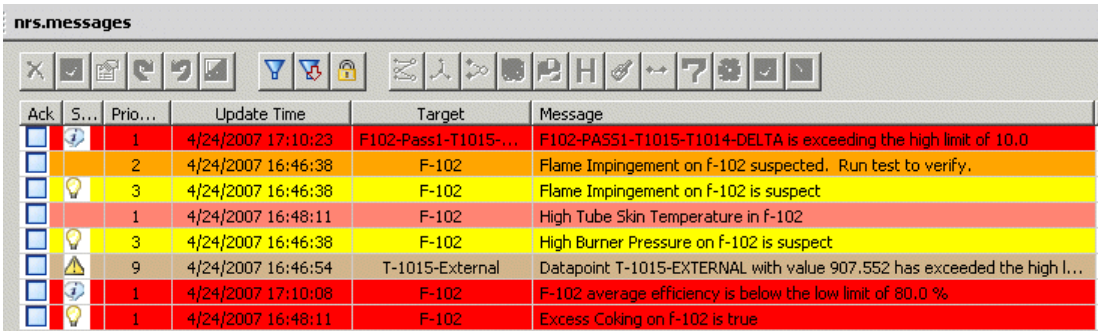
1 Choose View > Message Browser or click the equivalent toolbar button: 

2 Select a message in the browser and click a toolbar button.

To interact with the Message Browser:

➔ Select a message in the browser and click a toolbar button.

Here is the Message Browser for the *f102demo.kb* application after running the simulation and selecting a message:



20

Interacting with SymCure Diagnostic Console Browsers

Describes how to interact with the SymCure Alarms Browser, Root Causes Browser, Test Actions Browser, and Repair Actions Browser.

Introduction

You interact with specific events and actions through these built-in SymCure browsers:

- Alarms Browser — Shows specific events of type alarm.
- Root Causes Browser — Shows specific events of type root-cause.
- Test Actions Browser — Shows specific test actions.
- Repair Actions Browser — Shows specific repair actions.

The built-in SymCure browsers show various information about the specific event or specific action, depending on the browser, such as the event or action name, the target object, a text message, the event value, the action status and type, and the time at which the event or action was last updated.

You can interact with specific events and actions in various ways, such as:

- Showing the causal fault model for an alarm or root cause.
- Showing the root causes of an alarm
- Showing an event summary of symptoms, predications, root causes, and actions.
- Showing the sequence of events that led up to a specific alarm or root cause.
- Showing the specific event underlying a specific external action.
- Explaining and running external test and repair actions.

For a complete description of interacting with SymCure browsers, see Chapter 5 "Running SymCure Applications" in the *SymCure User's Guide*.

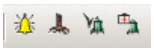
The SymCure browsers include all the buttons that are available in the default Message Browser, as well as those specific to SymCure. Just as you can log operator messages in the Message Browser, you can log messages for alarms, root causes, test actions, and repair actions.

You can configure the default browser to use for alarms, root causes, test actions, and repair actions. For example, you might want to display both alarms and root causes in the same browser, and both test actions and repair actions in the same browser. To do this, you configure SymCure initialization parameters. For details, see CDGUI (SymCure) in Configuring Startup Parameters

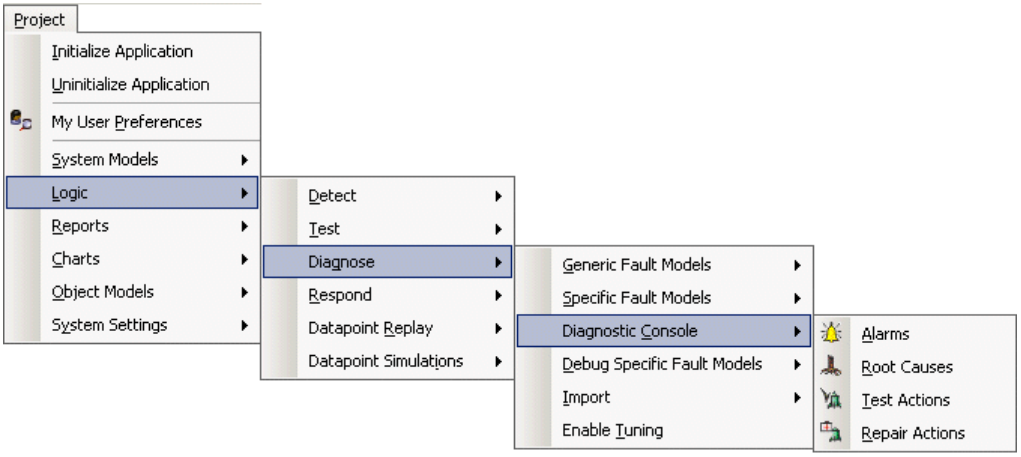
Displaying SymCure Browsers

For information about the default Message Browser, see Interacting with Operator Messages.

To display the SymCure browsers:

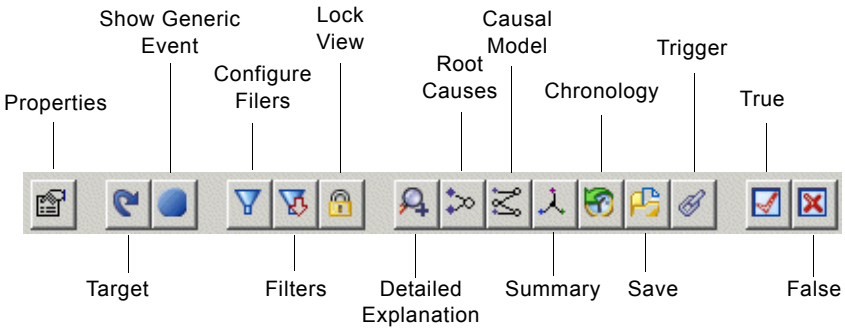
➔ Choose Project > Logic > Diagnose > Diagnostic Console and choose browser or click one of the equivalent Fault Modeling toolbar buttons: 

The Diagnostic Console submenu includes the four SymCure browsers—Alarms, Root Causes, Test Actions, and Repair Actions:



Interacting with the Alarms Browser

The Alarms Browser shows the target, event name, event value, message, and last update time. You interact with alarms in the Alarms Browser, using these toolbar buttons:



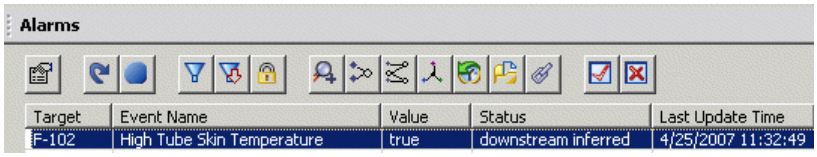
The Alarms Browser has these columns:

- Target — The domain object that is the target of the event.
- Event Name — The name of the specific event.
- Value — The value of the event, which is true, false, suspect, or unknown.
- Status — The status of the event, which is upstream inferred, downstream inferred, specified, or mutually exclusive.
- Last Update Time — The time at which the event value or status was last updated.

To interact with the Alarms Browser:

➔ Select an alarm in the Alarms Browser.

Here is the Alarms Browser with an alarm for the F-102 heater that is inferred to be true:



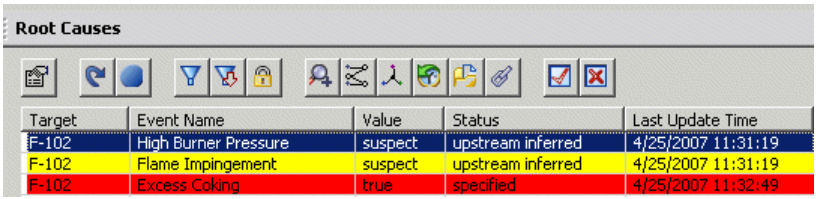
Interacting with the Root Causes Browser

The Root Causes Browser shows the target, event name, event value, message, and last update time. You interact with root causes, using these toolbar buttons. The buttons are identical to those in the Alarms Browser except that the Root Causes Browser does not include the Root Causes button. The Root Causes Browser has the same columns as the Alarms Browser.

To interact with the Root Causes Browser:

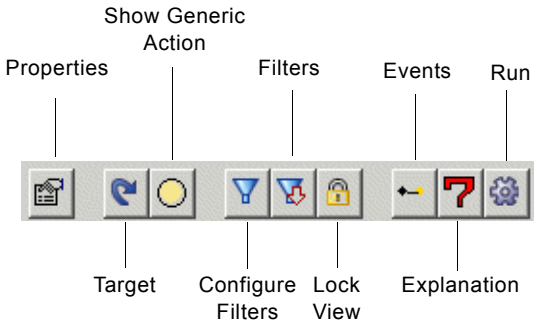
➔ Select a root cause in the Root Causes Browser.

Here is the Root Causes Browser with several root causes for the F-102 heater:



Interacting with the Test Actions Browser

The Test Actions Browser shows the target, test name, status, type, and last update time. You interact with tests, using the toolbar buttons:



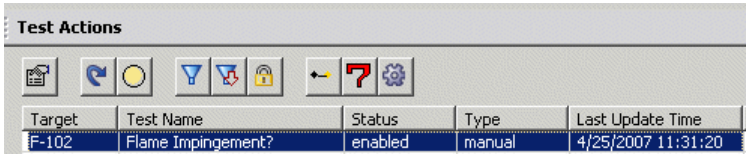
The Test Actions Browser has these columns:

- Target — The domain object that is the target of the test action.
- Test Name — The name of the specific test.
- Status — The status of the test. The options are: create, enabled, running, and inactive.
- Type — The type of test. The options are: manual and automatic.
- Last Update Time — The time at which the event status was last updated.

To interact with the Test Actions Browser:

➔ Select a test in the Test Actions Browser.

Here is the Test Actions Browser with a test action for the F-102 heater:



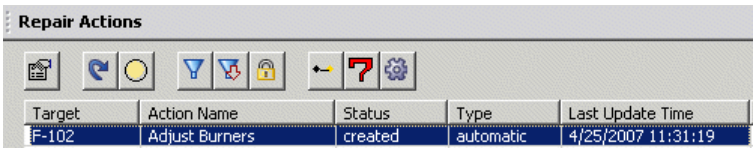
Interacting with the Repair Actions Browser

The Repair Actions Browser shows the target, test name, status, type, and last update time. You interact with the Repair Actions Browser, using the toolbar buttons, which are the same as in the Test Actions Browser. The Repair Actions Browser has the same columns as the Test Actions Browser.

To interact with the Repair Actions Browser:

➔ Select a repair action in the Repair Actions Browser.

Here is the Repair Actions Browser with a repair action for the F-102 heater:



21

Using Message Queues

Describes how to manage the message queues associated with the various types of browsers.

Introduction

Each type of browser is associated with a configurable message queue. You can configure logging for the built-in message queues. You can also create custom message queues.

Note To configure message queues, you must be in System-Administrator mode.

For information on creating custom message queues, see Custom Messaging.

Creating a New Message Queue

When creating a new message queue, you can configure the update latency, maximum entries in the queue, and the browser template. By default, message queues update their values once a second and allow a maximum of 10,000 messages before they start deleting messages. When the maximum is exceeded, the queue deletes the oldest messages first.

Note You cannot configure these attributes for the built-in queues; you can only configure them for custom queues.

To configure the general queue properties:

1 Choose Project > System Settings > Message Browsers > Queues > Manage and click the New button.

The dialog for configuring a new message queue appears.

2 Configure the following attributes of the queue:

Attribute	Description
Label	The name of the message queue.
Update Latency	The frequency with which to update the message queue. The default value is 1 second.
Maximum Entries in Queue	The maximum number of entries in the queue. The default value is 10,000.
Browser Template	The name of the browser template or access table to use for displaying queue messages in a browser.

Logging Messages

In System-Administrator mode, you can configure each message queue to log messages to a file, to a database, and/or to a JMS provider. You can log message additions, deletions, and changes. By default, logging is not enabled; thus, you must explicitly enable logging.

Optegrity creates log files in the `Archives` directory, using a default log filename. By default, it creates a new log file when the file size exceeds 1 MB or once a day, whichever comes first.

Various initialization parameters control the default behavior of message logging. For more information, see Appendix , Configuring Startup Parameters.

Logging Messages to a File

To log messages to a file:

1 Switch to System-Administrator mode.

For details, see Switching User Modes.

2 Choose Project > System Settings > Message Browsers > Queues > and choose the queue whose messages you want to log to a file.

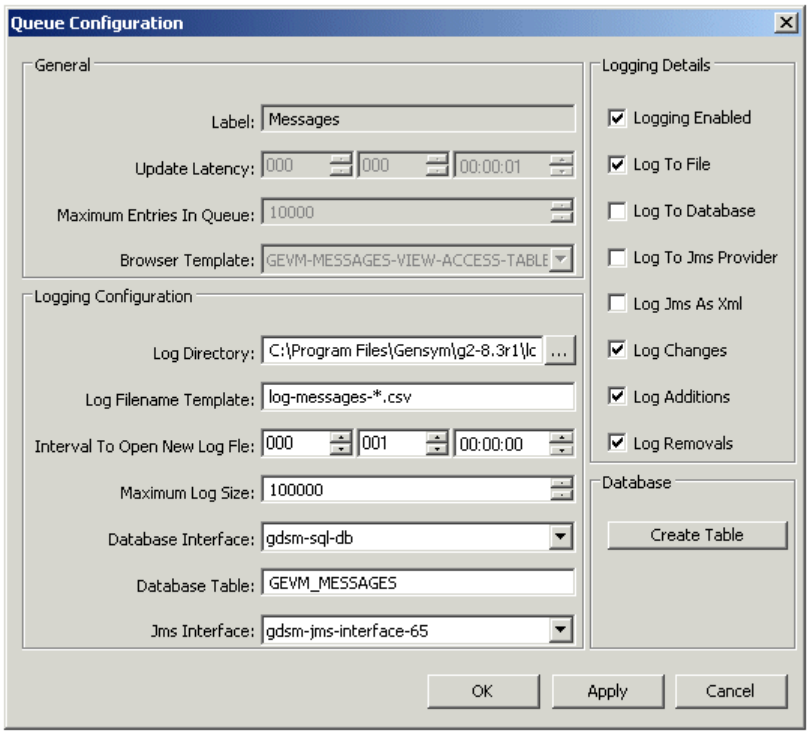
The dialog for configuring the message queue appears.

3 To log messages to a file, configure the following attributes of the queue:

Attribute	Description
Log Directory	The directory in which to create the log file, relative to current KB's directory. The default value is ..\Archives\.
Log Filename Template	A template for the log file, including a file extension and a wildcard for the unique ID. The unique ID consists of the year, date, and number. For example, the default value for the Alarms queue is log-alarms-*.csv; thus, a sample log file might be called *log-alarms-2004-11-27-7.csv*.
Interval to Open New Log File (s)	The time interval, in seconds, for creating a new log file, when neither the number of entries nor the file size has exceeded the maximum. The default value is 86400 seconds, or one day.
Maximum Log Size (Bytes)	The maximum size of the log file, in bytes. The default value is 100000 bytes.
Logging Enabled	Whether to enable message logging. By default, logging is disabled; thus, you must explicitly enable logging to log messages.
Log to File	Whether to log messages to a file. This option must be enabled to log messages to a file.
Log Changes	Whether to log changes to messages to the log file. Changes includes acknowledging messages and editing comments. By default, changes are logged.

Attribute	Description
Log Additions	Whether to log new messages to the log file. By default, new messages are logged.
Log Removals	Whether to log message deletions to the log file. By default, message deletions are logged.

Here is the properties dialog for the Messages queue, which logs message changes, additions, and removals to a file:



Logging Messages to a Database

You can log messages to an Oracle, Sybase, or ODBC database. To log messages to a database, first, you create a database interface, then you configure the message queue to log messages, using the database interface. You must also specify a database table. The database interface must exist before you can log messages to a database. In most cases, the database table must also exist. See note below.

