G2 Event and Data Processing

User's Guide

Version 2.3 Rev. 0

G2 Event and Data Processing User’s Guide, Version 2.3 Rev. 0

May 2007

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

Part Number: DOC129-230

Contents

Chapter 2 Block Reference *(continued)*

Chapter 2 Block Reference *(continued)*

Preface

Describes this guide and the conventions that it uses.

About this Guide

G2 Event and Data Processing (GEDP) simplifies the ability to detect events and manage alarms in real time by using a generic, graphical data flow diagramming language.

This guide provides:

- An introduction to the features and benefits of the GEDP graphical language.
- A description of each block in the language.

Audience

This guide is for application developers and system integrators, who are interested in using advanced graphical event detection techniques for their applications.

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
<u>*text-string*</u>	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 <u>*underlined*</u>. 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)
–> <u>*transferred-items*</u>: 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`

Introduction to G2 Event and Data Processing

Provides an overview of the features and benefits of the G2 Event and Data Processing (GEDP) block language.

Introduction

G2 Event and Data Processing (GEDP) is a multi-purpose graphical language. It is composed of graphical blocks that can be connected together to express a flow of data, perform calculations, execute functions, generate messages, and events. These graphically related blocks represent a declarative program or sequential flow of execute where each block represents an activity with a specific behavior. It has a clear starting point, typically the value of a sensor, also called an internal datapoint, associated with a domain object, such as a piece of equipment or a physical sensor. By default, the execution is event driven, that is, a value change of an internal datapoint triggers the execution of GEDP flow diagrams.

You typically place GEDP activity blocks on the details of a diagram folder. You can also place GEDP blocks on the subworkspace of a G2 object and refer to attributes of the object in the diagram. However, we recommend that you place your blocks in a diagram folder to facilitate easy access through built-in menus and tree views.

GEDP diagrams flow from left to right. The left-most blocks are entry points, which provide a source of data for the data flow diagram and perform these functions:

- Subscribe to datapoints in a process map to obtain data values from domain objects.
- Generate continuous data, for example, a real-time clock signal or white noise signal.

You can define generic GEDP templates, which apply to classes of domain objects. A domain object is any subclass of grtl-domain-object. Datapoints on domain objects provide an interface between external datapoints and GEDP diagrams.

Generic templates behave like methods, whereby each generic template is uniquely identified by its name and its target class, just as a method is defined by the method name and class.

Note In this document, we use GEDP diagrams to refer to diagrams for domain objects and GEDP templates to refer to diagrams for domain object classes.

A generic template can get and set values for domain objects. Setting values is useful when the diagram is performing some calculations and an external object, such as a procedure, needs to use this value.

GEDP diagrams also interface with SymCure (CDG), which performs diagnostic reasoning for abnormal condition management. GEDP diagrams can trigger SymCure diagnostics by sending SymCure events for domain objects.

GEDP diagrams can post messages for operators and other low-level notifications that are managed by the G2 Event Manager (GEVM).

GEDP Features

GEDP provides these categories of blocks:

- Data processing.
- Object processing.
- Event and alarm management.
- Domain object event management.
- SymCure fault model diagnostics.

Data Processing

Data processing blocks provide:

- Entry points for providing various types of data to a diagram, including float, integer, boolean, symbolic, and text.
- Signal generators for simulating real-time data, for example, using a real time clock, sine wave, or white noise.
- First-order exponential data filter for filtering real-time data.
- Arithmetic operators for performing various types of mathematical operations on data, including summation, multiplication, quotient, bias, gain, increment, decrement, change sign, inverse, and power.
- Arithmetic functions for performing statistical operations on data, including average, median, low selecting, high selecting, and user-defined functions.
- Data control blocks for controlling how data flows through a diagram, for example, delaying, inhibiting, overriding, and returning data values, as well as applying generic actions to data.
- Time series blocks for performing statistical operations on data histories, such as moving averages, rates of change, and standard deviations.
- Relational operators, which compare data to make inferences, for example, based on whether values are equal to, greater than, or less than a threshold.
- Logical operators, which perform boolean logic on true and false values.
- Path displays for any type of value.

Object Processing

GEDP object processing provides blocks for:

- Associating GEDP diagrams with specific objects.
- Getting named objects and superior objects.
- Getting and setting object properties.
- Getting data histories.

Event and Alarm Management

GEDP provides blocks for managing low-level notifications and operator messages that may be displayed in an application's message browsers. These blocks apply to individual objects as opposed to object classes. GEDP event and alarm management provides blocks for:

- Posting low-level notifications and operator messages.
- Acknowledging and deleting operator messages.
- Fetching notification events.

Domain Object Event Management

Domain object event management provides blocks to generate and manipulate low-level notifications and operator messages specified generically at the class level. GEDP manages events for domain objects by allowing you to define generic templates at the class level. When you initialize an application, GEDP creates a specific diagram for each instance of the specified class.

GEDP domain object event management provides blocks for:

- Associating generic templates with domain object classes.
- Fetching notification events.
- Property change notification entry points.
- Setting object properties.
- Executing GEDP diagrams.
- Sending SymCure events.
- Posting, checking for, and deleting low-level notifications.
- Posting, checking for, deleting, and acknowledging operator messages.

SymCure Fault Model Diagnostics

In addition to managing low-level notifications and operator messages, GEDP blocks for domain object event management can send SymCure events on domain object classes and individual domain objects.

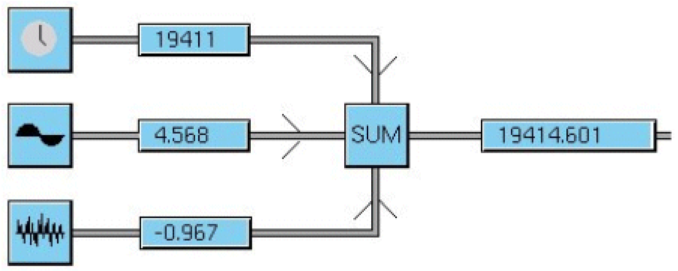
GEDP Diagrams

To create a GEDP diagram, you choose blocks from a palette, place them on the details of a GEDP diagram, and connect them together. Following are some examples of GEDP diagrams.

GEDP diagrams are associated with specific domain objects. You can also create generic GEDP templates that are associated with domain object classes. For more information, see “Generic GEDP Templates” on page 8.

Signal Generators as Entry Points

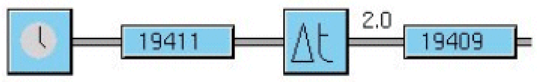
The following figure is an example of a GEDP diagram that contains three signal generators as entry points, and sums their outputs:



You use path displays to show intermediate and final values in the diagram. GEDP paths, or connections, can carry any type of item-or-value, including integers, floats, symbols, texts, objects, classes, rules, and procedures. Path displays adjust to their inputs to display the proper values. When the incoming item-or-value is a named object, the path display shows the name of the object.

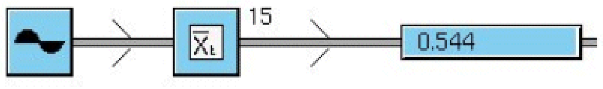
Data Delay

This diagram uses a Data Delay block, which delays data flow by 2 seconds:



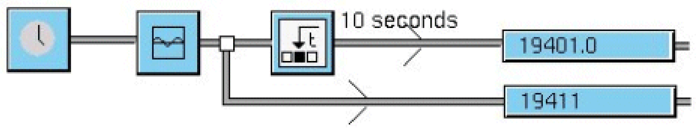
Statistical Analysis

This diagram calculates the moving average of a Sine Wave block over the last 15 seconds:



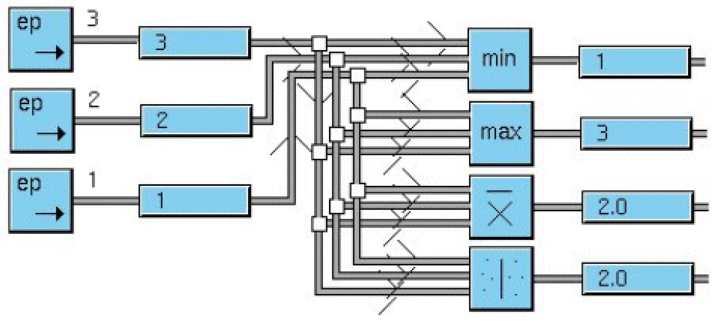
Historical Data

This diagram fetches the history value of a Float Variable block, which is set by a Real Time Clock, as of 10 seconds ago:



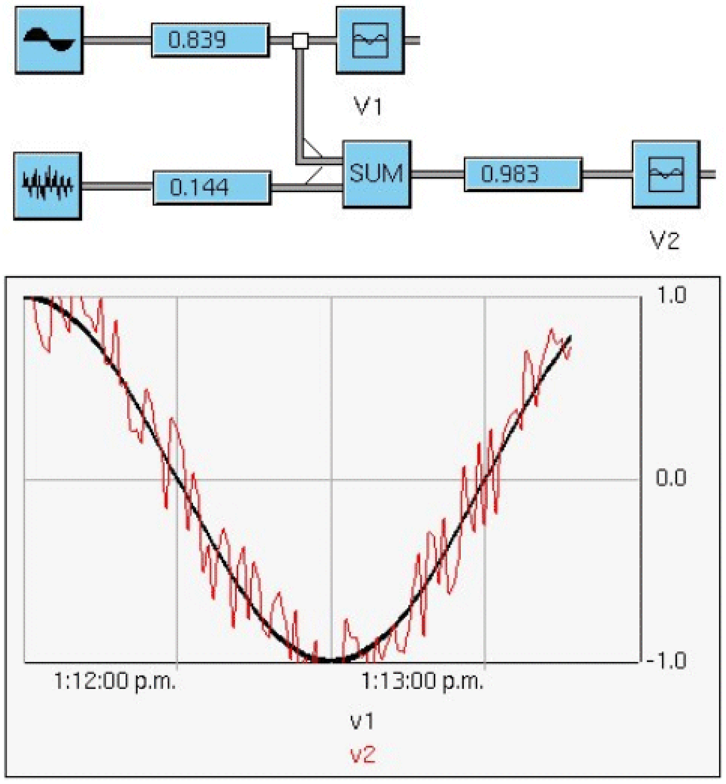
Multiple Downstream Blocks

Connections between blocks can propagate values to more than one downstream block. In this example, each of four blocks, respectively, evaluate the minimum of all entry points, the maximum of all entry points, the average value, and median value:



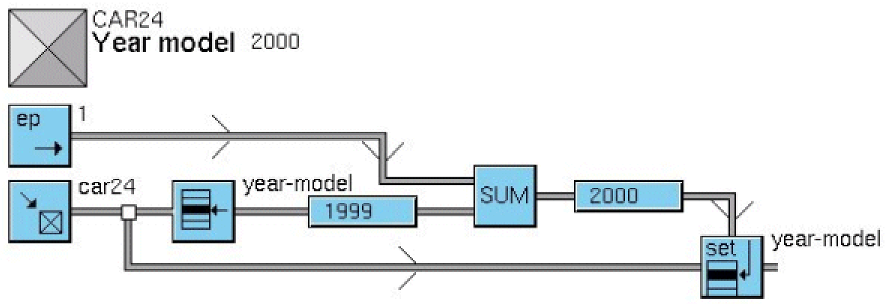
Variables

The Variable block allows you to inject an incoming value into a variable locally, then to use and reference this variable from any G2 element outside of the GEDP diagram. For example, in the following GEDP diagram, the blocks named V1 and V2 are Variable bocks. The trend chart references these variables to display the pure Sine Wave signal and the noisy signal:



Arbitrary G2 Objects

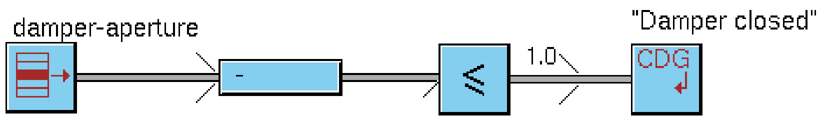
GEDP can access arbitrary G2 objects and their attributes. The following example shows how a GEDP diagram can fetch an object (car-24), then fetch an attribute of that object (year-model), and then add the constant 1 to the value of that attribute:



Domain Objects and SymCure

GEDP can access domain objects to get and set attribute values and to listen for property value changes. GEDP diagrams can send low-level notifications and operator messages, as well as SymCure events for domain objects. GEDP diagrams can also listen for low-level notifications. SymCure performs diagnostic reasoning about its events to determine root causes and to take corrective actions.

This example gets the value of the damper-aperture of the associated domain object and generates a SymCure Damper Closed event for the object:



Generic GEDP Templates

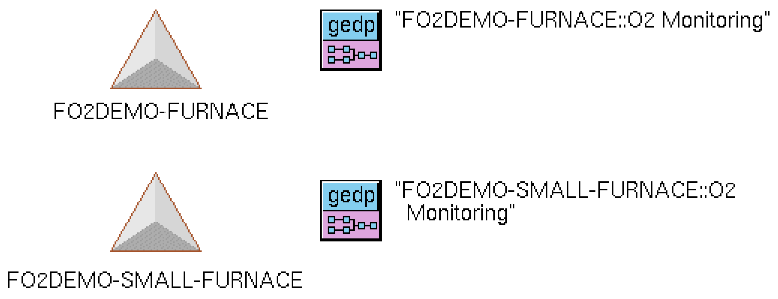
GEDP allows you to define generic templates, which are uniquely identified by their name and target class, similar to a method in traditional object-oriented programming. This feature enables you to define an object-oriented hierarchy of target classes, where each target class can define a generic template with the same name. When GEDP creates a specific diagram for a domain object, it uses method inheritance to determine which generic template to use.

For example, suppose you have a class hierarchy in which furnace is the superior class of small-furnace. You can define separate generic templates for each class, where each template has the same name, for example, O2 Monitoring. Each generic template is identified by its name and target class: furnace::O2 Monitoring and small-furnace::O2 Monitoring. When GEDP creates the specific diagram, it chooses the generic template whose target class most closely matches that of the domain object. Thus, if the domain object is a furnace, it would use the furnace::O2 Monitoring template, whereas if the domain object is a small-furnace, it would use the small-furnace::O2 Monitoring template.

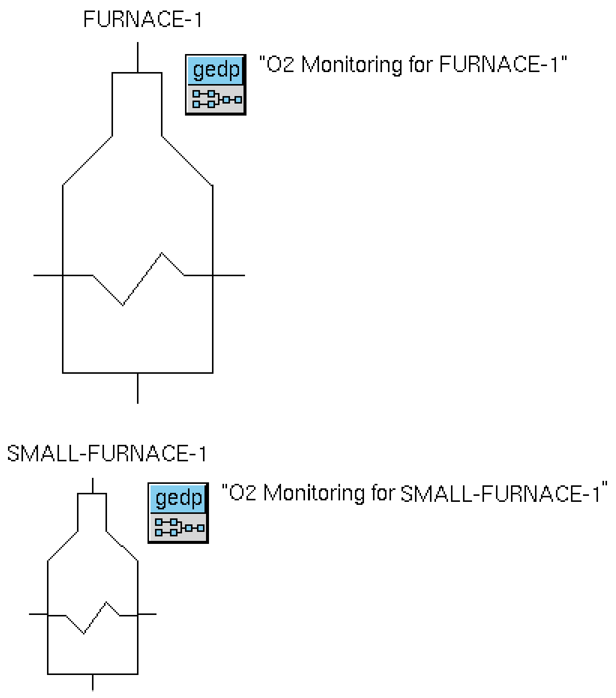
Note Generic templates do not support the call next method capability, where the diagram template for a subclass can invoke the template for its immediate parent class.

In the case of multiple inheritance, if multiple parent classes of a child class define a diagram template with the same name, consistent behavior is guaranteed only if the child class defines its own template.

This figure shows the domain object class definitions and their associated generic templates. Notice that the diagram templates have the same names, O2 Monitoring, but they are uniquely identified by their target classes, fo2demo-furnace and fo2demo-small-furnace:


FO2DEMO-FURNACE
gedp
"FO2DEMO-FURNACE::O2 Monitoring"
FO2DEMO-SMALL-FURNACE
gedp
"FO2DEMO-SMALL-FURNACE::O2 Monitoring"

This figure shows instances of each class, furnace-1 and small-furnace-1, each of which is associated with its own specific diagram, based on method inheritance:


FURNACE-1
gedp
"O2 Monitoring for FURNACE-1"
SMALL-FURNACE-1
gedp
"O2 Monitoring for SMALL-FURNACE-1"

GEDP Diagram Execution

GEDP diagram execution requires that diagrams be initialized. In addition, when defining generic templates for domain objects in a process map, you must also initialize the process map to create specific diagrams for each domain object.

