G2 Data Point Manager

User's Guide

Version 2.3 Rev. 0

G2 Data Point Manager User's Guide, Version 2.3 Rev. 0

May 2007

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

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

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

Part Number: DOC010-230

Contents

Preface

Describes this document and the conventions that it uses.

About this Guide

This guide describes the G2 Data Point Manager (GDPM) module. This module provides functionality for managing external datapoints.

Audience

This guide is for G2 developers who want to customize applications, using a set of standard application programmers' interface (API) procedures and methods, and built-in classes. It assumes familiarity with the G2 procedure language.

Conventions

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

Typographic

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

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

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

Procedure Signatures

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

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

Related Documentation

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`

1

Introduction to the G2 Data Point Manager

Describes the G2 Data Point Manager (GDPM) module, which provides functionality to manage external datapoints.

Introduction

G2 Data Point Manager (GDPM) provides functionality to configure, log, replay, and simulate datapoints, typically related to external sensors such as temperature, pressure, and flow. These external values are represented in GDPM as external datapoints and obtain their values typically via an OPC or PI interface and bridge. For example, to provide connectivity with DCS systems, you could use the G2 OPCLink and the G2-PI Bridge.

Within the G2 application, these external sensors (or referred as external datapoints in GDPM) propagate their value to internal datapoints that might be associate to equipment on process maps or other internal software sensors and analyzers. An internal datapoint can be stand-alone and related to a domain object graphically or it can be embedded within a domain object. Domain objects typically represent sensors or equipment, such as pumps, valves, or heaters.

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

Event detection and diagnostics analysis is beyond the scope of GDPM. For information on building graphical event-detection logic, see the *G2 Event and Data Processing User's Guide*. For information on building diagnostics logic, see the *SymCure User's Guide*.

G2 Data Point Manager (GDPM) provides functionality for:

- Configuring external datapoints from CSV files or manually through dialogs.
- Logging external datapoint values.
- Replaying external datapoint values.
- Simulating external datapoint values.
- Creating custom data sources.

Loading GDPM

To use the GDPM module, you must load or merge in `gdpm.kb`, which is located in the `g2i\kbs` directory.

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

2

Module Settings

Describes the G2 Data Point Manager (GDPM) module settings.

Introduction

The gdpm-module-settings object inherits GFR module settings. Upon startup, GFR locates one module settings object as the active setting, which is typically the instance in the highest level module. The active module is determined when G2 is started. Several APIs take the active module settings object into account during execution.

gdpm-module-settings

Manages system configurations for the GDPM module.

Class Inheritance Path

gfr-module-settings, object, item

Attributes

Attribute	Description
start-heartbeat-manager-on-startup	Whether to log datapoint values according to a schedule rather than each time the value changes.
Allowable values:	truth-value
Default value:	true
max-hearbeat-interval-in-minutes	The maximum value that any datapoint can use for the Default Heartbeat, in minutes.
Allowable values:	float
Default value:	10.0
heartbeat-scan-interval-in-minutes	The interval for logging external datapoints, in minutes, when the heartbeat manager is enabled.
Allowable values:	float
Default value:	1.0
comment-handler	The procedure to use for processing comments in the CSV file used for creating external datapoints. The last column in the file is for comments.
Allowable values:	symbol
Default value:	gdpm-dp-default-comment-handler

3

Configuring External Datapoints

Describes how to configure external datapoints.

Introduction

One of the most tedious tasks in building an intelligent fault management application is creating the hundreds and maybe thousands of G2 variables corresponding to the monitored tag variables in your external DCS system.

You use the G2 Data Point Manager (GDPM) module to create external variables for each DCS tag variable—automatically—using CSV files. You simply export to a file the OPC or PI configuration data for the tag variables you want to manage, specify a name for each external variable, and specify the interface type and datapoint type. GDPM reads the file and automatically creates variables of the correct type.

When creating external datapoints from a CSV file, you also have the option of automatically linking the external datapoints to internal datapoints in a process

map. To do this, for each external datapoint in the CSV file, you provide the name of an internal datapoint that will get its data from the external datapoint. You express the internal datapoint name, using **dot notation**, which concatenates the domain object name with its internal datapoint, for example, f-1001.pv where f-1001 is an object and pv is an attribute of f-1001. When GDPM reads the file, it automatically links the external and internal datapoints.

When creating the datapoints from a file, you can configure limits, targets, and deviations to perform data validation of external datapoint values. When a datapoint value is out of range, a data validation message occurs in the Messages browser.

You can configure engineering units for external datapoints in the CSV file. For details, see "Configuring External Datapoint Units in the CSV File" on page 25.

Once the datapoints have been created from the CSV file, you can change them through the external datapoints properties dialog, as needed.

In addition to creating external datapoints from a CSV file, you can create external datapoints from a palette.

You can also configure logging for external datapoints. For details, see "Configuring Datapoints for Logging" on page 30.

Creating External Datapoint Configuration Files

The CSV file that configures external datapoints contains the following information. The order of the information is preconfigured. Only some of the information is required; the rest is optional.

- Datapoint Name — A unique name for the external datapoint.
- Datapoint Server Information — Information about how the external datapoint is configured in the server, which includes:
  - Default Update Interval — How often to update data from the DCS system.
  - Datapoint Tag Type — The type of DCS tag variable the external datapoint represents.

  - Datapoint Type — The value type for the external datapoint. The options depend on the type of external system.
  - Datapoint Units — The engineering unit that the external datapoint uses. You can specify any of the built-in engineering unit or a user-defined synonym.
  - Related Process Map Datapoint Names — The name of an internal datapoint that gets its data from the external datapoint. If you do not want to associate external datapoints with internal datapoints when you create the external datapoints, you can leave this slot empty.
- Data validation limits, targets, and rates, which perform data validation on the external datapoint values. Data validation is optional.
- DCS configuration information for the external OPC or PI system that provides the external data.

You can configure the optional information in the external datapoints configuration CSV file, or you can configure it manually for individual internal datapoints in the process map.

Configuring the External Datapoint Name

You use the external datapoint name as the source datapoint of an internal datapoint in a process map. The external datapoint name must be a symbol, with no spaces.

To configure the external datapoint name:

➔ In the CSV file that configures external datapoints, configure the Datapoint Name column, the first column, to be a unique symbol.

Configuring the Default Update Interval

By default, external datapoints update their data whenever the value in the DCS system changes. To update external datapoints at regular intervals, you can configure the default update interval. The value is any time interval, such as 1 hour and 30 minutes or 1 minute.

Configuring the Datapoint Tag Type

Each external datapoint represents a DCS tag variable for a sensor or controller. For sensors, the datapoint tag type is pv, which represents the process value. Controllers can have the following datapoint tag types: pv, sp, which represents the controller setpoint, op, which represents the controller output, and mode, which represents the controller mode.

By configuring the type of datapoint tag type, you see only external datapoints of the required type when manually configuring the source datapoint of an internal datapoint.

Configuring the Datapoint Type

The external datapoint must define a data type, which depends on the external DCS system you are using:

- For OPC variables, the options are:
  - float
  - integer
  - logical
  - text
- For PI variables, the options are:
  - real
  - integer
  - digital

When you create the external datapoints, GDPM creates a datapoint of the proper class, based on the specified data type.

To configure the external datapoint type:

➔ In the CSV file that configures external datapoints, configure the Datapoint Type column to be one of the types listed above, depending on your DCS system.

Configuring the Datapoint Units

For information on configuring the datapoint units, see Configuring External Datapoint Units in the CSV File.

Configuring the Related Internal Datapoint

If you know ahead of time which internal datapoints should obtain their values from external datapoints, you can configure this information in the external datapoints configuration file. This step requires that you have already created and configured your process map.

If you do not have this information at the time that you create the external datapoints configuration file, you can configure each internal datapoint manually to refer to a particular external datapoint as the source datapoint.

Of course, if you have this information ahead of time, you will save time configuring this information from the CSV file. On the other hand, you might be creating your application in stages, adding domain objects to the process map incrementally as you manage more and more of your external process. You must decide which approach works best for your application. You can also use a combination of both approaches by automatically configuring some internal datapoints and adding others later.

To relate external datapoints to internal datapoints:

➔ In the CSV file that configures external datapoints, configure the Related Process Map Datapoint Names column to be the name of the internal datapoint that gets its data from the particular external datapoint.

When referring to the internal datapoint name, you must use dot notation, which concatenates the domain object name and its internal datapoint, using a period as a separator. For example, to relate the external datapoint named f-1001-external to the pv datapoint for the f-1001 flow sensor, you would use f-1001.pv.

Configuring Data Validation

In the CSV file that configures external OPC float and PI real datapoints, you can set up limits, targets, and rates for validating external data as it arrives. You can perform three types of data validation:

This type of data validation...	Validates external datapoint values based on...
Detected limit	Minimum and maximum limits.
Detected target	Minimum and maximum targets, based on a value.
Detected rate	Minimum and maximum rates, based on a time interval.

When an external data value does not conform to the specified data validation limits, targets, and/or rates, GDPM generates an operator message, which

indicates the reason the value is invalid. You can view these messages in the Message Browser. .

Just as you can relate external and internal datapoints manually for individual datapoints in a process map, you can configure data validation for internal datapoints manually.

You cannot configure data validation for integer, logical, text, or digital datapoint types.

To configure data validation:

➔ In the CSV file that configures external datapoints, configure some or all of these columns for a particular external datapoint, where the Enabled options are true or false:

Column	Value
Min-Max Enabled	true or false
Min-Max High-High	High-high limit value
Min-Max High	High limit value
Min-Max Low-Low	Low-low limit value
Min-Max Low	Low limit value
Target Enabled	true or false
Target High-High	High-high target value
Target High	High target value
Target Low-Low	Low-low target value
Target Low	Low target value
Target Value	The target value to which the external datapoint value is compared, relative to the high and low targets.
Rate Enabled	true or false
Rate High-High	High-high rate value
Rate High	High rate value
Rate Low-Low	Low-low rate value
Rate Low	Low rate value
Rate Interval	The rate at which the time interval between external datapoints is compared.

Configuring the DCS Datapoint Data

Depending on the type of DCS system you are using, you need to identify the actual OPC or PI tag variable for which you are creating an external variable. The configuration information varies, depending on your DCS system.

To configure the DCS datapoint data:

➔ Use the last columns in the CSV file to configure the datapoint data for your DCS system.

For example, to configure an OPC tag variable, you would configure these values:

- OPC Item ID
- OPC Access Path

For both OPC and PI tag variables, you can also configure these values in the second-to-last and last columns of the CSV file, respectively:

- High Process Limit — The high limit for external datapoint values.
- Low Process Limit — The low limit for external datapoint values.

Summary of the CSV File Format

The CSV file that configures external datapoints defines each external datapoint in its own row and the configuration information in each column. The following table defines the order of columns, from left to right, where the separators indicate groups of related configuration information.

Column	Description
Datapoint	
Datapoint Name	The name of the external datapoint.
Datapoint Server Configuration	
Default Update Interval	How often to update data from the DCS system. The value is any time interval, such as 1 minute or 1 hour and 30 minutes.

Column	Description
Datapoint Tag Type	The type of datapoint the external datapoint represents. The options are: pv, sp, op, or mode. Pv represents the process value for a sensor or controller; sp represents the setpoint of a controller; op represents the controller output; and mode represents the controller mode. By configuring the type of datapoint tag, GDPM shows you only datapoints of the required type when manually configuring the source datapoint of an internal datapoint.
Datapoint Type	The value type for the external datapoint. The options depend on the type of external system. For OPC variables, the options are: • float • integer • logical • text For PI variables, the options are: • real • integer • digital
Datapoint Units	The engineering unit that the external datapoint uses, for example, C or deg C.
Related Process Map Datapoint Names	The name of the internal datapoint that gets its value from the external datapoint.
Data Validation	
Min-Max Enabled High-High High Low-Low Low	Detected target settings for data validation.

Column	Description
Target Enabled High-High High Low-Low Low Value	Detected target settings for data validation.
Rate Enabled High-High High Low-Low Low Interval	Detected deviation settings for data validation.
Animation	
Animation Enabled	Animation Enabled is currently not supported.
Animation Object Name	Animation Object Name is currently not supported.
DCS Configuration	
OPC Item ID	The ID of the OPC tag variable from which the external variable gets its data.
OPC Access Path	The access path to the OPC tag variable in the external OPC system.
High Process Limit	The high limit for external data. Values above this limit are rejected.
Low Process Limit	The low limit for external data. Values below this limit are rejected.

Creating External Datapoints from a CSV File

Once you have created the CSV file for configuring external datapoints, you create an External Datapoints container that refers to this file. The container also refers to the OPC or PI Interface that it will use to obtain data through the bridge.

You can configure the external datapoints to propagate data when their values change or based on a schedule.

Once you have created the external datapoint configuration, you can create the external variables.

Note In the sections that follow, the menu choices assume that enable-menus-and-toolbars-upon-startup is enabled in the grtl-module-settings object. For more information, see the *G2 Run-Time Library User's Guide*.

Creating the External Datapoints Container

By default, the value of an external datapoint is propagated to its associated internal datapoint whenever the value changes. In some cases, the value of an external datapoint must be propagated to its internal datapoint at regular time intervals, based on a schedule. For example, this situation arises when using a bridge in synchronous mode.

You can configure External Datapoints containers to update, based on event-driven evaluation or based on a schedule. By default, external datapoints update their values based on event-driven evaluation, that is, when the external datapoint value changes. When configuring datapoints to update based on a schedule, you configure the update interval.

To create the External Datapoints container:

1. Choose Project > System Settings > External Datapoints > Manage and click the New button.

   The properties dialog for configuring external datapoints appears.

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

Tip We recommend that you prefix the name with your application name, for example, Myapp External Datapoints.