To log messages to a database:

1 Switch to System-Administrator mode.

For details, see Switching User Modes.

2 Create and configure a database interface for message logging.

For details, see Creating and Connecting Network Interfaces.

3 Choose Project > System Settings > Message Browsers > Queues > and choose the queue whose messages you want to log to a database.

The dialog for configuring the message queue appears.

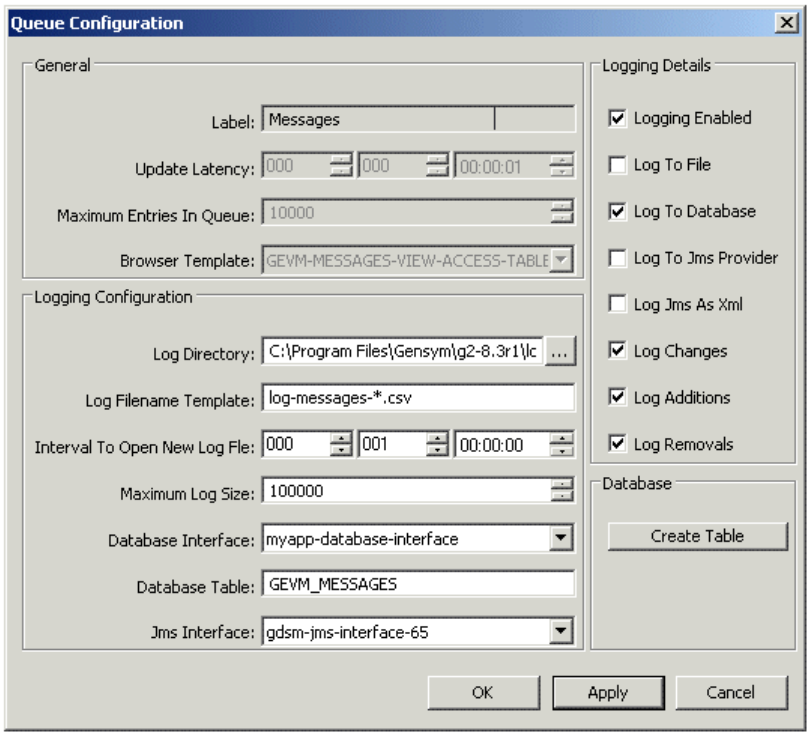
4 To log messages to a database, configure the following attributes of the queue:

Attribute	Description
Database Interface	The name of a database interface object to use for message logging.
Database Table	The name of a database table within the specified database interface object.
Logging Enabled	Whether to enable message logging. By default, logging is disabled; thus, you must explicitly enable logging to log messages.
Log to Database	Whether to log messages to a database. By default, logging to a database is disabled; thus, you must explicitly enable database logging to log messages.
Log Changes	Whether to log changes to messages to the database, which includes acknowledging messages. By default, changes are logged.
Log Additions	Whether to log new messages to the database. By default, new messages are logged.
Log Removals	Whether to log message deletions to the database. By default, message deletions are logged.

5 If the database table does not already exist, click Create Database Table to create the specified database table.

Note The Create Table button only works for the SQL Server database when using the G2-ODBC Bridge. If the G2-ODBC Bridge is connected to any other database, such as Access or Oracle, then you must create the database table manually, because the database data types of the fields in the table are not common to all databases. You can also customize the procedure that creates the database table for your particular database. For more information, see the *G2 Run-Time Library User's Guide*.

This dialog configures message logging in the Message Browser to log message changes, additions, and removals to a database, using the database interface named myapp-database-interface and the database table named gevm_messages, the default:



Logging Messages to a JMS Provider

You can log messages to a JMS provider. To log messages to a JMS provider, first, you create a JMS interface, then you configure the message queue to log messages, using the JMS interface. The database interface must exist before you can log messages to a database. You can also log messages to a JMS provider as XML.

To log messages to a JMS provider:

1 Switch to System-Administrator mode.

For details, see Switching User Modes.

2 Create and configure a JMS interface for message logging.

For details, see Creating and Connecting Network Interfaces.

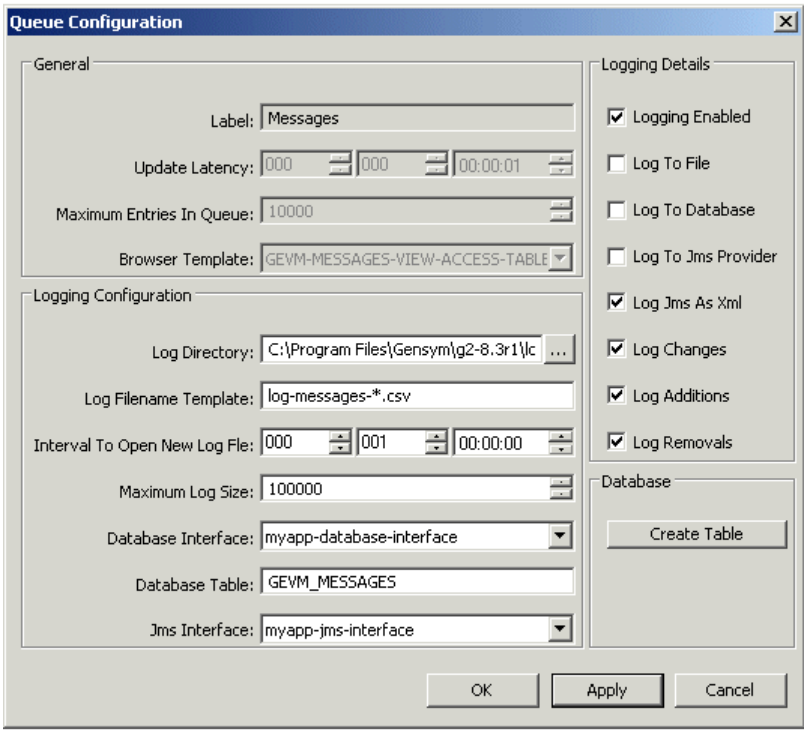
3 Choose Project > System Settings > Message Browsers > Queues > and choose the queue whose messages you want to log to a JMS provider.

The dialog for configuring the message queue appears.

4 To log messages to a JMS provider, configure the following attributes of the queue:

Attribute	Description
JMS Interface	The name of a JMS interface object to use for message logging.
Logging Enabled	Whether to enable message logging. By default, logging is disabled; thus, you must explicitly enable logging to log messages.
Log to JMS Provider	Whether to log messages to a JMS provider. By default, logging to a JMS provider is disabled; thus, you must explicitly enable JMS logging to log messages.
Log JMS as XML	Whether to log messages to the JMS provider as XML. By default, messages are logged as text.

This dialog configures message logging in the Message Browser to log message changes, additions, and removals to a JMS provider, using the JMS interface named myapp-jms-interface:



Contents of Log File

The log file contains the following columns for each entry. This is the same information that appears in the Message Browser and on the detail for each message. Some of this information, such as the message, category, details, and advice, is configured in the event source, for example, a GEDP Post Message block. Other information, such as user comments, is entered by the end user in the Message Browser. Still other information, such as the entry type, timestamp, target name, and source name, is generated by Optegrity.

Column	Description
Timestamp	The timestamp at which the message occurred. This value is determined by Optegrity.
Entry Type	A description of why the entry was logged. The options are: added, event-repetition, publish-event, comments-changed-event, deleting-event, and removed.

Column	Description
Target Name	The name of the domain object on which the event has occurred. This value is determined by Optegrity.
Source Name	The name of the event source. This value is determined by Optegrity.
Event Type	The class of message. This value is specified in the event source.
Priority	The message priority, which is a number from 1 to 9. This value is specified in the event source.
Repetition	The number of times that the same message has occurred on the same target object. This value is generated by Optegrity.
Category	The message category. This value is specified in the event source.
Message	The message text, with text substitutions. This value is specified in the event source.
Message Detail	The message detail text, with text substitutions. This value is specified in the event source.
Message Advice	The message advice text, with text substitutions. This value is specified in the event source.
Assigned to User	The value of the Assigned to User field on the message detail. This value is specified by the operator of the application at runtime.
Acknowledgement Required	Whether the message requires acknowledgement. This value is specified in the event source.
Acknowledged	Whether the message has been acknowledged. This value is determined by Optegrity.
Acknowledged By	The value of the Acknowledged By field on the message detail. This value is specified by the operator of the application at runtime.

Column	Description
Acknowledgement Timestamp	The timestamp at which the message was acknowledged. This value is determined by Optegrity.
User Comment	The value of the Comments on the message detail. This value is entered by the operator of the application at runtime.

Configuring the Browser Template for a Message Queue

A message queue uses a configurable browser template to display its messages. To do this, you create a custom message queue and associated browser template, then configure the queue to use the custom template.

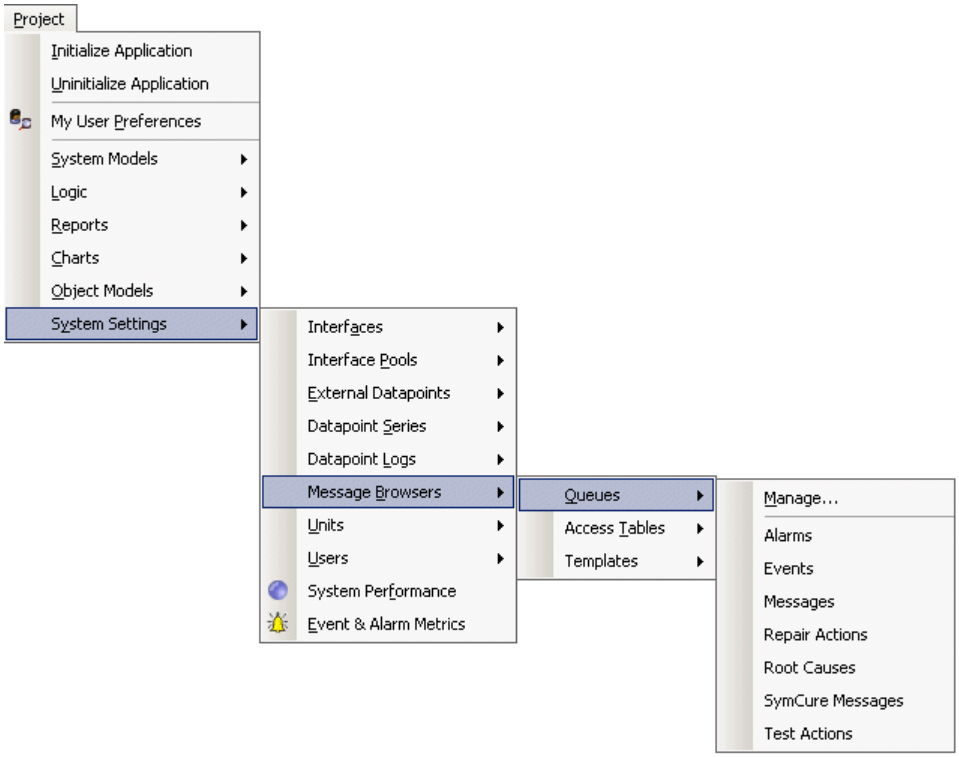
For information about creating custom browser templates and queues, see Custom Messaging.

Managing Message Queues

To manage message queues:

1 Choose Project > System Settings > Message Browsers > Queues.

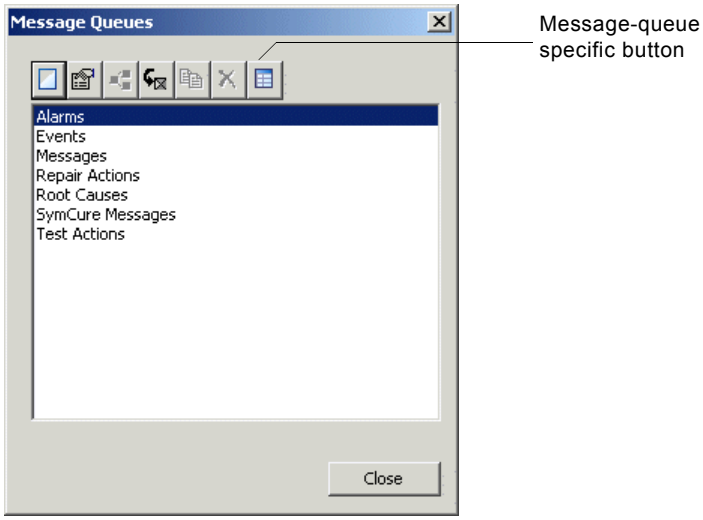
The Queues appear in the submenu:



2 To configure the properties for a message queue, choose it from the list.

3 To display a dialog for managing all message queues, choose Manage.

Here is the dialog for managing message queues:



For information on using this dialog and the Project menu to manage message queues, see Using the Project Menu.

For information on using the button specific to message queues, see Performing Specific Operations.

Part VII

Customization

Chapter 22: Creating Custom Event Detection

Describes how to create custom event detection for domain objects.

Chapter 23: Customizing Optegrity

Describes how to customize Optegrity.

Chapter 24: Configuring Startup Parameters

Describes the parameters that you can configure in the Optegrity startup file.

22

Creating Custom Event Detection

Describes how to create custom event detection for domain objects.

Introduction

To create custom event detection, you create:

- A custom domain object definition and its associated relations.
- A custom event object hierarchy.
- Event logic for each custom event.

Depending on the event, deploying a custom domain object with custom events involves:

- Configuring domain object parameters and related objects.
- Selecting and configuring specific domain object events for the domain object.

Creating Custom Domain Objects and Relations

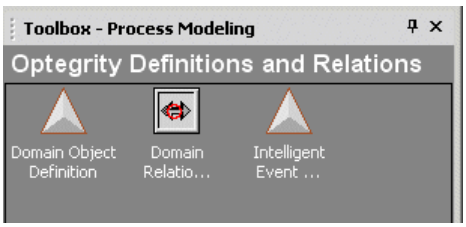
A domain object represents process equipment by defining its properties and its relationship to other domain objects in the process. Domain objects can also have methods that define the behavior of the object, including simulation, event monitoring, detection, and diagnostic activity. Optegrity provides built-in classes for defining custom domain objects and their relationship to other domain objects. The two built-in classes that you can subclass to define your own custom equipment hierarchy are opt-instrument and opt-equipment:

For information on how to create a custom domain object, see Creating Domain Object Definitions.

When creating event detection and fault models, it is often necessary to know how the domain object is related to other domain objects in the process. You define this relationship by creating a G2 connection on an Optegrity process map or by creating a G2 relation. To define a G2 relation between domain objects, you create a Domain Relation Definition, which is a subclass of g2-relation.

To create a custom domain object relation:

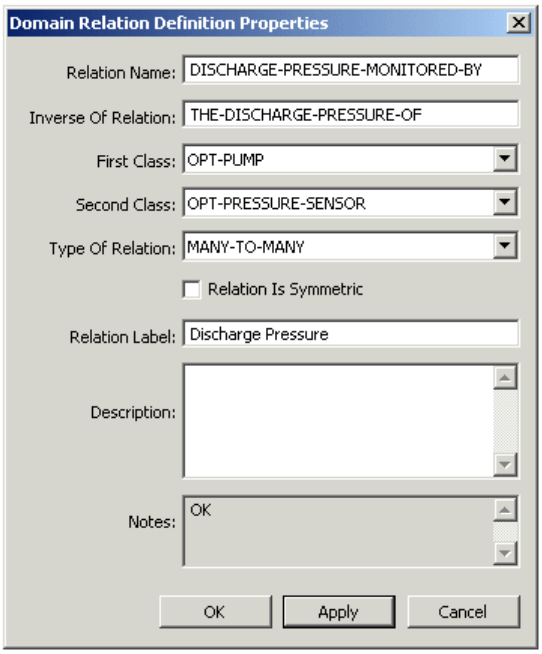
1 Choose View > Toolbox - Process Modeling, display the Optegrity Definitions and Relations palette, and create a Domain Relation Definition from the palette:



2 Configure the properties of the domain relation definition.

For example, here is a Domain Relation Definition that defines a relation between an opt-pump and an opt-pressure-sensor. The relation is named discharge-pressure-monitored-by and the inverse relation is named the-discharge-pressure-of.