Once diagrams and process maps are initialized, you have two choices for diagram execution:

- Using event-driven evaluation.
- Using scheduled evaluation.

Initializing GEDP Diagrams and Process Maps

Before a GEDP diagram can evaluate, the blocks in the diagram must be initialized. Initialization performs the following functions:

- Activates the block so it is ready to evaluate.
- Resets the current value of the block.

Typically, you create a process map that consists of a set of connected domain objects, and you create generic templates for those domain object classes.

Thus, before GEDP diagrams can execute, each domain object in a process map must also be initialized. By default, initializing process maps creates specific diagrams for each domain object in the map. It also initializes all the blocks in each specific diagram.

You might want to configure event limits in specific diagrams that persist through application initialization. You can accomplish this by configuring an option in the generic event detection diagram, in which case, initialization does not recreate specific diagrams each time.

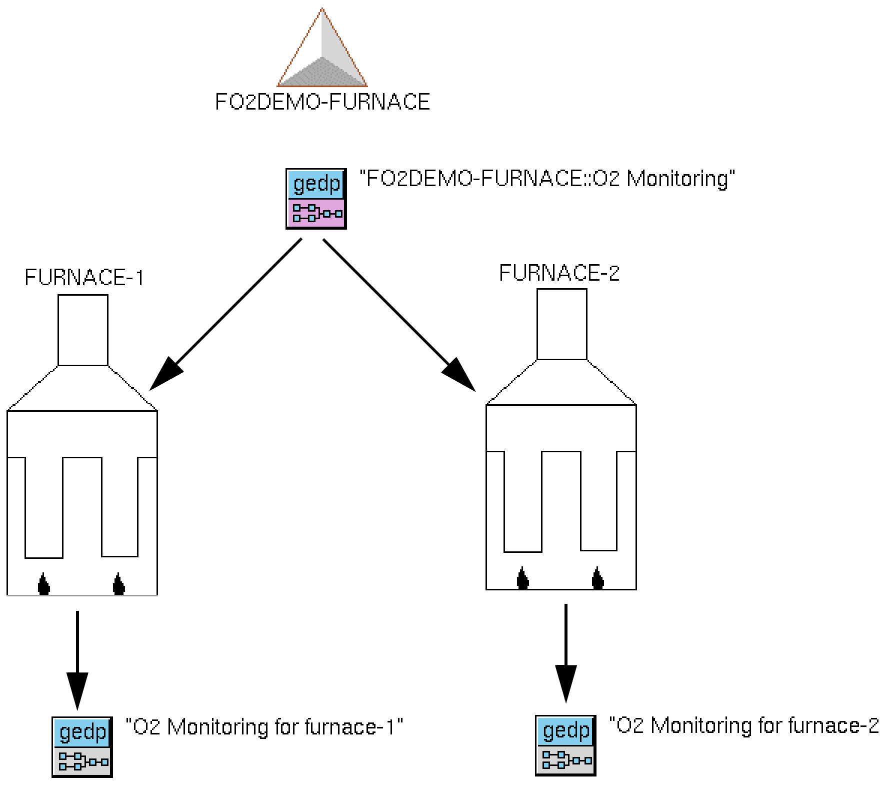
You initialize blocks and process maps through the GRTL menu bar.

To initialize GEDP diagrams and process maps:

➔ Choose Project > Initialize Application from the menu bar.

GEDP diagrams can now begin listening for events and processing data.

This diagram shows the results of initializing a process map that contains two instances of the fo2demo-furnace class, furnace-1 and furnace-2, for which a generic template named O2 Monitoring has been defined. Each instance defines a specific diagram, whose name reflects the object name.


FO2DEMO-FURNACE
gedp
"FO2DEMO-FURNACE::O2 Monitoring"
FURNACE-1
FURNACE-2
gedp
"O2 Monitoring for furnace-1"
gedp
"O2 Monitoring for furnace-2

Using Event-Driven Evaluation

When the blocks in a specific diagram have been initialized, by default, evaluation is event-driven, based on the entry point block in the diagram. Entry points can receive values from G2 variables, or they can receive event notifications from G2 objects, including domain objects. When the entry point block is evaluated, the output value is propagated downstream in the diagram, which causes the evaluation of each successive block in the diagram.

By default, entry points that pass data values, as opposed to events, are evaluated each time a value changes in the data source. You can also configure entry points to be evaluated manually. Entry points that listen for low-level notifications are evaluated whenever the event is sent.

You can disable and enable evaluation of individual blocks in a diagram to stop and start data flow, which is useful for debugging the execution of a diagram. You can also initialize, reset, and evaluate individual blocks in a diagram.

GEDP provides entry points that pass data values, as well as entry points that listen for events.

For information on entry points that...	See...
Pass data values	"Entry Points" on page 40.
Listen for low-level notifications	"Events and Alarm Management" on page 46.
Listen for property change events	"Generic Template Blocks" on page 68.

Using Scheduled Evaluation

GEDP diagrams can execute on a schedule, such as once every 30 seconds. They can also execute, activate, and/or deactivate, based on the value of an internal datapoint in a process map. For example, you can trigger the evaluation of a diagram that monitors temperature when the value of the associated datapoint reaches a certain level and deactivate the diagram when it goes below a certain level, thereby avoiding unnecessary evaluation.

You configure diagrams to execute on a schedule by configuring the properties of a generic template or specific diagram. For more information, see "Event Detection Diagrams" on page 52.

Accessing GEDP Blocks

To access GEDP blocks, you must load or merge in `gedp.kb`, which is located in the `g2i\kbs` directory.

The
`gedp-demo.kb` is located in the `g2i\examples` directory. On Windows, you can launch the demo from the Start menu.

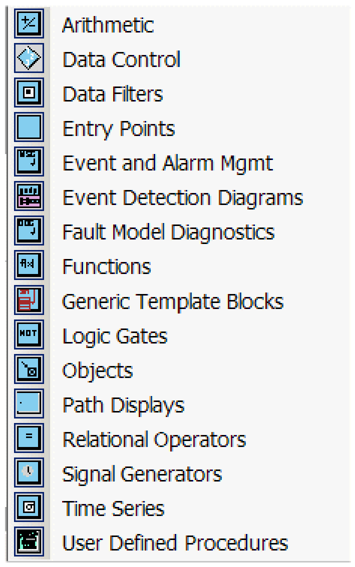
GEDP is available as part of the G2 bundle. You access GEDP blocks from the Event Detection toolbox.

Certain palettes in the Event Detection toolbox are only available when the intelligent object libraries are loaded.

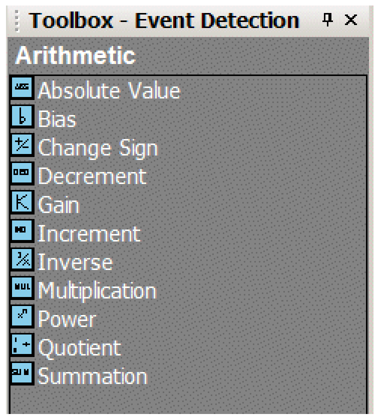
To access GEDP blocks:

➔ Choose View > Toolbox - Event Detection.

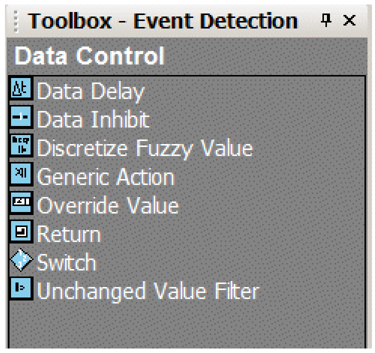
Here are the available GEDP palettes:



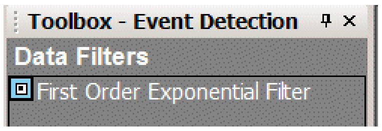
Here are the basic GEDP palettes in the order in which they appear in the toolbox:


Toolbox - Event Detection

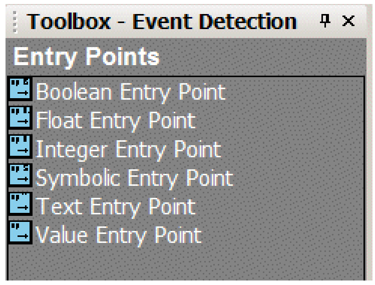
Arithmetic
- Absolute Value
- Bias
- Change Sign
- Decrement
- Gain
- Increment
- Inverse
- Multiplication
- Power
- Quotient
- Summation


Toolbox - Event Detection

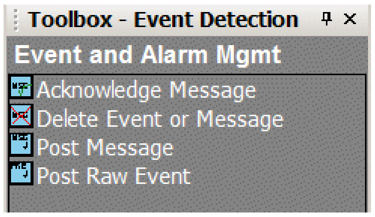
Data Control
- Data Delay
- Data Inhibit
- Discretize Fuzzy Value
- Generic Action
- Override Value
- Return
- Switch
- Unchanged Value Filter


Toolbox - Event Detection

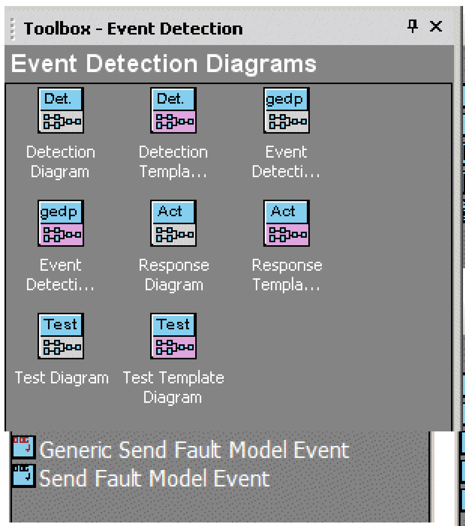
Data Filters
- First Order Exponential Filter


Toolbox - Event Detection

Entry Points
- Boolean Entry Point
- Float Entry Point
- Integer Entry Point
- Symbolic Entry Point
- Text Entry Point
- Value Entry Point

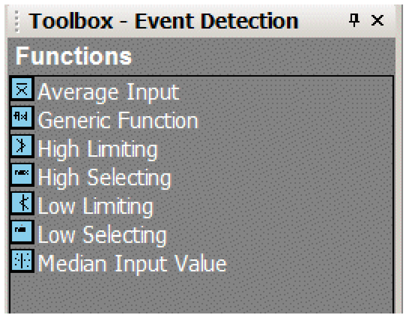

Toolbox - Event Detection

Event and Alarm Mgmt
- Acknowledge Message
- Delete Event or Message
- Post Message
- Post Raw Event

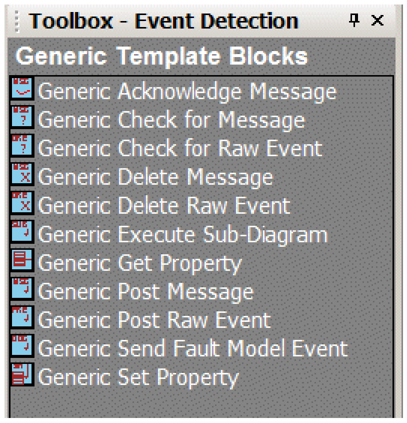

Toolbox - Event Detection

Event Detection Diagrams
- Detection Diagram
- Detection Templa...
- Event Detecti...
- Event Detecti...
- Response Diagram
- Response Templa...
- Test Diagram
- Test Template Diagram

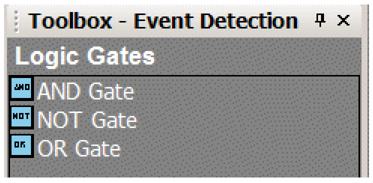
- Generic Send Fault Model Event
- Send Fault Model Event


Toolbox - Event Detection

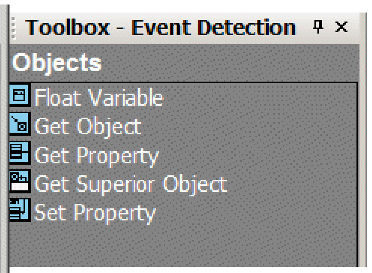
Functions
- Average Input
- Generic Function
- High Limiting
- High Selecting
- Low Limiting
- Low Selecting
- Median Input Value


Toolbox - Event Detection

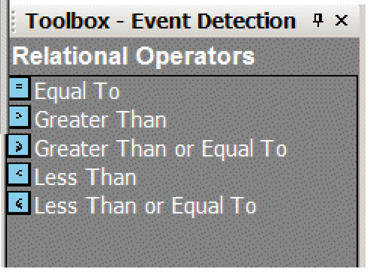
Generic Template Blocks
- Generic Acknowledge Message
- Generic Check for Message
- Generic Check for Raw Event
- Generic Delete Message
- Generic Delete Raw Event
- Generic Execute Sub-Diagram
- Generic Get Property
- Generic Post Message
- Generic Post Raw Event
- Generic Send Fault Model Event
- Generic Set Property


Toolbox - Event Detection

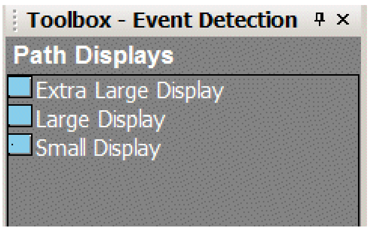
Logic Gates
- AND Gate
- NOT Gate
- OR Gate


Toolbox - Event Detection

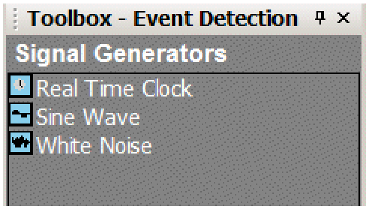
Objects
- Float Variable
- Get Object
- Get Property
- Get Superior Object
- Set Property


Toolbox - Event Detection

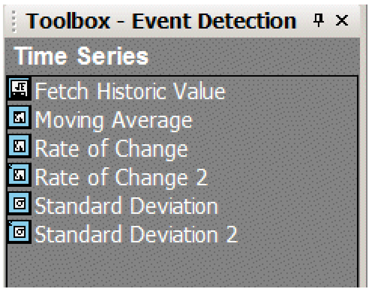
Relational Operators
- Equal To
- Greater Than
- Greater Than or Equal To
- Less Than
- Less Than or Equal To


Toolbox - Event Detection

Path Displays
- Extra Large Display
- Large Display
- Small Display


Toolbox - Event Detection

Signal Generators
- Real Time Clock
- Sine Wave
- White Noise


Toolbox - Event Detection

Time Series
- Fetch Historic Value
- Moving Average
- Rate of Change
- Rate of Change 2
- Standard Deviation
- Standard Deviation 2

For information on the blocks on each palette, see Chapter 2, “Block Reference” on page 17.

Block Reference

Provides a reference for the GEDP blocks.

Introduction

This chapter describes the behavior and specific properties of each GEDP block. Each individual block or each group of blocks defines one or more examples of how to use the block(s) in a diagram. Properties and menu choices that are common to all blocks are described separately.

GEDP blocks are organized by the palette on which they appear. The palettes are organized alphabetically.

Some blocks can appear in specific diagrams only, whereas other blocks can appear in generic templates only. Most blocks can appear in either specific diagrams or generic templates. This table describes which blocks you can use in each type of diagram:

Specific Diagrams or Generic Templates	Generic Templates Only	Specific Diagrams Only
Arithmetic	Generic Template Blocks	Events and Alarm Management
Data Control	Fault Model Diagnostics	Fault Model Diagnostics
Data Filters		Objects
Entry Points		
Functions		
Logic Gates		
Path Displays		
Relational Operators		
Signal Generators		
Time Series		

When running GEDP in a stand-alone environment, you create these objects from a palette.