---

3. Click the arrow to configure the Interface Name to refer to an existing network interface to use for obtaining external data.

4 Configure Value Propagation to determine how you want datapoint values to propagate, based on event-driven evaluation or based on a schedule.

   The default value is Event Driven.

5 If you choose Schedule Driven, configure Update Period to be the number of seconds between updates.

6 Accept the dialog.

To configure the CSV filename, you must be in developer mode.

To configure the CSV filename:

1 Switch to Developer mode.

2 Display the properties dialog for the External Datapoints container you want to configure.

3 Configure the Filename to be a complete path name to the CSV file that contains the external datapoints configuration data.

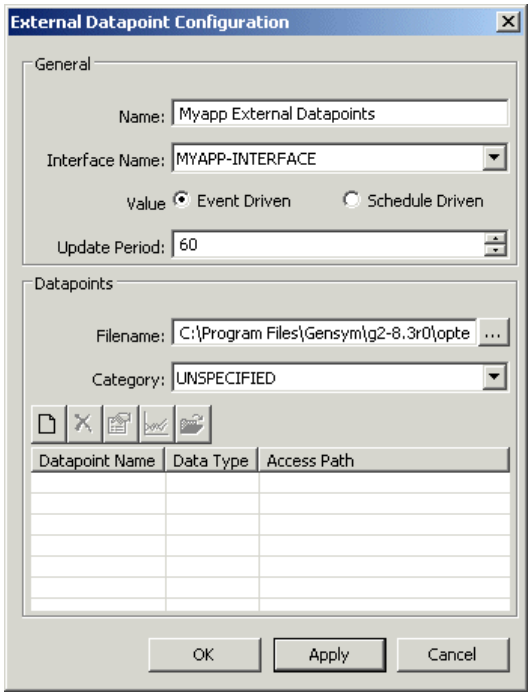
   For information on the format of the CSV file, see "Creating External Datapoint Configuration Files" on page 6.

4 Click OK.

You can create the external datapoints directly from the properties dialog by clicking the Create External Datapoints button. For details, see "Creating and Configuring External Datapoints" on page 16.

Once you have created the external datapoints, the properties dialog displays them in a spreadsheet. You can filter the external datapoints, based on the category.

Here is the properties dialog and associated External Datapoints container named Myapp External Datapoints, which obtains it data through the network interface named myapp-interface and from the specified CSV file. It propagates datapoint values once every 30 seconds, based on a schedule. This figure shows the dialog in Developer mode.



Creating and Configuring External Datapoints

You create the external datapoints from the External Datapoints container properties dialog.

GDPM reads the data from the CSV file and creates one external datapoint for each row in the file. The external datapoints appear on the External Datapoints container detail. The external datapoints are automatically related to the specified internal datapoint as specified in the CSV file.

Each external datapoint has an associated dialog, which contains all the information obtained from the CSV file. OPC float and PI real datapoints have two tabs, the General tab and the Alarm Limit Detection tab. The other datapoint types have only the General tab.

Once the external datapoints have been created, you can configure various additional information, such as the category, description, value translation

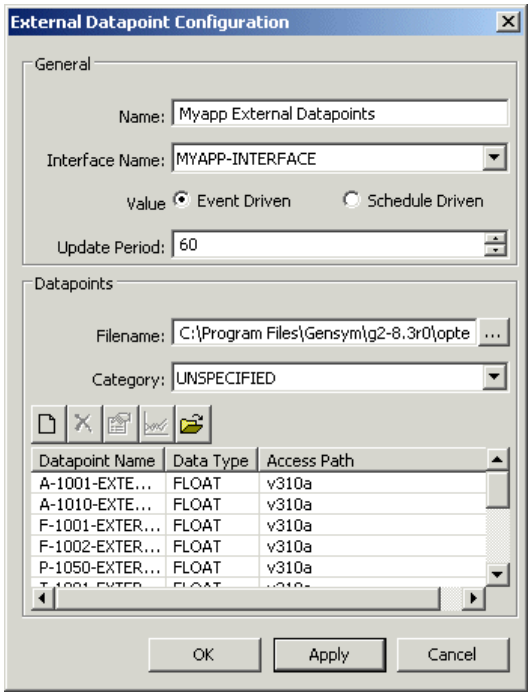
procedure, persistence, and data logging. You can also override values in the properties dialog.

Note Overriding values in the external datapoint properties dialog does not update the corresponding CSV file.

To create and configure the external datapoints:

1 Display the properties dialog for the External Datapoints container and click the Create from File button: 

All external datapoints specified in the CSV file appear in the spreadsheet at the bottom of the properties dialog. For example, here is the result of creating external datapoints from the `f102-external-datapoints-configuration.csv` file:



2 Select a row in the spreadsheet and click the Properties button to display the properties dialog for the external datapoint.

3 Configure the Category to filter external datapoints in the external datapoints configuration dialog.

You can define a category for external datapoints, then filter them in the external datapoint configuration properties dialog, based on the category. For example, you might want to specify the equipment subsystem as a category to show only those datapoints related to that subsystem, such as the fuel system.

You can also categorize the datapoints, based on the type of sensor, such as flow or temperature.

4 Configure the Description to provide a textual description of the external variable.

5 To translate the value, configure the Value Translation to be the name of an existing procedure.

For details, see "Translating External Datapoint Values" on page 22.

6 To filter values from the external system, configure the High Limit and Low Limit, and enable or disable the High Limit Check or Low Limit Check options, as needed.

You can also configure these values in the CSV file. When process limits are enabled, GDPM rejects data values that are above the high limit or below the low limit.

7 Configure the Min Persistence Value and Min Persistence Units to determine how long to keep operator messages in the Message Browser when an alarm limit is detected.

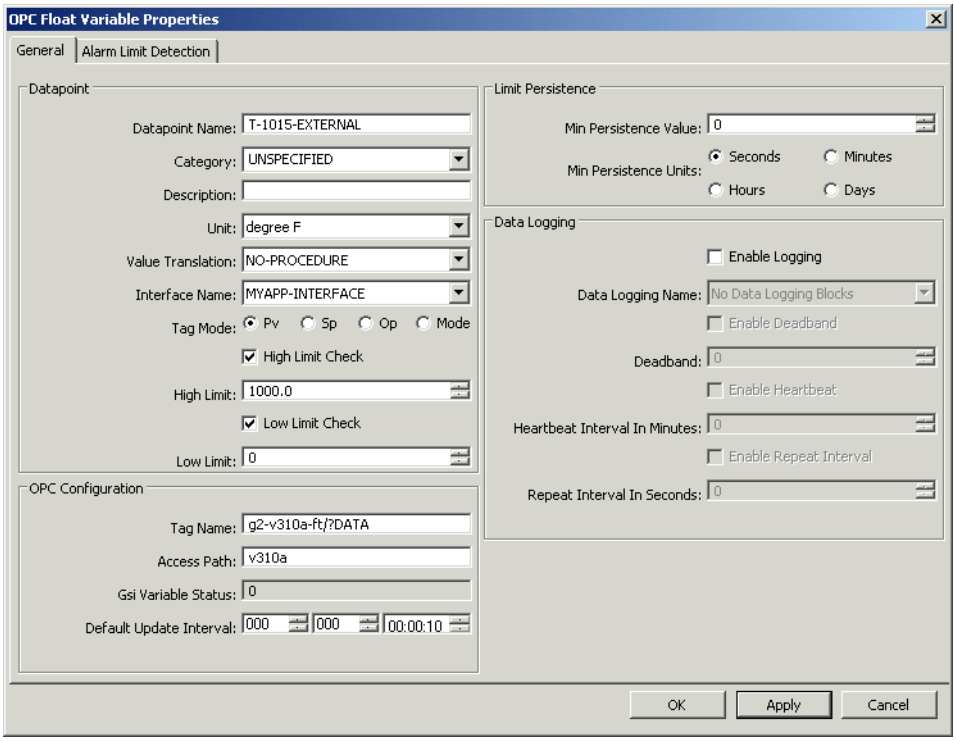
The default value is 0 seconds, which keeps the alarm limit detection message until the operator deletes it.

8 To log external datapoint values, configure Data Logging, as needed.

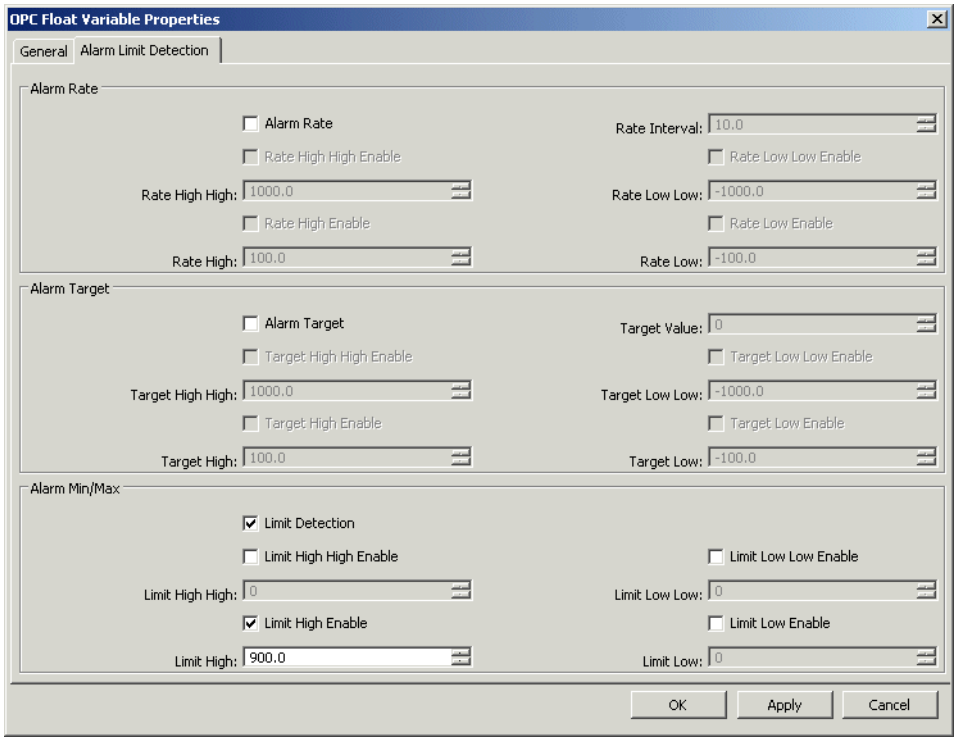
For details, see "Configuring Datapoints for Logging" on page 290.

9 For OPC float and PI real datapoints, click the Alarm Limit Detection tab and override the rate, target, and min-max limits for data validation, as needed.

Here is the General tab in the properties dialog for the external datapoint named t-1015-external, which is a type of OPC float. The Category has been specified as temperature. The OPC configuration identifies the tag variable in the external OPC system. The GSI Variable Status indicates the connection status of the external datapoint to the network interface, where 0 means inactive, 1 means in transition, 2 means ok, -1 means timeout, and -2 means error. The datapoint sets high and low process limits, and it keeps history for one day.



Here is the Alarm Limit Detection tab, which configures Alarm Limit Detection for data validation.



For an example of limit detection, see “Viewing Data Validation Alarms” on page 307.

Manually Relating External Datapoints

When you initialize process maps, GDPM automatically relates all internal datapoints to their source external datapoints, where an external datapoint can be the source datapoint for one or more internal datapoints. When you uninitialize process maps, GDPM de-links all internal and external datapoints.

You might want to manually relate an individual external datapoint to its associated internal datapoint(s), either to re-link the datapoints after they have already been linked or to link them after they have been de-linked. Manually relating external datapoints re-links the external datapoint with all of its associated internal datapoints.

To manually relate external datapoints to their internal datapoints:

1. Display the Navigator.
2. Choose Show Detail on an External Datapoints container in the Navigator to show its detail.

The detail contains the individual external datapoints in the container.

3 Choose Relate Sensors and Controllers on an individual external datapoint to relate the external datapoint to all internal datapoints that specify it as the source datapoint.

Creating Individual External Datapoints

In addition to creating external datapoints automatically from a CSV file, you can create individual external datapoints from a palette. You might want to do this if you add a datapoint after you have created them automatically. Alternatively, you might build your application by starting with the domain objects and adding the external datapoints individually later.

Note When creating individual external datapoints manually, be sure to configure the Source Datapoint of the internal datapoint that gets its values from the external datapoint you create.

To create individual external datapoints:

1 Create an External Datapoints container.

2 Display the properties dialog for the container and configure its properties.

For details, see "Creating the External Datapoints Container" on page 14.

3 Click the New button in the Datapoints area.

A dialog for choosing the external datapoint class appears.

4 Choose a class and click OK.

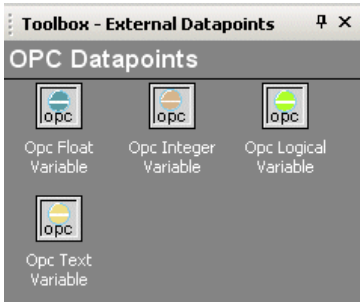
The properties dialog of the external datapoint appears.

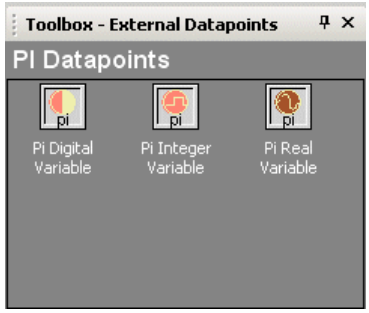
5 Configure the properties of the external datapoint manually.

For details, see "Creating and Configuring External Datapoints" on page 16.