DISCHARGE-PRESSURE-MONITORED-BY



Creating a Custom Event Object Hierarchy

Once you have defined a set of domain objects and domain object relations, the next step is to define the custom events for use in event detection on the domain object. To create and encapsulate complex event detection algorithms for a domain object, you create a custom event definition, which is a subclass of opt-domain-object-event. You then create instances of these event definitions and use them in generic event detection templates that apply to your custom domain object class.

You typically define class-specific attributes for event definitions, which specify parameters that determine when the event is true. To display class-specific attributes in the properties dialog of the custom event, configure the class-specific attribute of the definition to be a subclass of the g2-parameter class, for example, grtl-simple-quantitative-datapoint.

When configuring an event definition, you also specify the palette on which to place the custom event block for use in generic event detection templates.

Typically, you create a custom event definition hierarchy to organize events based on the type of event, as well as based on the class of domain object the event object monitors.

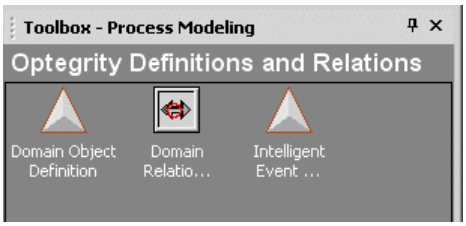
The inherited attributes of a custom event block are:

- Event Name (text) is the name of the event.
- Event Evaluation Time (read-only) is the time at which the Domain Object Event block was processed.
- Event Exists (truth-value) specifies whether the event exists after the Domain Object Event block is processed. This may be a fuzzy truth-value.
- Event Logic Status (read-only) displays any errors that occur when processing the Domain Object Event block.
- Description (text) is a textual description of the event.

For details on configuring the custom event block, see Configuring the Specific Event Detection Diagram for the Custom Event.

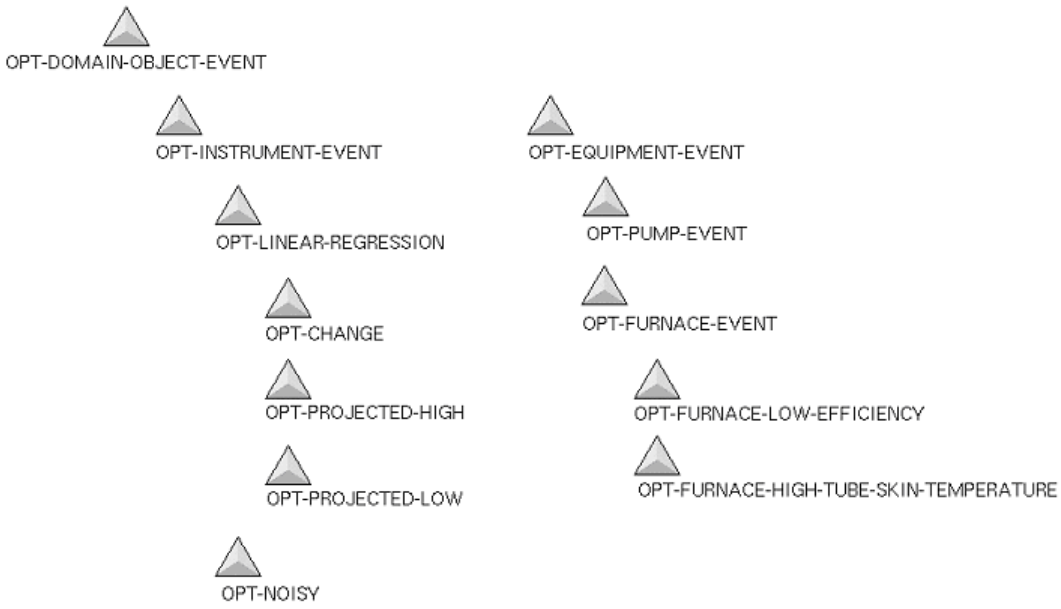
To create a custom event object hierarchy:

1 Choose View > Toolbox - Process Modeling, display the Optegrity Definitions and Relations palette, and create an Intelligent Event Definition from the palette:



2 Create a custom event definition hierarchy by creating subclasses of opt-domain-object-event.

For example, here is an event hierarchy for sensors and equipment:

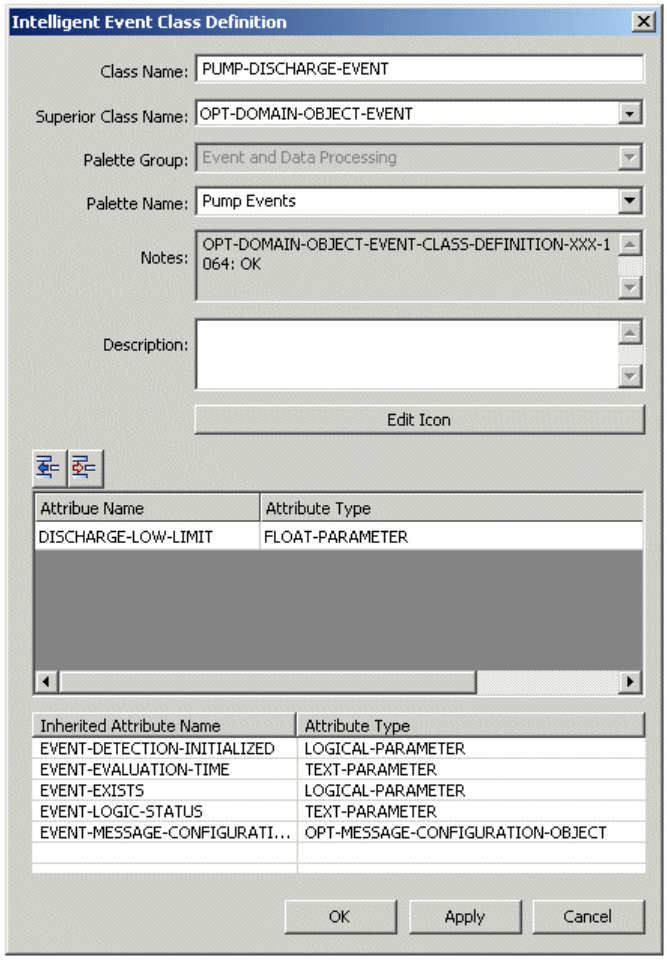


3 Configure the properties for each custom event definition in the hierarchy.

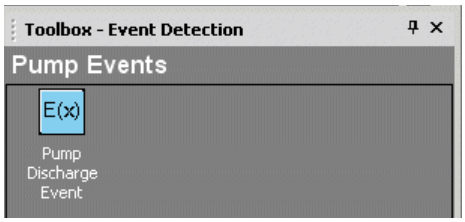
Configure a unique name, configure the icon for the class, and define class-specific attributes that the event requires. Any function, procedure, or method that you create for your new class can access the class-specific attributes. To display the event on a palette, configure the Palette Name. By default, the event appears in the specified palette in the Event Detection toolbox.

Here is the pump-discharge-event definition and its properties dialog. It defines the discharge-low-limit class-specific attribute, which is of type float-parameter. The event will appear in a Pump Events palette of the Event Detection toolbox.

PUMP-DISCHARGE-EVENT



Here is the Pump Events palette in the Event Detection toolbox, which includes the custom event block:



Configuring the Custom Event Logic

Once you have created a custom event hierarchy, the next step is to create the associated event logic for each event class in the hierarchy. Detecting events for domain objects can require complex mathematical analysis, which you typically perform within a method, procedure, or function.

The gedp-user-routine attribute of a domain object event specifies the name of the method, procedure, or function to execute when the block evaluates. By default, the gedp-user-routine is a method named opt-domain-object-event-logic with this signature:

```
opt-domain-object-event-logic
    (block: class opt-domain-object-event, input-value: item-or-value,
    domain-object: item-or-value)
    -> return: truth-value
```

where:

- *block* is the opt-domain-object-event block.
- *input-value* is the item or value on the input path to the block.
- *domain-object* is the domain object that is assigned to the GEDP diagram on which the opt-domain-object-event block resides or, if no domain object is assigned, the symbol none.

Thus, the custom event block takes any item or value on its input path, and it passes a discrete or fuzzy truth-value on its output path.

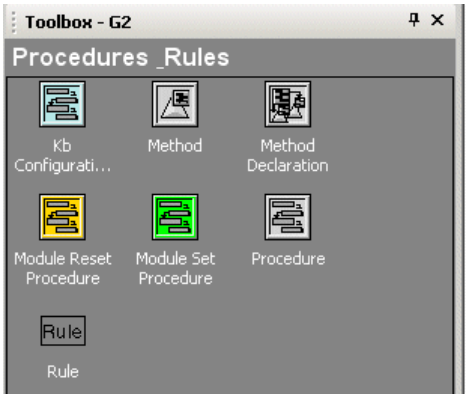
The easiest way to create the event logic for a custom event is to create a method for the custom event definition named opt-domain-object-event-logic. The block automatically calls this method when it executes.

If you do not want to implement a method, you can also create a procedure or function with the same signature of the opt-domain-object-event-logic method. You must then specify the gedp-user-routine attribute to be the name of your custom procedure or function in the attribute-initializations of the custom event

definition. The custom event block that appears on the palette will then use the user routine you initialized in the class definition instead of the default method.

To configure the custom event logic:

1 Choose View > Toolbox - G2, display the Procedure palette, and create a Method:



2 Choose Properties on the method and create a method named opt-domain-object-event-logic with the signature above that describes the event logic.

For example, here is the opt-domain-object-event-logic method for the pump-discharge-event custom event definition. The method checks for the existence of an opt-pressure-sensor that is the-discharge-pressure-of the input value of the event, which is an instance of opt-pump. If the pv of the pressure sensor is below the discharge-low-limit of the event, then the method returns true. The method is called for any pump-discharge-event instance.


pump-discharge-event::opt-domain-object-event-logic

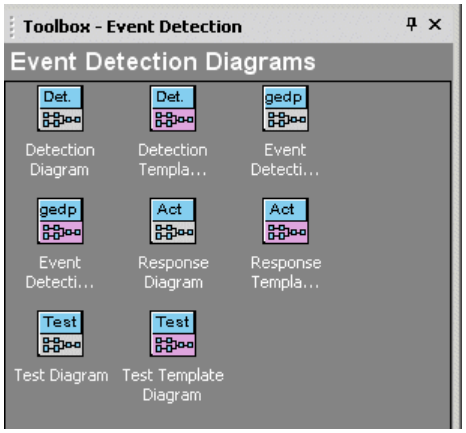
```
opt-domain-object-event-logic (event: class pump-discharge-event,
input-value: item-or-value, domain-object: item-or-value) = (item-or-value)
begin
    {Check for Discharge Pressure sensor existence and if pv violates low limit}
    if there exists an opt-pressure-sensor P that is
        the-discharge-pressure-of input-value
        and the pv of P < the discharge-low-limit of event
    then return true
    else return false
end
```

Configuring the Generic Event Detection Diagram for the Custom Event

Once you define the event logic for the classes in your event hierarchy, you can create custom event blocks and use them with other GEDP blocks to create generic dataflow templates for monitoring domain object classes for event detection, testing, or response.

To configure the generic event detection diagram for the custom event:

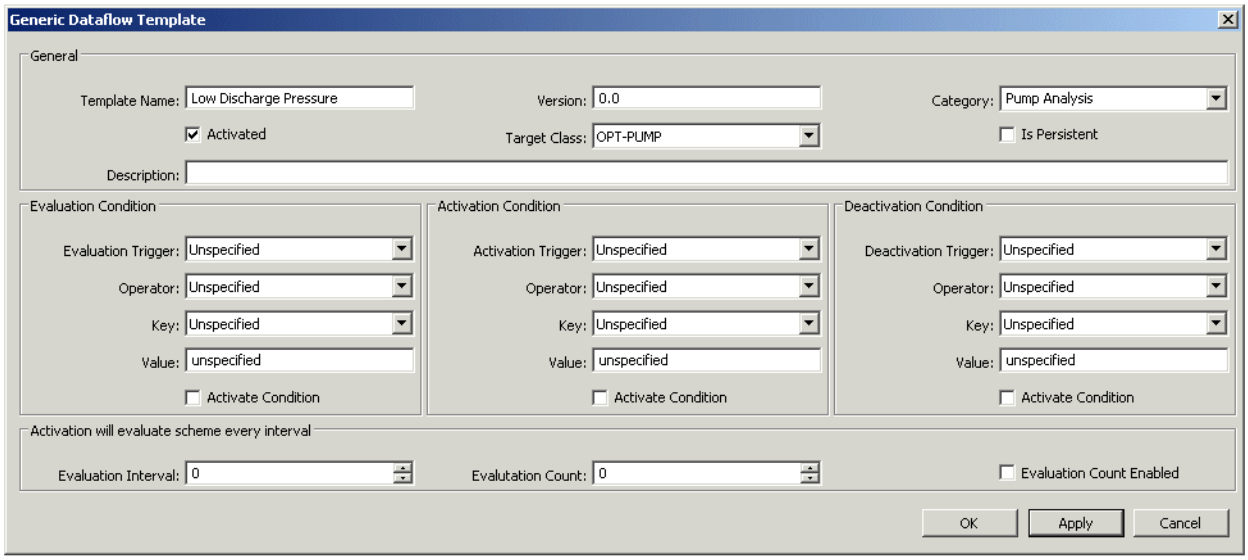
1 Choose View > Toolbox - Event Detection, display the Event Detection Diagrams palette, and create an Event Detection Template of the desired type:



2 Configure the properties of the Event Detection Template for the custom event.

Here is the Event Detection Template for the Low Discharge Pressure event:

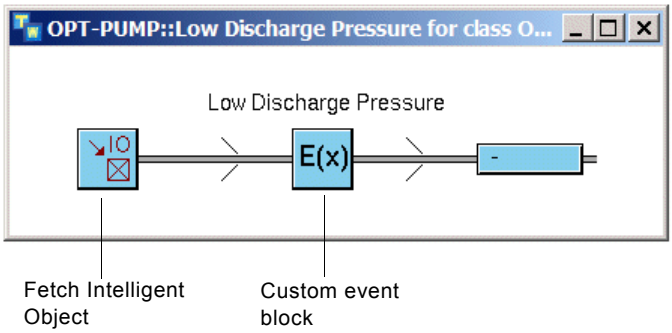
"OPT-PUMP::Low Discharge Pressure"



The Template Name is Low Discharge Pressure, the Target Class is opt-pump, the Category is Pump Analysis, the Evaluation Interval is 5. Enable the Activated option to activate the diagram. Enable the Is Persistent option to allow limits configured in the specific event detection diagram to persist through application initialization.