Arithmetic

Absolute Value
Bias
Change Sign
Decrement
Gain
Increment
Inverse
Multiplication
Power
Quotient
Summation

Data Control

Data Delay
Data Inhibit
Data Switch
Discretize Fuzzy Value
Override
Return
Generic Action

Data Filters

First Order Exponential Filter

Entry Points

Value Entry Point
Float Entry Point
Integer Entry Point
Boolean Entry Point
Symbolic Entry Point
Text Entry Point

Events and Alarm Management

Post Raw Event
Post Message
Delete Event or Message
Acknowledge Message

Event Detection Diagrams

Specific Event Detection Diagram
Generic Event Detection Template

Fault Model Diagnostics

Generic Send Fault Model Action Result
Generic Send Fault Model Event
Send Fault Model Event

Functions

Generic Function
Average Input
Median Input
Low Selecting
High Selecting

Generic Template Blocks

Generic Fetch Property
Set Property
Generic Send Fault Model Event
Generic Send Fault Model Action Result
Generic Post Raw Event
Generic Post Message
Generic Execute Subdiagram
Generic Check for Raw Event
Generic Delete Raw Event

Logic Gates

And Gate
Or Gate
Not Gate

Objects

Fetch Object
Fetch Superior Object
Fetch Property
Set Property
Float Variable

Path Displays

Small Path Display, Large Path Display, Extra-Large Path Display

Relational Operators

Less Than
Less Than Or Equal To
Greater Than
Greater Than Or Equal To
Equal To

Signal Generators

Real Time Clock
Sine Wave
White Noise

Time Series

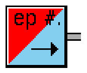
Rate of Change
Standard Deviation
Moving Average
Fetch History Value
Rate of Change 2
Standard Deviation 2

User Defined Procedures

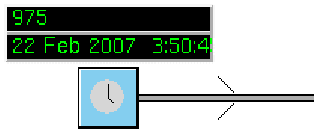
Generic Action Block Method
Generic Action Block Procedure

Common Menu Choices

All GEDP blocks define the following GEDP menu choices:

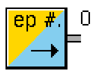
Menu Choice	Description
Properties	Displays the properties dialog for configuring the properties of the block.
Initialize	Validates, resets, and activates the individual block. A block can be invalid if it does not have data on one or more of its input paths. The block is not necessarily configured incorrectly if it is invalid. Invalid blocks look similar to this: 
Reset	Resets the block to its initial value.
Evaluate	Manually evaluates the block. This menu choice has an effect only when the block has been activated.

The paths between blocks define the following GEDP menu choice:

Menu Choice	Description
Create Output Display	Creates an output display that shows the current value and the current timestamp of the output value of the upstream block, and places it above the upstream block. For example: 

Common Properties

All GEDP blocks define the following properties:

Property	Description
Block Label	The label associated with the block, which appears next to the block in the Project hierarchy. By default, the label is system-defined. You can configure it to be any label.
Activated	Whether the block is currently activated. Before a block can evaluate and pass its data downstream, it must be activated. When you initialize a GEDP diagram, all the blocks in the diagram are automatically activated; thus, the Activated toggle button is enabled. You can manually deactivate individual blocks by clicking the Activated toggle button. When a block is not activated, it looks similar to this:  For information about initializing GEDP diagrams, see “Initializing GEDP Diagrams and Process Maps” on page 10.
Description	A textual description of the block.

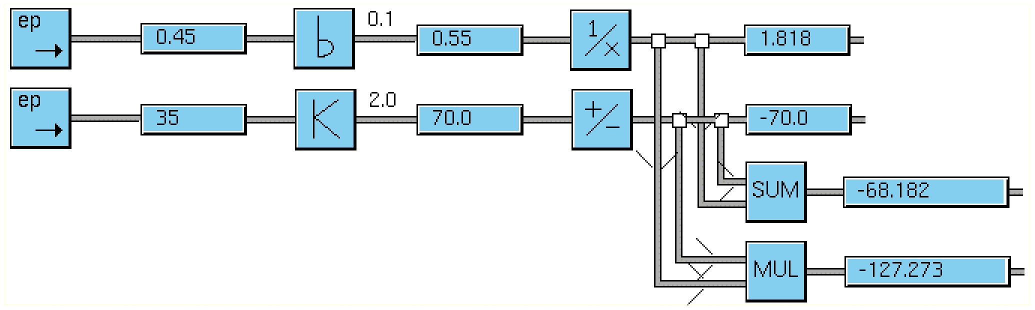
Arithmetic

These blocks perform arithmetic operations on data values. The descriptions of the mathematical formulas use the following notation, where *o* is the output value and *i* is the input value: *o* = f(*i*)

You can use these blocks in a specific diagram or in a generic template. For more information, see "Event Detection Diagrams" on page 52.

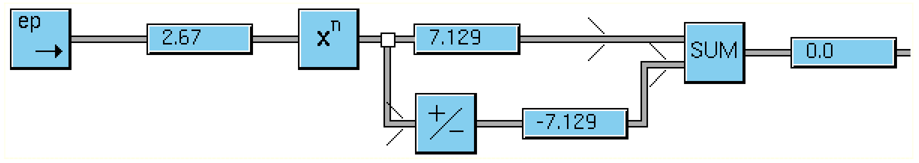
Arithmetic Blocks: Example 1

This example feeds two numeric values into the model, adds a bias value to the top number, and multiplies the lower number by a gain. You specify the bias and gain through the properties dialog. It then takes the inverse of the top number and changes the sign of the bottom number. Finally, it sums and multiples the two numbers and displays the result.



Arithmetic Blocks: Example 2

This example feeds a numeric value into the model and raises the number to a power. You configure the power through the properties dialog. It then changes the sign of the result and adds it to the computed number. The resulting value is always 0.



Absolute Value

Takes the absolute value of the input value:

$$o = |i|$$

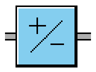
Bias

Adds the specified bias to the input value:

$$o = i + bias$$

Property	Description
Bias	The value to add to the input value.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Bias, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map, which you can choose from a dropdown list. The property name can also use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

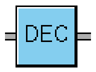
Change Sign



Changes the sign of the input value:

$$o = -i$$

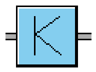
Decrement



Decrements the input value by one:

$$o = i - 1$$

Gain



Multiplies the input value by the specified gain factor:

$$o = i \times gain$$

Property	**Description**
Gain	The value by which to multiply the input value.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Gain, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map, which you can choose from a dropdown list. The property name can also use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

Increment

Increments the input value by one:

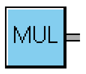
$$o = i + 1$$

Inverse

Calculates the inverse of the input value. The block passes the input value when the input value equals zero:

$$o = 1/i$$

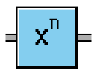
Multiplication



Calculates the product of all input values. The block accepts any number of inputs.

$$o = [i_1 \times i_2]\ldots$$

Power

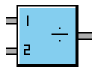


Calculates the input value raised to the specified power:

$$o = i^{power}$$

Property	Description
Power	The power to which to raise the input value.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Power, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

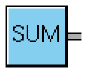
Quotient



Divides the input value on port 1 by the input value on port 2. The block does nothing if the input on port 2 equals zero.

$$o = i_1/i_2$$

Summation



Calculates the sum of all input values. The block accepts any number of inputs.

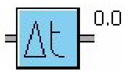
$$o = i_1 + i_2\ldots$$

Data Control

These blocks control how data flows through a GEDP diagram.

You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

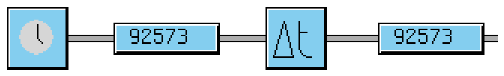
Data Delay



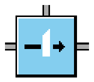
Delays propagation of the input by the specified number of seconds. The input can be any item or value.

Property	Description
Delay (seconds)	The number of seconds to delay before passing the input item or value.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Delay, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

This example shows how to delay a signal:



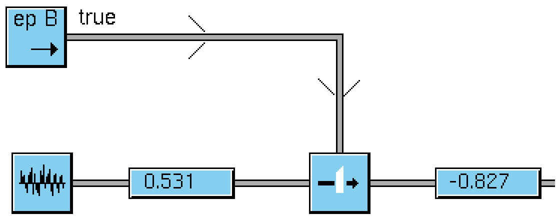
Data Inhibit



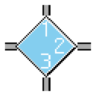
The Data Inhibit block lets a boolean path determine when to pass data on the output path. The block passes a data value on the horizontal input path and a boolean value on the top input path.

When the boolean value on the top input path is true, the block inhibits the input data value. When the boolean value is false, the block passes the data. The block evaluates whenever the data value or the boolean value changes.

This example shows how to inhibit a signal. Because the top input path is true, the signal is inhibited.



Data Switch

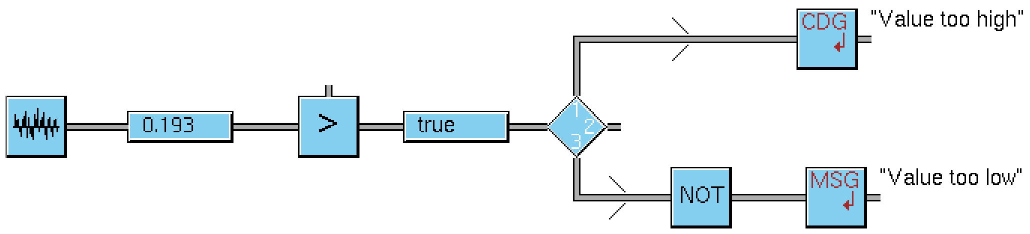


The Data Switch tests the input value to determine the path on which to send the value. This block is useful when you need to perform different actions depending on some condition. The block can pass numbers, symbols, text, and truth-values.

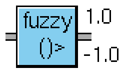
The block has one input path, which is unlabeled, and three output paths, which are labeled. You configure the data type of the input value, and the mathematical operator and value to be used as a test for each output path.

Property	Description
Value Type	The data type of the input value on the input path. The options are: quantity, symbol, text, and boolean.
Operator	The mathematical operator to apply to the input value on each input path. The options are: >, >=, <, <=, =, and unused. A value of unused means the input value is not passed.
Value	The value to be compared against the input value.

This example shows how to switch the output path, based on a test. The Data Switch block is configured to accepts boolean values. Output path 1 is configured to pass the input value when it equals true, and Output path 3 is configured to pass the input value when it equals false. When the value of the input signal is greater than the specified threshold, the observation passes a value of true to the Data Switch, which sends the “Value is too high” event to SymCure. When the value of the data signal is less than the threshold, the observation passes a value of false to the switch, which the Not Gate converts to true. When the converted value is true—the input signal is below the threshold—the Post Message block sends the “Value is too low” message.



Discretize Fuzzy Value



The Discretize Fuzzy Value block converts a fuzzy truth value into a discrete truth value, based on a maximum and minimum value, as follows:

if *i* >= *max* then *o* = true

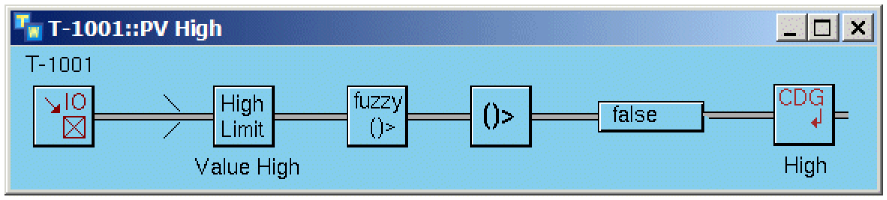
if *i* =< *min* then *o* = false

if *min* < *i* < *max* then *o* remains unchanged

By default, the block passes a value of true if the input fuzzy truth-value is equal to 1 and false if the input fuzzy truth-value is less than -1.

Property	Description
Max Threshold	The maximum threshold for passing a value of true.
Min Threshold	The minimum threshold for passing a value of false.

The Discretize Fuzzy Value block is used in the built-in event-detection diagrams for domain objects, for example, the PV High event of a sensor:

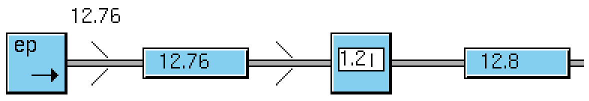


Override

Displays a dialog prompting the user for a manual override of the input value. The input can be any value. The overridden value is propagated downstream when the user clicks OK or Apply.

Property	Description
New Value	The value to output when the user clicks OK or Apply.

This example shows how to override a data value. The Entry Point specifies a value of 12.76, which triggers the Override block to display the dialog. The user specifies an override value of 12.8, which is the value that gets passed downstream.



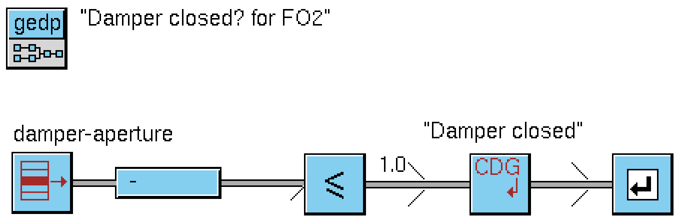
Return

Returns a value to the superior GEDP diagram, if it exists. The Return block is also useful when you want to evaluate a GEDP diagram and access the result. To do this, you can use this block in conjunction with the grtl-get-value API procedure, which extracts the current value of the diagram and returns it to the calling object. If the diagram continually evaluates as entry point values change, the API returns the last value set by the Return block.

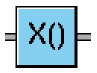
The Return block must be connected at the output of a block producing a value. In general, we recommend that you use only one Return block per diagram.

For information on how to associate generic templates with domain object classes, see "Event Detection Diagrams" on page 52.

For example, you can use a GEDP diagram as a test action when performing SymCure diagnostics. In the following example, the Property Change Entry Point block tests the value of the damper-aperture of the fo2 object associated with the GEDP diagram. If the value is less than 1.0, indicating the damper is closed, the Send Fault Model Event block sends a SymCure event to the domain object. If the event is true, the Return block then sends a value of true as the return value for the test; otherwise, it sends a value of false.



Generic Action



The Generic Action block lets you use a procedure, method, or function as a block in a diagram. The block applies the generic action to its input value and passes the return value of the action as its output value. You must specify the name of the procedure, method, or function in the User-Defined Procedure property of the block. The input value and output value for this block can be any item or value.

The signature of the generic action is:

> my-generic-action
>   (*block*: class gedp-generic-action-block,
>   *input-value*: item-or-value, *domain-object*: item-or-value)
>   –> *return*: item-or-value

where:

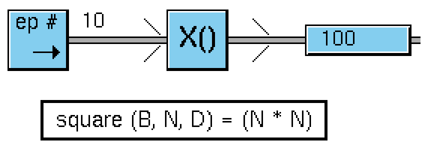
> *block* is the Generic Action block. You can subclass the block and refer to user-defined attributes of the block in the function, procedure, or method.
>
> *input-value* is the item or value on the input path to the block.
>
> *domain-object* is the domain object that is assigned to the GEPD diagram or if no domain object has been assigned, the symbol none.
>
> The generic action must return an item-or-value.

The generic action can be a user-defined procedure, method, or function that takes three arguments. You can create a user-defined procedure or method from the User Defined Procedures palette. For details, see “User Defined Procedures” on page 109.

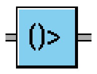
For information on how to create procedures and methods, see the *G2 Reference Manual*.

Property	**Description**
User Defined Procedure	The name of the function or procedure to execute. You can choose the user-defined procedure from a dropdown list of available procedures.

This example shows how to execute a user-defined function on an input data value.



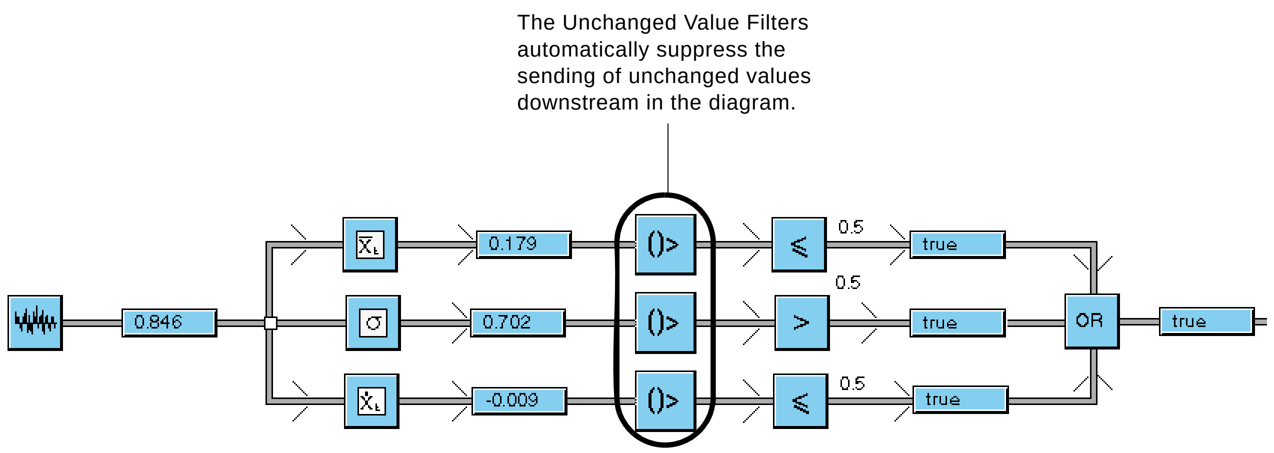
Unchanged Value Filter



By default, when GEDP evaluates a relational operator block in a diagram, it sends the truth value downstream, even if the truth value has not changed. Sometimes, this is not the desired behavior. For example, suppose the block tests whether an input value goes above 10. When the input value is 5, the block passes a truth value of false. Now suppose the input value becomes 2. By default, the block passes a truth value of false downstream, even though the truth value has not changed. This might or might not be the desired behavior; sometimes, you need to know the result of a new test, even if the value is the same, and sometimes you don’t.