You can also create individual external datapoints and manually add them to the External Datapoints container detail. To do this, show the detail of the External Datapoint container through the Navigator, create external datapoints from the

OPC Datapoints and PI Datapoints palettes in the External Datapoints toolbox, and place them on the detail. Here are the external datapoints palettes:





Translating External Datapoint Values

You can translate values for external datapoints by writing a custom procedure that takes a value, translating the value, then assigning the resulting value to the corresponding internal datapoints. The original external datapoint value is unchanged.

The signature for the translation procedure is:

my-translate-value-procedure
(*item*: class grtl-external-datapoint, *val*: value, *collection-time*: quantity)
–> *new-value*: value, *new-collection-time*: quantity

where:

- *item* is the external datapoint that received the value.
- *val* is the new raw value from the external datapoint.
- *collection-time* is the collection time of the new value.

The procedure returns these values:

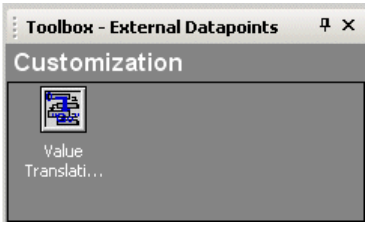
- *new-value* is the new translated value.
- *new-collection-time* is the collection time, if updated by this procedure.

Once the procedure has been defined, you can specify the translation procedure for any external datapoint by configuring the Value Translation Procedure in the properties dialog for the external datapoint. The dropdown list contains all the procedures that are a subclass of grtl-value-translation-procedure. Only procedures that are cloned from the palette appear in the dropdown list.

Translating external datapoint values requires knowledge of G2 and is, therefore, a developer task.

To translate external datapoint values:

1 Choose View > Toolbox - External Datapoints and display the Customization palette:



2 Create a Value Translation Procedure from the palette and place it within your application module.

3 Modify the procedure to perform the type of translation you want to occur.

Defining a procedure require knowledge of G2 and is an advanced feature. See the *G2 Reference Manual*.

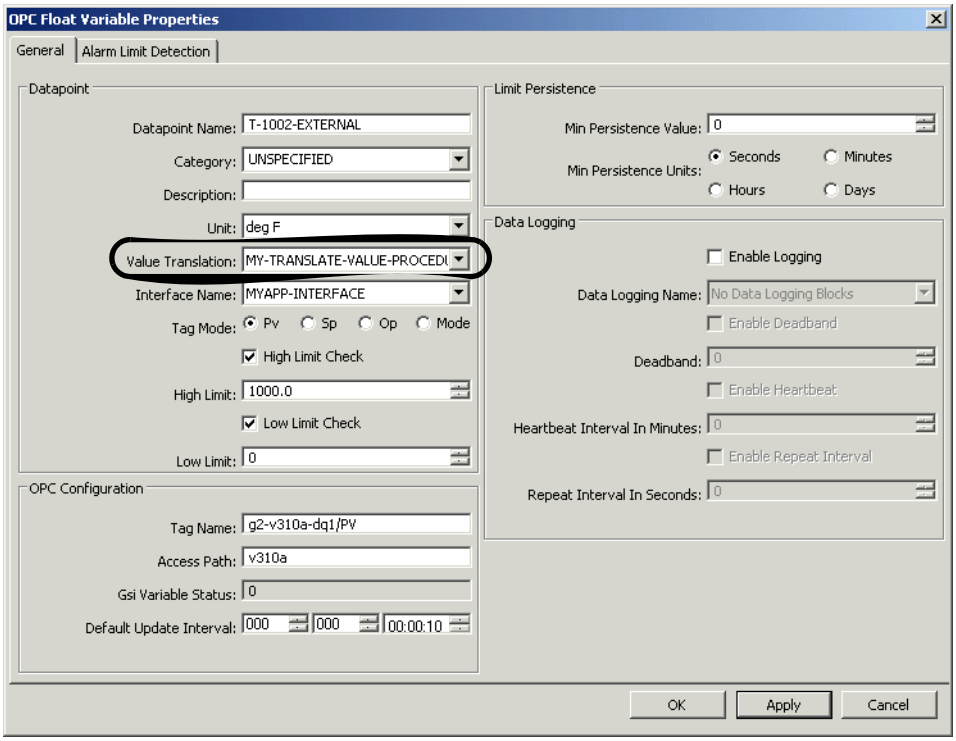
Be sure to change the name of the procedure to a unique name.

4 Display the properties dialog for the external datapoint whose value you want to translate and configure the Value Translation Procedure.

If you do not want any value translation to occur, choose no-procedure, which is the default for all external datapoints.

This dialog shows how to configure the Value Translation Procedure procedure for an OPC Float datapoint:

MY-TRANSLATE-VALUE-PROCEDURE

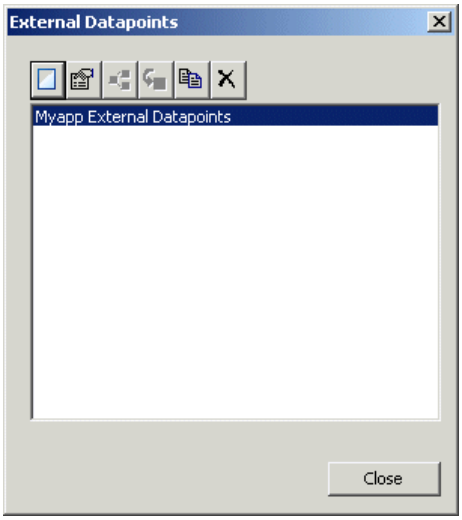


Managing External Datapoints

To manage external datapoints:

1. Choose Project > System Settings > External Datapoints.
2. All External Datapoints containers appear in the submenu.To configure the properties of an external datapoint container, choose one from the External Datapoints submenu.
3. To display a dialog for managing all external datapoint configuration objects, including displaying the detail, choose Manage.

Here is the External Datapoints Manage dialog:



Working with Engineering Unit Conversions

You work with engineering unit conversions in various ways in an application.

To work with engineering unit conversions, merge in `geuc.kb` from the `g2i kbs` directory.

Configuring External Datapoint Units in the CSV File

You can import units from the CSV file used for configuring external datapoints. You configure the Datapoint Units in the column to the right of the Datapoint Type column in the CSV file. The datapoint units that you configure are equivalent to the field units that domain objects use as sensor values.

You can specify any of the built-in engineering units or a synonym.

If you specify an engineering unit that does not exist, GDPM automatically creates a unit synonym definition and places it in the Undefined-Dimension category in the Unit Synonyms submenu.

Note Sometimes units provided in the DCS system are inaccurate, thus GDPM requires you to enter the units explicitly rather than obtaining them directly from the DCS system.

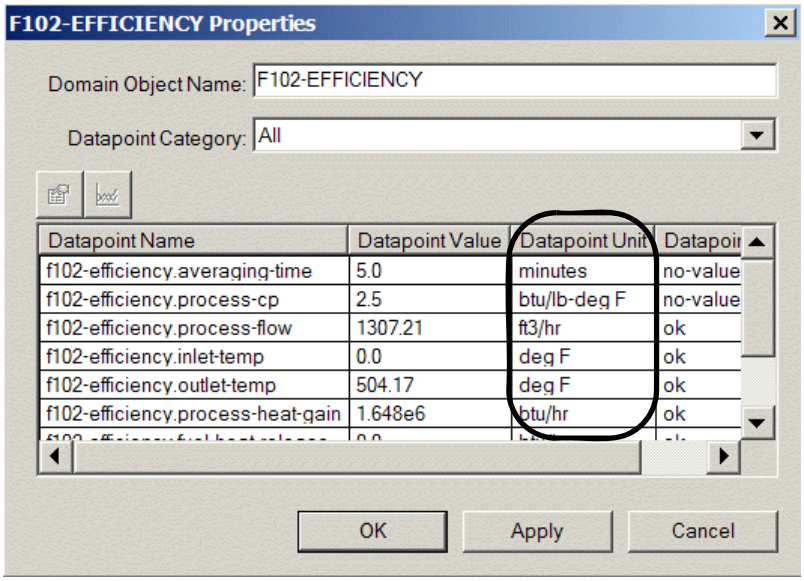
Here is part of an external datapoint configuration file, which defines field units for several datapoints in the metric system:

Datapoint	Datapoint Server Configuration				
Datapoint	Default	Datapoint	Datapoint	Datapoint	Related Process Map
Name	Update Interval	Tag Type	Type	Units	Datapoint Names
t1-dp	10 minutes	pv	float	deg C	t1.pv
t2-dp	10 minutes	pv	float	deg C	t2.pv
p1-dp	10 minutes	pv	float	kg/cm2	p1.pv
f1-dp	10 minutes	pv	float	m3/hr	f1.pv
f2-dp	10 minutes	pv	float	m3/hr	f2.pv
f3-dp	10 minutes	pv	float	m3/hr	f3.pv
a1-dp	10 minutes	pv	float	kj/m3	a1.pv

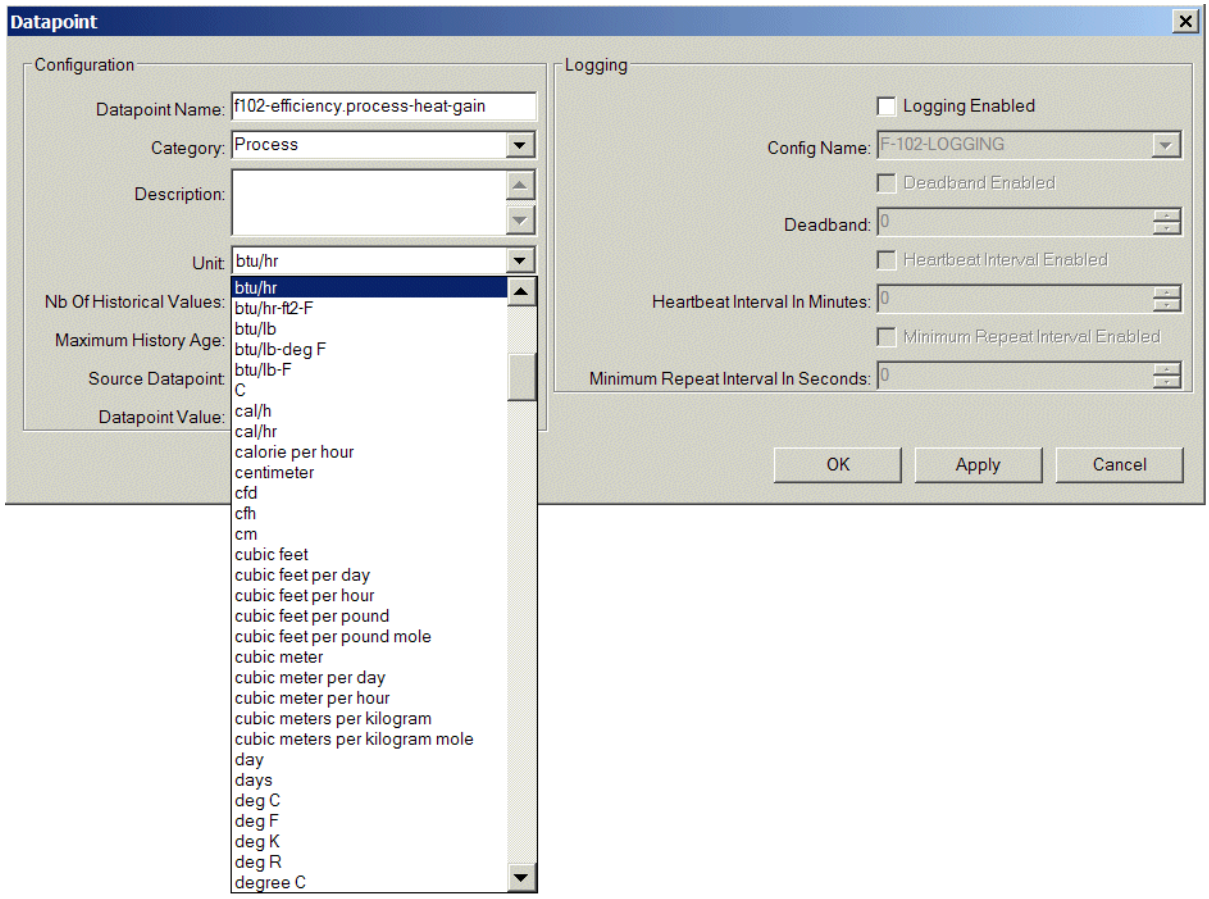
Caution When upgrading older versions of GDPM, you must add the Units column to the CSV file before creating external datapoints. Creating external datapoints from a CSV file that does not include the Units column generates an error.

Configuring Engineering Units for Domain Objects

You can configure engineering units for entering and displaying various event parameters and metrics. For example, here is the properties dialog for the Heater Efficiency derived sensor of a heater, which specifies units for the various internal datapoints of the sensor:



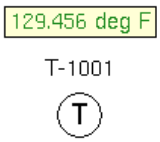
Here is the properties dialog for the calculated heat-gain internal parameter, which shows a dropdown list with some of the available engineering units:



Displaying Engineering Units for Datapoints

Units appear in the datapoint displays for internal and external datapoints.

For example, here is a display that shows the units of a flow sensor:



4

Logging Datapoints

Describes how to log external datapoint values.

Introduction

You can configure your application to log internal or external datapoints as the application runs. By default, GDPM logs data values as they arrive. You can also log data on a schedule. You can filter logged data, based on the rate at which it arrives and the variation of data values.

To log internal datapoints, you create a Data Logging configuration object:

Configuration Object	Description
Data Logging	Logs internal datapoints in a process map to a CSV file.

To configure data logging, you must specify the name of a CSV where data is to be logged. You also specify the name of a process map container object, whose internal datapoints you want to log. When logging is enabled, GDPM logs all internal datapoints in the process map, by default. You can selectively enable and disable logging for individual datapoints, through a spreadsheet. You can also configure logging for individual datapoints.

You can also log internal and external datapoint values to the default Message Browser.