3 Create a generic event detection template that incorporates your custom event block.

Here is the detail of the Low Discharge Pressure generic event detection template for an opt-pump, which uses the Low Discharge Pressure custom event block:

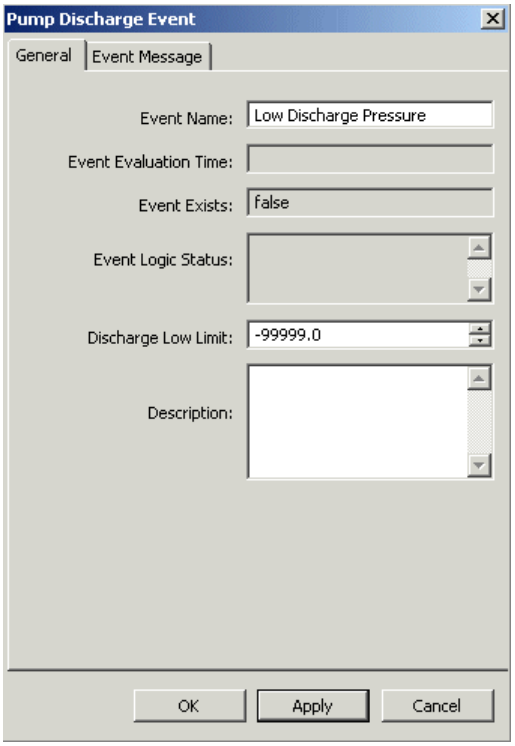


You create the Pump Discharge Event from the Pump Events palette in the Event Detection toolbox. The Fetch Intelligent Object block gets the opt-pump instance for the specific event detection diagram and passes it to the custom Low Discharge Pressure event. For details, see the *G2 Event and Data Processing User's Guide*.

When this event detection diagram is executed for an instance of opt-pump, the pump is passed as the input to the Low Discharge Event block, which executes the opt-domain-object-event-logic method to determine if a low discharge pressure exists.

4 Configure the user-defined attributes of the custom event block in the properties dialog.

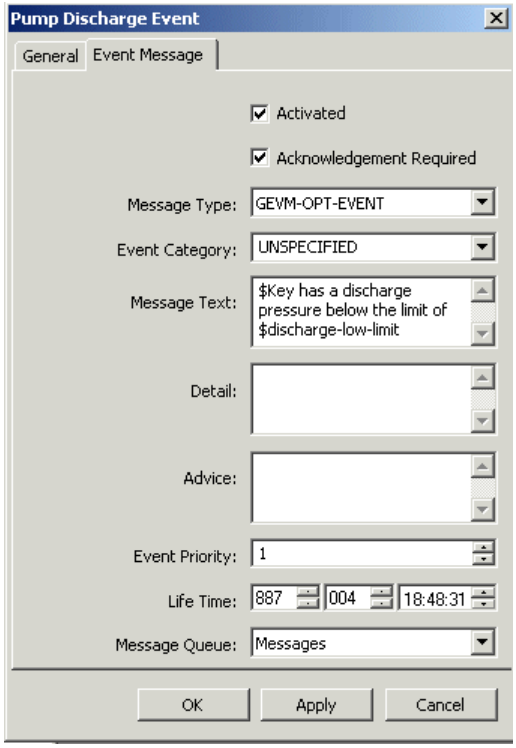
Here is the properties dialog for the Pump Discharge Event event block, which configures the Discharge Low Limit to be -99999.0:



You configure limits for the custom event block in the specific event detection diagrams for specific domain objects. See Configuring the Specific Event Detection Diagram for the Custom Event.

5 To generate a message when the event is true, click the Event Message tab in the properties dialog for the custom event block in the generic diagram and configure the Message Text and other properties, as needed.

Here is the Event Message tab for the properties dialog of the Low Discharge Pressure event block:



In the Message Text, the $Key and $discharge-low-limit references cause actual values for the name of the pump being monitored and the Discharge Low Limit of the event to be substituted in the message. The Event Priority is configured to be 1.

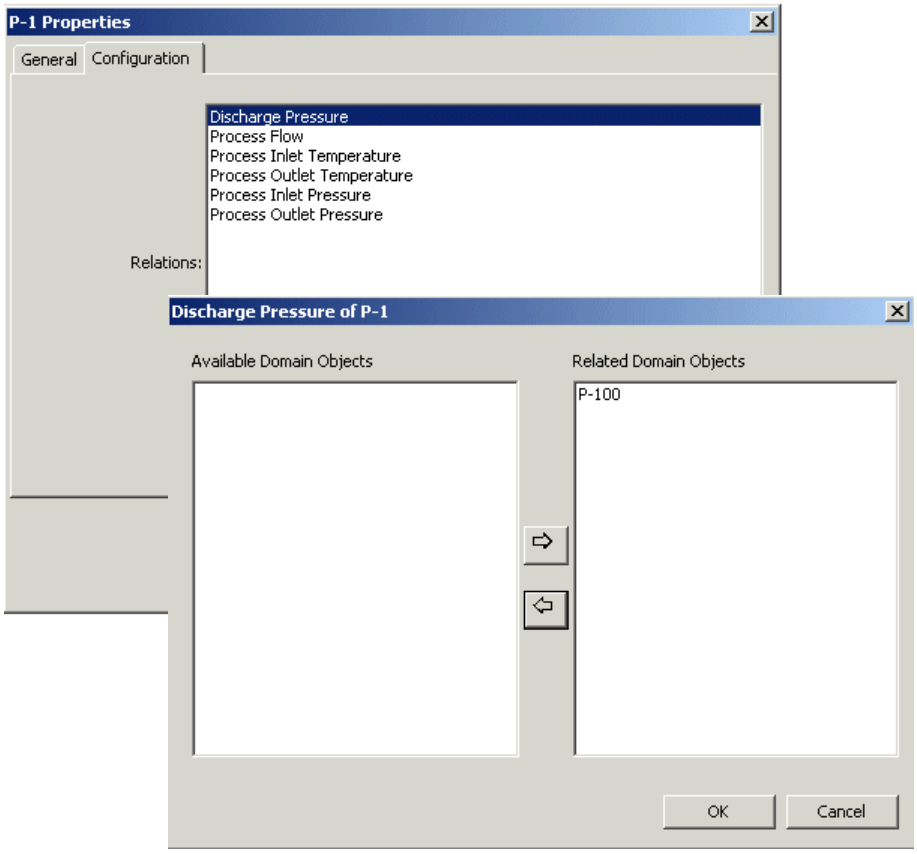
Configuring the Specific Event Detection Diagram for the Custom Event

Once you have created and configured the generic event detection template for the target class, you must configure the specific event detection diagram for each instance of the target class in your process map. For details, see Configuring Domain Objects.

To configure the specific event detection diagram for the custom event:

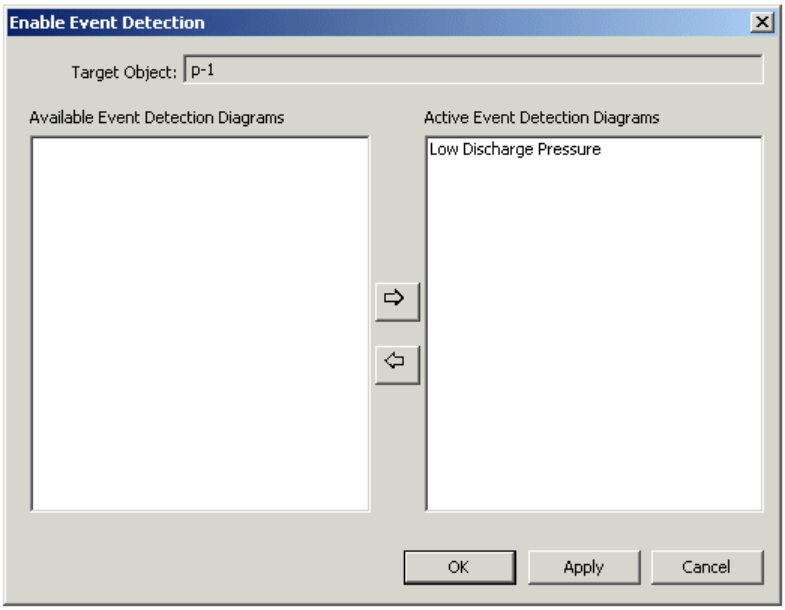
1 Create and configure a process map that includes instances of the target class and any related sensors that the custom event requires.

Here is a pump and a pressure sensor, where the P-100 pressure sensor is configured as the Discharge Pressure related sensor of the P-1 pump:


P-1 Properties
General
Configuration
Discharge Pressure
Process Flow
Process Inlet Temperature
Process Outlet Temperature
Process Inlet Pressure
Process Outlet Pressure
Relations:
Discharge Pressure of P-1
Available Domain Objects
Related Domain Objects
P-100
OK
Cancel

2 Enable the custom event detection diagram for the domain object in the process map.

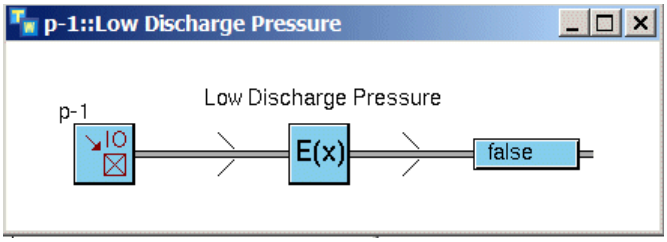
To enable the specific Low Discharge Pressure event for the P-1 pump, choose Enable Dataflow Event Detections and move the Low Discharge Pressure event from the Available list to the Active list:



You can also enable the custom event detection diagram by initializing the application.

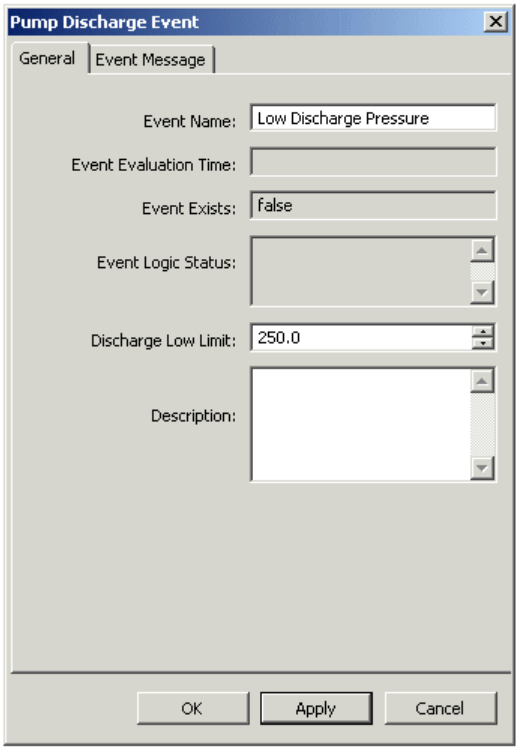
3 Choose Show Logic on the domain object and choose the specific event detection diagram to configure.

Here is the specific Low Discharge Pressure event detection diagram for the P-1 pump:



4 Configure the attributes of the custom event block in the specific diagram for the particular domain object.

To configure the Pump Discharge Event block for the P-1 pump, choose Properties on the Low Discharge Pressure Event block:

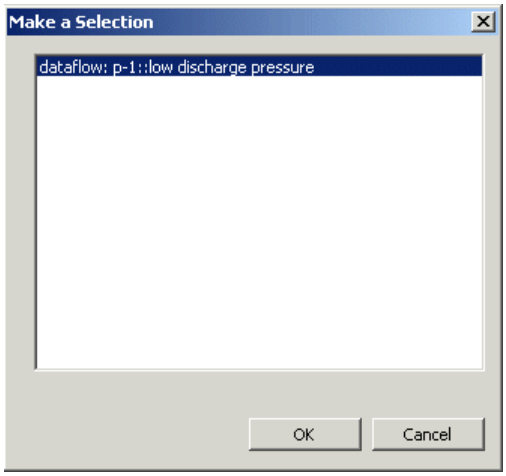


If the Discharge Low Limit is set to 250.0, a message is generated when the PV of the Discharge Pressure related sensor of the P-1 pump goes below 250.0.

Testing the Custom Event

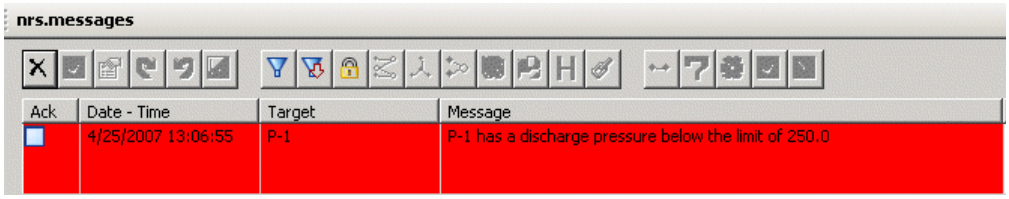
To test the custom event:

1. Display the properties dialog for the P-100 pressure sensor and display the properties dialog for the p-100.pv internal datapoint.
2. Configure the Datapoint Value to be a value below the Discharge Low Limit of the Low Discharge Pressure custom event.
3. Choose Run Detection Logic on the P-1 pump to test the logic and choose the specific event detection diagram to run:



4. Choose View > Message Browser.

Here is the message that occurs when the discharge pressure is 249:



23

Customizing Optegrity

Describes how to customize Optegrity.

Introduction

Optegrity allows you to customize and work with these features:

- Objects.
- User preferences.
- Application initialization.
- Data source integration with DCS systems.
- Engineering unit conversions.

- These aspects of messaging:
  - Message browsers.
  - Message log handling.
  - Message classes.
  - Message text substitution.
  - Message color lookup tables.
  - Timestamp format.
  - Message correlation.
  - Message priority escalation.
- Top menu bar and popup menus.

To customize Optegrity, you must switch to Developer mode. You might also be required to switch to System-Administrator or Administrator mode.

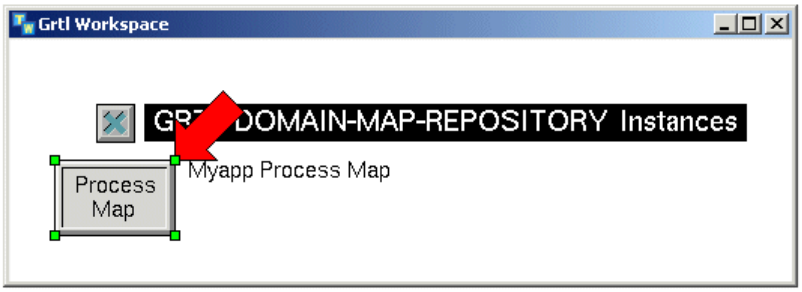
To customize Optegrity, you use objects in the various palettes of the G2 toolbox, for example, G2 procedures and methods.

Interacting with Objects in Developer Mode

In general, in Modeler mode, Optegrity hides container objects, such as the process map container. When you choose an object in the Project menu, Optegrity displays the properties dialog or the model detail, as appropriate for the type of object. For example, it displays the properties dialog for data replay, and it goes to the model detail for a process map. Similarly, you cannot go to these objects through the Manage dialogs.

You can access objects directly through the Navigator or search by using the Go To menu choice. You can also switch to Developer mode to enable the Go To menu choice in the Manage dialogs.

Depending on the type of object, when you go to the object, the object appears in a repository. For example, here is the result of choosing Go To on a Process Map container:



If you prefer to interact with objects directly, you can configure the Indicate Items user preference to go directly to the object itself in its repository when choosing objects from the Project menu. For details, see Configuring User Preferences,

Using the G2 Toolbox

Optegrity provides palettes for various core G2 objects for customizing applications. The Core G2 Objects palette is only available in Developer mode.

For information on using these G2 objects, see the *G2 Reference Manual*.