The Unchanged Value Filter block allows you to prevent values that have not changed from passing downstream. To use this feature, simply insert this block after any type of block whose unchanged values you want to filter.

This example uses Unchanged Value Filter blocks to suppress sending unchanged values downstream:


The Unchanged Value Filters automatically suppress the sending of unchanged values downstream in the diagram.
0.846
0.179
0.702
-0.009
()>
0.5
true
OR
true

Data Filters

This block appears after a data entry block to filter out noise and find data trends.

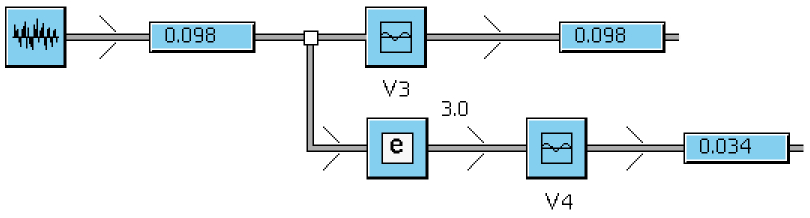
You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

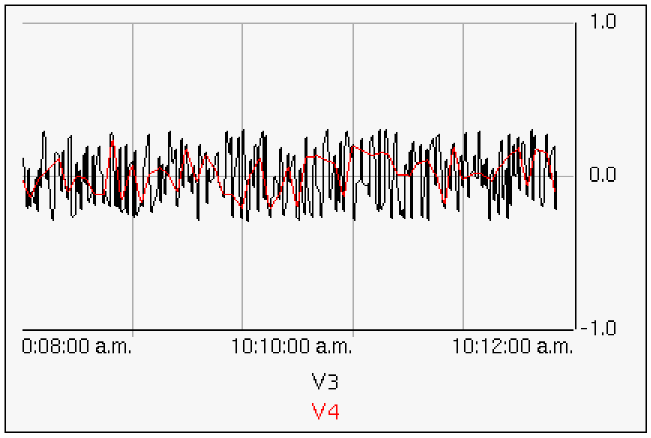
First Order Exponential Filter

Performs low-pass filtering on the input value to filter out high-frequency noise. The output depends on the previous output value and the current input value. You can change the Filter Constant value to specify how much the filter weights the previous value versus the current value. The larger the filter constant, the more weight the previous output values have. The filter does not require the values to be equally spaced in time.

Property	Description
Filter Constant	The relative weight of the previous value versus the current value when determining which values to filter out.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Filter Constant, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

This example shows the result of filtering high-frequency noise from the signal generated by a White Noise block and graphing the result against the original signal. The example uses Float Variable blocks to keep track of the histories, where V3 is the original signal and V4 is the filtered data.





Entry Points

Entry points act as a source for data in a GEDP diagram. They provide float, integer, boolean, text, and symbolic values to a GEDP diagram. You can specify a value for the entry point manually, in which case you must evaluate the block manually to propagate the data downstream. You can also specify a GRTL source datapoint from which the entry point obtains its data. By default, the datapoint source value is propagated downstream automatically when the value changes.

All entry points share the following properties:

Property	Description
Source Datapoint Name	A text string that refers to the name of a grtl-datapoint from which the entry point obtains its values. The Entry Point can refer to an internal datapoint of a domain object as the source datapoint, using dot notation. See “Entry Points: Example 4” on page 43.
Evaluate on Source Datapoint Change	When Source Datapoint Name is specified, whether to evaluate the datapoint automatically when the datapoint value changes. Set this option to false to evaluate the block manually, or when performing scheduled or datapoint-driven evaluations of the GEDP diagram. For more information, see “Specific Event Detection Diagram” on page 53.

You can also subclass entry point blocks to handle any custom datapoints that you might create.

You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

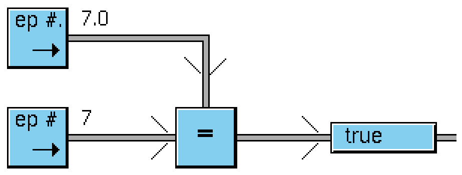
See also:

- “Fault Model Diagnostics” on page 60.
- “Events and Alarm Management” on page 46.
- “Generic Template Blocks” on page 68.

For information about creating grtl-datapoint objects, see the *Optegrity User’s Guide*.

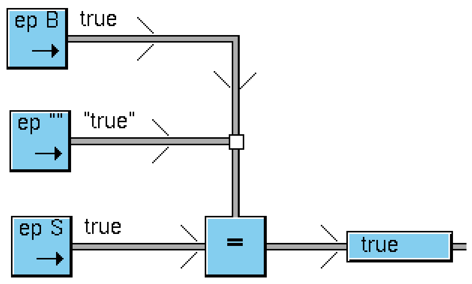
Entry Points: Example 1

This example shows how to provide float and integer values to a diagram. The diagram shows that a float value of 7.0 and an integer value of 7 are equal.



Entry Points: Example 2

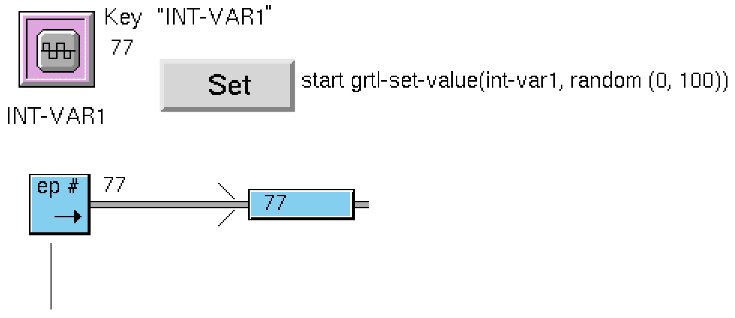
This example shows how to provide boolean, text, and symbolic values to a GEDP diagram. The diagram shows that a boolean value of true, a text value of "true", and a symbolic value of true are equal.



Entry Points: Example 3

You can use entry points to implement a GRTL listener interface to subscribe to and get external data. To do this, you configure the Source Datapoint Name property of the entry point to refer to a GRTL datapoint, which is a subclass of grtl-datapoint.

This example shows how to configure an Integer Entry Point to obtain its values from a GRTL datapoint named int-var1. This variable is an instance of a grtl-simple-integer-datapoint. The variable name is int-var1. The entry point specifies the Source Datapoint Name property as int-var1, which causes it to get its values from the variable. The action button calls the grtl-set-value API procedure to set the value of the variable to a random number between 0 and 100.



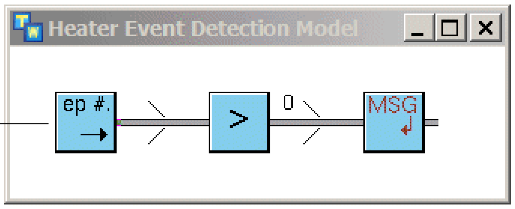
Source Datapoint Name is "int-var1".

Entry Points: Example 4

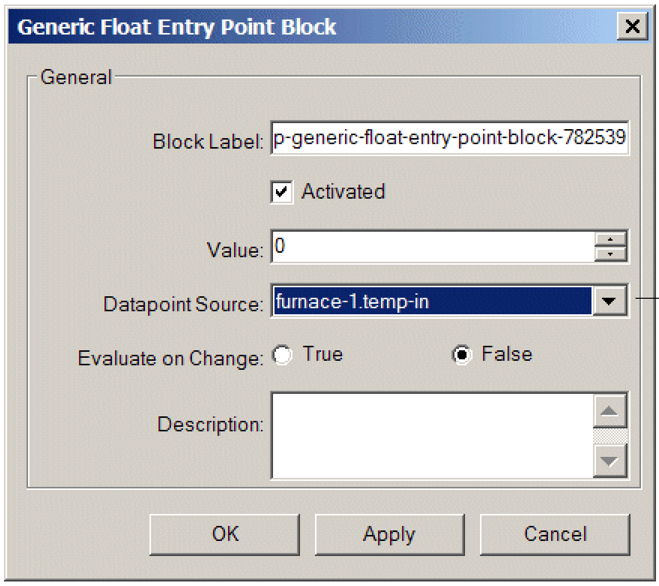
The Entry Point can refer to an internal datapoint of a domain object as the source datapoint. To refer to the internal datapoint of a domain object, the Source Datapoint Name must use notation, for example, fo2.temp-in. You can specify any level deep of embedded objects, using dot notation, for example, fo2.o2-sensor.pv.

When configuring the Source Datapoint Name of an Entry Point on the details of a specific diagram, you can click the Select button and choose from a list of available datapoints of the specific object, using dot notation. For example, here is a specific diagram with a Float Entry Point on its details. The properties dialog for the entry point specifies furnace-1.temp-in as the Source Datapoint Name, which appears in the list when you click the Select button.

Entry Point



Specific diagram for my-furnace instance.



Block Label: p-generic-float-entry-point-block-782539

[✓] Activated

Value: 0

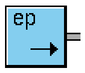
Datapoint Source: furnace-1.temp-in

Evaluate on Change: True [✓ False]

Description:

Source Datapoint Name refers to an internal datapoint of the associated domain object, using dot notation.

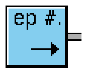
Value Entry Point



Provides values of arbitrary type to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	**Description**
Value	The value to propagate downstream, which can be a float, integer, boolean, text, or symbol.

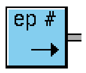
Float Entry Point



Provides floating point values to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	**Description**
Value	The floating point value to propagate downstream.

Integer Entry Point



Provides integer values to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	**Description**
Value	The integer value to propagate downstream.

Boolean Entry Point

Provides boolean (true or false) values to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	Description
Value	The boolean value to propagate downstream.

Symbolic Entry Point

Provides symbolic values to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	Description
Value	The symbolic value to propagate downstream.

Text Entry Point

Provides text values to the diagram. For a description of the common properties of entry points, see the introduction to “Entry Points” on page 40.

Property	Description
Value	The text value to propagate downstream.

Events and Alarm Management

These blocks generate, manage, and listen for low-level notifications and operator messages related to individual objects. A **low-level notification** is an event that describes the state of an object, based on event detection logic. An **operator message** is an event that describes an alarm condition for a domain object.

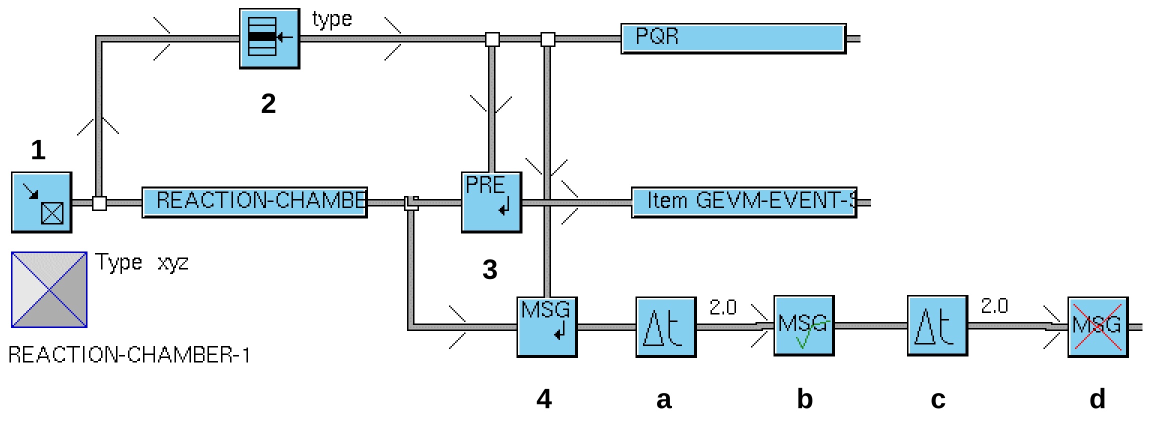
You can interact with low-level notifications and operator messages, using the G2 Event Management (GEVM) module, which provides a default Message Browser for viewing low-level notifications and messages. It also provides an API for implementing listener interfaces.

You can use the blocks on this palette in a specific diagram only. For more information, see the Specific Event Detection Diagram block on page 53.

To post low-level notifications and messages that apply generically to any instance of a class of domain objects, use the Generic Post Raw Event and Generic Post Message blocks on the Generic Template Blocks palette. For details, see “Generic Template Blocks” on page 68.

Events and Alarm Management Example

This example shows how to post low-level notifications and operator messages, and to configure an escalation scheme for operator messages. The figure labels each part of the diagram.



1 The Fetch Object block gets the object named reaction-chamber-1 and displays it in the path display.

2 The Fetch Property block gets the value of the type property of the object and displays it in the path display.

3 The Post Raw Event block posts a low-level notification for the fetched object and displays the event in the path display.

4 The Post Message block posts a message for the fetched object and passes the message downstream. The message is acknowledged and deleted, using the default escalation scheme, as follows:

  a The Data Delay block waits, then passes the message downstream.

  b The Acknowledge Message block acknowledges the message and passes it downstream.

  c The Data Delay block waits, then passes the message downstream.

  d The Delete Event or Message Block deletes the message from the queue.

Post Raw Event

Posts a low-level notification for the input object on the horizontal input path, using the value on the vertical input path, then passes the event downstream. Typically, the operator does not interact with low-level notifications. Instead, you can use low-level notifications to manage application states, for example, across multiple G2s.

You can configure the event category, message, detail, priority, and life time. The message and detail can refer to any attribute of the input object, using the syntax $*attribute-name*, where *attribute-name* is any attribute of the input object. For example, if the input object defines an attribute named temperature, you could refer to the value of this attribute by using $temperature. At run time, the block substitutes the value of the attribute into the text.

The message and detail can also refer to these keywords for text substitution at run time:

- $TARGET-OBJECT: The name of the target object for the message.
- $BLOCK-LABEL: The label of the Post Message block, which appears next to the block.
- $BLOCK-COMMENT: The value of the user-defined comment associated with the message. The operator enters comments through the Message Browser at run time.
- $MODEL-NAME: The name of the GEDP diagram in which the Post Message block appears.
- $MODEL-CATEGORY: The category of the GEDP diagram or template

- $INPUT-VALUE: The input value that is being passed to the Post Message block.
- $INPUT-VALUE-COLLECTION-TIME: The time at which the input value arrived at the block.

Property	Description
Post When Input is True	Whether to post the event only when the input value is a truth-value and is true. By default, the block posts the event whenever any input value arrives at the block.
Category	A symbol that defines the event category, which you can use to filter events.
Message	A text string to display in the specified queue when the event is posted.
Detail	A text string that describes the event.
Priority	An integer that defines the event priority.
Life Time	The duration of the event in the specified queue, in seconds.

Post Message

Posts an operator message for the input object on the horizontal input path, using the value on the vertical input path, then passes the message downstream. You can configure the message category, message, detail, advice, priority, life time, and message queue.

The message and detail can refer to any attribute of the input object, using the syntax $*attribute-name*, where *attribute-name* is any attribute of the input object. For example, if the input object defines an attribute named temperature, you could refer to the value of this attribute by using $temperature. At run time, the block substitutes the value of the attribute into the text.

The message, detail, and advice can refer to these keywords for text substitution at run time:

- $TARGET-OBJECT: The name of the target object for the message.
- $BLOCK-LABEL: The label of the Post Message block, which appears next to the block.
- $BLOCK-COMMENT: The value of the user-defined comment associated with the message. The operator enters comments through the Message Browser at run time.
- $MODEL-NAME: The name of the GEDP diagram in which the Post Message block appears.
- $MODEL-CATEGORY: The category of the GEDP diagram or template.
- $INPUT-VALUE: The input value that is being passed to the Post Message block.
- $INPUT-VALUE-COLLECTION-TIME: The time at which the input value arrived at the block.

Property	**Description**
Post When Input is True	Whether to post the message only when the input value is a truth-value and is true. By default, the block posts the message whenever any input value arrives at the block.
Acknowledgement Required	Whether the message requires acknowledgement before it can be deleted. By default, messages do require acknowledgement.
Type	A symbol that identifies the message class. The default value is gevm-message. Using this message type automatically posts the message to the default GEVM Message Browser.
Category	A symbol that defines the message category, which you can use to filter messages.
Message	A text string to display in the specified queue when the message is posted.
Detail	A text string that describes the message.
Advice	A text string that provides advice to the operator about the message.
Priority	An integer that defines the message priority.
Life Time	The duration of the message in the specified queue, in seconds.
Send to Queue	A text string that specifies the name of the queue in which to display the message. By default, the message goes to the default Messages queue and appears in the Messages Browser; thus, unless you want to send the message to another queue, you do not need to configure this property.