One use of a log file is for data replay. For more information, see Chapter 5, “Replaying Data” on page 37.

Configuring Datapoints for Logging

You can configure logging for internal and external datapoints to:

- Post log messages to the Message Browser.
- Post log messages to a CSV file.
- Log data values each time they change, the default, or on a schedule, which you might do when the values change infrequently.

To configure datapoints for logging, you can assign all internal datapoints in a process map and/or all external datapoints in one or more External Datapoints containers. Once you assign internal and external datapoints for logging, you can configure logging parameters for individual datapoints.

By default, GDPM does not perform data logging. You must explicitly enable data logging to begin logging.

You can also configure datapoints to filter logged values by enabling and configuring these logging parameters:

- Heartbeat Interval, which logs data according to a schedule, rather than each time the value changes. The value cannot exceed the Maximum Heartbeat given in the Data Logging configuration object. The default value is the Default Heartbeat in Minutes given in the Data Logging configuration object.
- Repeat Interval, which allows you to reduce the number of logged datapoints, based on the rate of incoming data. The datapoint logs value is logged each time it changes or once every Repeat Interval, whichever is longer. The data is not logged if the elapsed time from the last logged value is less than the Repeat Interval. You configure this attribute when the rate of incoming data is greater than the rate at which you want to log data.
- Deadband, which only logs a new value if that value differs from the last logged value by more than the deadband.

You can configure logging parameters for all assigned datapoints through a spreadsheet, or you can configure logging for individual datapoints through the individual properties dialogs.

For information on configuring the properties of external datapoints, see Chapter 3, “Creating and Configuring External Datapoints” on page 16.

To configure datapoints for logging:

1. Choose Project > System Settings > Datapoint Logs > Manage and click the New button.

   The properties dialog for configuring data logging appears.

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

   The name must be a symbol, without spaces.

Tip We recommend that you prefix the name with your application name, for example, myapp-logging-configuration.

---

3. To send log messages to the Message Browser, enable the Enable Message Browser Logging option.
4. To send log messages to a log file, enable the Enable CSV Logging option and click the browse button to configure the Filename to be a complete path name to a CSV file to use for logging.

   The Filename can refer to an existing file, or you can specify a new file, in which case GDPM creates the file.

5. To log data on a schedule, enable the Enable Heartbeat Manager option, and configure the Maximum Heartbeat and Default Heartbeat in Minutes.

   Maximum Heartbeat is the maximum value that any internal or external datapoint can use for the Default Heartbeat.

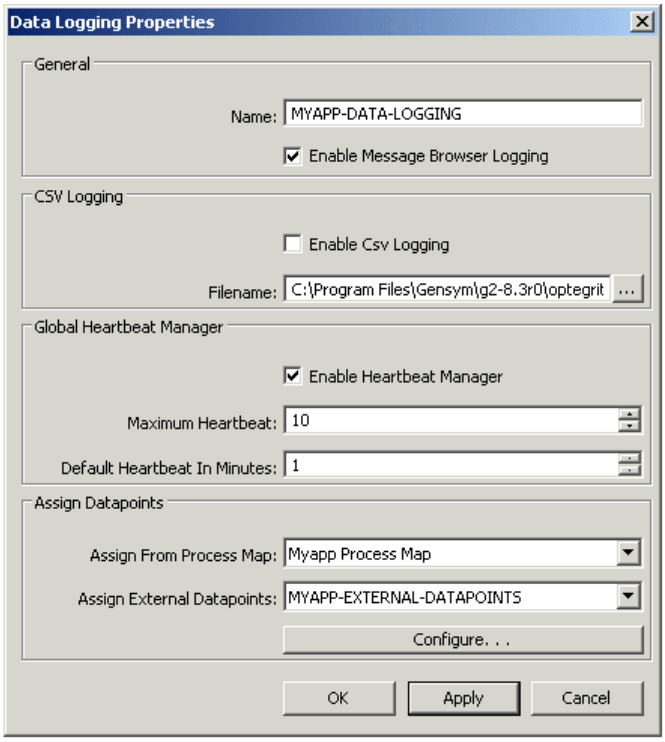
   Default Heartbeat in Minutes is the default heartbeat for each internal or external datapoint, in minutes.

6. To assign internal datapoints for logging, configure the Assign from Process Map to be a process map whose internal datapoints you want to assign.

   You can assign internal datapoints from as many process maps as you need.

7 To assign external datapoints for logging, configure the Assign External Datapoints to be the External Datapoints container whose external datapoints you want to assign.

Here is the properties dialog for the Data Logging configuration object named myapp-data-logging, which logs all internal datapoints in the Myapp Process Map and Myapp External Datapoints to the specified CSV file and to the Message Browser:



8 To assign the internal and external datapoints, click Apply.

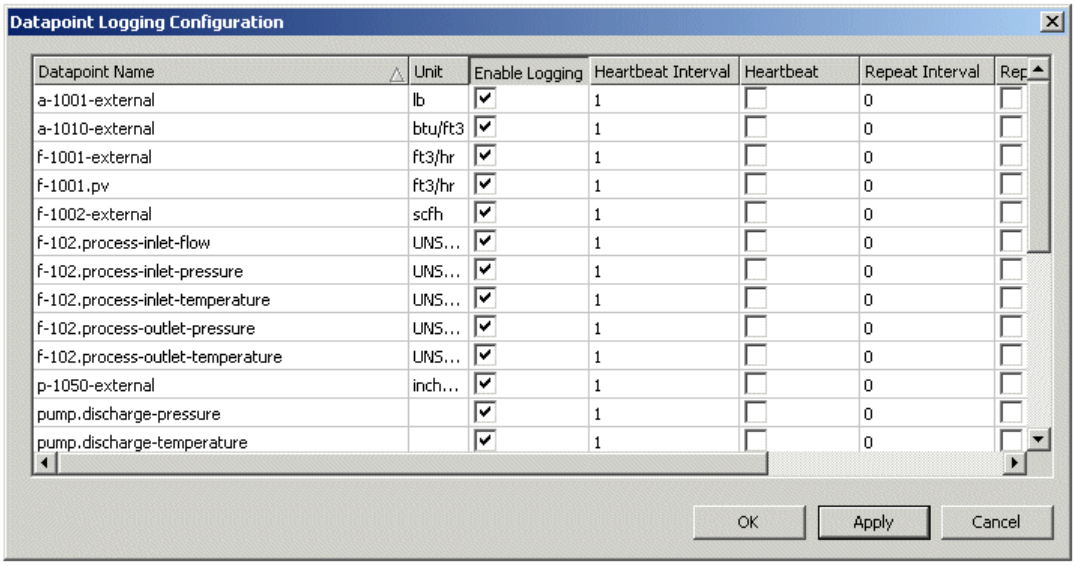
Now that you have assigned internal and external datapoints for logging, you can configure logging parameters for each assigned datapoint through a spreadsheet.

9 Click the Configure button to display a spreadsheet of all assigned internal and external datapoints, then configure the individual logging parameters for each datapoint.

You must specifically enable logging for each datapoint. You can also enable, disable, and configure the Heartbeat Interval, Repeat Interval, and Deadband for each datapoint.

To enable a logging parameter, click the spreadsheet cell labeled Disabled to toggle it to Enabled. To disable it, click the Enabled label to toggle it back to Disabled.

For example, this spreadsheet enables logging for all internal datapoints in the Myapp Process Map and all associated external datapoints:



Configuring logging parameters in this spreadsheet automatically updates the logging specification in the properties dialogs for the individual internal and external datapoints.

Log File Format

The first column in the log file is the time at which the value was logged. Each subsequent column in the log file corresponds with an external or internal datapoint whose value is being logging. Each row represents the logged value for that datapoint.

By default, GDPM appends data to the log file.

Here is a sample log file for the Myapp Process Map application, which logs both external and internal datapoints:

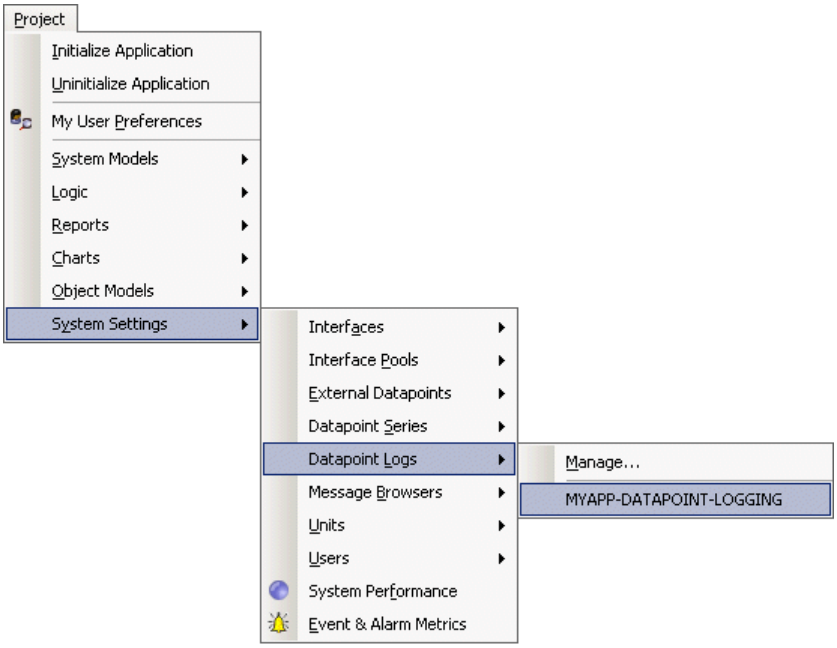
	A	B	C	D	E	F	G	H	I	J
1		FO2-IO-O	FO2-IO-TI	FO2-IO-F	FO2-IO-TI	fo2.T-IN-(	fo2.T	f_101.PV	t_102.PV	fo2.O2-SE
2	0	2.403	550.26	74.949	497.673		550.26	74.949	497.673	2.403
3	0.017	1.976	506.835	71.982	538.293		506.835	71.982	538.293	1.976
4	0.033	1.038	557.284	69.999	539.081		557.284	69.999	539.081	1.038
5	0.05	2.503	580.281	68.685	509.207		580.281	68.685	509.207	2.503
6	0.067	1.073	511.767	67.955	501.302		511.767	67.955	501.302	1.073
7	0.083	1.656	538.692	69.323	513.968		538.692	69.323	513.968	1.656
8	0.1	1.365	542.9	67.153	520.527		542.9	67.153	520.527	1.365
9	0.117	2.327	506.091	69.916	451.553		506.091	69.916	451.553	2.327
10	0.133	1.146	529.965	69.912	516.274		529.965	69.912	516.274	1.146
11	0.15	2.921	512.78	70.465	478.23		512.78	70.465	478.23	2.921
12	0.167	1.562	599.378	73.545	522.112		599.378	73.545	522.112	1.562
13	0.183	1.184	519.731	66.85	477.222		519.731	66.85	477.222	1.184
14	0.2	1.088	512.066	65.537	450.092		512.066	65.537	450.092	1.088
15	0.217	1.95	548.395	72.037	457.349		548.395	72.037	457.349	1.95
16	0.233	1.729	561.626	69.981	455.609		561.626	69.981	455.609	1.729
17	0.25	2.777	597.235	65.945	533.391		597.235	65.945	533.391	2.777
18	0.267	2.541	526.451	71.125	547.95		526.451	71.125	547.95	2.541
19	0.283	1.445	574.82	74.116	471.94		574.82	74.116	471.94	1.445
20	0.3	2.987	524.158	73.742	521.48		524.158	73.742	521.48	2.987
21	0.317	1.108	568.792	73.451	538		568.792	73.451	538	1.108
22	0.333	1.42	530.216	69.27	491.927		530.216	69.27	491.927	1.42
23	0.35	2.51	555.321	70.716	469.735		555.321	70.716	469.735	2.51

Managing Data Logging

To manage data logging:

1 Choose Project > System Settings > Datapoint Logs.

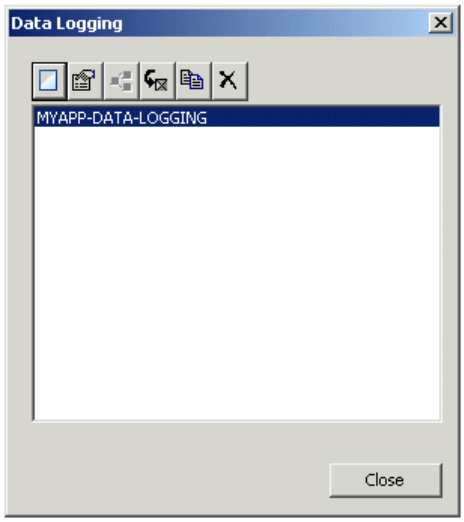
All Data Logging configuration objects appear in the submenu, for example:



2 To configure the properties of a data logging configuration object, choose one from Data Logging submenu.

3 To display a dialog for managing data logging, choose Manage.

Here is the Data Logging Manage dialog:



5

Replaying Data

Describes how to replay external datapoint values.

Introduction

You replay data to:

- Test an application in offline mode.
- Validate control algorithms prior to placing a system online.

You can simulate data for internal datapoints or external datapoints.

You use these configuration objects for replaying data from CSV files:

Configuration Object	Description
Data Replay	Configures the data file(s) to use for replaying data, and controls when to start and stop replaying data.
Data File	Configures various information about the CSV file to use for data replay, depending on the type.

You must configure a Data Replay object to specify the data file. You can replay data from one or more CSV files at once.

You can use datapoint displays to see how internal and external data values are updated. Datapoint displays are available in the G2 toolbox.

When simulating data or when the application is online, GDPM performs data validation, depending on how the external datapoints are configured.

For information on configuring external datapoints for data validation, see "Configuring Data Validation" on page 9.