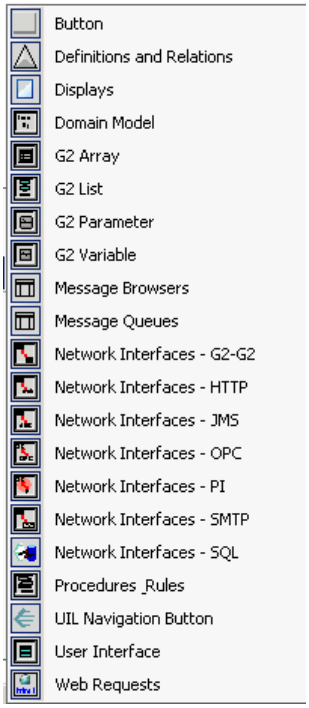
To display the Core G2 Objects palettes:

1 Switch to Developer mode.

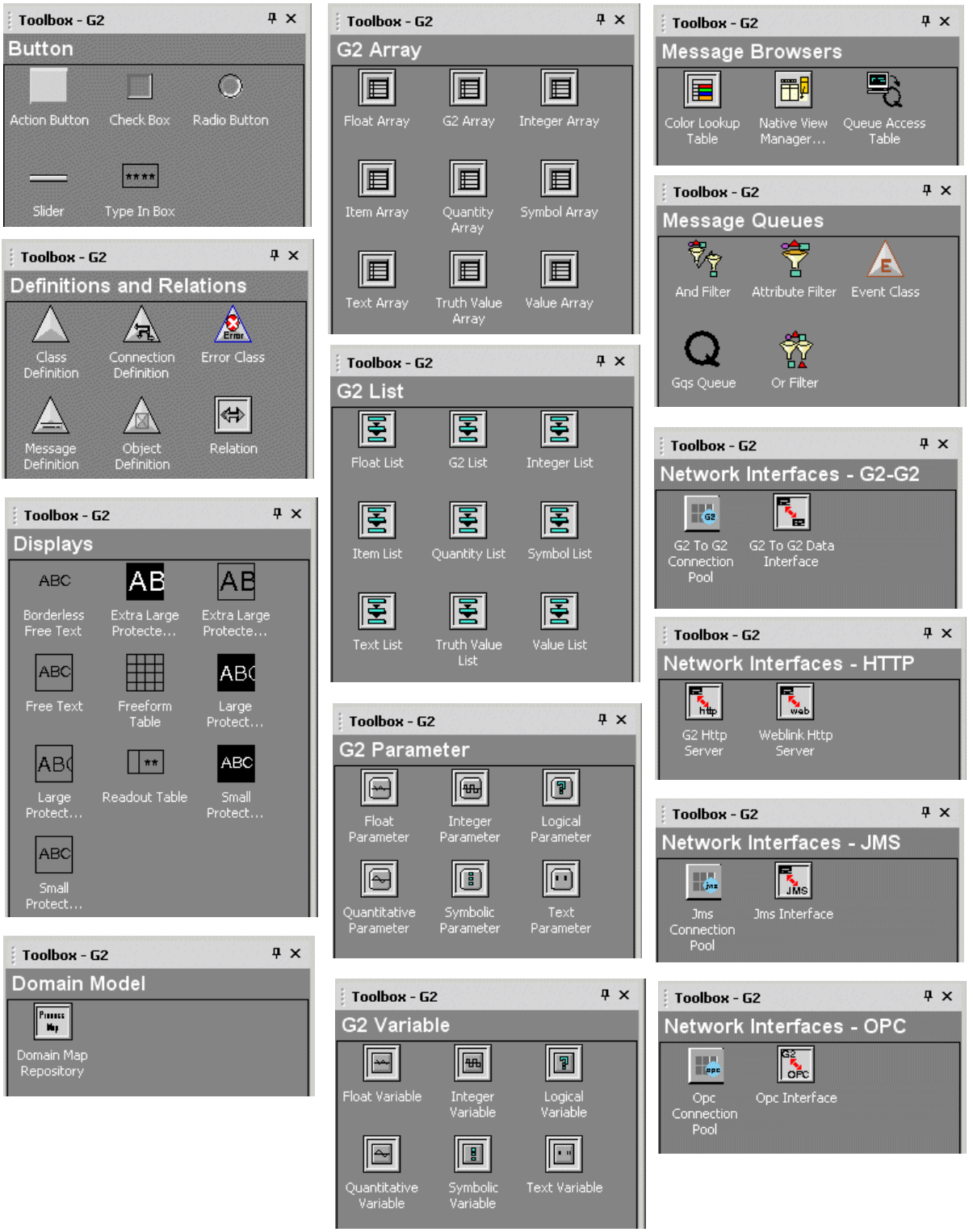
For details, see Switching User Modes.

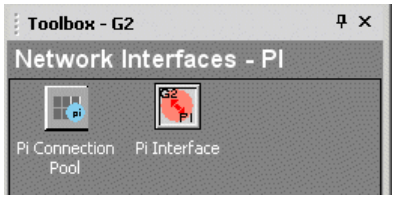
2 Choose View > Toolbox - G2.

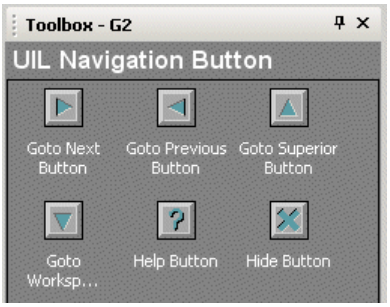
Here are the palettes from which to choose:

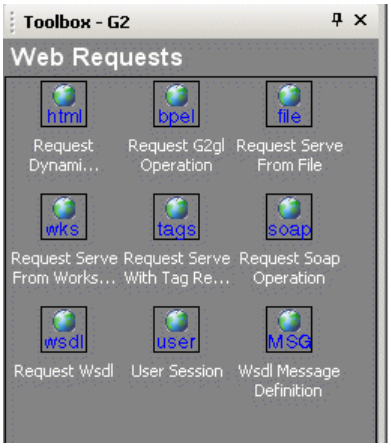


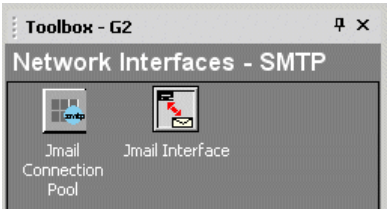
Here are the palettes in the G2 toolbox:

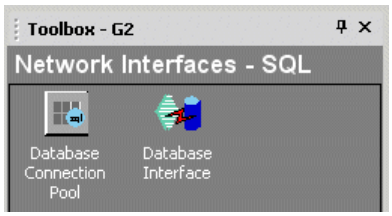


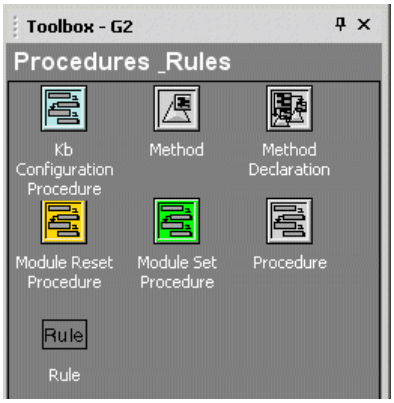












For information on this palette...	See...
Button Definitions and Relations Displays G2 Array G2 List G2 Parameter G2 Variable Procedures and Rules	*G2 Reference Manual*
Domain Model	Building a Process Map

For information on this palette...	See...
Network Interfaces	Configuring Network Interfaces
Interface Pools	Using Interface Pools
Message Browsers Message Queues	*G2 Event Manager User's Guide*
User Interface	*G2 Run-Time Library User's Guide*
UIL Navigation Buttons	*G2 GUIDE User's Guide*
Web Requests	*G2 Web User's Guide*

Configuring User Preferences

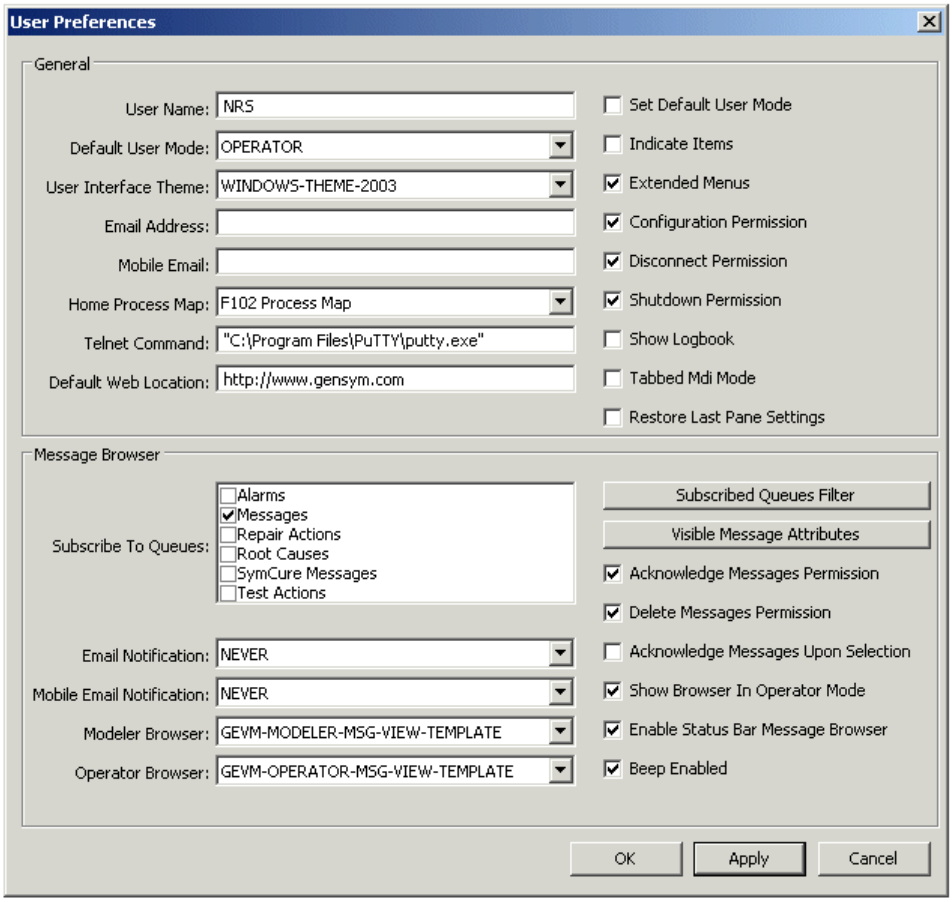
In System-Administrator and Administrator modes, you can configure additional attributes for each user preference. For information about basic user preferences, see Configuring User Preferences.

In addition to configuring user preferences that Optegrity creates automatically when you start the server and client, you can also create new user preferences for specific clients, based on their user name.

To configure user preferences:

1. Switch to System-Administrator or Administrator mode.
2. Choose Project > System Settings > Users and choose the user preference to configure or create a new user preference, using the Manage dialog.

The User Preferences dialog appears:



3 Configure the customization attributes, as follows:

Attribute	Description
General	
User Name	The user name associated with the user preference. The default User Name is the user name for the current user. To create a user preference for a new user, enter the user name of a user in the *g2.ok* file, which must be a symbol. For details, see Chapter 62 "Licensing and Authorization" in the *G2 Reference Manual*.
Configuration Permission	Whether to allow the user to switch to configure the application in Modeler mode. By default, Configuration Permission is enabled, which means when the operator clicks the close button in the operator interface, Optegrity switches to Modeler mode. In Modeler mode, you can create and configure applications, using the top menu bar. When Configuration Permission is disabled, Optegrity closes the client when the operator clicks the close button in the operator interface. We recommend that you disable this option for operators.
Disconnect Permission	Whether to allow the user to disconnect the client from the server, using the File > Close menu choice. By default, all users can disconnect the client from the server.
Shutdown Permission	Whether to allow the user to shut down the server, using the File > Exit menu choice. By default, modelers and operators cannot shut down the server.
Show Logbook	Whether to show the G2 Logbook when an error occurs.

Attribute	Description
Message Browser	
Subscribe to Queues	The message queues to which the specified user subscribes. By subscribing to a queue, the user sees messages associated with that queue in the Message Browser view of the operator interface. Messages for the Messages queue appear in the Message Browser. These message include intelligent object messages; SymCure messages configured for root causes, alarms, test actions, and repair actions; messages generated from GEDP diagrams; and general informational messages. **Note**: SymCure queues specific events and actions in the Alarms, Root Causes, Test Actions, and Repair Actions queues, and displays them in the four diagnostic console browsers. SymCure also allows you to configure operator messages for individual events and actions, which provide richer information to the operator, including message contents, advice, and detail. Operator messages for events and actions are displayed in the Message Browser. If you configure operator messages for SymCure events and actions, we recommend that you do not subscribe to the four SymCure queues, because this may result in duplicated messages in the Message Browser.
Subscribed Queues Filter	The default filter to apply for filtering messages in the subscribed queues. For details, see Configuring Filters.
Visible Message Attributes	The properties to show in the message details. By default, all properties are showing. For details, see Configuring Message Details.
Acknowledge Messages Permission	Whether to allow the user to acknowledge messages in the Message Browser view of the operator interface. By default, operators can acknowledge messages.
Delete Messages Permission	Whether to allow operators to delete messages in the Message Browser view of the operator interface. By default, users can delete messages.

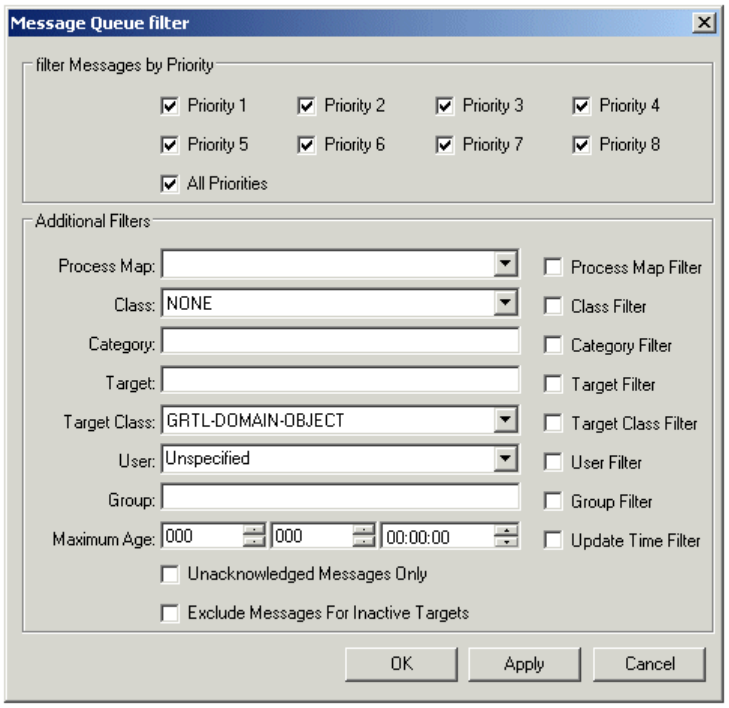
Configuring Filters

By default, the Message Browser shows all messages. You might want to restrict the messages that appear for a particular user in a given user mode. You can filter messages, based on a variety of criteria, including priority, object type, category, and age.

To configuring filters:

➔ In the user preferences dialog, click the Subscribed Queues Filter button and configure the filter criteria.

Here is the default filter dialog:



Attribute	Description
Filter Messages by Priority	The priority of the messages to show.
Process Map Process Map Filter	The process map for which to show messages when Process Map Filter is enabled.
Class Class Filter	The classes for which to show messages when Class Filter is enabled.

Attribute	Description
Category Category Filter	The category of messages to show when Category Filter is enabled.
Target Target Filter	The target object for which to show messages when Target Filter is enabled.
Target Class Target Class Filter	The target class for which to show messages when Target Class Filter is enabled.
User User Filter	The user for which to show messages when User Filter is enabled.
Group Group Filter	The group for which to show messages when Group Filter is enabled.
Maximum Age Update Time Filter	The maximum age of messages to show when Update Time Filter is enabled.
Unacknowledged Messages Only	Whether to show unacknowledged messages only. By default, acknowledged messages are visible.
Exclude Messages For Inactive Targets	Whether to exclude messages if the target object status is inactive.