You can configure a plot specification for the block, which allows you to display a trend chart of up to four values when the message appears in the Message Browser.

To configure the trend chart, click the Plot Specification button and configure these attributes for each plot:

Attribute	Description
Title	The label for the plot.
Item	The datapoint to plot.
Significant Digits	The number of significant digits to round the value.
Use History	Whether to plot a history of the value.
Color	The plot color.

To display the plot, when the message appears in the Messages Browser, select the message and click the Additional Information button. For more information, see the *Optegrity User's Guide*.

Delete Event or Message

Deletes the input event or message. The block passes nothing downstream.

Acknowledge Message

Acknowledges the input event or message. The block passes the event or message downstream.

Event Detection Diagrams

An event detection diagram provides a container for creating a GEDP diagram that applies to a single domain object or an arbitrary G2 object. A generic event detection template provides a container for creating a generic diagram that applies to all instances of a target class of domain objects.

You place GEDP blocks on the diagram details. You can configure how the GEDP diagram evaluates.

Certain GEDP blocks are designed for use in specific diagrams only, while others are designed for use in generic templates only. You can use most GEDP blocks in either type of diagram. For information on which blocks you can use in specific vs. generic diagrams, see the Introduction.

When running GEDP in a stand-alone environment, you create GEDP diagrams and templates from a palette.

You can use a GEDP diagram as the activation procedure for a SymCure generic action. Such a GEDP diagram must use the Return block to return a value to its generic diagram. Also, ensure that the Activated option is disabled for the specific diagram or generic template to deactivate the diagram after it has finished execution. For more information, see the *SymCure User's Guide*.

GEDP provides three general categories of diagrams and templates:

- Detection diagrams and templates.
- Response diagrams and templates.
- Test diagrams and templates.

In addition, you can create a general-purpose event detection diagram from the Event Detection Diagrams palette.

To create an event detection diagram:

1 Choose Project > Logic, choose Detect, Test, or Respond, choose Dataflow Templates or Data Flow Instances, choose Manage, and click the New button.

GEDP creates a diagram or template and displays its properties dialog.

Tip You can also create event detection diagrams and templates from the Navigator by expanding the tree under the Logic node, or from the Event Detection Diagrams palette of the Event Detection toolbox.

2 Configure the properties of the GEDP diagram or template.

For a description of the properties, see “Specific Event Detection Diagram” on page 53 and “Generic Event Detection Template” on page 58.

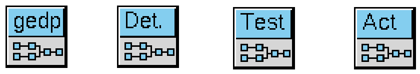
To display the event detection diagram details:

➔ Choose Project > Event Detection, choose Generic Event Detection or Specific Event Detection, and choose a diagram whose details you want to display.

or

➔ Choose Show Details on a GEDP diagram or template.

Specific Event Detection Diagram



Provides a container in which you create a GEDP diagram that applies to a single domain object or an arbitrary G2 object.

Specific diagrams can monitor internal datapoints contained in the process map and can trigger events, based on datapoint values. They can also listen for low-level notifications, messages, and SymCure events for instances of the target domain object class.

By default, GEDP diagrams evaluate on event-driven basis, based on the entry points on the details. You can configure the diagram to evaluate on a schedule, with an optional maximum number of evaluations. You can also configure the diagram to execute, activate, and/or deactivate, based on the value of an internal datapoint in a process map.

Properties

This table describes the properties of the block:

Property	Description
Diagram Name	A text string that identifies the specific diagram.
Version	A user-defined revision number, which you can use for model management, for example, 1.1 and 1.2.
Category	A user-defined text string, which you can use to organize models in the menus. The categories appear as submenus in the Project > Event Detection > Specific Event Detection menu. If you do not define a category, the specific diagram appears in the Unspecified submenu.

Property	**Description**
Evaluation Interval	The time interval at which to schedule the evaluation of the diagram. By default, this value is `0`, which means evaluation is event-driven. To configure the diagram to evaluate on a schedule, set the Evaluation Interval to the rate at which evaluation should occur, in seconds. When Evaluation Interval is set to a number greater than 0, the Evaluate on Source Datapoint Change property of the entry point must be `no`.
Evaluation Repeat Count	The maximum number of times the diagram should execute. This property is only relevant when the Evaluation Interval is set to a number greater than 0.
Repeat Count Enabled	Whether to enable the Evaluation Repeat Count.
Evaluation Condition	A specification of when the diagram should evaluate, based on the value of an internal datapoint in a process map. See below for a description of the properties you can configure.
Activation Condition	A specification of when the diagram should activate, based on the value of an internal datapoint in a process map. See below for a description of the properties you can configure.
Deactivation Condition	A specification of when the diagram should deactivate, based on the value of an internal datapoint in a process map. See below for a description of the properties you can configure.

To specify an Evaluation Condition, Activation Condition, and/or Deactivation Condition, you configure these properties:

Property	Description
Evaluation Trigger Activation Trigger Deactivation Trigger	The name of the internal datapoint to evaluate. The datapoint must be a subclass of grtl-datapoint.
Operator	The mathematical operator to use. The default value is true, which means the test is always true. The options are: =, >, <, >=. <=. /=
Key	The name of another datapoint whose value should be compared against the value of Evaluation Trigger, Activation Trigger, or Deactivation Trigger.
Value	A static value to compare against the value of Evaluation Trigger, Activation Trigger, or Deactivation Trigger.
Activate Condition	Whether to activate the Evaluation Condition, Activation Condition, or Deactivation Condition.

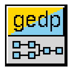
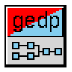
This table provides a summary of how to configure the various evaluation options for a GEDP diagram:

Evaluation Type	Evaluate Source Datapoint Change of Entry Point	Evaluation Interval	Evaluation Repeat Count Enabled	Evaluation Condition
Event-driven	true	0	false	unspecified
Demand-driven	false	0	false	unspecified
Scheduled	false	*quantity*	false	unspecified
Bound scheduled	false	*quantity*	*integer*	unspecified
Datapoint-driven	false	0	false	A specification for Evaluation Condition

To configure the Activation Condition and Deactivation Condition, you use the same specification for the first three properties in the table above as the Datapoint Value evaluation type.

Menu Choices

This table describes the menu choices for a specific event detection diagram:

Menu Choices	Description
Show Details	Shows the diagram details on which you place GEDP blocks.
Initialize	Validates, resets, and activates the blocks on the diagram details.
Activate	Activates all the blocks on the diagram details. All blocks in the diagram must be active before the diagram can be evaluated.
Deactivate	Deactivates all the blocks on the diagram details. When a diagram has been deactivated, the diagram looks like this: 
Validate	Validates the blocks on the diagram details to verify that they are configured correctly. If a diagram is invalid, the diagram looks like this: 
Evaluate	Manually evaluates the blocks on the diagram details. This menu choice only appears when the diagram has been activated.

Generic Event Detection Template

Provides a template in which you create a generic event detection template that applies to all instances of a target class of domain objects. The GEDP blocks on the details apply to instances of the target class.

When you initialize a process map that contains domain objects, GEDP creates specific event detection diagrams for each instance of the target class.

If you delete a generic event detection template, all specific event detection diagrams created from the generic template are automatically deleted. Similarly, if you activate or deactivate a generic event detection template by enabling or disabling the Activate option in the properties dialog, all specific event detection diagrams created from the generic template are automatically activated or deactivated as well.

Properties

A generic event detection template defines the same properties as a specific event detection diagram with the following changes:

Property	Description
Template Name	The name of the generic template.
Target Class	The name of the domain object class to which the generic event detection template applies. The target class is any subclass of the grtl-domain-object class, including any subclass of opt-domain-object.
Is Persistent	Whether the specific diagrams created for instances of the target class persist through initialization. By default, specific diagrams do not persist. Enable this option when creating generic event detection diagrams whose specific diagrams you want to persist, for example, in custom event detection templates. By default, Is Persistent is enabled for all the built-in event detection templates that are available when the intelligent object libraries are loaded.

For information on these properties, see the Specific Event Detection Diagram block on page 53.

For an example, see “Generic Template Blocks Example” on page 68.

Menu Choices

A Generic Event Detection Template defines these menu choices only; it does not define the same menu choices as a Specific Event Detection Diagram.

Menu Choices	Description
Assign Diagrams to Domain Objects	Creates specific event detection diagrams for all instances of the target class. If an instance of the target class already has a specific diagram, this menu choice does not create a new diagram for this instance.
Delete Diagrams on Domain Objects	Deletes all existing specific event detection diagrams for all instances of the target class.

Fault Model Diagnostics

The Generic Send Fault Model Event and Send Fault Model Event blocks trigger SymCure events that apply generically to all instances of a domain object class and to specific domain objects, respectively. Thus, these blocks can appear in a generic template and in a specific diagram, respectively. For more information, see “Event Detection Diagrams” on page 52.

The Generic Send Fault Model Action Result block

When changing an event name on the SymCure model, the event name change propagates to all corresponding Send Fault Model Event blocks within a generic template.

Generic Send Fault Model Action Result

Sends a result to a SymCure action. A GEDP diagram is frequently used to obtain the result of a SymCure test. If the test is associated with multiple events, the GEDP diagram must send a value to each of the associated events. A Generic Send Fault Model Action Result block would communicate the result of the diagram to the SymCure test, which would then update the values of the associated events, thus simplifying the GEDP diagram.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	Description
Action Name	The name of the SymCure action whose result to send. You choose the action from a dropdown list of existing SymCure actions; thus, the action must already be defined.
Send True Result	Whether to send a result of true to the action.
Send Suspect Result	Whether to send a result of suspect to the action.
Send False Result	Whether to send a result of false to the action.
Cost	The cost of executing the action.

You can navigate to the SymCure action from the diagram by choosing Show Fault Model Action on the block.

This is the same as the Send Fault Model Action Result block on the Generic Template Blocks palette.

Generic Send Fault Model Event

Sends a SymCure event on the associated domain object, then passes the event downstream. You must pass the value of the event on the horizontal input path, which can be the symbol true, false, suspect, or unknown; the integer 1 for true, -1 for false, and 0 for suspect; or the text "true", "false", "suspect", or "unknown".

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

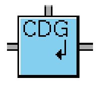
Property	**Description**
Event Name	The name of the SymCure event to send. You choose the event from a dropdown list of existing SymCure events that have been defined for the target class of the event detection template; thus, the event must already be defined.
Sent True Event	Whether to send events whose value is true. By default, the block sends only true events.
Send Suspect Event	Whether to send events whose value is suspect. By default, the block does not send suspect events.

Property	Description
Send False Events	Whether to send events whose value is false. By default, the block does not send false events.
If Event Exists	Checks for the existence of true, false, and suspect events before sending the event to SymCure. By default, the existence check is enabled for false events only, which prevents sending a new false event to SymCure when a false event already exists. By default, suspect and true events are sent regardless of the event's existence. In general, whether SymCure should be sent a new false event or not depends on the nature of the event detection mechanism. If the underlying event is being monitored on a regular basis, there is no need to build a new false event, because if SymCure needs it later, it is guaranteed that SymCure will receive it. On the other hand, if the event detection is asynchronous and SymCure has no idea if or when a new value for the event will be available, it does make sense to send the new false event to SymCure.

This is the same as the Send Fault Model Event block on the Generic Template Blocks palette.

You can navigate to the SymCure event from the diagram by choosing Show Fault Model Event on the block.

Send Fault Model Event



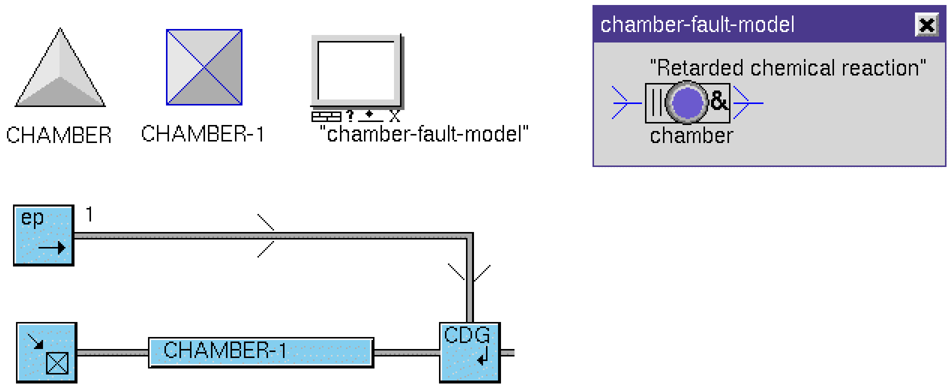
Sends a SymCure event on the input object, then passes the event downstream. You must pass the object to which the event applies on the horizontal input path and the value of the event on the vertical input path. The value of the event can be the symbol true, false, suspect, or unknown; the integer 1 for true, -1 for false, and 0 for suspect; or the text "true", "false", "suspect", or "unknown".

Property	Description
Event Name	The name of the SymCure event to send. You choose the event from a dropdown list of existing SymCure events that have been defined for the domain object; thus, the event must already be defined.
Send True Events	Whether to send events whose value is true. By default, the block sends only true events.
Send Suspect Events	Whether to send events whose value is suspect. By default, the block does not send suspect events.
Send False Events	Whether to send events whose value is false. By default, the block does not send false events.
If Event Exists	Checks for the existence of true, false, and suspect events before sending the event to SymCure. By default, the existence check is enabled for false events only, which prevents sending a new false event to SymCure when a false event already exists. By default, suspect and true events are sent regardless of the event's existence. In general, whether SymCure should be sent a new false event or not depends on the nature of the event detection mechanism. If the underlying event is being monitored on a regular basis, there is no need to build a new false event, because if SymCure needs it later, it is guaranteed that SymCure will receive it. On the other hand, if the event detection is asynchronous and SymCure has no idea if or when a new value for the event will be available, it does make sense to send the new false event to SymCure.

You can navigate to the SymCure event from the diagram by choosing Show Fault Model Event on the block.

This example uses the Fetch Object block to fetch the object named chamber-1, which is a domain object. The GEDP diagram uses a Value Entry Point to pass an event value of 1 for true. It then sends the SymCure event named

"Retarded chemical reaction" to the fetched object. The fault model shows the SymCure event, whose event type is alarm.



When the “Retarded chemical reaction” event is sent, it automatically appears in the SymCure diagnostic console Alarms Browser.

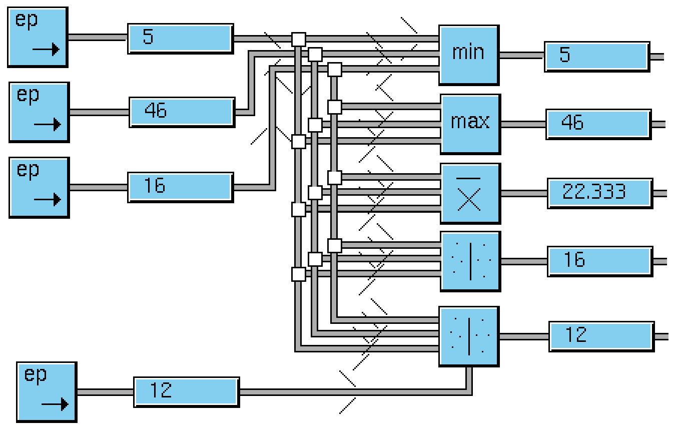
Functions

These blocks perform statistical operations on data values. You can also execute user-defined functions.

You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

Functions Example

This example computes the minimum, maximum, average, and median values of the top three input values. It also computes the median value of all four input values.

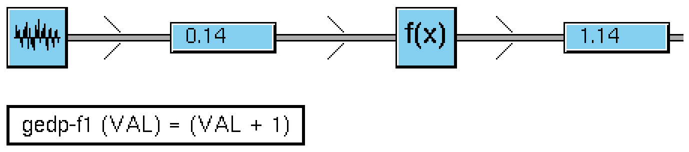


Generic Function

Executes a user-defined function on the input value.

Property	Description
Function Name	The name of the user-defined function to execute.

This example executes a user-defined function on an incoming signal:



```
gedp-f1 (VAL) = (VAL + 1)
```

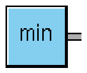
Average Input

Computes the average of all the input values. The block accepts any number of inputs.

Median Input

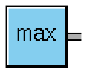
Passes the median of all the input values. The block accepts any number of inputs. If the block has an odd number of input values, it passes the middle input value. If the block has an even number of input values, it passes the minimum of the two middle input values.