Creating Data Series

You can create one of two types of data series for data replay, depending on the type of process you want to simulate:

To simulate...	Use this type of data series...	Which sends new values...
A continuous process	Continuous	At regular time intervals, 5 seconds, by default.
A batch process	Differential	At a specified timestamp, based on an acceleration factor.

Creating a Continuous Data Series

To create a continuous data series:

1. Choose Project > System Settings > Datapoint Series > Continuous Data Series > Manage and click the New button.

   The properties dialog for configuring the data series appears.

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

   The name must be a symbol without spaces.

3. Configure the File Name to be a complete path to a CSV file to use for data simulation.

   By default, the first row of the data replay file contains the name of the internal or external datapoint whose values should be replayed, and the second row of the file contains the data.

4. If necessary, configure the Tagname Row and First Data Row to refer to different rows.

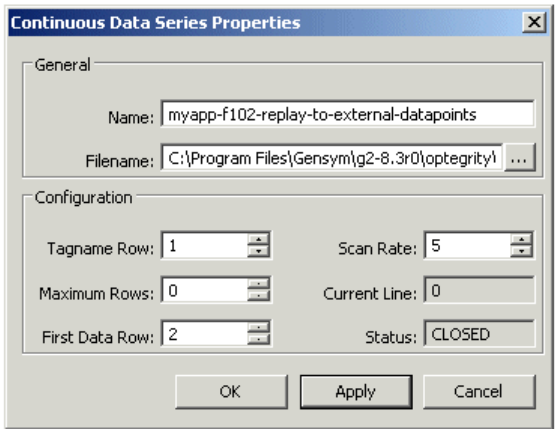
   By default, the data replay file sends a value once every 5 seconds.

5. To increase or decrease the rate at which the data file sends values, configure the Scan Rate, in seconds.

   By default, the data replay file sends all the values in the file.

6. To limit the number of rows of data to send, configure the Maximum Rows.

Here is the properties dialog for a continuous data file named myapp-f102-replay-data-to-external-datapoints:



Creating a Differential Data Series

For a differential data series, you can configure the following attributes, in addition to those you configure for a continuous data series:

- Start Time allows you to skip the beginning portion of the data series up to the first row greater than or equal to the specified time. The start time must use the same time format as the timestamp in the first column of the CSV file.
- Acceleration Factor specifies the speed at which to send data to the specified datapoints, which is 1.0, by default. For example, an acceleration of 20 means that each timestamp specified in the CSV file is divided by 20 to determine the actual time at which the data is sent. For example, if the CSV file specifies a timestamp of 2 minutes (120 seconds), the data is actually sent at 6 seconds (120/20 seconds).
- Time Format indicates how to interpret time values specified in the CSV file. By default, the data series assumes time values are in decimal hours. For example, 90 minutes is expressed as 1.5 hours. You can specify time values, using one of the other formats, including a custom format. If you use a custom time format, you must provide a Time Conversion Procedure that determines how to interpret time values in the CSV file.
- Time Conversion Procedure is the name of a G2 procedure that interprets the time value in the differential data series.

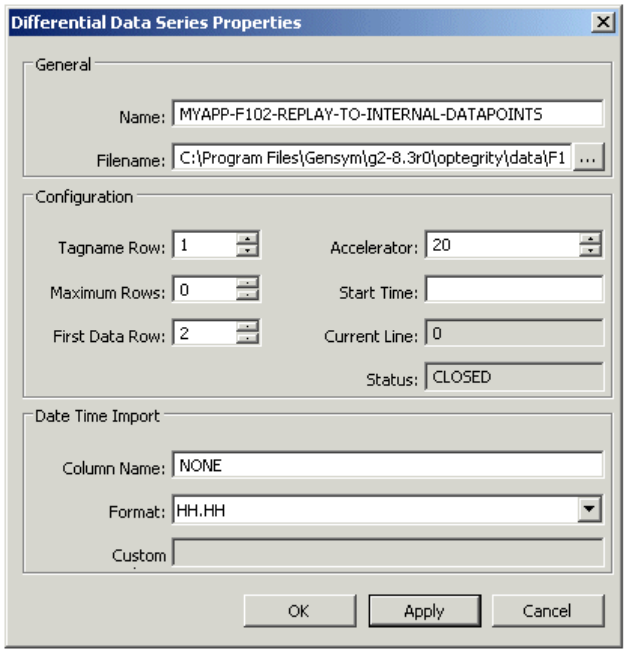
To create a differential data series:

1. Choose Project > System Settings > Datapoint Series > Differential Data Series > Manage and click the New button.

   The properties dialog for configuring the data file appears.

2. Follow steps 2 through 6 under “Creating a Continuous Data Series” on page 39.
3. Configure the additional attributes, described above, as needed.

Here is the properties dialog for a differential data file named myapp-f102-replay-data-to-internal-datapoints. The data file assumes timestamps are expressed in decimal hours, with an acceleration factor of 20.0.



Creating Data Replay Files

The format of a data replay file is similar to the format of a log file, where each column represents data values for an internal or external datapoint, and each row represents a new value. By default, the first row contains the name of the internal or external datapoint, and the second row contains the first row of data.

For differential data series, the first column of the data file provides a timestamp that determines the rate at which to replay the data, expressed in decimal hours. The rate is determined by calculating the difference in time between consecutive rows. The time interval between rows does not need to be evenly spaced. For continuous data files, the first column is ignored.

Tip You can use a log file as a data replay file. For details, see Chapter 4, "Logging Datapoints" on page 29.

To create a data replay file:

1. Create a CSV file in which the first row contains the names of each internal or external variable whose value you want to replay, and the subsequent rows contain the data values to replay.

   To replay external datapoint values, use the names of external datapoints in the External Datapoints container, for example, t-1001-external.

   To replay internal datapoint values, use dot notation to refer to the name of the internal datapoint and its corresponding domain object, for example, t-1001.pv.

2. For differential data files, the first column must be a timestamp.

Here is a partial CSV file for replaying external datapoint values:

	A	B	C	D	E	F
1	T-1015-EXTERNAL	T-1014-EXTERNAL	A-1010-EXTERNAL	F-1002-EXTERNAL	T-1016-EXTERNAL	P-1050-EXTERNAL
2	880.559	867.319	924.36	1513.582	910.896	-185.823
3	884.124	867.128	921.363	1514.142	909.295	-196.09
4	883.222	869.856	927.397	1527.227	906.567	-196.309
5	881.445	869.339	930.449	1521.695	910.683	-180.239
6	879.705	870.293	928.661	1518.973	908.261	-196.178
7	882.302	867.933	920.235	1515.675	909.133	-182.598
8	880.012	868.06	938.024	1515.178	906.819	-183.601
9	881.591	867.024	939.798	1515.43	908.614	-193.986
10	880.036	867.042	925.199	1527.177	907.133	-198.784
11	884.382	868.551	928.61	1516.293	909.524	-193.921
12	883.023	870.725	929.824	1525.676	907.284	-181.704
13	884.181	867.846	922.269	1525.748	910.635	-188.431
14	883.859	869.679	925.713	1521.503	908.859	-180.553
15	880.727	870.73	922.924	1513.038	906.346	-188.007

Configuring Data Replay

To configure data replay, you create a Data Replay configuration object to determine which data series to use for data replay.

To configure data replay:

1 Choose Project > Logic > Datapoint Replay > Manage and click the New button.

The properties dialog for configuring data replay appears.

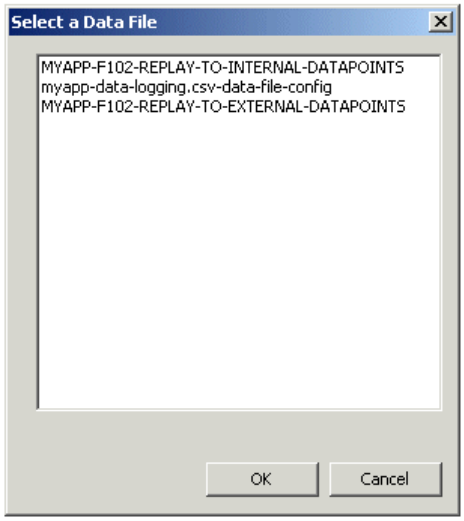
2 Configure a unique Name, which is system-generated, by default.

Tip We recommend that you prefix the name with your application name, for example, Myapp Data Replay.

---

3 Click the Add File button () to display a list of available data files to replay.

The list includes all continuous and differential data files, and all log files that have been created. For example:



4 Select a data file to use for data replay.

You can add as many data series as you want to the dialog. For continuous data series, when data replay finishes replaying from the first data series, it begins replaying data from the next data series, and so on. For differential data series, it simulates data from both data series simultaneously according to the specified time differentials. You can use this feature to create more complex simulations.

5 To remove a data series from the simulation, select a file and click the Remove File button ( ) in the dialog.

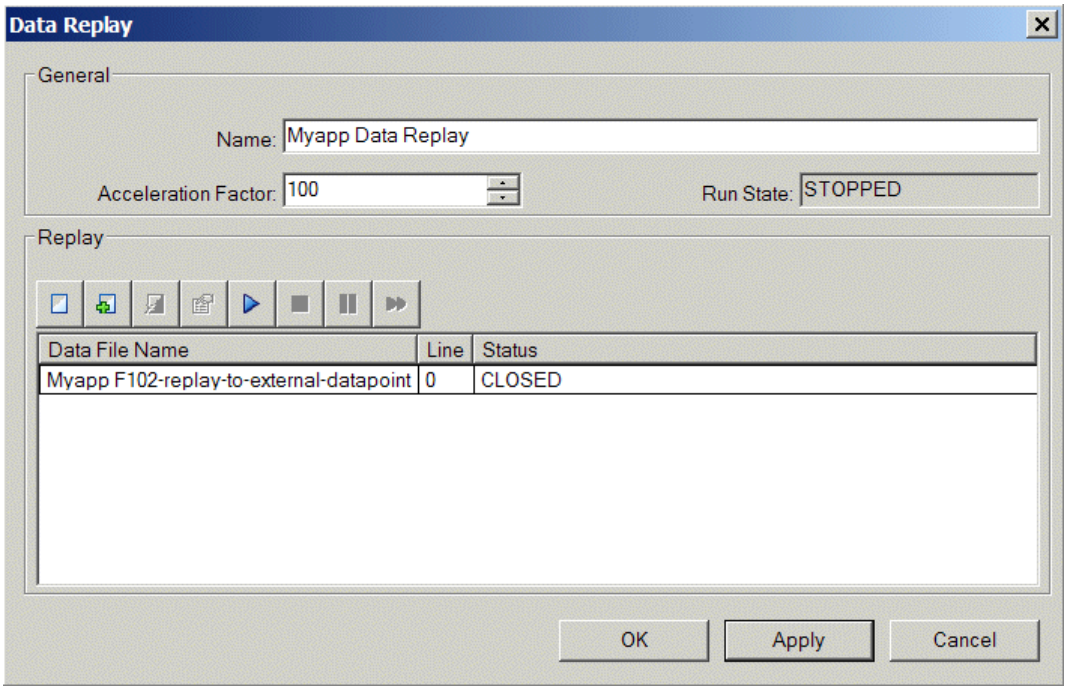
6 To create a new data series, click the New button ( ), choose the type of data series to create, and click OK.

7 Configure the properties of the new data series and add it to the list in the Data Replay dialog.

8 Configure the Acceleration Factor to speed up the rate at which data is sent.

For details, see "Creating Data Series" on page 38.

Here is the properties dialog for the Data Replay configuration object named Myapp Data Replay configured to replay data from a single differential data file that simulates internal datapoint values. The Acceleration Factor is set to 100.0, which, combined with an Acceleration Factor of 20.0 in the data file, means the timestamps in the CSV file are divided by 200.0.



Replaying Data from CSV Files

You control data replay from the Data Replay properties dialog or manage dialog. You can start, pause, and stop the replaying of data at any time.

Before you can replay data, you must create and configure at least one data series, create the associated CSV file, and create and configure Data Replay to use a specific data series.

When replaying data, GDPM detects errors in the associated CSV data series, for example, a bad file name or bad data. For information about the error, see the Message Browser.

To replay data from a CSV file:

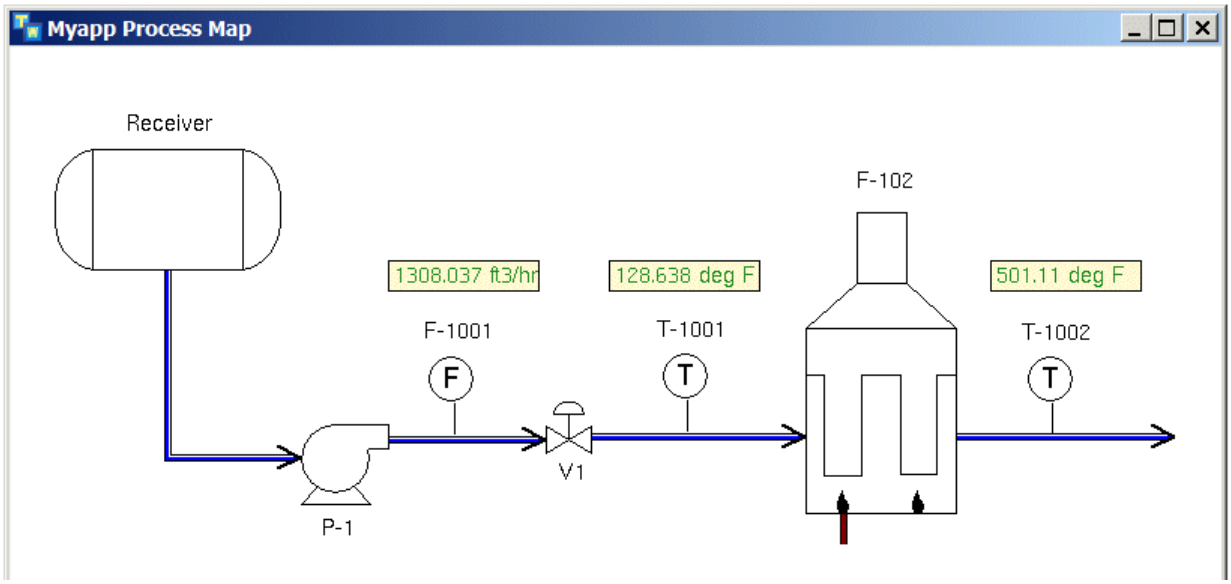
1. After initializing the process map, click the Start button (▶) in the data replay properties dialog.