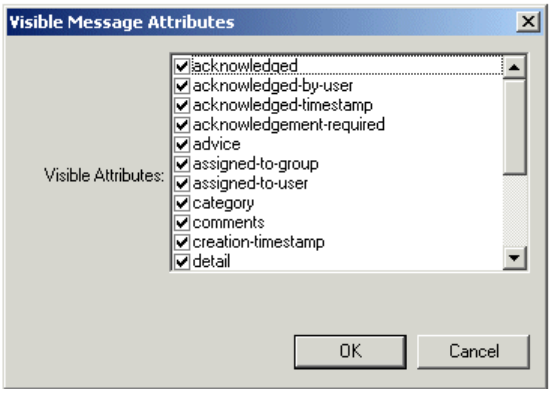
Configuring Message Details

By default, when you click the Properties button for a message in the Message Browser, all message details appear. You can restrict the contents of the message details dialog.

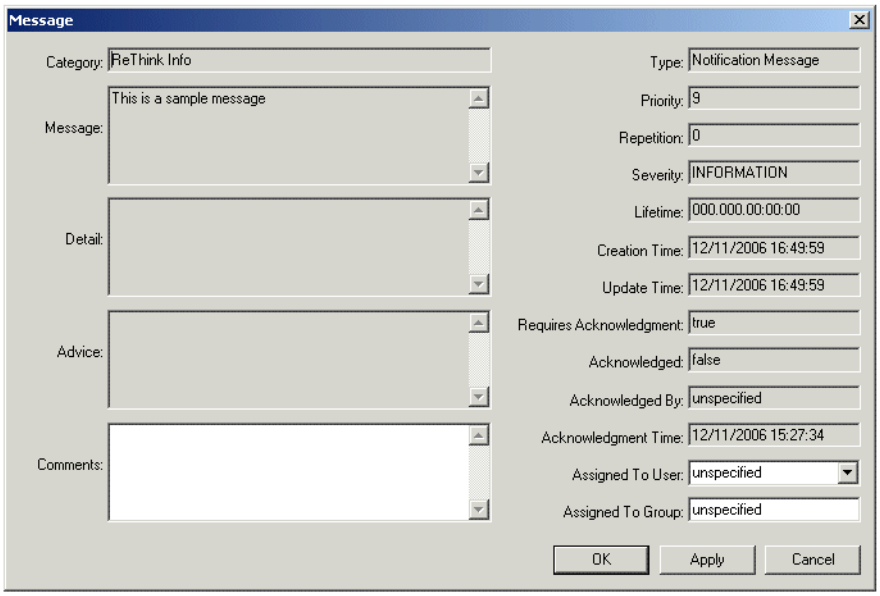
To configure message details:

➔ In the user preferences dialog, click the Message Details button and configure the attributes to appear in the Message Detail Selection dialog by removing attributes from the Selected Attributes column, as needed.

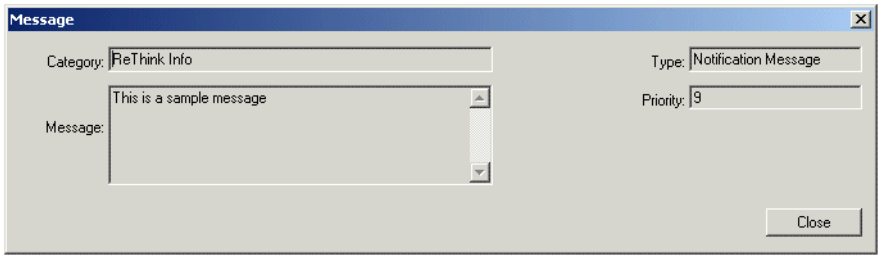
By default, all message details appear:



Here is the default message details for a message with all attributes showing:



Here is the message details for a message with just four attributes visible:



Application Initialization

You can call the following APIs to initialize and uninitialize applications programmatically.

guif-initialization-process-maps
(*win*: class ui-client-item)

Initializes process maps and domain objects, creates external datapoints, recompiles SymCure diagrams, and clears message browsers. Initializing process maps creates specific GEDP diagrams for each domain object in a process map.

guif-uninitialization-process-maps
(*win*: class ui-client-item)

Deletes specific event detection diagrams for domain objects in all process maps. You might need to uninitialize a process map after making changes to the GEDP diagram for a domain object.

Custom Data Source Integration

Optegrity supports custom data source integration for PLC or DCS systems other than OPC and PI, which are built into Optegrity. Custom network interfaces support the same functionality as the built-in network interfaces, namely creating external datapoints from a specification in a CSV file, relating those datapoints to internal datapoints, and replaying datapoint from a CSV file.

Creating a custom data source requires the G2 Data Source Manager (GDSM) and G2 Run-Time Library (GRTL) modules. Here are the high-level steps for creating a custom data source.

To create a custom data source:

1. Create a new KB that requires the GDSM and GRTL modules to contain the custom data source and external datapoint definitions.
2. Create a GDSM network interface class and implement its methods.

   For details, see Creating the Custom Network Interface Class.
3. Create GRTL external datapoint classes of the required types and implement their methods.

   For details, see Creating the Custom Network Interface Class.
4. Create a CSV template file to import the external datapoints.

Creating the Custom Network Interface Class

The network interface class definition must inherit from gdsm-dcs-interface. The class can also multiply inherit from the built-in OPC or PI interface classes, gdsm-opc-interface or gdsm-pi-interface.

You must implement the following method for your custom network interface class:

gdpm-get-external-datapoint-class-name
(*io*: class gdsm-external-system-interface, *datapoint-type*: symbol)
-> *external-datapoint*: symbol

Returns a valid grtl-external-datapoint subclass given a GDSM network interface and a datapoint type. Values for *datapoint-type* are: real, float, integer, text, logical, or digital.

You can optionally implement the following method for your custom network interface class:

gdsm-network-interface-configure
(*io*: class gdsm-external-system-interface,
network-pool: class gdsm-network-connection-pool)

Configures a GDSM network interface, using a network pool. A network pool can contain multiple network interface objects on its detail. When allocating resources for load balancing, Optegrity chooses a network interface, based on availability.

gdsm-network-interface-get-status
(io: class gdsm-external-system-interface)
-> *connection-status*: symbol

Determines the status of the network connection between a GDSM network interface and the external bridge process, refreshes the icon of the interface, based on the status, and returns the status. The return values are one of these

symbols: connected, not-connected, in-transition, timed-out, or connection-lost.

gdsm-network-interface-connect
(*io*: class gdsm-external-system-interface, *host*: text, *port*: integer, *connection-timeout*: integer)

Connects a GDSM network interface to an external bridge process, given a host and port. The connection times out after *connection-timeout* seconds.

gdsm-network-interface-disconnect
(*io*: class gdsm-external-system-interface)

Disconnects a GDSM network interface from the bridge process.

gdsm-network-interface-animate
(*io*: class gdsm-external-system-interface, *allocated*: truth-value)

Animates a GDSM network interface as it gets allocated and deallocated for communication via the bridge, where *allocated* is true when the interface is allocated for communication.

grtl-show-properties
(*item*: class gdsm-external-system-interface,
client: class ui-client-item)
-> *exists*: truth-value

Displays the properties dialogs of a GDSM network interface in a given client window. The method returns true if the interface exists; otherwise, it returns false.

gdsm-network-interface-handle-connection-timeout
(*io*: class gdsm-external-system-interface)

Implements the connection timeout behavior of a GDSM network interface.

gdsm-network-interface-handle-connection-failure
(*io*: class gdsm-external-system-interface)

Implements the connection failure behavior of a GDSM network interface.

Creating Custom External Datapoint Classes

Creating Custom External Datapoint Classes

When creating a custom external datapoint class, we recommend inheriting from these classes:

- gsi-data-service, gdpm-io-variable, variable-or-parameter
- The specific GRTL external datapoint type: grtl-external-float-datapoint, grtl-external-integer-datapoint, grtl-external-text-datapoint, grtl-external-symbolic-datapoint, or grtl-external-logical-datapoint.

Your custom class should define attributes to uniquely identity variables and their associations in the control system.

You must implement the following methods for your external datapoint classes:

gdpm-io-configure-variable
(*io-variable*: class gdpm-opc-variable, *block*: class gdpm-io-block, *tokens*: sequence, *client*: class ui-client-item)

Creates and configures external datapoints from a CSV file and places them on the detail of an External Datapoints container. The method should extract specific configuration information from the sequences derived from the CSV file and assign them to the newly created external datapoints.

grtl-show-properties
(*item*: class gdpm-opc-variable, *client*: class ui-client-item)
-> truth-value

Displays the properties dialog of an external datapoint. The method returns true if the user presses OK to exit the dialog, and false otherwise.

Example: TDC Data Source Integration

Here are the key classes and methods for defining a custom DCS interface class and associated external datapoints for a TDC system.

The complete example is available in *gdpm-demo.kb*, which is located in the *g2i examples* directory.

Custom Network Interface Class

Here is a custom DCS interface class for interfacing with a TDC system:

GDPM-G2TDC-GATEWAY-INTERFACE

Direct superior classes gdsm-dcs-interface

Class specific attributes g2tdc-configuration is an instance of a g2tdc-configuration, initially is an instance of a g2tdc-configuration;
g2tdc-stats is an instance of an item-list, initially is an instance of an item-list

Here is the implementation of the gdpm-get-external-datapoint-class-name method for the custom TDC interface class:

gdpm-g2tdc-gateway-interface::gdpm-get-external-datapoint-class-name

```
gdpm-get-external-datapoint-class-name(io: class gdpm-g2tdc-gateway-interface ,
datapoint-type: symbol) = (symbol)
{
This method should return a valid grtl-external-datapoint subclass given the the io
interface type and the datapoint type. Values for datapoint-type maybe real, float,
integer, text, logical, or digital.
}
variable-class: symbol ;
begin
    case (datapoint-type) of
        REAL:  variable-class = the symbol gdpm-g2tdc-pv;
        FLOAT: variable-class = the symbol gdpm-g2tdc-pv;
        INTEGER: variable-class = the symbol gdpm-g2tdc-integer ;
        otherwise: variable-class = the symbol gdpm-g2tdc-pv ;
    end;
    return variable-class;
end
```

Custom External Datapoint Classes

Here is the class hierarchy for the custom external datapoint classes required by the TDC interface:

GDPM-G2TDC-DATA-POINT

Direct superior classes: gsi-data-service, gdpm-io-variable, variable-or-parameter

Class specific attributes: g2tdc-tag is a symbol, initially is none;
g2tdc-parameter is a symbol, initially is pv;
g2tdc-parameter-index is an integer, initially is 0;
g2tdc-value-type is a symbol, initially is none

GDPM-G2TDC-INTEGER

Direct superior classes: gdpm-g2tdc-data-point, grtl-external-integer-datapoint

GDPM-G2TDC-REAL

Direct superior classes: gdpm-g2tdc-data-point, grtl-external-float-datapoint

GDPM-G2TDC-PV

Direct superior classes: gdpm-g2tdc-real

Here is the implementation of the method for the custom TDC interface class:

gdpm-g2tdc-data-point::gdpm-io-configure-variable

```
gdpm-io-configure-variable(io-variable: class gdpm-g2tdc-data-point , block: class
gdpm-io-block, tokens: sequence, win: class ui-client-item)

{
Configures the external datpoint TDC specific configurations based on the values
from the csv file. Starting at the index 24 within the tokens start the column in the
spreadsheet that are specific to each gsi interface. In this example we extract the tag
and the index.
}

bridge-name: symbol = the bridge-name of block;
interface: class gdpm-g2tdc-gateway-interface;
tag: symbol;
index: integer;

begin

    call next method;

    { --- Setup the TDC specific fields using the values from the csv file }

    tag = call grtl-uppercase-symbol( "[TOKENS[24]]" );
    if text-begins-with-quantity( "[TOKENS[25]]" ) then
        index = quantity( "[TOKENS[25]]" )
    else
        index = 0;
```

```
    conclude that the g2tdc-tag of io-variable = tag;
    conclude that the g2tdc-parameter-index of io-variable = index;

    { --- The interface object must exist, otherwise create it }

    if not(there exists a gdpm-g2tdc-gateway-interface interface named by
        bridge-name) then
        call gdpm-io-create-interface(block, the symbol
            gdpm-g2tdc-gateway-interface, win);

end
```

Here is the implementation of the method for the custom TDC interface class:

gdpm-g2tdc-data-point::grtl-show-properties

```
grtl-show-properties (Itm: class gdpm-g2tdc-data-point, Client: class ui-client-item) =
(truth-value)

{
This method will open the property dialogs of an object if defined
}

ret: truth-value = true;
Dlg: item-or-value;
Btn: item-or-value;

begin
    Dlg, Btn = call uil-control-dialog-callback
        ("gdpm-external-datapoint-tdc-configuration-dialog" , the symbol none,
        gdpm-external-datapoint-tdc-variable-configuration-dialog-actions,
        Itm, Client);

    if Btn exists and Btn is an uil-button and the label of Btn = "OK" then ret =
        true;
    if Dlg exists and Dlg is a uil-dialog then call
        grtl-cleanup-and-release-dialog (Dlg, Client);
    return ret;

end
```

To view the properties dialog definition, see the *gdpm-demo.kb* example KB.

Working with Engineering Unit Conversions

You can use the following API procedures and functions to work with engineering unit conversions programmatically your Optegrity or G2 application. They provide the ability to:

- Convert values from one engineering unit to another for a given dimension.
- Convert from internal to external units, and from external to internal units for a given dimension.
- Get the internal units of a given dimension in either the metric or English system.
- Get the units for a given parameter of a sensor or controller.

Dimension Types

Optegrity defines built-in engineering units for a number of dimension types, which it uses for displaying engineering units. The dimension types categorize the units that the various parameters and metrics require.

In the API procedures and functions that follow, the *dimension-type* is one of these symbols:

pressure
length
area
volume
volumetric-flow
volumetric-heat-capacity,
volumetric-enthalpy
mass
mass-flow
mass-heat-capacity
mass-enthalpy,
density
density-slope
specific-volume
temperature
power
heat-transfer,
time
molar-volume
voltage
current

Dimension Units

Optegrity allows you to configure the units of a given dimension for entering parameters and displaying metrics for domain objects, in a given unit system. For example, if the unit system is metric and you are configuring the units for the process-pc of the Heater Efficiency derived sensor of a heater, you can choose from the following metric units:

kilojoule per cubic meter degree celsius	kj/m3-C	kj/m3-deg C
joule per cubic meter degree celsius	j/m3-C	j/m3-deg C
kilocalorie per cubic meter degree celsius	kcal/m3-C	kcal/m3-deg C
calorie per cubic meter degree celsius	cal/m3-C	cal/m3-deg C

Similarly, if the unit system is english, you would choose from these units:

btu per cubic foot degree fahrenheit	btu/ft3-F	btu/ft3-deg F

In the API procedures and functions that follow, the dimension units that you specify as arguments should be one of the built-in synonyms, as a text. These are either input or output units, or internal or external units, depending on the API.

Note that spaces are stripped from the units, so the spaces you enter do not matter.

Note If you enter a dimension unit that is not one of the built-in synonyms, Optegrity automatically creates the specified unit synonym and places it in the Undefined-Dimensions category in the Unit Synonyms dialog. Typically, undefined synonyms constitute typographical errors in the API procedure code.

Conversion Status

The API procedures that convert engineering units all return the following status values, as a text, to indicate whether the conversion was successful:

Status	Description
converted	A conversion definition was found and the input value was successfully converted.
undefined	A conversion definition was not found because of an unrecognized *dimension-type*, *input-units*, or *output-units*. This status can mean that either a conversion has not been defined, or that the user is supplying an input or output synonym for an existing conversion that is not yet recognized and needs to be added to the appropriate synonym definition. Check the API call for correct units, or create custom synonyms or custom conversion definitions, as needed.
unrecognized equation type	The equation type for the conversion definition was not recognized as one of the following symbols: MULTIPLIER-ONLY MULTIPLY-FIRST OFFSET-ONLY OFFSET-FIRST For custom conversions, modify the Equation Type. For built-in conversions, contact Gensym support.
missing eu organizer	The GEUC organizer object is missing. This error can occur only if a user or procedure specifically deletes the packaged organizer object. Report occurrences of this status to Gensym support.
no sequence	The requested dimension could not be matched in the GEUC conversion organizer object. Check the API call for the correct dimension. The API automatically creates an empty custom conversion definition, which you can edit through the manager.
zero multiplier	The Multiplier for the requested conversion is zero, which would create a divide-by-zero error if executed. For custom conversions, update the Multiplier parameter and retry the unit conversion. For built-in conversion, contact Gensym support.