Low Selecting



Passes the minimum of all the input values. The block accepts any number of inputs.

High Selecting



Passes the maximum of all the input values. The block accepts any number of inputs.

Generic Template Blocks

These blocks allow you to listen for property changes for instances of domain object classes, and to set property values of those objects. These blocks also allow you to generate low-level notifications, messages, and SymCure events for instances of domain object classes, and to listen for low-level notifications.

You can use these blocks in a generic template only. For more information, see the Generic Event Detection Template block on page 58.

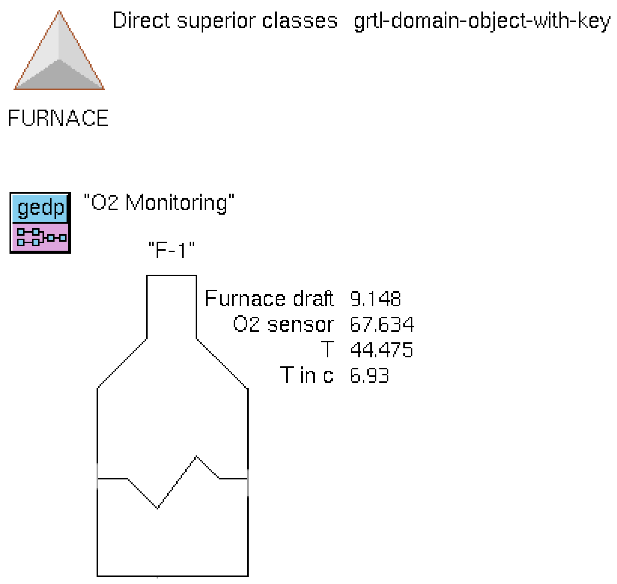
The domain object class is the target class of a generic template. The target class is any subclass of grtl-domain-object, which includes any subclass of opt-domain-object.

For more information about GEDP diagram templates, see “Generic Event Detection Template” on page 58.

For information on blocks that perform the same functions on individual objects, see “Fault Model Diagnostics” on page 60 and “Events and Alarm Management” on page 46, and “Objects” on page 82.

Generic Template Blocks Example

This example associates a domain object class with a generic template that listens for property change events on any furnace. Here is the generic template named "O2 Monitoring", whose target-class is furnace. The furnace class is a subclass of grtl-domain-object-with-key, which provides the mechanism for associating domain objects with GEDP diagrams.

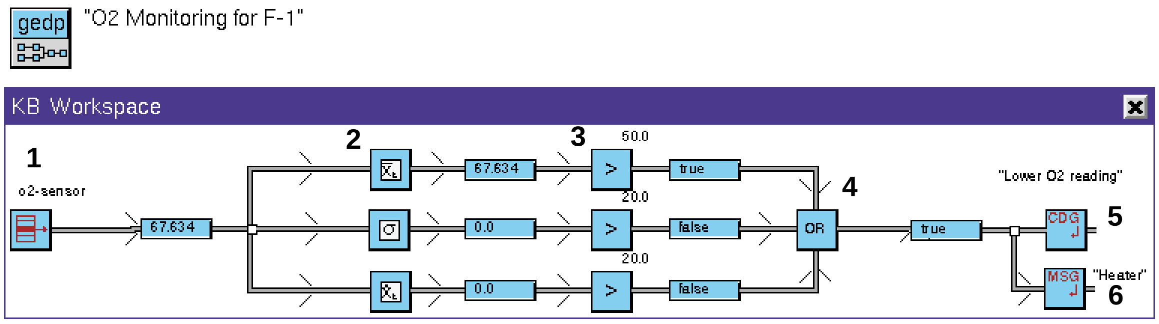


Direct superior classes grtl-domain-object-with-key

```
Furnace draft  9.148
    O2 sensor  67.634
            T  44.475
       T in c  6.93
```

To associate the generic template with the furnace class, you must initialize the process map by choosing Project > Initialize Application.

Here is the specific diagram associated with the f-1 furnace named "O2 Monitoring for F-1":



The GEDP diagram uses these blocks to perform these tasks:

1 The Generic Fetch Property block listens for changes to the o2-sensor attribute of the associated domain object, in this case, the f-1 furnace.

2 The Moving Average, Standard Deviation, and Discrete Rate of Change blocks perform statistical analysis on the input values.

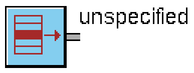
3 The Greater Than blocks tests the computed values against different thresholds.

4 The Or Gate tests whether any of the computed values is true.

5 If true, the Generic Send Fault Model Event block generates a Low O2 reading event for the f-1 furnace.

6 If true, the Generic Post Message block generates a heater message for the f-1 furnace.

Generic Fetch Property

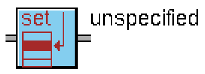


Listens for changes to the value of the specified attribute of the associated domain object and passes the value downstream. The block triggers only when the attribute value changes.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	**Description**
Property Name	The name of the property of the domain object to which to listen for changes. The property name can use dot notation to refer to embedded datapoints of embedded objects of domain objects, for example, plant.furnace.o2-sensor or temp.pv.average.

Set Property

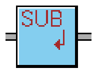


Sets the specified attribute of the associated domain object to the value on the input path, then passes the value downstream.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	**Description**
Property Name	The name of the property of the domain object to set. The property name can use dot notation to refer to embedded datapoints of embedded objects of domain objects, for example, plant. furnace.o2-sensor or temp.pv.average.

Generic Execute Subdiagram

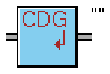


Allows you to sequence the execution of other GEDP diagrams. Sequencing GEDP diagrams provides several benefits, such as limiting processing overhead and operator messages by starting with basic sensor analysis, then, depending on their results, evaluating more complex event detections.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	**Description**
Sub Diagram Name	The name of a generic template whose specific diagram should be executed.

Generic Send Fault Model Event



Sends a SymCure event on the associated domain object, then passes the event downstream. You must pass the value of the event on the horizontal input path, which can be the symbol true, false, suspect, or unknown; the integer 1 for true, -1 for false, and 0 for suspect; or the text "true", "false", "suspect", or "unknown".

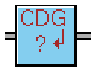
You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	Description
Event Name	The name of the SymCure event to send. You choose the event from a dropdown list of existing SymCure events that have been defined for the target class of the event detection template; thus, the event must already be defined.
Sent True Event	Whether to send events whose value is true. By default, the block sends only true events.
Send Suspect Event	Whether to send events whose value is suspect. By default, the block does not send suspect events.
Send False Events	Whether to send events whose value is false. By default, the block does not send false events.
If Event Exists True False Suspect	Checks for the existence of true, false, and suspect events before sending the event to SymCure. By default, the existence check is enabled for false events only, which prevents sending a new false event to SymCure when a false event already exists. By default, suspect and true events are sent regardless of the event's existence. In general, whether SymCure should be sent a new false event or not depends on the nature of the event detection mechanism. If the underlying event is being monitored on a regular basis, there is no need to build a new false event, because if SymCure needs it later, it is guaranteed that SymCure will receive it. On the other hand, if the event detection is asynchronous and SymCure has no idea if or when a new value for the event will be available, it does make sense to send the new false event to SymCure.

This is the same as the Generic Send Fault Model Event block on the Fault Model Diagnostics palette.

To go to the generic fault model that contains the specified event, choose Show Fault Model Event on the block

Generic Send Fault Model Action Result



Sends a result to a SymCure action. A GEDP diagram is frequently used to obtain the result of a SymCure test. If the test is associated with multiple events, the GEDP diagram must send a value to each of the associated events. A Generic Send Fault Model Action Result block would communicate the result of the diagram to the SymCure test, which would then update the values of the associated events, thus simplifying the GEDP diagram.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

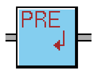
Property	**Description**
Action Name	The name of the SymCure action whose result to send. You choose the action from a dropdown list of existing SymCure actions; thus, the action must already be defined.
Send True Result	Whether to send a result of true to the action.
Send Suspect Result	Whether to send a result of suspect to the action.
Send False Result	Whether to send a result of false to the action.
Cost	The cost of executing the action.

You can navigate to the SymCure action from the diagram by choosing Show Fault Model Action on the block.

This is the same as the Send Fault Model Action Result block in the Fault Model Diagnostics palette.

To go to the generic fault model that contains the specified action, choose Show Fault Model Action on the block.

Generic Post Raw Event



Posts a low-level notification for the associated domain object, using the value on the input path, then passes the event downstream. Typically, the operator does not interact with low-level notifications. Instead, you can use low-level notifications to manage application states, for example, across multiple applications.

You can configure the event category, message, detail, priority, life time, event queue, and escalation scheme. The message and detail can refer to any attribute of the input object, using the syntax $*attribute-name*, where *attribute-name* is any attribute of the input object. For example, if the input object defines an attribute named temperature, you could refer to the value of this attribute by using $temperature. At run time, the block substitutes the value of the attribute into the text.

The message and detail can refer to these keywords for text substitution at run time:

- $TARGET-OBJECT: The name of the target object for the message.
- $BLOCK-LABEL: The label of the Post Message block, which appears next to the block.
- $BLOCK-COMMENT: The value of the user-defined comment associated with the message. The operator enters comments through the Message Browser at run time.
- $MODEL-NAME: The name of the specific diagram in which the Post Message block appears.
- $MODEL-CATEGORY: The category of the generic template.
- $INPUT-VALUE: The input value that is being passed to the Post Message block.
- $INPUT-VALUE-COLLECTION-TIME: The time at which the input value arrived at the block.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see "Generic Event Detection Template" on page 58.

Property	Description
Post When Input is True	Whether to post the low-level notification only when the input value is a truth-value and is true. By default, the block posts the event whenever any input value arrives at the block.
Category	The symbol that defines the event category.
Message	A text message to display in the specified queue when the event occurs, as a string.
Detail	A detailed description of the event, as a string.
Priority	The priority of the event, as an integer.
Life Time	The duration of the event, in seconds.

Generic Check for Raw Event

Checks if a low-level notification for the associated domain object exists. The low-level notification can be generated by the Post Raw Event block on the Generic Template Blocks palette or on the Event and Alarm Mgmt palette. The block outputs a value of true if the event exists and false if the event does not exist

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	Description
Category	The category of the low-level notification to be checked. The low-level notification must exist for the domain object associated with the block.

Generic Delete Raw Event

Deletes all low-level notifications in a specified category that occur on the associated domain object.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	Description
Category	The category of the low-level notification to be deleted. The low-level notification must exist for the domain object associated with the block. If the domain object has multiple low-level notifications in the same category but from different senders, the block deletes all the low-level notifications.

Generic Post Message

Posts an operator message for the associated domain object, using the value on the input path, then passes the message downstream. You can configure the message category, message, detail, advice, priority, life time, event queue, and escalation scheme.

The message and detail can refer to any attribute of the input object, using the syntax $*attribute-name*, where *attribute-name* is any attribute of the input object. For example, if the input object defines an attribute named temperature, you could refer to the value of this attribute by using $temperature. At run time, the block substitutes the value of the attribute into the text.

The message and detail can refer to these keywords for text substitution at run time:

- $TARGET-OBJECT: The name of the target object for the message.
- $BLOCK-LABEL: The label of the Post Message block, which appears next to the block.

- $BLOCK-COMMENT: The value of the user-defined comment associated with the message. The operator enters comments through the Message Browser at run time.
- $MODEL-NAME: The name of the specific diagram in which the Post Message block appears.
- $MODEL-CATEGORY: The category of the generic template.
- $INPUT-VALUE: The input value that is being passed to the Post Message block.
- $INPUT-VALUE-COLLECTION-TIME: The time at which the input value arrived at the block.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see "Generic Event Detection Template" on page 58.

Property	Description
Post When Input is True	Whether to post the message only when the input value is a truth-value and is true. By default, the block posts the message whenever any input value arrives at the block.
Acknowledgement Required	Whether the message requires acknowledgement before it can be deleted. By default, messages do require acknowledgement.
Type	A symbol that identifies the message class. The default value is gevm-message. Using this message type automatically posts the message to the default GEMV Message Browser.
Category	The symbol that defines the message category.
Message	The message text to display in the specified queue when the message occurs, as a string.
Detail	A detailed description of the message, as a string.
Advice	A description of advice to take when the message occurs, as a string.
Priority	The priority of the message, as an integer.

Property	Description
Life Time	The duration of the message, in seconds.
Send to Queue	A text string that specifies the name of the queue in which to display the message. By default, the message goes to the Messages queue, which displays message in the Messages Browser; thus, unless you want to send the message to another queue, you do not need to configure this property.

Generic Check for Message

Checks if a message for the associated domain object exists. The message can be generated by the Post Message block on the Generic Template Blocks palette or on the Event and Alarm Mgmt palette. The block outputs a value of true if the message exists and false if the message does not exist

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see “Generic Event Detection Template” on page 58.

Property	Description
Type	A symbol that identifies the message class. The default value is gevm-message.
Category	The category of the message to be checked. The message must exist for the domain object associated with the block.

Generic Delete Message

Deletes the input message for the associated domain object. The block passes the message downstream.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see "Generic Event Detection Template" on page 58.

Generic Acknowledge Message

Acknowledges the input message for the associated domain object. The block passes the message downstream.

You use this block in a generic template. The associated domain object is an instance of the target class of the generic template. The target class is any subclass of grtl-domain-object. For more information about generic templates, see "Generic Event Detection Template" on page 58.

Logic Gates

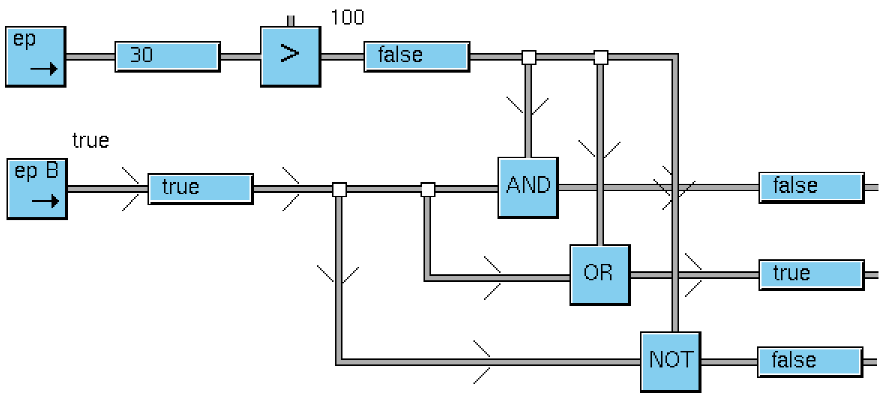
These blocks perform logical operations on boolean values. Typically, you use logic gates with observation blocks, which test numeric values and return boolean values. You can also provide logical values, using a Boolean Entry Point.

You can use these blocks in a specific diagram or in a generic template. For more information, see "Event Detection Diagrams" on page 52.

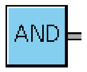
Logic Gates Example

This example tests whether a number is above a threshold, using a Greater Than block. It then compares the boolean result with the value of a Boolean Entry Point, as follows:

- The output value of the And Gate is false, because at least one input value is false.
- The output value of the Or Gate is true, because at least one input value is true.
- The output value of the Not Gate is false, because the most recent input value was true.

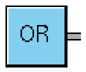


And Gate



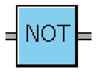
Performs a logical "and" on all the input boolean values, which returns true if all input values are true; otherwise returns false. This block accepts any number of inputs.

Or Gate



Performs a logical "or" on all the input boolean values, which returns true if any input value is true; otherwise returns false. This block accepts any number of inputs.

Not Gate



Performs a logical "not" on the input boolean value, which returns true if the input value is false and returns false if the input value is true. We recommend that you use the block with a single input only, because we cannot guarantee which input is chosen for the logical NOT operation.