   You can also click the Start button in the manage dialog. For details, see "Managing Data Replay" on page 49.

   The Data Replay object sends values to the specified internal or external datapoints at the specified time interval and acceleration, depending on the type of data file. Datapoint values update.

2. To pause the simulation, click the Pause button (⏸).
3. To continue the simulation, click the Resume button (⏩).
4. To stop the simulation, click the Stop button (■).

Here is the Myapp process map with datapoint displays that show the values of internal datapoints in the process map after running a simulation:



Displaying Trend Charts of Datapoint Values

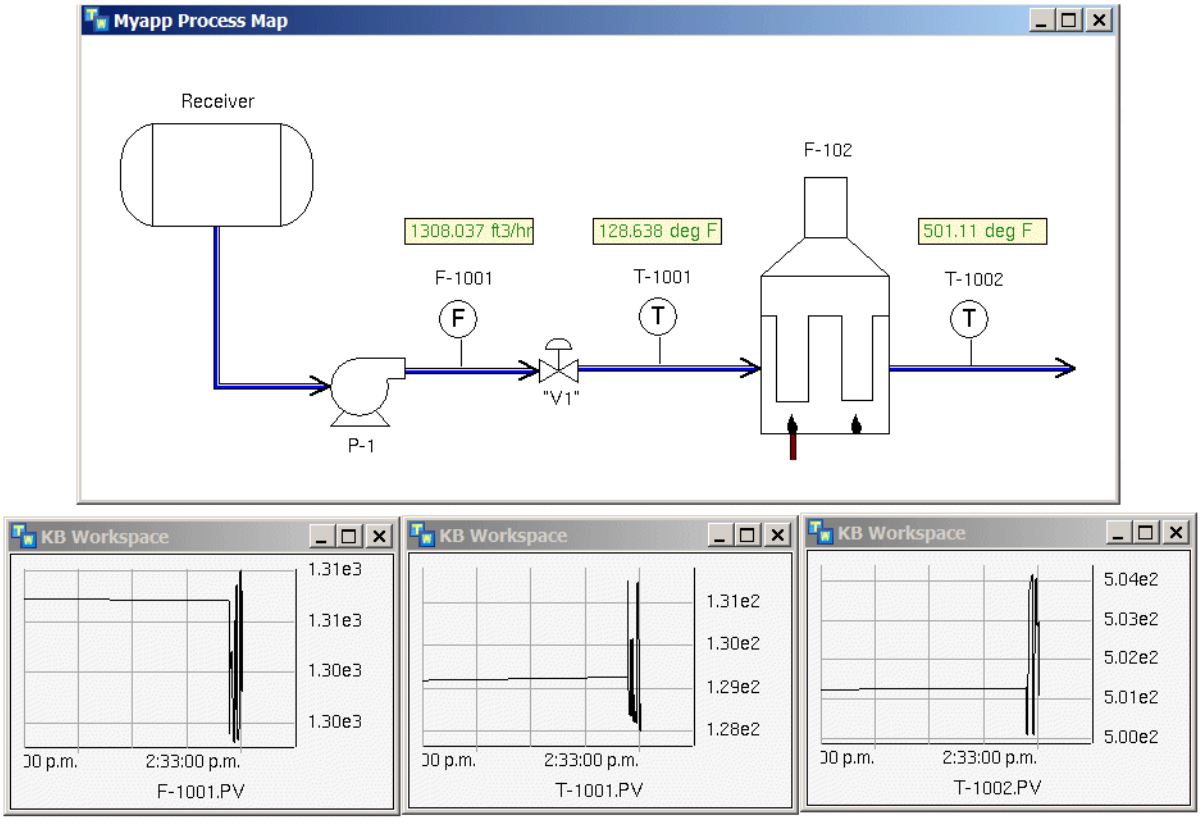
In addition to viewing datapoint values in displays, you can display trend charts of internal datapoint values so you can see a visual representation of the data as it arrives.

To display trend charts of datapoint values:

1. Display the properties dialog for the domain object whose internal datapoint you want to view in a trend chart.
2. Select a datapoint and click the Show Trend button.

Note If you create multiple trend charts, the charts are placed on top of each other. You must move the new chart to uncover the exiting charts.

When the internal datapoint receives values, the trend chart is automatically updated. For example, here is the result of running the simulation for Myapp Process Map with trend charts:



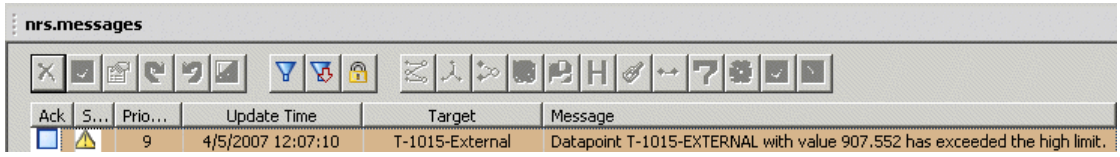
Viewing Data Validation Alarms

When simulating data, GDPM performs data validation on all external datapoints that configure validation limits and targets. If the external datapoint value violates these limits, GDPM generates an operator message, which you can view in the Message Browser.

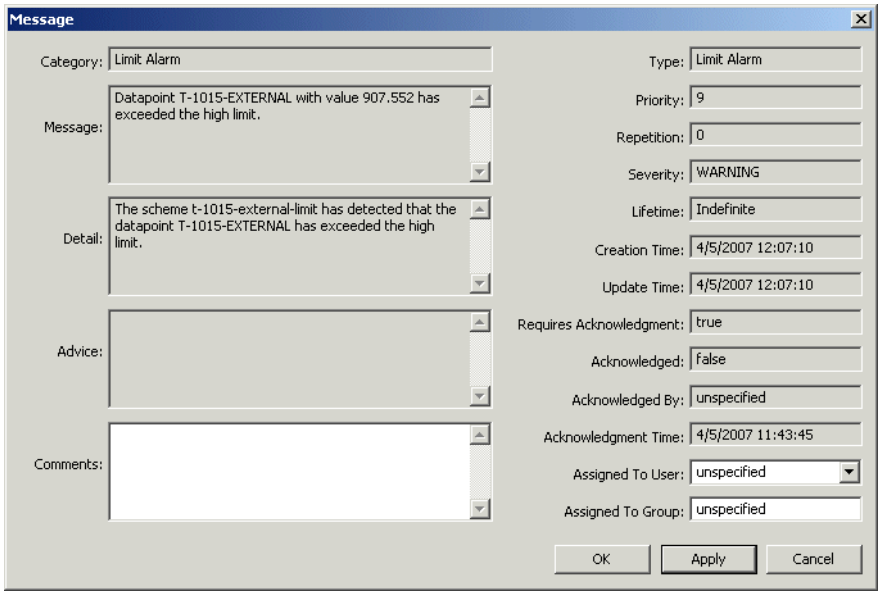
To view data validation alarms:

1. Run a simulation in which the simulated value for an external datapoint violates a data validation limit.
2. Choose View > Message Browser to view the data validation alarm.

Here is the Message Browser with a data validation message for the F-102 demo:



Here is the alarm properties:

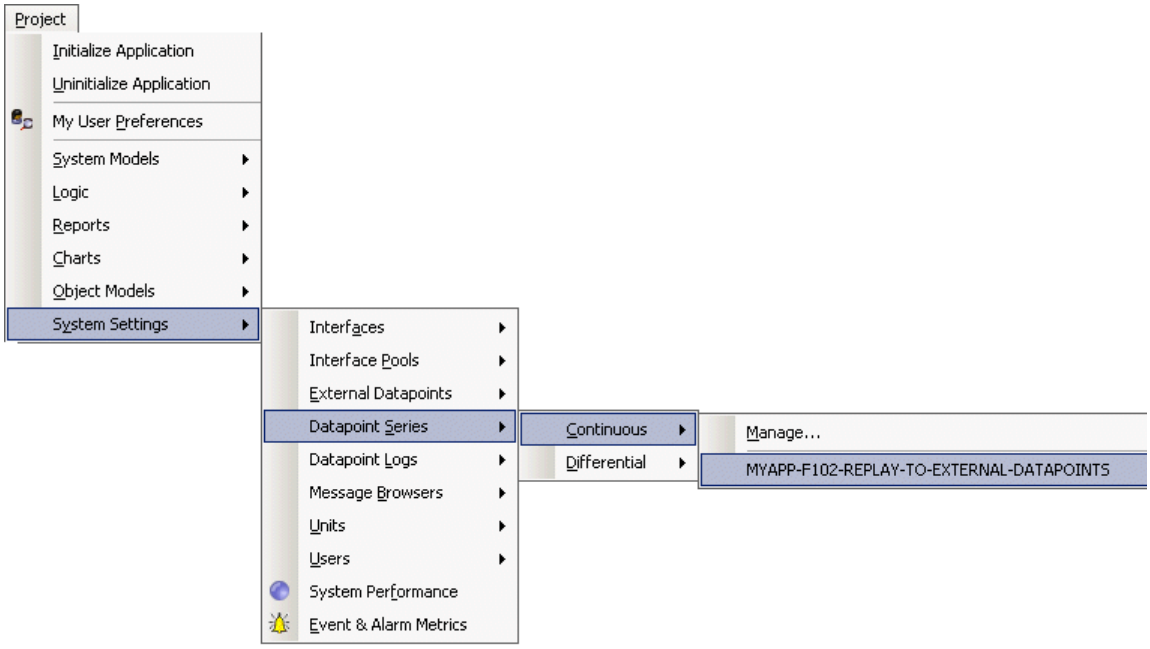


Managing Data Series

To manage data series:

1 Choose Project > System Settings > Datapoint Series > Continuous Data Series or Differential Data Series.

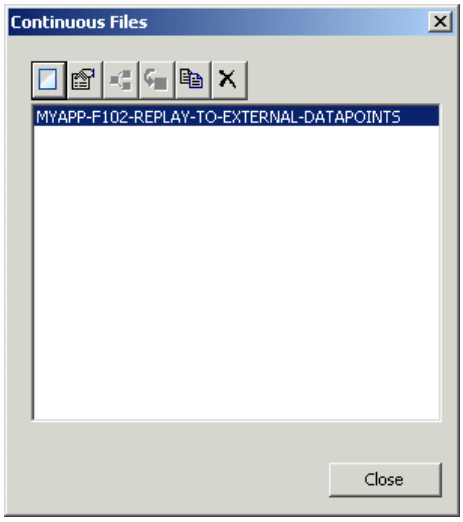
All data series appear in the submenu, for example:



2 To go to a data series, choose one from appropriate submenu of the Data Series menu.

3 To display a dialog for managing data files, choose Manage.

Here is the dialog for managing data series:

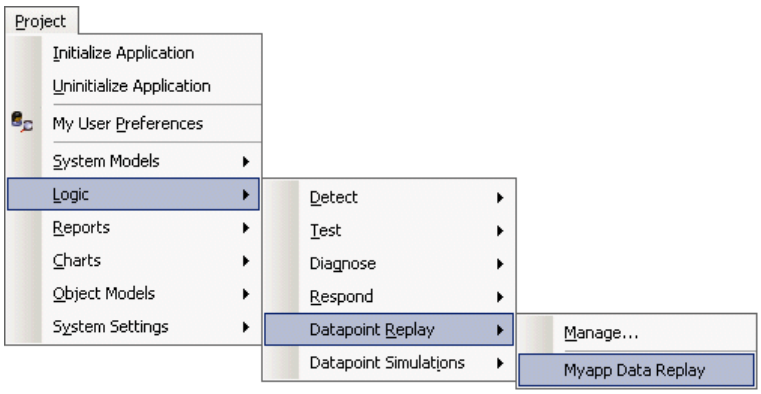


Managing Data Replay

To manage data replay:

1 Choose Project > Logic > Datapoint Replay.

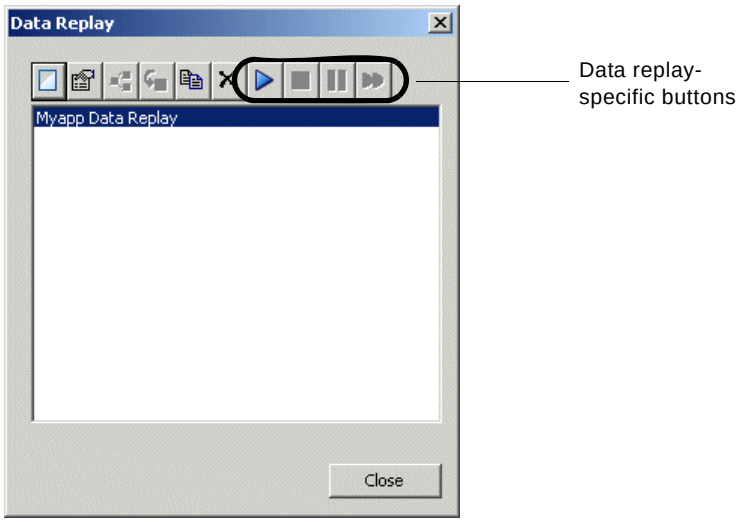
All Data Replay configuration objects appear in the submenu, for example:



2 To configure the properties of a data replay configuration object, choose one from the Data Replay submenu.

3 To display a dialog for managing data replay, choose Manage.

Here is the Data Replay Manage dialog:



6

Simulating External Datapoint Values

Describes how to simulate external datapoint values.