Status	Description
procedure abort	A serious programming error occurred. Send G2 Logbook error messages to Gensym support for analysis.
abort	A serious programming error occurred. Send G2 Logbook error messages to Gensym support for analysis.

API Procedures

geuc-convert-engineering-units
(*dimension-type*: symbol, *input-value*: quantity,
input-units: text, *output-units*: text)
-> *output-value*: quantity, *status*: text

Converts an input value to an output value of a given dimension, given the input and output units.

Argument	Description
dimension-type	The dimension type, as a symbol. See Dimension Types.
input-value	The input value to convert.
input-units	The input units to use for the conversion, as a text.
output-units	The output units to use for the conversion, as a text.

Return Value	Description
output-value	The converted output value in the given output units.
status	The status of the conversion. See Conversion Status.

Custom Messaging

You can customize these aspects of messaging:

- The message browser.
- Message logging behavior.
- Message classes, message text, message color, and timestamp format.
- Message correlation.
- Message priority escalation.

For a complete description of how to customize messaging, see the *G2 Event Manager User's Guide*.

Custom Menus

You can customize the menus that Optegrity displays in the top menu bar, as well as popup menus on items.

To customize menus in the Telewindows user interface, you use API procedures provided by GRTL. For details, the *G2 Run-Time Library User's Guide*.

To customize menus in the classic user interface, you must be familiar with G2 Menu System (GMS), which provides graphical tools and API procedures for extending the top menu bar and popup menus for various classes. For details, see the *G2 Menu System User's Guide*.

Custom Popups

Suppose you implement a custom DCS interface, and you want to add menu choices to the popup menu for the network interface or external datapoint classes. To do this, you must implement the following methods:

grtl-item-exec-menu-callback
(*target*: class gdpm-g2tdc-gateway-interface, *label*: symbol, *action*: symbol, *win*: class ui-client-item)

Called after the user selects a popup menu.

grtl-object-popup-menu-constructor
(*target*: class gdpm-g2tdc-gateway-interface, *user-mode*: symbol, *cascading-property-menu*: truth-value, *templates-list*: class item-list, *win*: class ui-client-item)

Called when a popup menu needs to be displayed. This procedure builds the popup menu, using the GMS features that dynamically create menus. The method can take contextual information into account to configure the item popup menu.

For a complete example, load the *gdpm-demo.kb* located in the *g2i examples* directory.

Implementing the Popup Constructor

This example shows a popup constructor that builds a popup menu for a custom DCS interface class. See Example: TDC Data Source Integration.


gdpm-g2tdc-gateway-interface::_guif-object-popup-menu-constructor

```
grtl-object-popup-menu-constructor(target: class gdpm-g2tdc-gateway-interface,
    user-mode: symbol, cascading-property-menu: truth-value,
    TemplatesList: class item-list, win: class ui-client-item)
{
This method is called whenever a popup menu of this class needs to be displayed.
The purpose of this procedure is to dynamically build the menu per item class. It
recursively walks through the class hierarchy to build the emnu structure and can into
account any contextual information to build and configure the popup menu system.
}
mct: class gms-choice-template;
cdt: class gms-dynamic-cascade-template;
sep: class gms-separator-template;
isModeler: truth-value = false;
isOperator: truth-value = false;

begin
    call next method;

    if user-mode = the symbol administrator or user-mode = the symbol
        developer or user-mode = the symbol modeler then
        isModeler = true;

    if isModeler or user-mode = the symbol operator then
        isOperator = true;

    create a gms-separator-template sep;
    insert sep at the end of TemplatesList;

    create a gms-choice-template mct;
    conclude that the gms-text-resource-group of mct = the symbol
        gdpm-demo-text-resources;
    conclude that the gms-label of mct = the symbol gdpm-demo-toggle-
        has-external-scheduler ;
    conclude that the gms-initially-enabled of mct = (if isModeler and the
        names of target exists then true else false);
    insert mct at the end of TemplatesList;
end
```

Implementing the Callback that Executes the Popup

This example shows the callback when displaying a popup menu for a custom DCS interface class. See Example: TDC Data Source Integration.


gdpm-g2tdc-gateway-interface::_guif-item-exec-menu-callback

```
grtl-item-exec-menu-callback(target: class gdpm-g2tdc-gateway-interface,
    label: symbol, action: symbol, Win: class ui-client-item)
{
This method is called when a popup menu is selected and needs to be excecuted.
The action specifies the menu and therefore the action to execute. This example is
very simple and the intend is to show the overall architecture.
}
begin

    { --- Execute action }
    case (action) of

    gdpm-demo-toggle-has-external-scheduler:
        begin
            conclude that the external-system-has-a-scheduler of target =
                not(the external-system-has-a-scheduler of target);
        end;
    otherwise :
        begin
            call next method;
        end;
    end
end
```

24

Configuring Startup Parameters

Describes the parameters that you can configure in the Optegrity startup file.

Introduction

You can configure a number of parameters in a startup file to define the default behavior of the various Optegrity modules. You configure these parameters in the *config.txt* file, which is located in the *g2i\kbs* directory.

The *config.txt* file defines default values for all these parameters, which you can modify, as needed. The parameters are organized in the file according to the module they control.

This chapter documents each parameter and it default value.

Installation Directory

INSTALL-DIR=C:\Program Files\Gensym\g2-2011

The default installation directory for the application.

GRTL

Applications

APPLICATION-ERROR-ENABLED=true

Configures the gfr-error-handling-enabled attribute in the gfr-startup-settings. For more information, see the *G2 Error Handling Foundation User's Guide*.

APPLICATION-ERROR-INFORM-ENABLED=true

If true, enables error logging to the G2 Logbook.

APPLICATION-ERROR-LOG-ENABLE=true

If true, enables error logging to the log file specified by APPLICATION-ERROR-LOG-FILE.

APPLICATION-ERROR-LOG-FILE=$APPLICATION-ROOT-DIRECTORY\logs\kb-errors.log

Specifies the default location of the log file. The directory can refer to $INSTALLATION-DIRECTORY, which is the default installation directory.

APPLICATION-LOCALIZATION-FILES=

The pathname to the GRTL resource file specified as a command-line option to G2 at startup. The pathname may contain an asterisk (*) in the file name, which is replaced with the language of the resource file to load, for example, english. There is no default.

APPLICATION-ROOT-DIRECTORY=$INSTALLATION-DIRECTORY

The root directory of the application. The default value is $INSTALLATION-DIRECTORY, which is the default user installation directory.

APPLICATION-URL=

The default URL to access the application via the Web. There is no default.

APPLICATION-IS-WEB-HOSTED=false

If true, specifies that the application is running in a hosted environment, which means, for example, it can restrict access to some functionality.

User Preferences

USER-PREFERENCES-CONFIGURATION-FILE=$APPLICATION-ROOT-DIRECTORY/g2i/data/user-preferences.txt

The location of the configuration file for all grtl-user-preferences. The file contains the preferences for all defined user preferences, each in its own section. The settings are imported from the file to configure the user preference objects in G2.

User Audit Files

USER-AUDIT-FILE-ENABLED=false

If true, logs all user log in, log out and change mode activities. The log file is specified in the user-audit-file.

USER-AUDIT-FILE=$APPLICATION-ROOT-DIRECTORY\logs\user-audit-trail.csv

The audit log file to use to log activities. The path name can start with the pattern $APPLICATION-ROOT-DIRECTORY or $INSTALLATION-DIRECTORY, which is replaced at runtime by their appropriate values.

UTC Offset

UTC-OFFSET=0

An integer giving the time offset for the current time zone from the UTC time, for example, -5 for the East Coast of the US. This attribute is currently used to generate the time axis for charts.

Repository

REPOSITORY-MODULE=top-level

The module name where Optegrity stores objects that are created dynamically, using the Project menu. By default, Optegrity stores these objects in the top-level module. You can also provide any valid module name.

Indicator Arrows

INDICATOR-DELETE-BY-DEFAULT=true

Determines the behavior of the indicator arrow Optegrity uses when you go to various objects. By default, the indicator is deleted after the timeout given by indicator-default-timeout. Set this parameter to false to cause the arrow to remain visible until the user clicks it.

INDICATOR-DEFAULT-TIMEOUT=60

The timeout after which the indicator arrow is removed, in seconds.

INDICATOR-DEFAULT-COLOR=red

The default color of the indicator arrow. You can set this parameter to any valid G2 color.

INDICATE-ITEMS=true

The default setting for the Indicate Items option in the user preferences properties dialog, which determines whether to go to objects directly or to show their properties dialog or detail, depending on the type of object. For details, see Configuring User Preferences.

Timestamp Format

Optegrity provides the following parameters in the configuration file for configuring the timestamp format that Optegrity uses in the Message Browser:

DATE-TIME-FORMAT=MONTH-DAY-YEAR-HOUR-MM-SS

Determines the date and time format. The default value shows the month, day, year, hour, minute, and seconds, using this format: 11/27/03 11:27:03. You can set it to any of these formats: year-month-day-hour-mm-ss, month-day-year-hour-mm-ss, or day-month-year-hour-mm-ss.

DATE-TIME-FORMAT-DATE-DELIMITER=/

Determines the delimiter to use for separating the day, month, and year. The default value is slash, as in 11/27/03.

DATE-TIME-FORMAT-TIME-DELIMITER=:

Determines the delimiter to use for separating the hour, minutes, and seconds. The default value is a colon, as in 11:27:03.

User Interface Refresh

USER-INTERFACE-REFRESH-PERIOD=15

The update frequency for updating message details in any of the message browsers. For example, when the same message arrives on the same domain object, the Repeat Count on the message details now updates to indicate the new message count.

GDSM

Network Connections

NETWORK-CONNECTION-FAULT-CATEGORY=Network Connection

The category of the errors generated in GDSM.

CREATE-MESSAGE-UPON-CONNECTION-SUCCESS=false

If true and upon successful connection to the bridge process, causes an operator message to be generated.

MINIMUM-PERSISTENCE-INTERVAL=5

As rules detect changes, gdsm-handle-bridge-connection waits this amount of time to confirm the status change prior to posting messages. This delay might help avoid actions when states change rapidly.

AUTO-CONNECT-INTERVAL=15

As rules detect changes, gdsm-handle-bridge-connection waits this amount of time after the minimum-persistence-interval to confirm the status change prior to scheduling auto-recovery actions. This delay might help to clear states, for example, sockets in the OS or processes shutting down.

DEFAULT-HTTP-INTERFACE-IS-G2-HTTP-SERVER=true

Whether the default HTTP interface is the G2 HTTP server.

Enable Interfaces

ENABLE-DEFAULT-OPC-INTERFACE=false

Whether to enable the default OPC interface.

ENABLE-DEFAULT-PI-INTERFACE=false

Whether to enable the default PI interface.

ENABLE-DEFAULT-SQL-INTERFACE-POOL=false

Whether to enable the default SQL interface pool.

ENABLE-DEFAULT-SMTP-INTERFACE-POOL=false

Whether to enable the default SQL interface pool.

ENABLE-DEFAULT-HTTP-INTERFACE=true

The default HTTP interface name.

ENABLE-DEFAULT-SNMP-INTERFACE=false

Whether to enable the default SNMP interface.

ENABLE-DEFAULT-SNMP-TRAP-RECEIVER-INTERFACE=false

Whether to enable the default SNMP trap receiver interface.

GEVM

Messages

MESSAGE-REEVALUATION-ENABLED=false

Determines whether the priority of an operator message in the Message Browser escalates automatically over time. If this parameter is set to true, the message priority is reevaluated every period, according to the message-reevaluation-period, and the priority is decreased by one level. If the life time of a message has expired, the message is automatically deleted. By default, message reevaluation is disabled.

MESSAGE-REEVALUATION-PERIOD=3600

The message reevaluation period, in seconds.

MAXIMUM-EVENT-HISTORY-STATES=10

The maximum number of state changes to keep in history for each message. State changes include information about when the message is created, acknowledged, and deleted, changes to the repetition count and priority, and

information on when a message is subsumed by another message, based on message correlation.

Message Color

Optegrity provides several parameters in the configuration file for configuring message colors.

REVERSE-MESSAGE-COLOR-IF-ACKNOWLEDGED=true

Determines whether the message color is reversed when the message has been acknowledged. By default, when a message is acknowledged, the text color changes to become what was the background color. To avoid contrast problems, the background color of acknowledged messages is smoke, by default.

You can customize the background color of acknowledged messages. For more information, see Custom Messaging.

CDG-MESSAGES-USE-STANDARD-COLOR-MAPPING=true

Determines whether messages about SymCure events use the SymCure color mappings or whether they use the standard color mapping for operator message. The default value is true, which uses the standard SymCure color mappings. Set this parameter to false to provide a unified color mapping for operators, based on the message-color-based-on parameter.

By default, SymCure uses red for specified as true, yellow for suspect, and green for specified as false. When the event is inferred to be true or false, the colors are salmon and lime-green, respectively.

For more information about what these values and status values mean, see the *SymCure User's Guide*.

MESSAGE-COLOR-BASED-ON=priority

Sets the mode for determining message color, which is priority, by default. The default color mapping is based on priority, as follows:

1	red
2	orange
3	yellow
4	thistle
5	salmon
6	green
7	wheat
8	sienna
9	tan
10	sky-blue

You can also define message color, based on the colors defined in the process map object containing the target object of the message. To do this, set this property to process-map. For more information, see Creating a Process Map.

You can also define message color, based on message type. This configuration requires knowledge of G2. For more information, see Custom Messaging.

Logbook and Message Board Handlers

REGISTER-LOGBOOK-MESSAGE-HANDLER=false

When true, messages posted to the logbook are rerouted to become GEVM messages (gevm-notification-message) and inserted into the primary message queue labelled Messages.

LOGBOOK-MESSAGES-PRIORITY=8

The initial priority of logbook messages rerouted as GEVM messages.

REGISTER-MESSAGE-BOARD-HANDLER=false

When true, messages posted to the message board are rerouted to become operator messages (gevm-notification-message) and inserted into the primary message queue labelled Messages.

MESSAGE-BOARD-MESSAGES-PRIORITY=9

The initial priority of message board messages rerouted as GEVM messages.

Message Browser

JMAIL-INTERFACE-NAME=none

Specifies the default JMail interface to use for sending e-mail messages.

GEUC

GEUC-DATA-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/g2i/data

The default directory for custom conversions and synonyms for engineering units.

GRLB

GRLB-DELETE-HIDDEN-WORKSPACES-MONITOR-INTERVAL=300

The refresh time, in seconds, for a procedure monitoring the display of workspace views created by the relation browser to detect workspaces that are not visible on any G2 window and can, therefore, be deleted. If these workspaces are not deleted, memory leaks can occur.

_GRLB-NAMES-VISIBLE=false

When true, the relation browser shows a name for every item it displays. This parameter should always be false.

GRLB-DEFAULT-LAYOUT-ALGORITHM=grlb-circular-layout

Controls the layout for the display of relations. By default, the layout is circular, that is, the selected item is shown at the center of a circle and its related items are shown along the circumference of the circle. The radius of the circle is determined by _GRLB-RADIUS. The other option is grlb-default-layout, which shows the related objects above the selected item.

_GRLB-RADIUS=400

When using a circular layout, the radius of the circle.

GRLB-DELETE-PROXY-CLASS-DEFINITIONS-ON-STARTUP=true

By default, proxy class definitions are deleted whenever there is a restart. Set this parameter to *false* to cache proxy class definitions within repositories in the application KB, in which case they are never deleted.