Objects

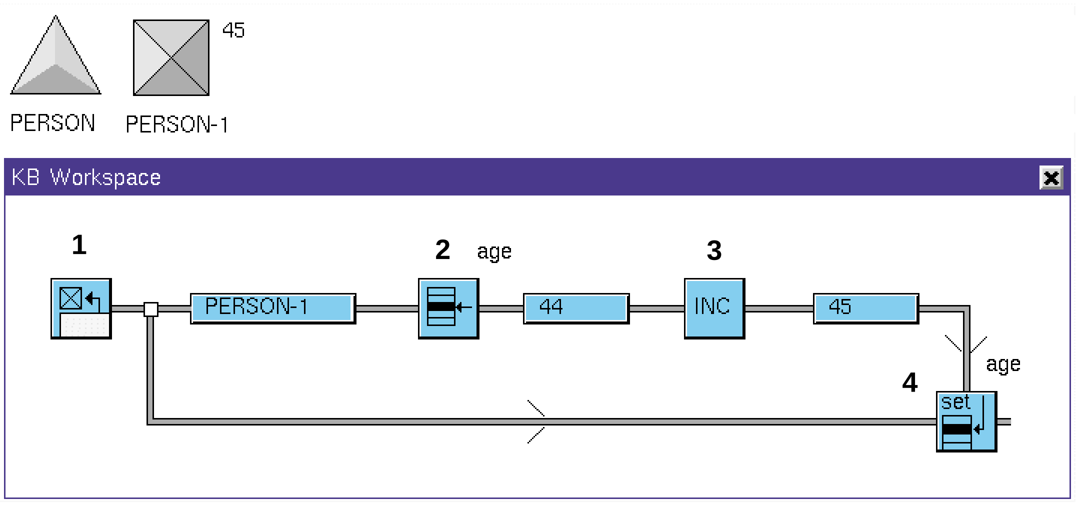
These blocks allow you to get specific objects, and to get and set property values of those objects. This category of blocks also provides a float variable in which to store a history of property values for an object.

You can use these blocks in a specific diagram only. For more information, see the Specific Event Detection Diagram block on page 53.

Objects: Example 1

This example shows how to create a GEDP diagram on the subworkspace of an object, then get and set an attribute of the object. The diagram uses these blocks, which are labeled in the diagram below:

1. The Fetch Superior Object block gets the superior object, which is person-1.
2. The Fetch Property block gets the value of the age attribute of the object.
3. The Increment block increments the value.
4. The Set Property block sets the new age value back into the age attribute of the object.



Objects: Example 2

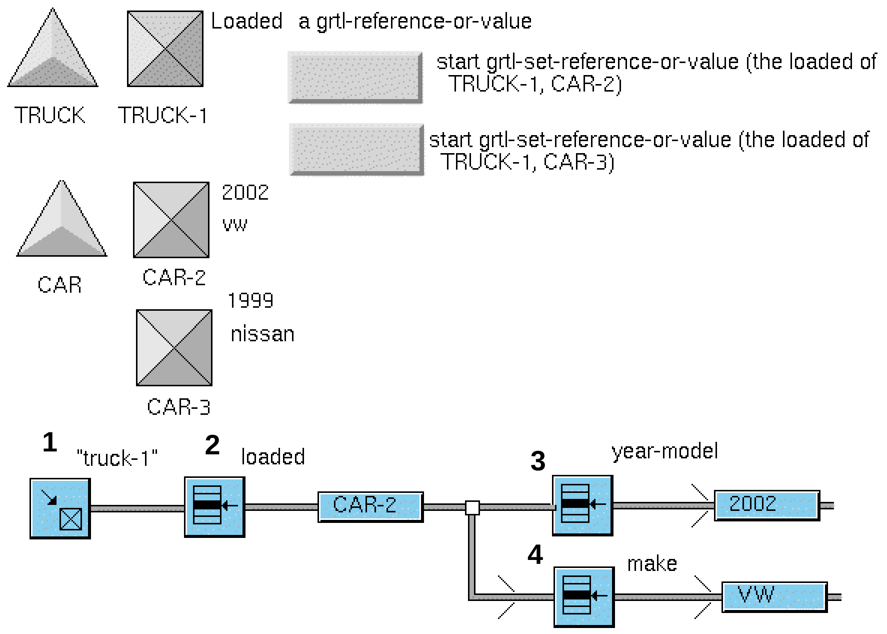
The next example shows how to fetch a property of an object, then fetch a property of a subobject of the object. The diagram uses these blocks, which are labeled in the diagram below:

1 The Fetch Object block gets the object named truck-1.

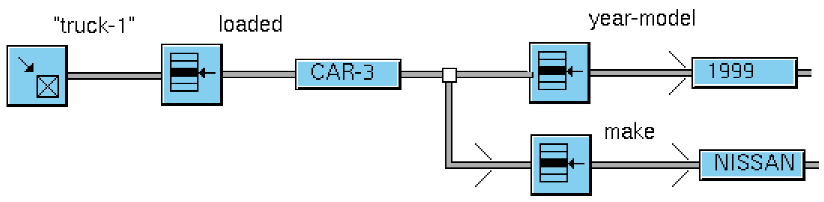
2 The Fetch Property block gets the value of the loaded attribute of the object, which is another object, in this case, car-2.

3 The top Fetch Property block gets the value of the year-model attribute of the subobject, in this case, 2002.

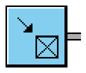
4 The bottom Fetch Property block gets the value of the make attribute of the subobject, in this case, vw.


Loaded a grtl-reference-or-value
start grtl-set-reference-or-value (the loaded of TRUCK-1, CAR-2)
start grtl-set-reference-or-value (the loaded of TRUCK-1, CAR-3)
TRUCK
TRUCK-1
2002
vw
CAR
CAR-2
1999
nissan
CAR-3
1
"truck-1"
2
loaded
CAR-2
3
year-model
2002
4
make
VW

The loaded attribute of the truck is an instance of the grtl-reference-or-value class. You can use the grtl-set-reference-or-value procedure to set the value of the subobject to either car-2 or car-3. By using this subclass, you can use the Fetch Property block to get an attribute of the subobject. Here is the diagram that results when you set the subobject to car-3:



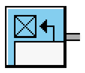
Fetch Object



Fetches an object of a given class, by its name or a key, and passes the object downstream. You can use this block in conjunction with the Fetch Property block to get the value of an attribute of an object.

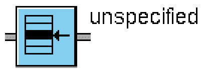
Property	**Description**
Name	A text string that identifies the name of the object to fetch.
Object Class	A symbol that defines the class name of the object to fetch.

Fetch Superior Object



Fetches the domain object that is superior to the workspace on which the block appears and passes the object downstream. You can use this block in conjunction with the Fetch Property block to get the value of an attribute of an object. This block does nothing if the superior object does not exist.

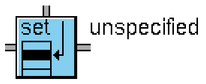
Fetch Property



Fetches the value of the specified attribute from the input object and passes the object downstream. This block does nothing if the specified attribute does not exist. You can use this block in conjunction with the Fetch Object and Fetch Superior Object blocks to get the value of an attribute of an object. You can also string together multiple Fetch Property blocks to get the attribute of a subobject.

Property	Description
Property Name	The name of the property to fetch from the input object.

Set Property



Sets the specified attribute of the input object on the vertical path to the value on the horizontal input path. Once the property is set, the updated object is passed downstream.

Property	Description
Property Name	The name of the property to set in the input object.

Float Variable

Stores the incoming numerical value locally and passes it unchanged to downstream blocks. This block is useful for displaying the incoming value on trend charts. It is also useful in conjunction with the Rate of Change 2 and Standard Deviation 2 blocks, where you set the history-keeping-spec of the variable to the required history depth of the downstream block.

You must configure the attributes of the variable through its table. To do this, you must switch to Administrator mode. For information on configuring variables, see the *G2 Reference Manual*.

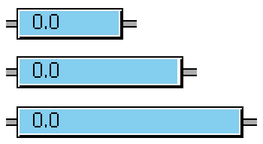
For an example of the Float Variable block, see “First Order Exponential Filter” on page 38.

Path Displays

These blocks display the content of any path.

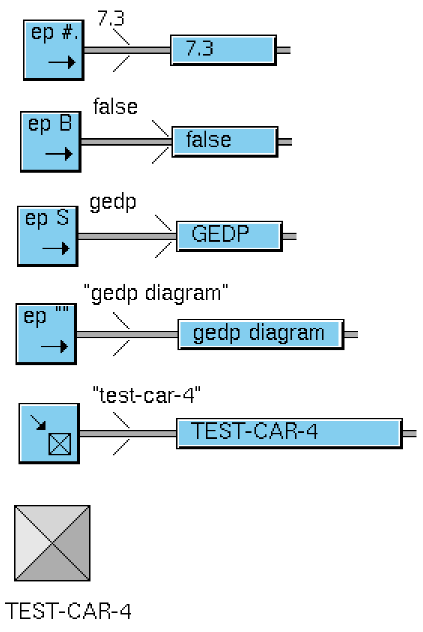
You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

Small Path Display, Large Path Display, Extra-Large Path Display



Displays the output value of an upstream block. If the input value is a value, the path display shows the integer, float, text, boolean, or symbolic value. If the input value is an item, the path display shows the name of the item, if the item has a name; otherwise, it displays an item. If the upstream block has no current output value, the path display shows null.

This example uses different-sized path displays to show a float, boolean, symbolic, and text value, and an object named test-car-4:



Relational Operators

These blocks perform tests on incoming data values and pass a boolean value downstream, based on the result of the test.

By default, when an observation block evaluates, it sends the truth value downstream, even if the truth value has not changed. Sometimes, this is not the desired behavior. For example, suppose the block tests whether an input value goes above 10. When the input value is 5, the block passes a truth value of false. Now suppose the input value becomes 2. By default, the block passes a truth value of false downstream, even though the truth value has not changed. This might or might not be the desired behavior; sometimes, you need to know the result of a new test, even if the value is the same, and sometimes you don't.

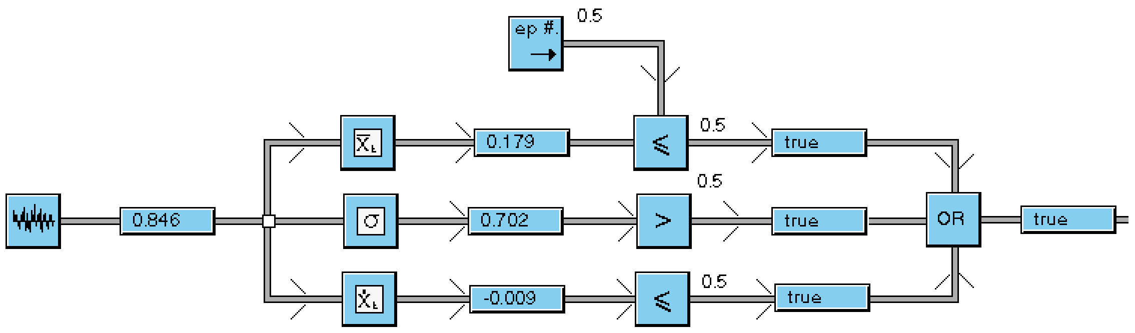
All observation blocks provide the Sync option, which determines whether the block passes new values downstream when the value has not changed. The behavior of the block depends on how you configure the Threshold, as well as on the value of the Sync option, as follows:

- If you configure the Threshold at design time through the properties dialog, the block always passes truth values downstream, even when the truth value has not changed. In this mode, the value of the Sync option is ignored.
- If you configure the Threshold at run time by connecting a block to the top input path, and if the Sync option is enabled, the block passes the truth value downstream, even when the truth value has not changed. However, if the Sync option is disabled, the block passes truth values downstream only when the truth value has changed. By default, the Synch value is enabled; thus, by default, observation blocks behave the same as they did in the previous release.

You can use these blocks in a specific diagram or in a generic template. For more information, see "Event Detection Diagrams" on page 52.

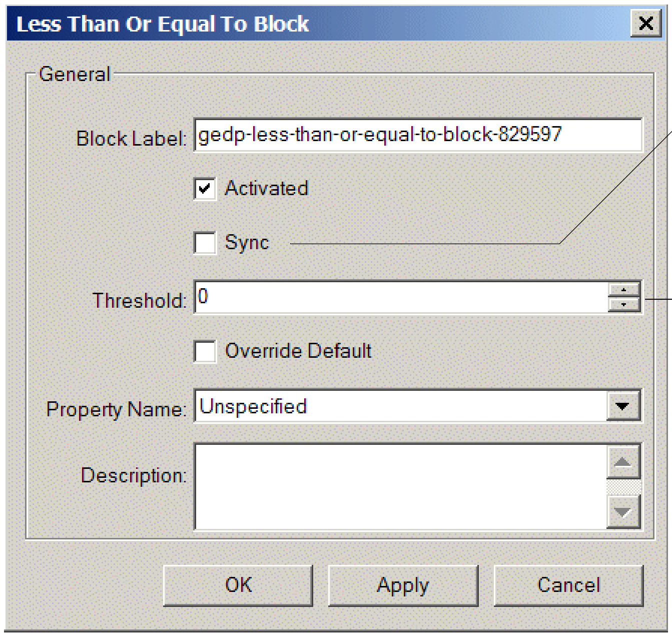
Relational Operators: Example 1

This example computes the moving average, standard deviation, and discrete rate of change of a signal, and tests the values against three thresholds. It passes a value of true if any of the computed values is true. The diagram provides a way of configuring the threshold for the Less Than or Equal To block at run-time, using the Entry Point. The other relational operators specify the threshold through their properties dialogs.



Relational Operators: Example 2

Here is a Greater Than block and its properties dialog configured to suppress sending unchanged values downstream:



Less Than Or Equal To Block

General

Block Label: gedp-less-than-or-equal-to-block-829597

[✓] Activated

[ ] Sync

Threshold: 0

[ ] Override Default

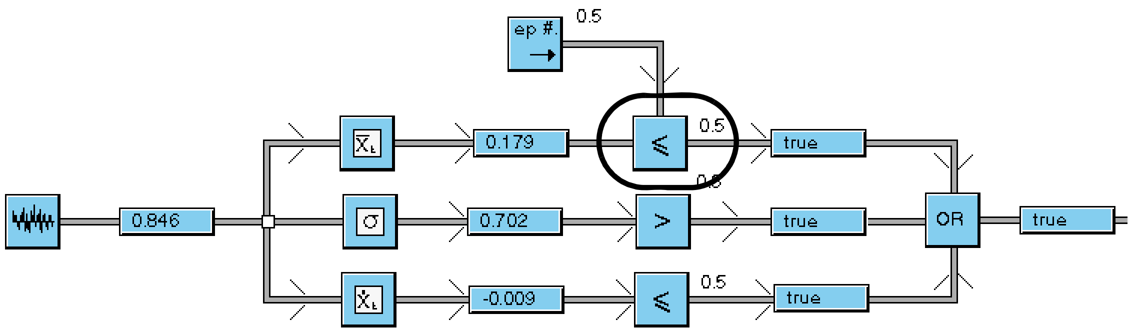
Property Name: Unspecified

Description:

OK Apply Cancel

Sync option is disabled, which suppresses sending unchanged values downstream.

To suppress sending unchanged values, the Threshold of the Less Than block must be configured at run time.



Less Than

Passes true if the input value on the horizontal input path is less than the specified threshold; otherwise passes false. You can specify the threshold at design time through the properties dialog or at run-time through the vertical input path.

Property	Description
Sync	Whether the block passes new values downstream when the value has not changed. By default, this option is disabled, which suppresses sending unchanged values downstream. See the introduction to this section for a description of how to set this option.
Threshold	The value below which the input value must be for the block to pass true.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Threshold, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

Less Than Or Equal To

Passes true if the input value on the horizontal input path is less than or equal to the specified threshold; otherwise passes false. You can specify the threshold at design time through the properties dialog or at run-time through the vertical input path.

Property	**Description**
Sync	Whether the block passes new values downstream when the value has not changed. By default, this option is disabled, which suppresses sending unchanged values downstream. See the introduction to this section for a description of how to set this option.
Threshold	The value below or equal to which the input value must be for the block to pass true.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Threshold, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

Greater Than

Passes true if the input value on the horizontal input path is greater than the specified threshold; otherwise passes false. You can specify the threshold at design time through the properties dialog or at run-time through the vertical input path.

Property	Description
Sync	Whether the block passes new values downstream when the value has not changed. By default, this option is disabled, which suppresses sending unchanged values downstream. See the introduction to this section for a description of how to set this option.
Threshold	The value above which the input value must be for the block to pass true.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Threshold, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

Greater Than Or Equal To

Passes true if the input value on the horizontal input path is greater than or equal to the specified threshold; otherwise passes false. You can specify the threshold at design time through the properties dialog or at run-time through the vertical input path.

Property	Description
Sync	Whether the block passes new values downstream when the value has not changed. By default, this option is disabled, which suppresses sending unchanged values downstream. See the introduction to this section for a description of how to set this option.
Threshold	The value above or equal to which the input value must be for the block to pass true.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Threshold, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

Equal To

Passes true if the two input values are identical, within a specified tolerance; otherwise passes false.