Introduction

You can use data drivers to simulate internal or external datapoint data. This feature provides an alternative to using data replay to simulate datapoint data from a CSV file.

The data driver provides the ability to:

- Simulate external or internal datapoint data.
- Specify the time interval for updating datapoint values.
- Specify the average value and noise.
- Simulate state transitions.
- Selectively activate and deactivate the simulation.

You can create one or more data drivers for individual datapoints.

Creating a Simple Data Simulation

Before you create a data simulation, you must first create either:

- A sensor or controller whose process value (pv), set point (sp), or controller output (op) you want to simulate.
- An external datapoint that is the source datapoint for the internal datapoint of any type of domain object.

The simplest way to simulate data is to create a single data driver that simulates datapoint values around an average, with noise. The data driver configures:

- The name of the sensor object and its associated datapoint (pv, sp, or op), or the name of the external datapoint whose values you want to simulate.
- The average value and the signal noise.

For information on creating external datapoints, see Chapter 3, "Configuring External Datapoints" on page 5.

To create a simple data simulation:

1. Create a sensor or controller whose datapoint you want to simulate, or create an external datapoint that is the source datapoint for an internal datapoint of any type of domain object.
2. Choose Project > Logic > Datapoint Simulations > Manage and click the New button.

   The properties dialog for configuring the Data Driver appears.
3. Configure the Organizer Name.
4. Click the New button ( ) to create a new data driver.

   The properties dialog for configuring the data driver appears.
5. Configure these attributes, depending on the type of data driver:

Attribute	Description
Datapoint Name	The name of an internal or external datapoint whose data should be simulated.
Active	Whether the simulation is currently active.
Average Value	The average value for the specified datapoint.

Attribute	Description
Noise	The maximum value above and below the average that represents noise in the current datapoint value.
Interval Seconds	The time interval, in seconds, for updating datapoint values.

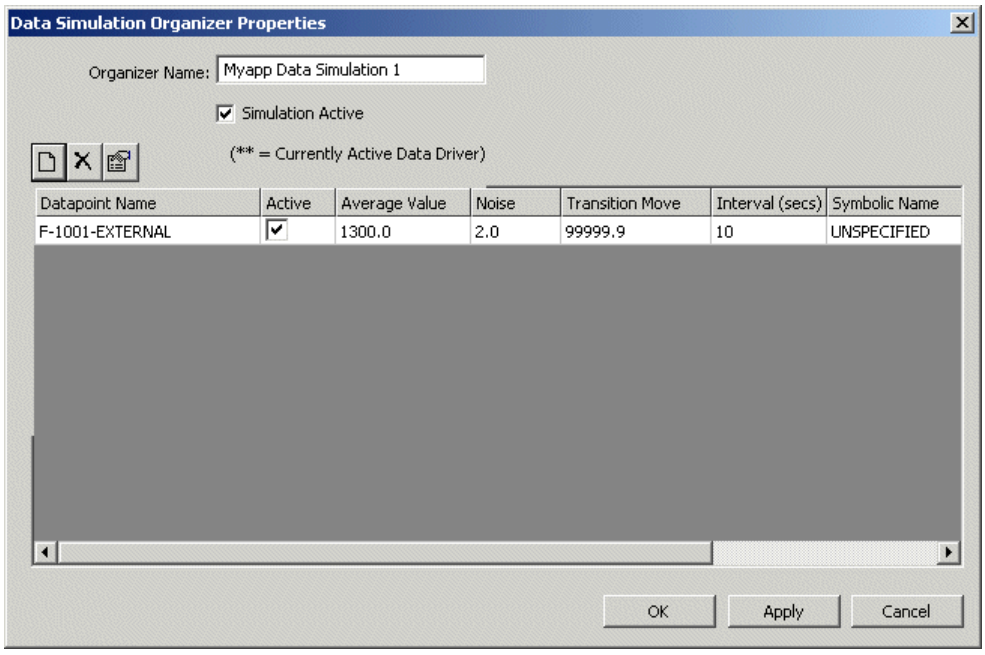
6 Create as many data drivers as you need.

The simulation is disabled, by default. You must explicitly enable it to simulate data.

7 Enable the Simulation Active toggle button and click the Apply button.

Tip You can also activate and deactivate the simulation from the Manage dialog by selecting a Data Simulation and clicking the Activate and Deactivate buttons.

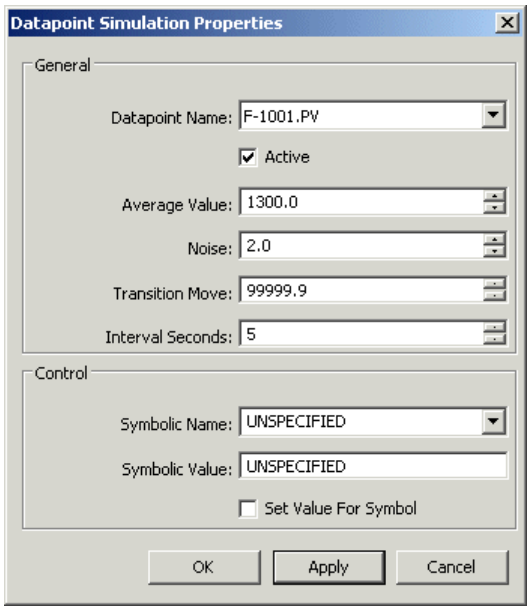
The properties dialog for the data simulation shows all associated data drivers. You can also delete a data driver from the data simulation and display its properties, using the Delete and Properties toolbar buttons. For example:



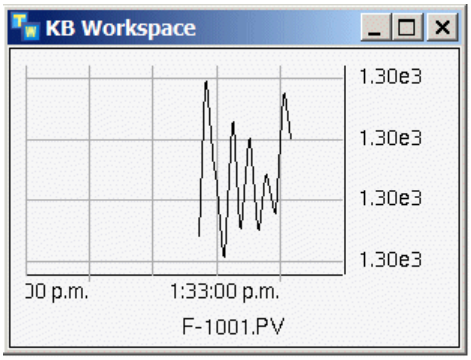
For information on configuring the other data driver attributes and on using the other toolbar buttons, see "Creating a Data Simulation with Transitions" on page 56.

Example: Internal Datapoint Simulation for a Sensor

This example shows a simple data simulation that is configured to simulate data for the pv of a flow sensor named f-1001. The data values update once every five seconds, and the simulated values range between 1300 +/- 2.

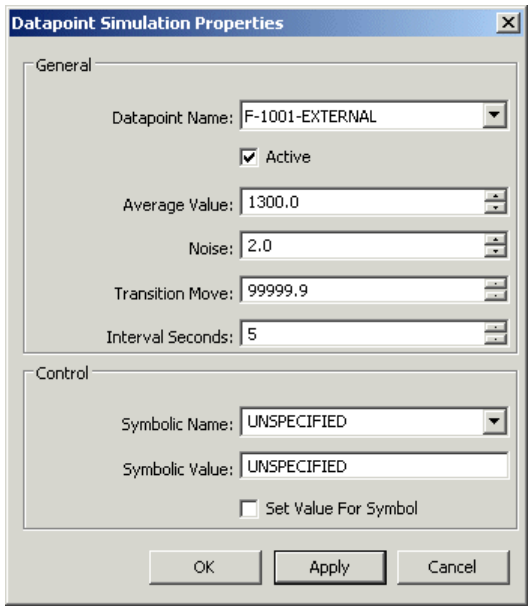


Here is the trend chart for the f-1001 sensor, which shows the history of the pv datapoint:

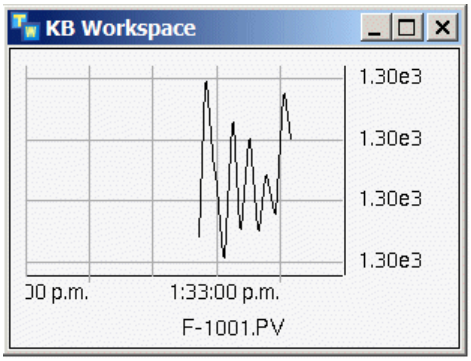


Example: External Datapoint Simulation for a Sensor

This example shows a simple data simulation for an external datapoint named f-1001-external, which is the source datapoint of the flow sensor named f-1001. The pv of the f-1001 sensor gets its data from the external datapoint. The data driver is configured to simulate data for the f-1001-external, using the same update interval, average, and noise as the internal data simulation.



Here is the properties dialog for the flow-sensor, which shows the history of the pv datapoint, which gets its data from the external datapoint named flow-dp:



Creating a Data Simulation with Transitions

Rather than creating a simple data simulation with a single data driver, you might want to create multiple data drivers for a single datapoint, each of which represents a transition from one state to another. For example, you might create a data driver that represents the normal state, another that represents a high state, another that represents a severe change, another that represents a noisy signal, and another that represents a flat-line.

You create data simulations with transitions by creating a data driver for each state, a symbolic parameter that represents the current state, and buttons for choosing each state. In addition to the basic configuration information, you must configure each data driver with the following information:

- The name of a symbolic parameter that determines the current state.
- The symbolic state associated with the data driver.
- The transition value to use when switching to the specified symbolic state.

Before you create a data simulation with transitions, you must first create the internal or external datapoint whose data you want to simulate. For details, see "Creating a Simple Data Simulation" on page 52.

To create a data simulation with transitions:

1. Create and configure a data driver for an internal or external datapoint that represents a normal state.

   For details, see "Creating a Simple Data Simulation" on page 52.

2. Configure these additional attributes for the data driver:

Attribute	Description
Transition Move	A value by which the datapoint should increment, either up or down, when making a state transition.
Symbolic Name	The name of a symbolic parameter that provides state transition values for the specified datapoint.
Symbolic Value	The value of the specified symbolic parameter associated with the specified state.

Note If the Symbolic Name does not exist, Optegrity creates a symbolic parameter of that name.

For example, you might specify Symbolic Name of flow-status, whose Symbolic Value is normal to represent the normal operating state of the datapoint.

The Transition Move is the value to increment the datapoint when transitioning back to a normal state from some other state. Typically, you specify a large transition, relative to the average values of each transition state, so as to return to the normal average relatively quickly.

For example, if the normal average is 100 and the high average is 120, a Transition Move of 10 would cause the datapoint value to return to the normal average in approximately two transitions, depending on the specified noise. On the other hand, a Transition Move of 2 would cause the datapoint value to return to normal in approximately ten transitions.

3 Create and configure a data driver for the same datapoint to represent the transition to another state.

For example, to simulate a high state, the Average Value might be 120, and the Symbolic Value might be high. The Transition Move determines how quickly the simulation moves from the normal state to the high state. To simulate a projected high, you would configure the transition to be a small increment, relative to the average values of each state. Thus, a Transition Move of 1.0 would cause the datapoint value to rise to the high average in approximately 20 steps.

4 Create and configure additional data drivers for the same datapoint to represent transitions to additional states, as needed.

For example, to simulate a severe change, configure the Transition Move to be a large number, relative to the average values of each state, such as 20, which would cause the datapoint value to spike to the specified average in one transition.

To simulate a noisy signal, configure the Noise to be a large number relative to the average, and to simulate a flat-line, configure the Noise to be zero. In both cases, you would configure the Average Value and Transition Move to be the same as the normal signal, 100.0 and 10.0, so the noisy and flat-line signal are based on a normal signal.

Tip In general, you should set the Noise to be less than the Transition Move; otherwise, the noise counteracts the transition and causes the datapoint value to take a long time to transition to the new state.

---

5 To set the current state, display the properties dialog for a particular data driver and enable the Set Value for Symbol option.

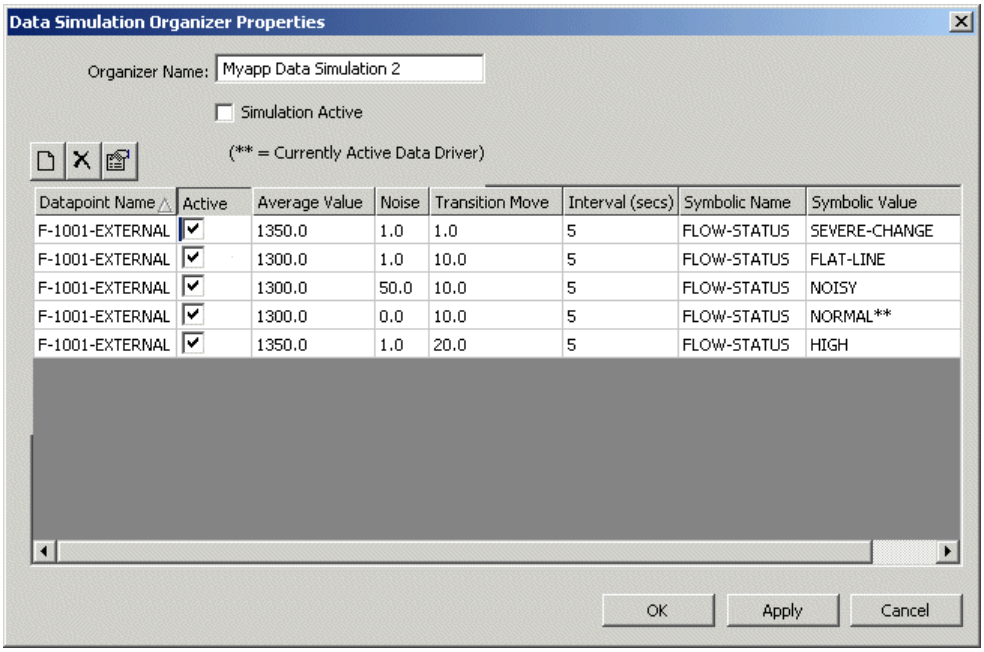
In the properties dialog for the Data Simulation, the Symbolic Value has ** next to the value to indicate that it is the currently active data driver.