CDG (SymCure)

These parameters control the default behavior of the SymCure diagnosis manager. For a description of these parameters, see Chapter 6, "Configuring a SymCure Application" in the *SymCure User's Guide*.

CDG-TERMINATE-DIAGNOSIS-EARLY=true

CDG-UPSTREAM-LIMIT=1000

CDG-DOWNSTREAM-LIMIT=1000

CDG-DIAGNOSIS-DELETION-INTERVAL=300

CDG-DIAGNOSIS-DELETION-MONITOR-INTERVAL=300

CDG-INCREMENTAL-DIAGNOSIS-MONITOR-INTERVAL=120

CDG-COMPUTE-PRIORITY-PROCEDURE=unspecified

CDG-DEFAULT-TARGET-PRIORITY=1

CDG-UNCHANGED-EVENTS-MONITOR-NAME=cdg-unchanged-events-monitor

CDG-UNCHANGED-EVENTS-FILTER=false suspect unknown

CDG-UNCHANGED-EVENTS-MONITOR-INTERVAL=21600

CDG-AUDIT-INCOMING-EVENT-PROCEDURE=unspecified

CDG-AUDIT-ROOT-CAUSE-PROCEDURE=unspecified

CDG-AUDIT-ALARM-PROCEDURE=unspecified

CDG-AUDIT-DIAGNOSIS-BEFORE-DELETION-PROCEDURE=unspecified

CDG-AUDIT-DIAGNOSIS-STATUS-PROCEDURE=unspecified

CDG-AUDIT-DIAGNOSIS-AFTER-MERGER-PROCEDURE=unspecified

CDG-HORIZONTAL-DISTANCE=300

CDG-VERTICAL-DISTANCE=100

CDG-DISPLAY-ANIMATED-SPECIFIC-FAULT-MODEL=false

CDG-USER-DEFINED-SCHEDULING-PROCEDURE=unspecified

CDG-ENABLE-DEBUGGING=false

CDG-ALLOW-UNSPECIFIED-EVENT-TO-BE-ROOT-CAUSE=false

CDG-SPECIFIC-FAULT-MODEL-ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/archives

CDG-GENERIC-FAULT-MODEL-ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/archives

CDG-ROOT-CAUSE-EPISODES-ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/archives

CDG-EPISODE-DELETION-MONITOR-INTERVAL=86400

CDG-ARCHIVE-GENERIC-FAULT-MODELS-ON-COMPILATION=true

CDG-ENABLE-CHECK-FOR-CHATTERING-EVENTS=true

CDG-LOOKBACK-FOR-CHATTERING=30

CDG-MAX-CHATTERING-REPETITIONS=10

CDG-MESSAGE-SUBSTITUTION-VERB-TAGS=$BECOMES $OCCURS

CDG-LOOKBACK-FOR-CHARTING-ROOT-CAUSE-EPISODES-DISTRIBUTIONS=86400

CDG-INTERVAL-FOR-CHARTING-ROOT-CAUSE-EPISODES-DISTRIBUTIONS=3600

CDGUI (SymCure)

These parameters control the default behavior of the SymCure user interface. For a description of these parameters, see Chapter 6, "Configuring a SymCure Application" in the *SymCure User's Guide*.

RESOURCES-SUBDIRECTORY=resources/symcure

CDG-ALARM-MESSAGE-QUEUE=Alarms

CDG-ROOT-CAUSES-MESSAGE-QUEUE=Root Causes

CDG-TEST-ACTIONS-MESSAGE-QUEUE=Test Actions

CDG-REPAIR-ACTIONS-MESSAGE-QUEUE=Repair Actions

GEVM-GQS-QUEUE Instances

The following sections are used to initialize the configuration of GEVM-GQS-QUEUE instances. The name in bracket needs to match the key of the queue.

Events

These parameters set the default behavior for archiving raw events. Raw events do not appear in any browser, by default.

MAX-ENTRIES-IN-MEMORY=10000

The default maximum number of raw events to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

UPDATE-LATENCY=1.0

How often to update the queue, in seconds.

ARCHIVING-ENABLED=false

The default behavior for logging raw events. Set to true to log raw events. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

ARCHIVING-LOG-TO-FILE-ENABLED=false

Whether logging to a file is enabled, by default.

ARCHIVING-LOG-TO-DATABASE-ENABLED=false

Whether logging to a database is enabled, by default.

ARCHIVING-LOG-TO-JMS-ENABLED=false

Whether logging to a JMS provider is enabled, by default.

ARCHIVING-LOG-CHANGES-ENABLED=true

Whether to log changes, by default.

ARCHIVING-LOG-ADDITIONS-ENABLED=true

Whether to log additions, by default.

ARCHIVING-LOG-REMOVAL-ENABLED=true

Whether to log removals, by default.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving raw events, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log_events_*.csv

The default file name template to use for logging raw events. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=500000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

ARCHIVING-DATABASE-INTERFACE=

The default database interface for logging to a database.

ARCHIVING-DATABASE-TABLE=gevm_events

The default table for logging to a database.

ARCHIVING-LOG-JMS-INTERFACE=

The default jms-interface for logging to a JMS provider.

ARCHIVING-LOG-JMS-AS-XML=false

Whether to log to JMS as XML.

Messages

These parameters set the default behavior for archiving operator messages, which appear in the Message Browser.

MAX-ENTRIES-IN-MEMORY=10000

The default maximum number of operator messages to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

UPDATE-LATENCY=1.0

How often to update the queue, in seconds.

ARCHIVING-ENABLED=false

The default behavior for logging operator messages. Set to true to log operator messages. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

ARCHIVING-LOG-TO-FILE-ENABLED=false

Whether logging to a file is enabled, by default.

ARCHIVING-LOG-TO-DATABASE-ENABLED=false

Whether logging to a database is enabled, by default.

ARCHIVING-LOG-TO-JMS-ENABLED=false

Whether logging to a JMS provider is enabled, by default.

ARCHIVING-LOG-CHANGES-ENABLED=true

Whether to log changes, by default.

ARCHIVING-LOG-ADDITIONS-ENABLED=true

Whether to log additions, by default.

ARCHIVING-LOG-REMOVAL-ENABLED=true

Whether to log removals, by default.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving operator messages, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log-messages-*.csv

The default file name template to use for logging operator messages. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=100000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

Alarms

These parameters set the default behavior for archiving alarms, which appear in the SymCure Alarms Browser.

MAX-ENTRIES-IN-MEMORY=100

The default maximum number of alarms to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving alarms, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log-messages-*.csv

The default file name template to use for logging alarms. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=100000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

ARCHIVING-ENABLED=false

The default behavior for logging alarms. Set to true to log operator messages. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

Root Causes

These parameters set the default behavior for archiving root causes, which appear in the SymCure Root Causes Browser.

MAX-ENTRIES-IN-MEMORY=100

The default maximum number of root causes to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

UPDATE-LATENCY=1.0

How often to update the queue, in seconds.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving root causes, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log-messages-*.csv

The default file name template to use for logging root causes. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=100000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

ARCHIVING-ENABLED=false

The default behavior for logging root causes. Set to true to log operator messages. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

Test Actions

These parameters set the default behavior for archiving test actions, which appear in the SymCure Test Actions browser.

MAX-ENTRIES-IN-MEMORY=100

The default maximum number of test actions to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

UPDATE-LATENCY=1.0

How often to update the queue, in seconds.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving test actions, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log-messages-*.csv

The default file name template to use for logging test actions. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=100000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

ARCHIVING-ENABLED=false

The default behavior for logging test actions. Set to true to log operator messages. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

Repair Actions

These parameters set the default behavior for archiving repair actions, which appear in the SymCure Repair Actions browser.

MAX-ENTRIES-IN-MEMORY=100

The default maximum number of repair actions to keep in memory. When the number of raw events exceeds the maximum, the oldest events are deleted.

UPDATE-LATENCY=1.0

How often to update the queue, in seconds.

ARCHIVING-DIRECTORY=$APPLICATION-ROOT-DIRECTORY/logs

The default directory for archiving repair actions, relative to the *optegrity\kbs* directory.

ARCHIVING-FILENAME-TEMPLATE=log-messages-*.csv

The default file name template to use for logging repair actions. The * is replaced by a key composed of a timestamp corresponding when the log file was created and an index.

ARCHIVING-INTERVAL-TO-OPEN-NEW-LOG-FILE=86400

The default time interval to force the creation of a new log file, in seconds.

ARCHIVING-MAXIMUM-FILE-SIZE=100000

The default maximum size of a log file, in bytes. If the length of the log file exceeds this size, a new log file is created.

ARCHIVING-ENABLED=false

The default behavior for logging repair actions. Set to true to log operator messages. You can configure additional logging behavior in the properties dialog for the queue object. See Logging Messages.

User Interface

The following sections are used to configure user interface components such as menubars, toolbars, status bar and pane windows. For details, see Part III, User Interface Operations in the *G2 Run-Time Library User's Guide*.

Default-Menubar

ENABLED=true

Default-Status-Bar

ENABLED=true

INITIALLY-VISIBLE=true

MINIMUM-HEIGHT=-1

Toolbars

Standard

ENABLED=true

INITIALLY-VISIBLE=true

INITIAL-DOCK=top

INITIAL-DOCK-PRIORITY=0

NEIGHBOR-DOC=left

NEIGHBOR-TOOLBAR=none

ENABLE-TOOLTIPS=true

Layout

enabled=true

INITIALLY-VISIBLE=true

INITIAL-DOCK=top

INITIAL-DOCK-PRIORITY=5

NEIGHBOR-DOC=left

NEIGHBOR-TOOLBAR=web

ENABLE-TOOLTIPS=true

Web

ENABLED=true

INITIALLY-VISIBLE=true

INITIAL-DOCK=top

INITIAL-DOCK-PRIORITY=10

NEIGHBOR-DOC=right

NEIGHBOR-TOOLBAR=layout

ENABLE-TOOLTIPS=true

Child Windows

Project-Hierarchy

ENABLED=true

INITIALLY-VISIBLE=true

WINDOW-PRIORITY=0

STATE=normal

CLOSEABLE=true

MINIMIZEABLE=true

MAXIMIZEABLE=true

RESIZEABLE=true

FLOATABLE=true

AUTOHIDEABLE=true

DRAGGABLE=true

INITIAL-DOCK=left

NEIGHBOR-DOCK=within

NEIGHBOR-WINDOW-NAME=

LEFT=0

TOP=0

WIDTH=250

HEIGHT=350

Class-Hierarchy

ENABLED=true

INITIALLY-VISIBLE=false

WINDOW-PRIORITY=10

STATE=normal

CLOSEABLE=true

MINIMIZEABLE=true

MAXIMIZEABLE=true

RESIZEABLE=true

FLOATABLE=true

AUTOHIDEABLE=true

DRAGGABLE=true

INITIAL-DOCK=left

NEIGHBOR-DOCK=within

NEIGHBOR-WINDOW-NAME=

LEFT=0

TOP=0

WIDTH=240

HEIGHT=350

Module-Hierarchy

ENABLED=true

INITIALLY-VISIBLE=false

WINDOW-PRIORITY=20

STATE=normal

CLOSEABLE=true

MINIMIZEABLE=true

MAXIMIZEABLE=true

RESIZEABLE=true

FLOATABLE=true

AUTOHIDEABLE=true

DRAGGABLE=true

INITIAL-DOCK=left

NEIGHBOR-DOCK=within

NEIGHBOR-WINDOW-NAME=

LEFT=0

TOP=0

WIDTH=240

HEIGHT=350

Toolbox-G2

ENABLED=true

INITIALLY-VISIBLE=false

WINDOW-PRIORITY=35

STATE=normal

LARGE-ICON-SIZE=true

CLOSEABLE=true

MINIMIZEABLE=true

MAXIMIZEABLE=true

RESIZEABLE=true

FLOATABLE=true

AUTOHIDEABLE=true

DRAGGABLE=true

INITIAL-DOCK=left

NEIGHBOR-DOCK=bottom

NEIGHBOR-WINDOW-NAME=project-hierarchy

LEFT=0

TOP=0

WIDTH=240

HEIGHT=350

[html-browser]

ENABLED=true

INITIALLY-VISIBLE=false

WINDOW-PRIORITY=0

STATE=normal

CLOSEABLE=true

MINIMIZEABLE=true

MAXIMIZEABLE=true

RESIZEABLE=true

FLOATABLE=true

AUTOHIDEABLE=true

DRAGGABLE=true

INITIAL-DOCK=left

NEIGHBOR-DOCK=within

NEIGHBOR-WINDOW-NAME=

LEFT=10

TOP=10

WIDTH=650

HEIGHT=450

Intelligent Objects

IOC-INTERNAL-UNIT-SYSTEM=english

F102Demo

These initialization parameters are used for the *f102demo.kb*.

CDG-DIAGNOSIS-DELETION-INTERVAL=10

CDG-DIAGNOSIS-DELETION-MONITOR-INTERVAL=5

Network Interface Connections

The following sections specify the default network interface connections.

default-opc-interface

BRIDGE-HOST-NAME=localhost

BRIDGE-HOST-PORT=22040

BRIDGE-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

default-pi-interface

BRIDGE-HOST-NAME=localhost

BRIDGE-HOST-PORT=22041

BRIDGE-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

default-sql-interface-pool

NETWORK-INITIAL-INTERFACE-COUNT=1

NETWORK-DEFAULT-HOST-NAME=localhost

NETWORK-BASE-PORT-NUMBER=22060

NETWORK-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

USER-NAME=

USER-PASSWORD=

DATABASE-CONNECT-STRING=

DATABASE-MAXIMUM-DEFINABLE-CURSORS=100

DATABASE-BIND-VARIABLE-PREFIX=:

default-smtp-interface-pool

NETWORK-INITIAL-INTERFACE-COUNT=1

NETWORK-DEFAULT-HOST-NAME=localhost

NETWORK-BASE-PORT-NUMBER=22050

NETWORK-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

USER-NAME=

USER-PASSWORD=

INCOMING-EMAIL-HOST=localhost

INCOMING-EMAIL-PROTOCOL=pop3

INCOMING-EMAIL-FOLDER=INBOX

INCOMING-EMAIL-DELETE-MESSAGES-ON-HOST=false

OUTGOING-EMAIL-HOST=localhost

OUTGOING-EMAIL-FROM-ADDRESS=g2@localhost

default-http-interface

BRIDGE-HOST-NAME=localhost

BRIDGE-HOST-PORT=22042

BRIDGE-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=true

LAUNCH-REMOTE-PROCESS=true

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=true

LOGGING-ENABLED=false

ADD-HTTP-REQUEST-ATTRIBUTES-TO-LOG=false

LOG-FILE=$APPLICATION-ROOT-DIRECTORY/logs/g2-http-server-log.txt

HTTP-SERVER-PORT=8085

HTTP-SERVER-SSL-ENABLED=false

HTTP-SERVER-SSL-CERTIFICATE-FILE=

HTTP-SERVER-ROOT-DIRECTORY=$INSTALLATION-DIRECTORY/g2i/data/http_root

default-snmp-interface

BRIDGE-HOST-NAME=localhost

BRIDGE-HOST-PORT=22043

BRIDGE-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

REMOTE-PROCESS-INITIALIZATION-STRING=-p 2 -t 8 -d

default-snmp-trap-receiver-interface

BRIDGE-HOST-NAME=localhost

BRIDGE-HOST-PORT=22044

BRIDGE-CONNECTION-TIMEOUT=15

AUTO-CONNECT-TO-REMOTE-PROCESS=false

LAUNCH-REMOTE-PROCESS=false

SHUTDOWN-REMOTE-PROCESS-UPON-DISCONNECT=false

REMOTE-PROCESS-INITIALIZATION-STRING=-p 1 -v 2 -d

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z