Property	Description
Sync	Whether the block passes new values downstream when the value has not changed. By default, this option is disabled, which suppresses sending unchanged values downstream. See the introduction to this section for a description of how to set this option.
Threshold	The value to use as a tolerance to consider the input values equal.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Threshold, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

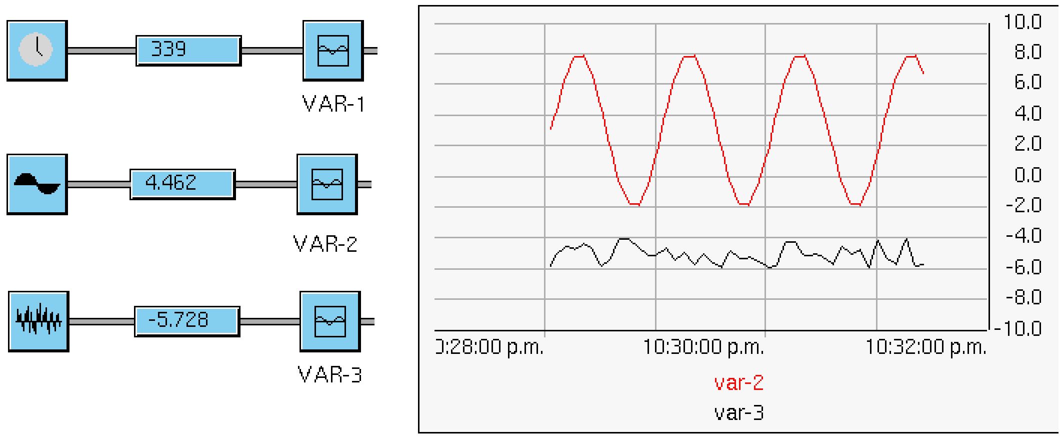
Signal Generators

These blocks provide sample data for testing GEDP diagrams. However, you can also use them in conjunction with real-time data.

You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

Signal Generators Example

This example generates a real-time signal, a sine wave, and a white noise signal, collects the histories in a Float Variable block, and plots the sine wave and white noise. For information on the variable block, see “Float Variable” on page 86.



Real Time Clock

Outputs the G2 time once per second. G2 time is the number of seconds since G2 was last started. This block can be useful to trigger periodic events or to reason about time.

Property	Description
Sample Period	The frequency with which the block passes a new value. If this value is large, the block produces a coarser signal and your application runs faster.

Sine Wave

Generates a sine wave signal, defined by its period, amplitude, bias, and phase.

Property	Description
Period	The amount of time that the block takes to complete a cycle.
Amplitude	Half the difference between the minimum and maximum values.
Bias	The value between the minimum and the maximum.
Phase	A number of degrees between 0 and 360, which determines where the block starts its cycle.
Sample Period	The frequency with which the block passes a new value. If this value is large, the block produces a coarser signal and your application runs faster.
Reset Phase	Whether to start the signal over again when the sample period has passes.

White Noise

Generates a white noise signal, defined by its mean and variance.

Property	Description
Mean	The mean output value of the block.
Variance	The variance of the output values of the block.
Sample Period	The frequency with which the block passes a new value. If this value is large, the block produces a coarser signal and your application runs faster.

Time Series

These blocks perform operations on data histories. You can use these blocks in conjunction with the Float Variable block on the Objects palette, which stores data histories. For more information, see “Float Variable” on page 86.

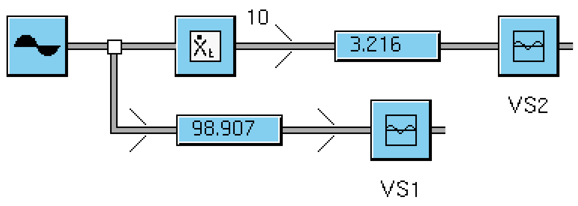
You can use these blocks in a specific diagram or in a generic template. For more information, see “Event Detection Diagrams” on page 52.

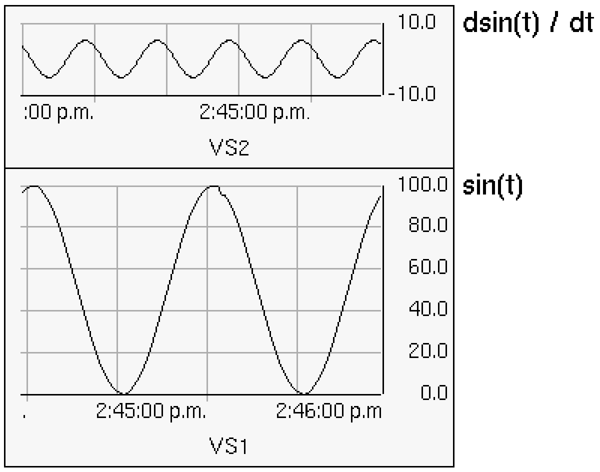
Rate of Change

Computes the instantaneous rate of change of the input value over the specified number of points.

Property	Description
Number of Points	The number of points over which to compute the rate of change.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Number of Points, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

This example computes the rate of change of a sine wave over the last 10 points. Notice that VS2 crosses zero each time VS1 reaches a maximum or minimum value, therefore behaving as dsin(t)/dt:



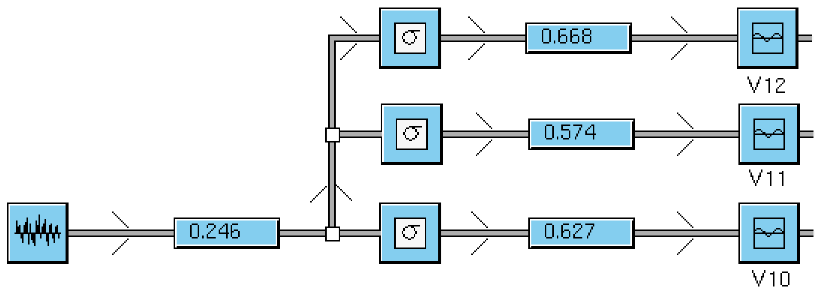


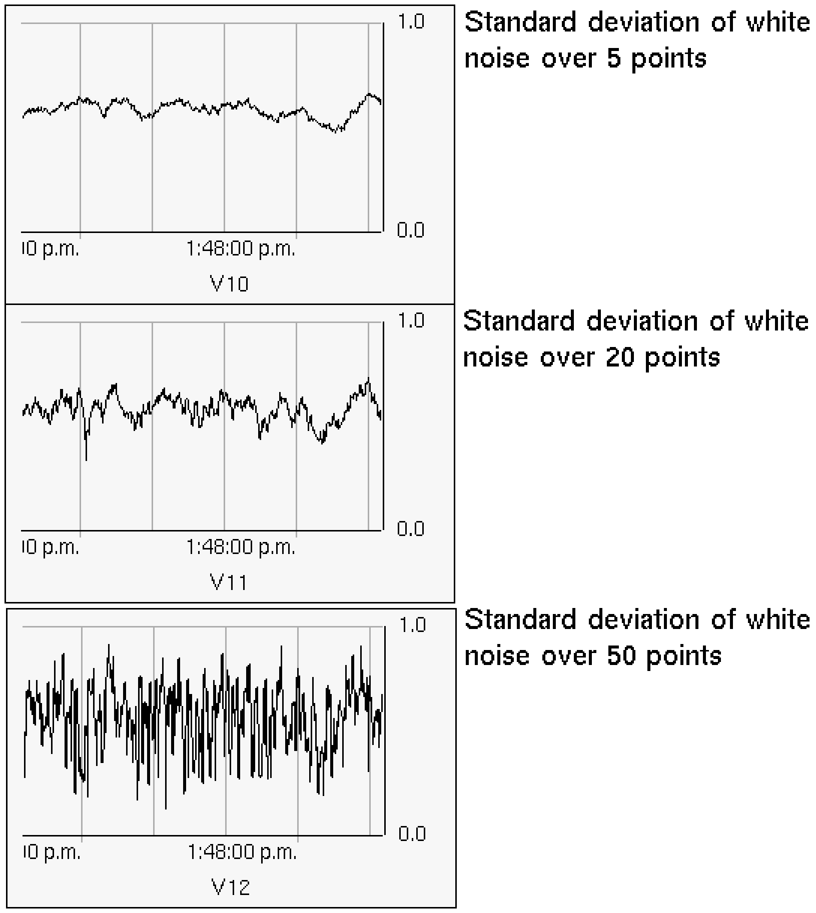
Standard Deviation

Computes the variance of the input value over the specified number of points.

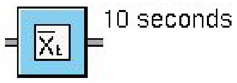
Property	Description
Number of Points	The number of points over which to compute the standard deviation.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Number of Points, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

In this example, V10 is the standard deviation of the input white noise calculated over the last 5 points, V11 represents the standard deviation over the last 20 points, and V13 represents the standard deviation over the last 50 points.




1.0
0.0
0 p.m.
1:48:00 p.m.
V10
Standard deviation of white noise over 5 points
1.0
0.0
0 p.m.
1:48:00 p.m.
V11
Standard deviation of white noise over 20 points
1.0
0.0
0 p.m.
1:48:00 p.m.
V12
Standard deviation of white noise over 50 points

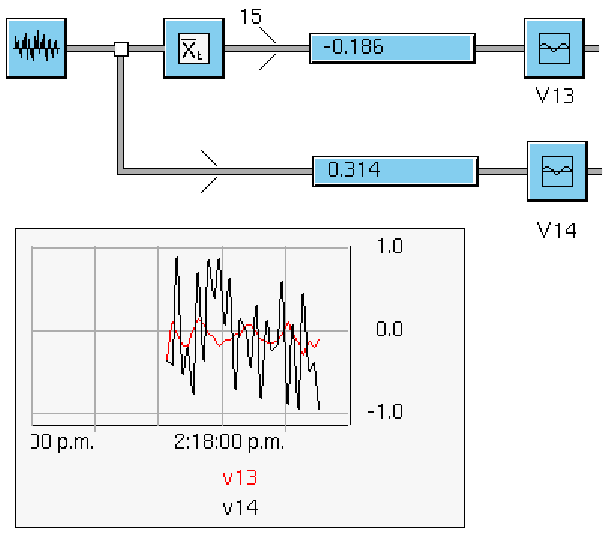
Moving Average



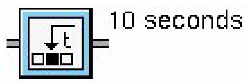
Computes the moving average of the input value over the specified history, specified as a time interval.

Property	Description
Lookback (seconds)	The number of seconds of history over which to compute the moving average.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Lookback, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

In this example, the Moving Average block calculates the moving average of a white noise signal over the last 15 seconds (red) and plots it against the original signal (black):



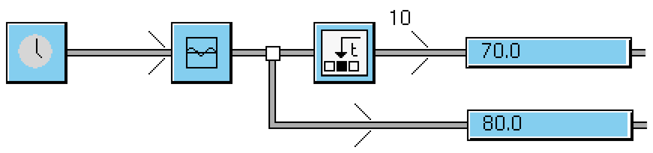
Fetch History Value



Fetches the value of the Float Variable block connected at the input, at the specified time in history. The Float Variable block must set its history-keeping-spec to keep enough history for the block to operate.

Property	Description
Lookback (seconds)	The number of seconds in the past at which to fetch a value.
Override Default	Whether to listen for changes in the domain object property specified in the Property Name. When enabled, this option updates the value of the Lookback, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv.

This example fetches the value of the Float Variable block as of 10 seconds ago. The value originates from a Real Time Clock block; thus, the fetched value is actually that of the Float Variable block as of 10 seconds ago. The history-keeping-spec of the Float variable is set to keep history with maximum age of data points = 15 seconds.



Rate of Change 2

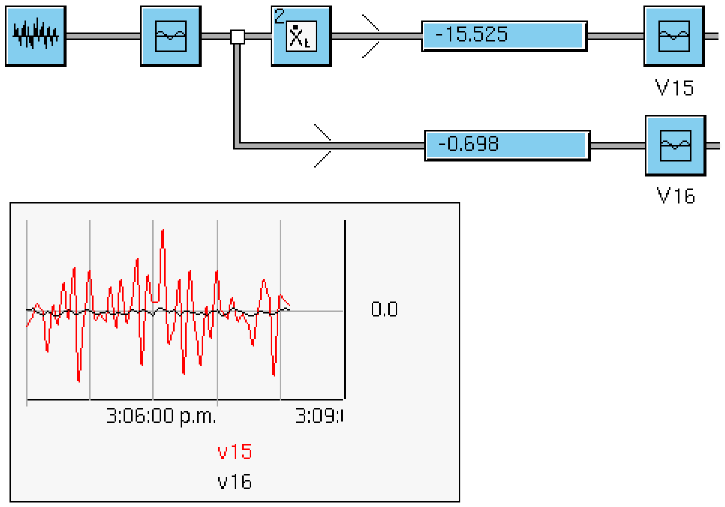
Computes the instantaneous rate of change of the input value over the specified time window. This block must be connected at the output of a Float Variable block. The Float Variable block must set its history-keeping-spec to the start time of the block.

The block can perform the calculation as a rate per second, minute, or hour. To specify the time window, specify the start time as the starting point in history and the end time as the ending point in history. For instance, if you want to calculate the rate of change of the input value over 10 minutes, beginning 5 minutes ago, specify the start time as 10 minutes and the end time as 5 minutes. If you want to calculate the rate of change over the last 3 minutes, specify start time as 3 minutes and the end time as 0.

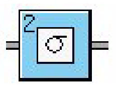
Property	Description
Start Time	The time in history at which to start computing the rate of change.
End Time	The time in history at which to end computing the rate of change.
Rate Per	The time period to use for computing the rate of change. The options are seconds, minutes, or hours.
Override Default Start Time	Whether to listen for changes in the domain object property specified in the associated Property Name. When enabled, this option updates the value of the Start Time, using the specified property value, whenever the property value changes.
Override Default End Time	Whether to listen for changes in the domain object property specified in the associated Property Name. When enabled, this option updates the value of the End Time, using the specified property value, whenever the property value changes.

Property	Description
Override Default Rate Type	Whether to listen for changes in the domain object property specified in the associated Property Name. When enabled, this option updates the value of the Rate Type, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv. You specify a Property Name for each override option that is enabled.

This example calculates the rate of change of the input white noise signal over 10 minutes (red), beginning 5 minutes ago, against the original signal (black):



Standard Deviation 2

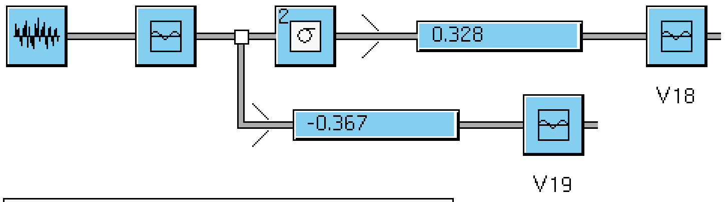


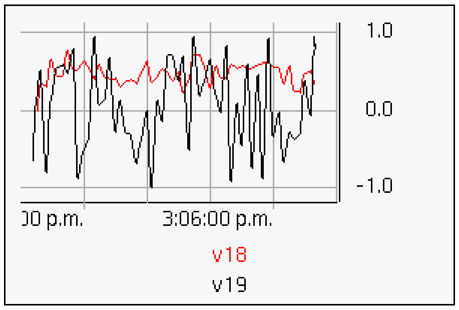
Computes the variance of the input value over the specified time window. This block must be connected at the output of a Float Variable block. The history-keeping-spec of the Float Variable block must be set to the start time of the block.

To specify the time window, specify the start time as the starting point in history and the end time as the ending point in history. See the description of Rate of Change 2 for examples.

Property	Description
Start Time	The time in history at which to start computing the rate of change.
End Time	The time in history at which to end computing the rate of change.
Override Default Start Time	Whether to listen for changes in the domain object property specified in the associated Property Name. When enabled, this option updates the value of the Start Time, using the specified property value, whenever the property value changes.
Override Default End Time	Whether to listen for changes in the domain object property specified in the associated Property Name. When enabled, this option updates the value of the End Time, using the specified property value, whenever the property value changes.
Property Name	The property name of an internal datapoint defined for a domain object in a process map. The property name can use dot notation to refer to embedded datapoints of the object, for example, fo2.o2-sensor or f_101.pv. You specify a Property Name for each override option that is enabled.

This example calculates the standard deviation of the input white noise signal over 10 minutes (red), beginning 5 minutes ago, against the original signal (black):





User Defined Procedures

You use these blocks with the Generic Action block.

Generic Action Block Method

Provides a template for creating a method that the Generic Action block can execute. The method appears in the dropdown list of the User Defined Procedure property of the Generic Action block. When defining a generic method, you must also create a method declaration from the G2 toolbox.

For details, see "Generic Action" on page 35.

Generic Action Block Procedure

Provides a template for creating a procedure that the Generic Action block can execute. The method appears in the dropdown list of the User Defined Procedure property of the Generic Action block.

For details, see "Generic Action" on page 35.

Index