6 Enable the Simulation Active toggle button in the Data Simulation properties dialog to activate the data drivers, and click OK or Apply.

The data driver with the default state determines the datapoint values to use for the simulation.

Example: External Datapoint Simulation with Transitions

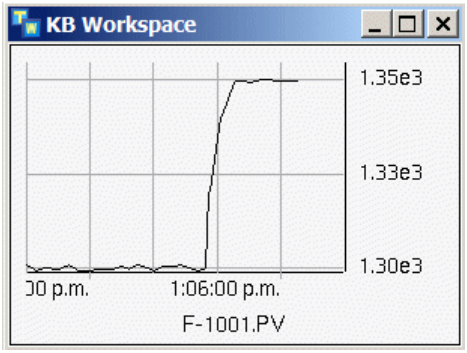
This example shows a data simulation for an external datapoint with five states for normal, high, severe change, noisy, and flat-line, where the current state is normal:



This table summarizes the values of the relevant attributes for each data driver:

	Normal	High	Severe Change	Noisy	Flat-Line
Symbolic Parameter Value	normal	high	severe-change	noisy	flat-line
Average Value	1300.0	1350.0	1350.0	1300.0	1300.0
Noise	1.0	1.0	1.0	50.0	0.0
Transition Move	10.0	1.0	20.0	10.0	10.0

Here is a trend chart for the f-1001 sensor, which shows the pv history for a normal state that transitioned to a severe change state:

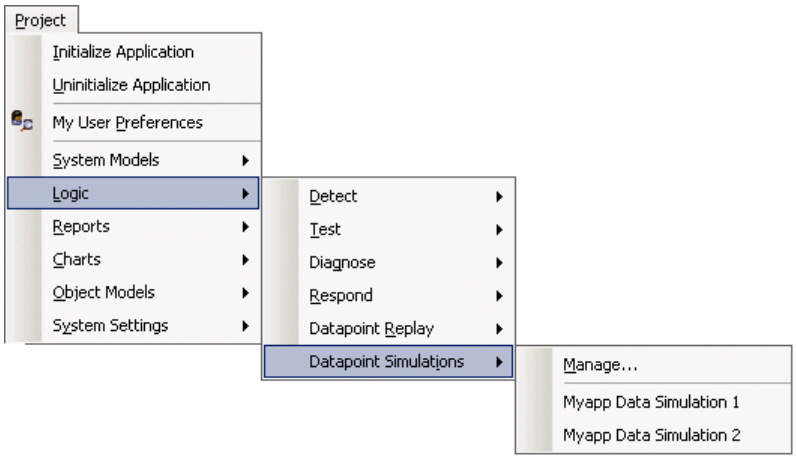


Managing Data Simulations

To manage data simulations:

1 Choose Project > Logic > Datapoint Simulations.

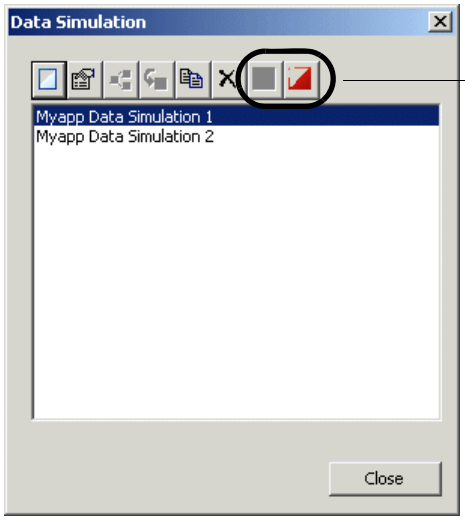
All data simulations appear in the submenu, for example:



2 To configure the properties of a data simulation, choose one from the Data Simulations submenu.

3 To display a dialog for managing all data simulations, choose Manage.

Here is the Data Simulation Manage dialog:



7

Custom Data Source Integration

Describes how to create custom data sources.

Introduction

GDPM supports custom data source integration for PLC or DCS systems other than OPC and PI, which are built into GDPM. Custom network interfaces support the same functionality as the built-in network interfaces, namely creating external datapoints from a specification in a CSV file, relating those datapoints to internal datapoints, and replaying datapoint from a CSV file.

Creating a custom data source requires the G2 Data Source Manager (GDSM) and G2 Run-Time Library (GRTL) modules. Here are the high-level steps for creating a custom data source.

Creating a Custom Data Source

To create a custom data source:

1. Create a new KB that requires the GDSM and GRTL modules to contain the custom data source and external datapoint definitions.
2. Create a GDSM network interface class and implement its methods.

   For details, see "Creating the Custom Network Interface Class" on page 62.
3. Create GRTL external datapoint classes of the required types and implement their methods.

   For details, see "Creating the Custom Network Interface Class" on page 62.
4. Create a CSV template file to import the external datapoints.

Creating the Custom Network Interface Class

The network interface class definition must inherit from gdsm-dcs-interface. The class can also multiply inherit from the built-in OPC or PI interface classes, gdsm-opc-interface or gdsm-pi-interface.

You must implement the following method for your custom network interface class:

gdpm-get-external-datapoint-class-name
(*io*: class gdsm-external-system-interface, *datapoint-type*: symbol)
–> *external-datapoint*: symbol

Returns a valid grtl-external-datapoint subclass given a GDSM network interface and a datapoint type. Values for *datapoint-type* are: real, float, integer, text, logical, or digital.

You can optionally implement the following method for your custom network interface class:

gdsm-network-interface-configure
(*io*: class gdsm-external-system-interface,
network-pool: class gdsm-network-connection-pool)

Configures a GDSM network interface, using a network pool. A network pool can contain multiple network interface objects on its detail. When allocating resources for load balancing, GDPM chooses a network interface, based on availability.

gdsm-network-interface-get-status
(io: class gdsm-external-system-interface)
–> *connection-status*: symbol

Determines the status of the network connection between a GDSM network interface and the external bridge process, refreshes the icon of the interface, based on the status, and returns the status. The return values are one of these symbols: connected, not-connected, in-transition, timed-out, or connection-lost.

gdsm-network-interface-connect
(*io*: class gdsm-external-system-interface, *host*: text, *port*: integer, *connection-timeout*: integer)

Connects a GDSM network interface to an external bridge process, given a host and port. The connection times out after *connection-timeout* seconds.

gdsm-network-interface-disconnect
(*io*: class gdsm-external-system-interface)

Disconnects a GDSM network interface from the bridge process.

gdsm-network-interface-animate
(*io*: class gdsm-external-system-interface, *allocated*: truth-value)

Animates a GDSM network interface as it gets allocated and deallocated for communication via the bridge, where *allocated* is true when the interface is allocated for communication.

grtl-show-properties
(*item*: class gdsm-external-system-interface,
client: class ui-client-item)
–> *exists*: truth-value

Displays the properties dialogs of a GDSM network interface in a given client window. The method returns true if the interface exists; otherwise, it returns false.

gdsm-network-interface-handle-connection-timeout
(*io*: class gdsm-external-system-interface)

Implements the connection timeout behavior of a GDSM network interface.

gdsm-network-interface-handle-connection-failure
(*io*: class gdsm-external-system-interface)

Implements the connection failure behavior of a GDSM network interface.

Creating Custom External Datapoint Classes

When creating a custom external datapoint class, we recommend inheriting from these classes:

- gsi-data-service, gdpm-io-variable, variable-or-parameter
- The specific GRTL external datapoint type: grtl-external-float-datapoint, grtl-external-integer-datapoint, grtl-external-text-datapoint, grtl-external-symbolic-datapoint, or grtl-external-logical-datapoint.

Your custom class should define attributes to uniquely identity variables and their associations in the control system.

You must implement the following methods for your external datapoint classes:

gdpm-io-configure-variable
(*io-variable*: class gdpm-opc-variable, *block*: class gdpm-io-block, *tokens*: sequence, *client*: class ui-client-item)

Creates and configures external datapoints from a CSV file and places them on the detail of an External Datapoints container. The method should extract specific configuration information from the sequences derived from the CSV file and assign them to the newly created external datapoints.

grtl-show-properties
(*item*: class gdpm-opc-variable, *client*: class ui-client-item)
–> truth-value

Displays the properties dialog of an external datapoint. The method returns true if the user presses OK to exit the dialog, and false otherwise.

Example: TDC Data Source Integration

Here are the key classes and methods for defining a custom DCS interface class and associated external datapoints for a TDC system.

The complete example is available in `gdpm-demo.kb`, which is located in the `g2i examples` directory.

Custom Network Interface Class

Here is a custom DCS interface class for interfacing with a TDC system:

GDPM-G2TDC-GATEWAY-INTERFACE

Direct superior classes gdsm-dcs-interface

Class specific attributes g2tdc-configuration is an instance of a g2tdc-configuration, initially is an instance of a g2tdc-configuration;
g2tdc-stats is an instance of an item-list, initially is an instance of an item-list

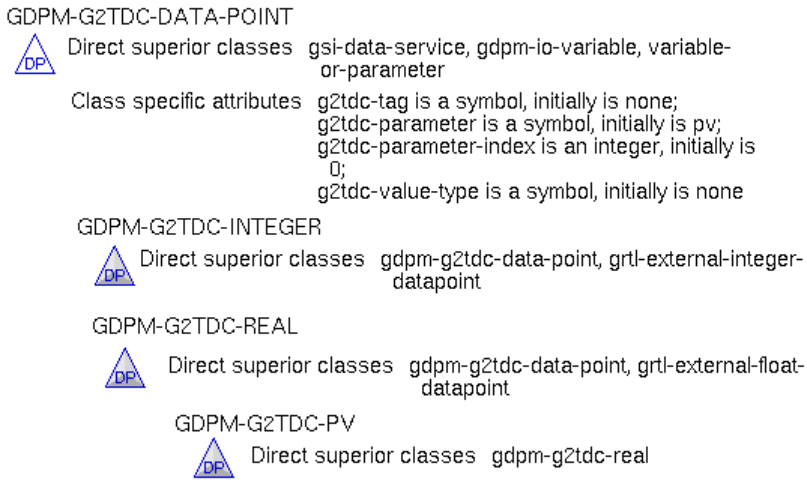
Here is the implementation of the gdpm-get-external-datapoint-class-name method for the custom TDC interface class:

gdpm-g2tdc-gateway-interface::gdpm-get-external-datapoint-class-name

```
gdpm-get-external-datapoint-class-name(io: class gdpm-g2tdc-gateway-interface ,
datapoint-type: symbol) = (symbol)
{
This method should return a valid grtl-external-datapoint subclass given the the io
interface type and the datapoint type. Values for datapoint-type maybe real, float,
integer, text, logical, or digital.
}
variable-class: symbol ;
begin
     case (datapoint-type) of
          REAL:  variable-class = the symbol gdpm-g2tdc-pv;
          FLOAT: variable-class = the symbol gdpm-g2tdc-pv;
          INTEGER: variable-class = the symbol gdpm-g2tdc-integer ;
          otherwise: variable-class = the symbol gdpm-g2tdc-pv ;
     end;
     return variable-class;
end
```

Custom External Datapoint Classes

Here is the class hierarchy for the custom external datapoint classes required by the TDC interface:



Here is the implementation of the method for the custom TDC interface class:


gdpm-g2tdc-data-point::gdpm-io-configure-variable

```
gdpm-io-configure-variable(io-variable: class gdpm-g2tdc-data-point , block: class
gdpm-io-block, tokens: sequence, win: class ui-client-item)

{
Configures the external datpoint TDC specific configurations based on the values
from the csv file. Starting at the index 24 within the tokens start the column in the
spreadsheet that are specific to each gsi interface. In this example we extract the tag
and the index.
}

bridge-name: symbol = the bridge-name of block;
interface: class gdpm-g2tdc-gateway-interface;
tag: symbol;
index: integer;

begin

    call next method;

    { --- Setup the TDC specific fields using the values from the csv file }

    tag = call grtl-uppercase-symbol( "[TOKENS[24]]" );
    if text-begins-with-quantity( "[TOKENS[25]]" ) then
        index = quantity( "[TOKENS[25]]" )
    else
        index = 0;
```

```
    conclude that the g2tdc-tag of io-variable = tag;
    conclude that the g2tdc-parameter-index of io-variable = index;

    { --- The interface object must exist, otherwise create it }

    if not(there exists a gdpm-g2tdc-gateway-interface interface named by
        bridge-name) then
        call gdpm-io-create-interface(block, the symbol
            gdpm-g2tdc-gateway-interface, win);

end
```

Here is the implementation of the method for the custom TDC interface class:

gdpm-g2tdc-data-point::grtl-show-properties

```
grtl-show-properties (Itm: class gdpm-g2tdc-data-point, Client: class ui-client-item) =
(truth-value)

{
This method will open the property dialogs of an object if defined
}

ret: truth-value = true;
Dlg: item-or-value;
Btn: item-or-value;

begin
    Dlg, Btn = call uil-control-dialog-callback
        ("gdpm-external-datapoint-tdc-configuration-dialog" , the symbol none,
        gdpm-external-datapoint-tdc-variable-configuration-dialog-actions,
        Itm, Client);

    if Btn exists and Btn is an uil-button and the label of Btn = "OK" then ret =
        true;
    if Dlg exists and Dlg is a uil-dialog then call
        grtl-cleanup-and-release-dialog (Dlg, Client);
    return ret;

end
```

To view the properties dialog definition, see the `gdpm-demo.kb` example KB.

Index

E

G

H

I

L

M

N

O

P

R

S

T

V